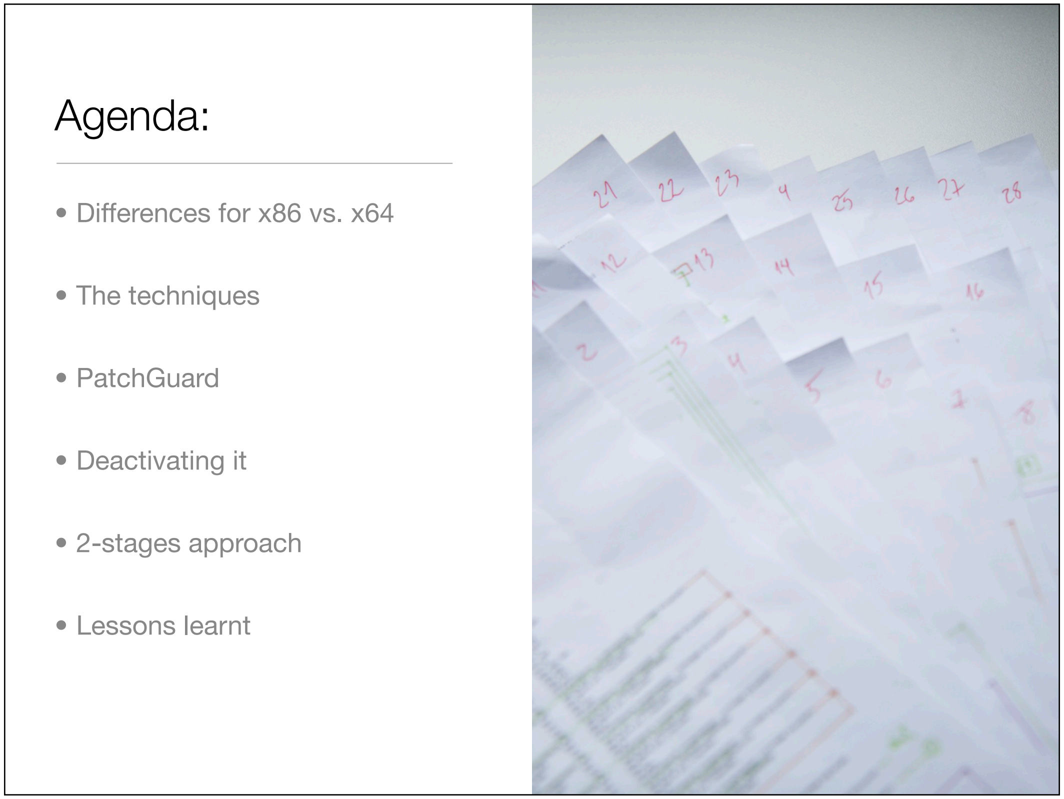
0-RING x64

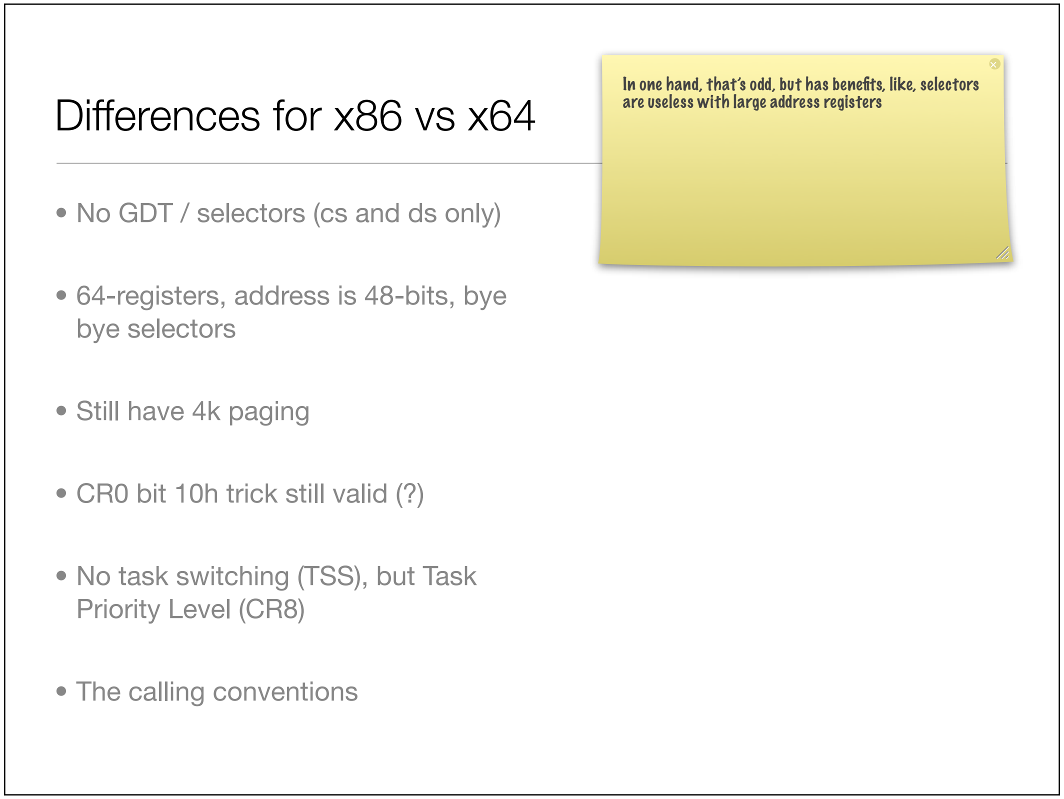
Gustavo Scotti,
Immunity, Inc.



Agenda:

- Differences for x86 vs. x64
- The techniques
- PatchGuard
- Deactivating it
- 2-stages approach
- Lessons learnt

Photo: Organize the prints with index numbers shown altogether.



Differences for x86 vs x64

In one hand, that's odd, but has benefits, like, selectors are useless with large address registers

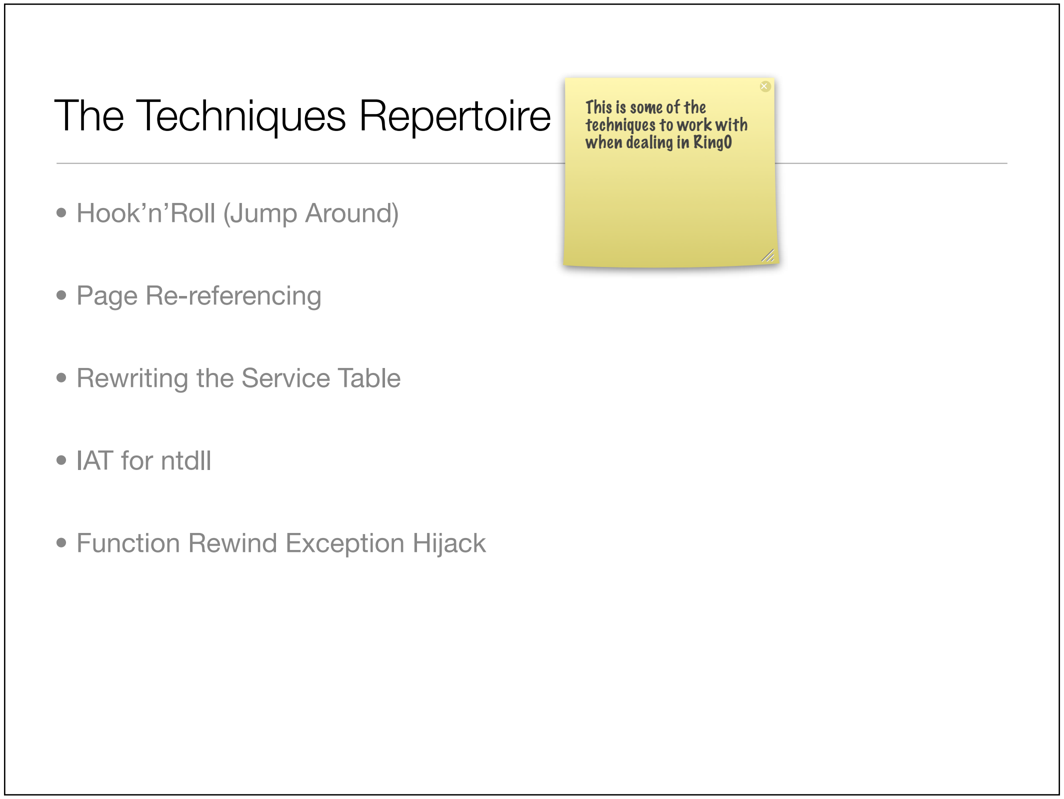
- No GDT / selectors (cs and ds only)
- 64-registers, address is 48-bits, bye bye selectors
- Still have 4k paging
- CR0 bit 10h trick still valid (?)
- No task switching (TSS), but Task Priority Level (CR8)
- The calling conventions

Silly joke: “To operate in 64-bits, you would need twice as much RAM.” (unknown source)
Calling conventions: introduce the subject, how Intel suggested to use rcx/rdx/r8.../ then stack.
    The challenge to debug and doing stack backtrace (hooray to windbg)
Selectors: Why do I hate selectors so much? Selectors are friendly fire for those in the development front. Context switches and gdt change hurt my balls off.

Why insisting on paging? Nice speech to make the time runs up. Explain how the mapping process is done, and raise the question why Intel allows 4k, 2M and 4M pages?

Code Signing

- In 64-bit, all drivers need to be signed
- PnP has a mandatory catalog (.cat) companion, which is also signed
- Non-PnP or boot drivers, embedded signature is enough
- Have you bought your ticket to the Ring-0?



The Techniques Repertoire

This is some of the techniques to work with when dealing in Ring0

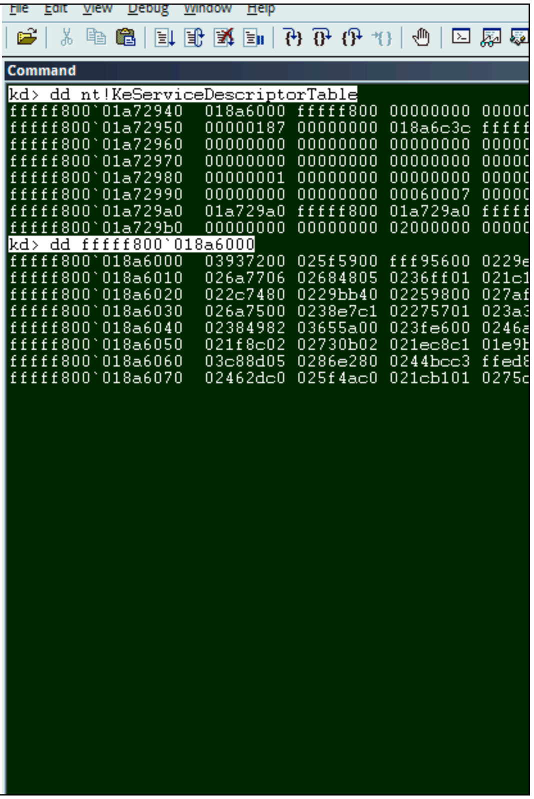
- Hook’n’Roll (Jump Around)
- Page Re-referencing
- Rewriting the Service Table
- IAT for ntdll
- Function Rewind Exception Hijack

Silly joke for “Hook’n’Roll”... “Sing the rap ‘Jump Around’ from House of Pain”

Explain the IAT and why it is so important. Dll loading is also a good call.

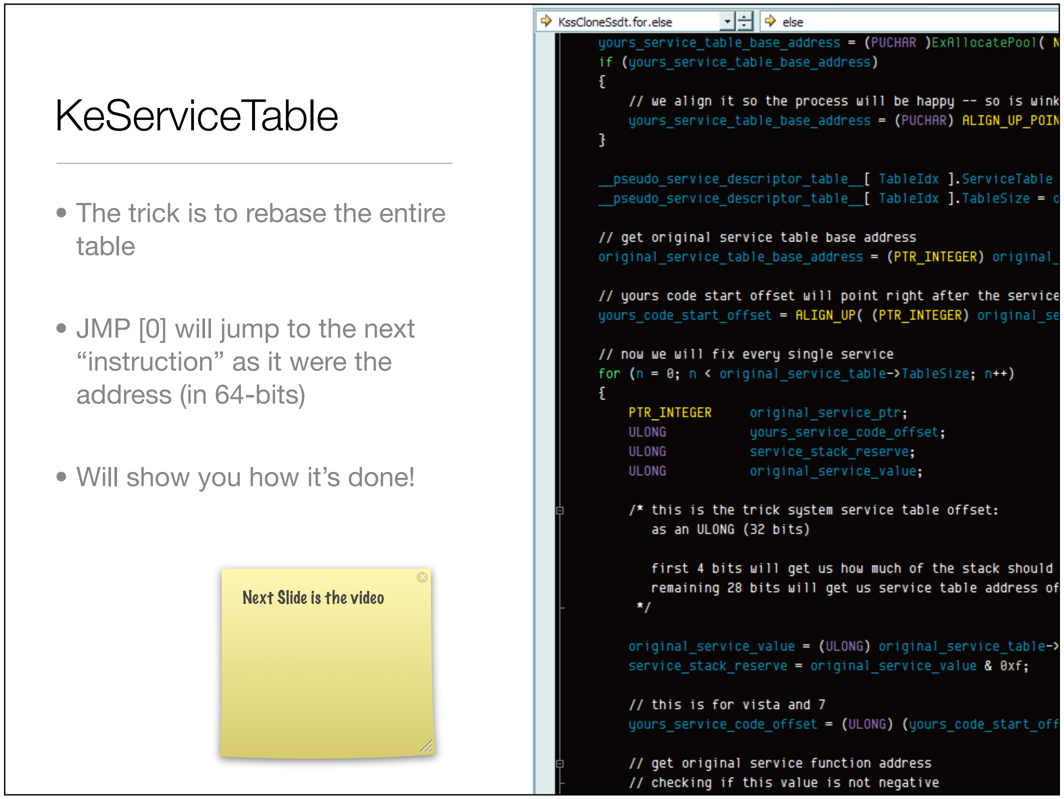
KeServiceTable

- #1 choice of 32-bitters
- X64 uses a different approach
- Offset to the service is 32-bit
- Base address is the table
- 0x10 aligned, last 4 bits used to stack alignment



```
kd> dd nt!KeServiceDescriptorTable
fffff800`01a72940  018a6000 fffff800 00000000 00000
fffff800`01a72950  00000187 00000000 018a6c3c fffff
fffff800`01a72960  00000000 00000000 00000000 00000
fffff800`01a72970  00000000 00000000 00000000 00000
fffff800`01a72980  00000001 00000000 00000000 00000
fffff800`01a72990  00000000 00000000 00060007 00000
fffff800`01a729a0  01a729a0 fffff800 01a729a0 fffff
fffff800`01a729b0  00000000 00000000 02000000 00000
kd> dd fffff800`018a6000
fffff800`018a6000  03937200 025f5900 fff95600 0229e
fffff800`018a6010  026a7706 02684805 0236ff01 021c1
fffff800`018a6020  022c7480 0229bb40 02259800 027af
fffff800`018a6030  026a7500 0238e7c1 02275701 023a3
fffff800`018a6040  02384982 03655a00 023fe600 0246a
fffff800`018a6050  021f8c02 02730b02 021ec8c1 01e9b
fffff800`018a6060  03c88d05 0286e280 0244bcc3 ffed8
fffff800`018a6070  02462dc0 025f4ac0 021cb101 0275c
```

Why is the service table the #1 choice? Explain the export table in PE format and why not having the table is bad to figure out function addresses and it’s painful to signature process for public symbol addresses.



KeServiceTable

- The trick is to rebase the entire table
- JMP [0] will jump to the next “instruction” as it were the address (in 64-bits)
- Will show you how it’s done!

Next Slide is the video

```
    yours_service_table_base_address = (PUCHAR )ExAllocatePool( N
    if (yours_service_table_base_address)
    {
        // we align it so the process will be happy -- so is wink
        yours_service_table_base_address = (PUCHAR) ALIGN_UP_POIN
    }

    __pseudo_service_descriptor_table__[ TableIdx ].ServiceTable
    __pseudo_service_descriptor_table__[ TableIdx ].TableSize = o

    // get original service table base address
    original_service_table_base_address = (PTR_INTEGER) original_

    // yours code start offset will point right after the service
    yours_code_start_offset = ALIGN_UP( (PTR_INTEGER) original_se

    // now we will fix every single service
    for (n = 0; n < original_service_table->TableSize; n++)
    {
        PTR_INTEGER     original_service_ptr;
        ULONG           yours_service_code_offset;
        ULONG           service_stack_reserve;
        ULONG           original_service_value;

        /* this is the trick system service table offset:
           as an ULONG (32 bits)

           first 4 bits will get us how much of the stack should
           remaining 28 bits will get us service table address of
         */

        original_service_value = (ULONG) original_service_table->
        service_stack_reserve = original_service_value & 0xf;

        // this is for vista and 7
        yours_service_code_offset = (ULONG) (yours_code_start_off

        // get original service function address
        // checking if this value is not negative
```

Next slide is the video copying the service table and having the driver running.
Silly joke: “your debit card is about to be charged $100 bucks for this exhibition (DRM protected).
Later link the Patchguard protection mechanism, linked to the DRM modules.

```
{
    // we align it so the process will be happy -- so is winkernel
    yours_service_table_base_address = (PUCHAR) ALIGN_UP_POINTER( yours_ser
```

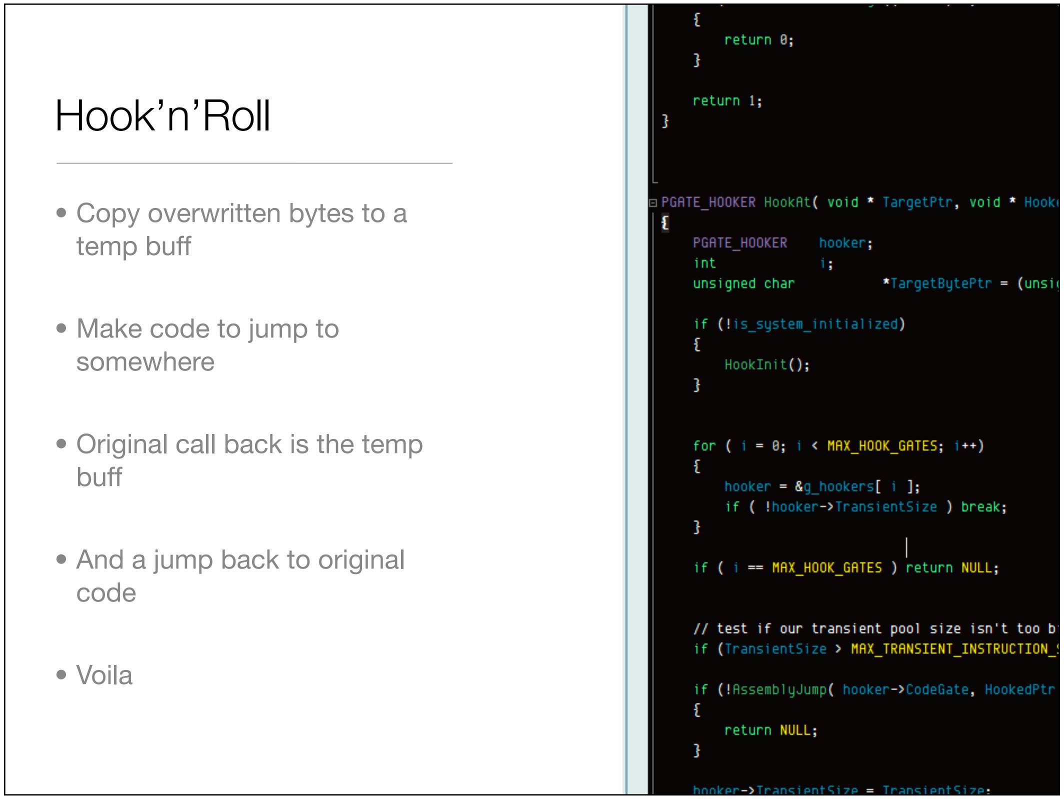
Windows Vista and Windows Server 2008 x64 Checked Build Environment

```
Z:\h2hc\sample1>
```

```
    // this is for vista and 7
    yours_service_code_offset = (ULONG) (yours_code_start_offset + ( * SERVICE
```

KeServiceTable - In Action

Comment the video as it plays.
You can do some more silly jokes, like “See me, without hands!”
Or shout “oops..” when mistypes is played back.



Hook’n’Roll

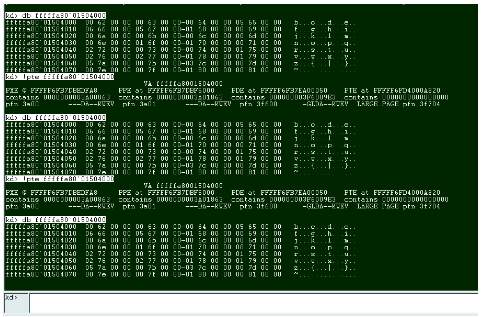
- Copy overwritten bytes to a temp buff
- Make code to jump to somewhere
- Original call back is the temp buff
- And a jump back to original code
- Voila

```
    {
        return 0;
    }

    return 1;
}


PGATE_HOOKER HookAt( void * TargetPtr, void * Hooke
{
    PGATE_HOOKER    hooker;
    int             i;
    unsigned char           *TargetBytePtr = (unsi

    if (!is_system_initialized)
    {
        HookInit();
    }


    for ( i = 0; i < MAX_HOOK_GATES; i++)
    {
        hooker = &g_hookers[ i ];
        if ( !hooker->TransientSize ) break;
    }

    if ( i == MAX_HOOK_GATES ) return NULL;


    // test if our transient pool size isn't too b
    if (TransientSize > MAX_TRANSIENT_INSTRUCTION_

    if (!AssemblyJump( hooker->CodeGate, HookedPtr
    {
        return NULL;
    }

    hooker->TransientSize = TransientSize;
```

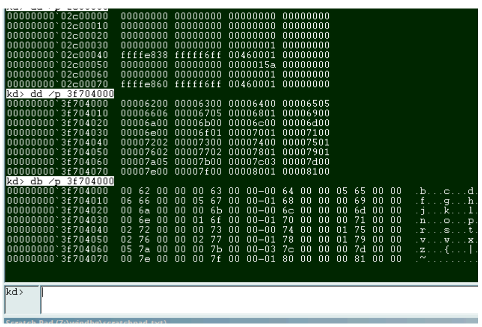
Explain this is as old as assembler exists.
Or another silly joke: “as old as the real Rock’n’Roll, kid”.
TODO: You can make an animation for this thing. Amuse us, dude!

Page Referencing

- Figure out where in physical memory it is
- You can remap the same physical address
- You can copy your content to another physical memory
- Reference a virtual address to another physical page
- That's the PAGE fun!



```
kd> db fffffa80`01504000
fffffa80`01504000  00 62 00 00 00 63 00 00-00 64 00 00 05 65 00 00  .b...c...d...e..
fffffa80`01504010  06 66 00 00 05 67 00 00-01 68 00 00 00 69 00 00  .f...g...h...i..
fffffa80`01504020  00 6a 00 00 00 6b 00 00-00 6c 00 00 00 6d 00 00  .j...k...l...m..
fffffa80`01504030  00 6e 00 00 01 6f 00 00-01 70 00 00 00 71 00 00  .n...o...p...q..
fffffa80`01504040  02 72 00 00 00 73 00 00-00 74 00 00 01 75 00 00  .r...s...t...u..
fffffa80`01504050  02 76 00 00 02 77 00 00-01 78 00 00 01 79 00 00  .v...w...x...y..
fffffa80`01504060  05 7a 00 00 00 7b 00 00-03 7c 00 00 00 7d 00 00  .z...{...|...}..
fffffa80`01504070  00 7e 00 00 00 7f 00 00-01 80 00 00 00 81 00 00  .~..............
kd> !pte fffffa80`01504000
                                           VA fffffa8001504000
PXE @ FFFFF6FB7DBEDFA8     PPE at FFFFF6FB7DBF5000    PDE at FFFFF6FB7EA00050    PTE at FFFFF6FD4000A820
contains 000000003A00863  contains 000000003A01863  contains 000000003F6009E3  contains 0000000000000000
pfn 3a00       ---DA--KWEV  pfn 3a01       ---DA--KWEV  pfn 3f600      -GLDA--KWEV  LARGE PAGE pfn 3f704

kd> db fffffa80`01504000
fffffa80`01504000  00 62 00 00 00 63 00 00-00 64 00 00 05 65 00 00  .b...c...d...e..
fffffa80`01504010  06 66 00 00 05 67 00 00-01 68 00 00 00 69 00 00  .f...g...h...i..
fffffa80`01504020  00 6a 00 00 00 6b 00 00-00 6c 00 00 00 6d 00 00  .j...k...l...m..
fffffa80`01504030  00 6e 00 00 01 6f 00 00-01 70 00 00 00 71 00 00  .n...o...p...q..
fffffa80`01504040  02 72 00 00 00 73 00 00-00 74 00 00 01 75 00 00  .r...s...t...u..
fffffa80`01504050  02 76 00 00 02 77 00 00-01 78 00 00 01 79 00 00  .v...w...x...y..
fffffa80`01504060  05 7a 00 00 00 7b 00 00-03 7c 00 00 00 7d 00 00  .z...{...|...}..
fffffa80`01504070  00 7e 00 00 00 7f 00 00-01 80 00 00 00 81 00 00  .~..............
kd> !pte fffffa80`01504000
                                           VA fffffa8001504000
PXE @ FFFFF6FB7DBEDFA8     PPE at FFFFF6FB7DBF5000    PDE at FFFFF6FB7EA00050    PTE at FFFFF6FD4000A820
contains 000000003A00863  contains 000000003A01863  contains 000000003F6009E3  contains 0000000000000000
pfn 3a00       ---DA--KWEV  pfn 3a01       ---DA--KWEV  pfn 3f600      -GLDA--KWEV  LARGE PAGE pfn 3f704

kd> db fffffa80`01504000
fffffa80`01504000  00 62 00 00 00 63 00 00-00 64 00 00 05 65 00 00  .b...c...d...e..
fffffa80`01504010  06 66 00 00 05 67 00 00-01 68 00 00 00 69 00 00  .f...g...h...i..
fffffa80`01504020  00 6a 00 00 00 6b 00 00-00 6c 00 00 00 6d 00 00  .j...k...l...m..
fffffa80`01504030  00 6e 00 00 01 6f 00 00-01 70 00 00 00 71 00 00  .n...o...p...q..
fffffa80`01504040  02 72 00 00 00 73 00 00-00 74 00 00 01 75 00 00  .r...s...t...u..
fffffa80`01504050  02 76 00 00 02 77 00 00-01 78 00 00 01 79 00 00  .v...w...x...y..
fffffa80`01504060  05 7a 00 00 00 7b 00 00-03 7c 00 00 00 7d 00 00  .z...{...|...}..
fffffa80`01504070  00 7e 00 00 00 7f 00 00-01 80 00 00 00 81 00 00  .~..............

kd>
```

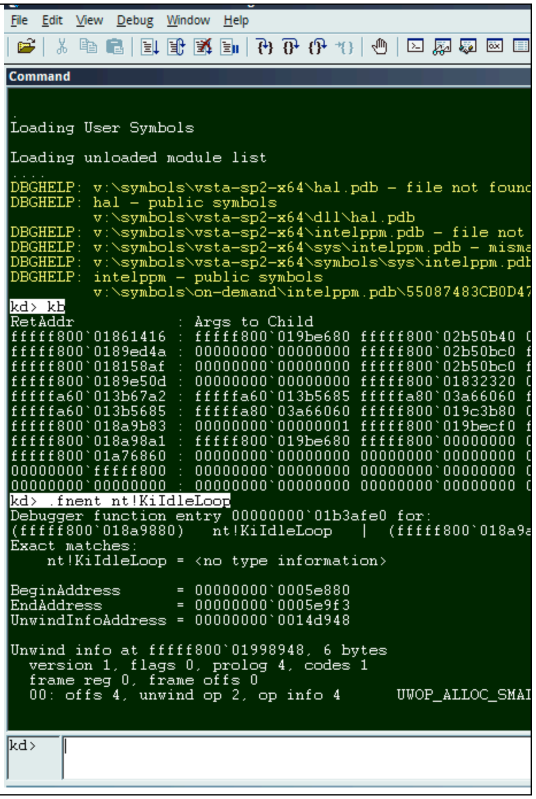


```
00000000`02c00000  00000000 00000000 00000000 00000000
00000000`02c00010  00000000 00000000 00000000 00000000
00000000`02c00020  00000000 00000000 00000000 00000000
00000000`02c00030  00000000 00000000 00000001 00000000
00000000`02c00040  ffffe838 fffff6ff 00460001 00000000
00000000`02c00050  00000000 00000000 0000015a 00000000
00000000`02c00060  00000000 00000000 00000001 00000000
00000000`02c00070  ffffe860 fffff6ff 00460001 00000000
kd> dd /p 3f704000
00000000`3f704000  00006200 00006300 00006400 00006505
00000000`3f704010  00006606 00006705 00006801 00006900
00000000`3f704020  00006a00 00006b00 00006c00 00006d00
00000000`3f704030  00006e00 00006f01 00007001 00007100
00000000`3f704040  00007202 00007300 00007400 00007501
00000000`3f704050  00007602 00007702 00007801 00007901
00000000`3f704060  00007a05 00007b00 00007c03 00007d00
00000000`3f704070  00007e00 00007f00 00008001 00008100
kd> db /p 3f704000
00000000`3f704000  00 62 00 00 00 63 00 00-00 64 00 00 05 65 00 00  .b...c...d.
00000000`3f704010  06 66 00 00 05 67 00 00-01 68 00 00 00 69 00 00  .f...g...h.
00000000`3f704020  00 6a 00 00 00 6b 00 00-00 6c 00 00 00 6d 00 00  .j...k...l.
00000000`3f704030  00 6e 00 00 01 6f 00 00-01 70 00 00 00 71 00 00  .n...o...p.
00000000`3f704040  02 72 00 00 00 73 00 00-00 74 00 00 01 75 00 00  .r...s...t.
00000000`3f704050  02 76 00 00 02 77 00 00-01 78 00 00 01 79 00 00  .v...w...x.
00000000`3f704060  05 7a 00 00 00 7b 00 00-03 7c 00 00 00 7d 00 00  .z...{...|.
00000000`3f704070  00 7e 00 00 00 7f 00 00-01 80 00 00 00 81 00 00  .~.........

kd>
```

PLEASE FINISH ME!!!

Exception Handler

- Calling Convention for x64
- How to backtrace?
- Several general purpose regs
- The opposite happens: regs are stored in local stack



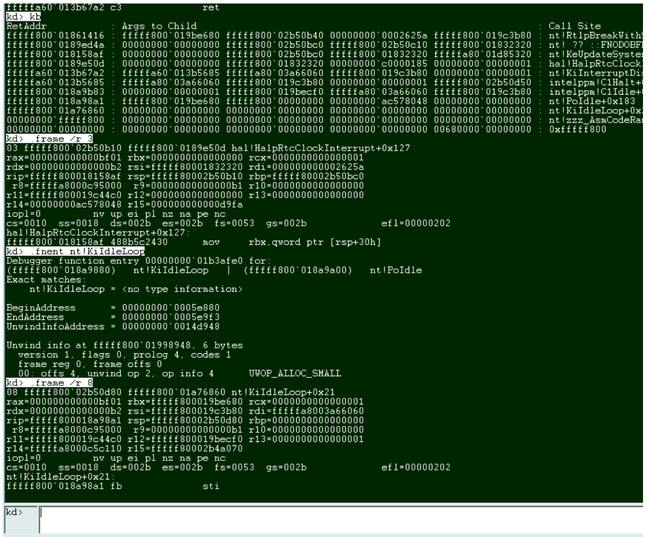
```
.
Loading User Symbols

Loading unloaded module list
....
DBGHELP: v:\symbols\vsta-sp2-x64\hal.pdb - file not found
DBGHELP: hal - public symbols
         v:\symbols\vsta-sp2-x64\dll\hal.pdb
DBGHELP: v:\symbols\vsta-sp2-x64\intelppm.pdb - file not
DBGHELP: v:\symbols\vsta-sp2-x64\sys\intelppm.pdb - misma
DBGHELP: v:\symbols\vsta-sp2-x64\symbols\sys\intelppm.pdb
DBGHELP: intelppm - public symbols
         v:\symbols\on-demand\intelppm.pdb\55087483CB0D47
kd> kb
RetAddr           : Args to Child
fffff800`01861416 : fffff800`019be680 fffff800`02b50b40 0
fffff800`0189ed4a : 00000000`00000000 fffff800`02b50bc0 f
fffff800`018158af : 00000000`00000000 fffff800`02b50bc0 f
fffff800`0189e50d : 00000000`00000000 fffff800`01832320 0
fffffa60`013b67a2 : fffffa60`013b5685 fffffa80`03a66060 f
fffffa60`013b5685 : fffffa80`03a66060 fffff800`019c3b80 0
fffff800`018a9b83 : 00000000`00000001 fffff800`019becf0 f
fffff800`018a98a1 : fffff800`019be680 fffff800`00000000 0
fffff800`01a76860 : 00000000`00000000 00000000`00000000 0
00000000`fffff800 : 00000000`00000000 00000000`00000000 0
00000000`00000000 : 00000000`00000000 00000000`00000000 0
kd> .fnent nt!KiIdleLoop
Debugger function entry 00000000`01b3afe0 for:
(fffff800`018a9880)   nt!KiIdleLoop   |  (fffff800`018a9a
Exact matches:
    nt!KiIdleLoop = <no type information>

BeginAddress      = 00000000`0005e880
EndAddress        = 00000000`0005e9f3
UnwindInfoAddress = 00000000`0014d948

Unwind info at fffff800`01998948, 6 bytes
  version 1, flags 0, prolog 4, codes 1
  frame reg 0, frame offs 0
  00: offs 4, unwind op 2, op info 4      UWOP_ALLOC_SMAL
```

```
kd>
```

Explain how it works, and why it’s better than the stack model. And why the stack changes inside the code and how it can help in reversing x64 code.

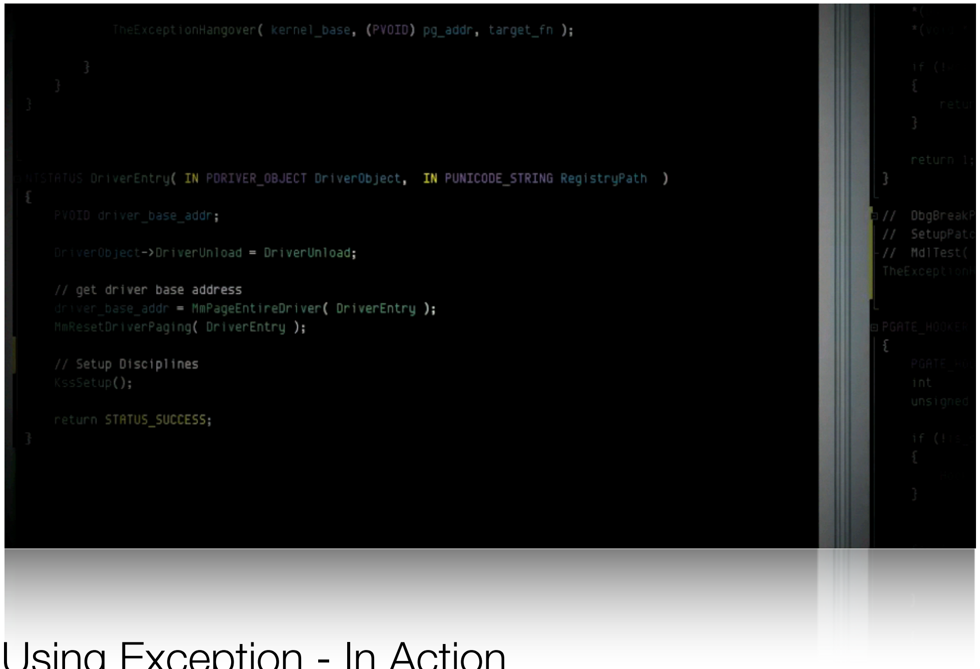
Exception Handler

- Unwind Info holds:
- Stacked saved parameters
- Internal stack changes at code flow
- Holds function Exception Handler for SEH
- Function information is inside PE’s Directory



```
fffffa60`013b67a2 c3              ret
kd> kb
RetAddr           : Args to Child                                                           : Call Site
fffff800`01861416 : fffff800`019be680 fffff800`02b50b40 00000000`0002625a fffff800`019c3b80 : nt!RtlpBreakWith
fffff800`0189ed4a : 00000000`00000000 fffff800`02b50bc0 fffff800`02b50c10 fffff800`01832320 : nt! ?? ::FNODOBF
fffff800`018158af : 00000000`00000000 fffff800`02b50bc0 fffff800`01832320 fffffa80`01d85320 : nt!KeUpdateSyste
fffff800`0189e50d : 00000000`00000000 fffff800`01832320 00000000`c0000185 00000000`00000001 : hal!HalpRtcClock
fffffa60`013b67a2 : fffffa60`013b5685 fffffa80`03a66060 fffff800`019c3b80 00000000`00000001 : nt!KiInterruptDi
fffffa60`013b5685 : fffffa80`03a66060 fffff800`019c3b80 00000000`00000001 fffff800`02b50d50 : intelppm!C1Halt+
fffff800`018a9b83 : 00000000`00000001 fffff800`019becf0 fffffa80`03a66060 fffff800`019c3b80 : intelppm!C1Idle+
fffff800`018a98a1 : fffff800`019be680 fffff800`00000000 00000000`ac578048 00000000`00000000 : nt!PoIdle+0x183
fffff800`01a76860 : 00000000`00000000 00000000`00000000 00000000`00000000 00000000`00000000 : nt!KiIdleLoop+0x
00000000`fffff800 : 00000000`00000000 00000000`00000000 00000000`00000000 00000000`00000000 : nt!zzz_AsmCodeRa
00000000`00000000 : 00000000`00000000 00000000`00000000 00000000`00000000 00680000`00000000 : 0xfffff800
kd> .frame /r 3
03 fffff800`02b50b10 fffff800`0189e50d hal!HalpRtcClockInterrupt+0x127
rax=000000000000bf01 rbx=0000000000000000 rcx=0000000000000001
rdx=00000000000000b2 rsi=fffff80001832320 rdi=000000000002625a
rip=fffff800018158af rsp=fffff80002b50b10 rbp=fffff80002b50bc0
 r8=fffffa8000c95000  r9=00000000000000b1 r10=0000000000000000
r11=fffff800019c44c0 r12=0000000000000000 r13=0000000000000000
r14=00000000ac578048 r15=00000000000d9fa
iopl=0         nv up ei pl nz na pe nc
cs=0010  ss=0018  ds=002b  es=002b  fs=0053  gs=002b             efl=00000202
hal!HalpRtcClockInterrupt+0x127:
fffff800`018158af 488b5c2430      mov     rbx,qword ptr [rsp+30h]
kd> .fnent nt!KiIdleLoop
Debugger function entry 00000000`01b3afe0 for:
(fffff800`018a9880)   nt!KiIdleLoop   |  (fffff800`018a9a00)   nt!PoIdle
Exact matches:
    nt!KiIdleLoop = <no type information>

BeginAddress      = 00000000`0005e880
EndAddress        = 00000000`0005e9f3
UnwindInfoAddress = 00000000`0014d948

Unwind info at fffff800`01998948, 6 bytes
  version 1, flags 0, prolog 4, codes 1
  frame reg 0, frame offs 0
  00: offs 4, unwind op 2, op info 4      UWOP_ALLOC_SMALL
kd> .frame /r 8
08 fffff800`02b50d80 fffff800`01a76860 nt!KiIdleLoop+0x21
rax=000000000000bf01 rbx=fffff800019be680 rcx=0000000000000001
rdx=00000000000000b2 rsi=fffff800019c3b80 rdi=fffffa8003a66060
rip=fffff800018a98a1 rsp=fffff80002b50d80 rbp=0000000000000000
 r8=fffffa8000c95000  r9=00000000000000b1 r10=0000000000000000
r11=fffff800019c44c0 r12=fffff800019becf0 r13=0000000000000001
r14=fffffa8000c5c110 r15=fffff80002b4a070
iopl=0         nv up ei pl nz na pe nc
cs=0010  ss=0018  ds=002b  es=002b  fs=0053  gs=002b             efl=00000202
nt!KiIdleLoop+0x21:
fffff800`018a98a1 fb              sti

kd>
```

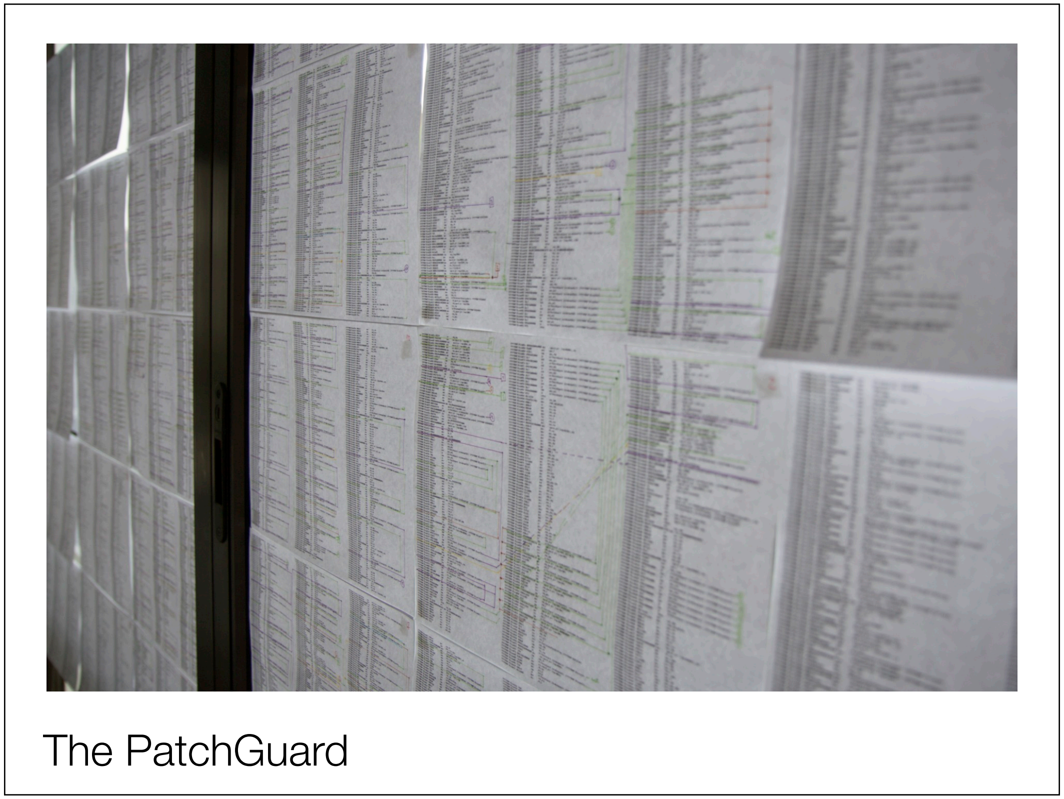
You can talk one whole day here.
Maybe a quick explanation of the Kernel exception handling.
This is interesting to reverse engineering.



```
            TheExceptionHangover( kernel_base, (PVOID) pg_addr, target_fn );

        }
    }
}


NTSTATUS DriverEntry( IN PDRIVER_OBJECT DriverObject,  IN PUNICODE_STRING RegistryPath  )
{
    PVOID driver_base_addr;

    DriverObject->DriverUnload = DriverUnload;

    // get driver base address
    driver_base_addr = MmPageEntireDriver( DriverEntry );
    MmResetDriverPaging( DriverEntry );

    // Setup Disciplines
    KssSetup();

    return STATUS_SUCCESS;
}
```

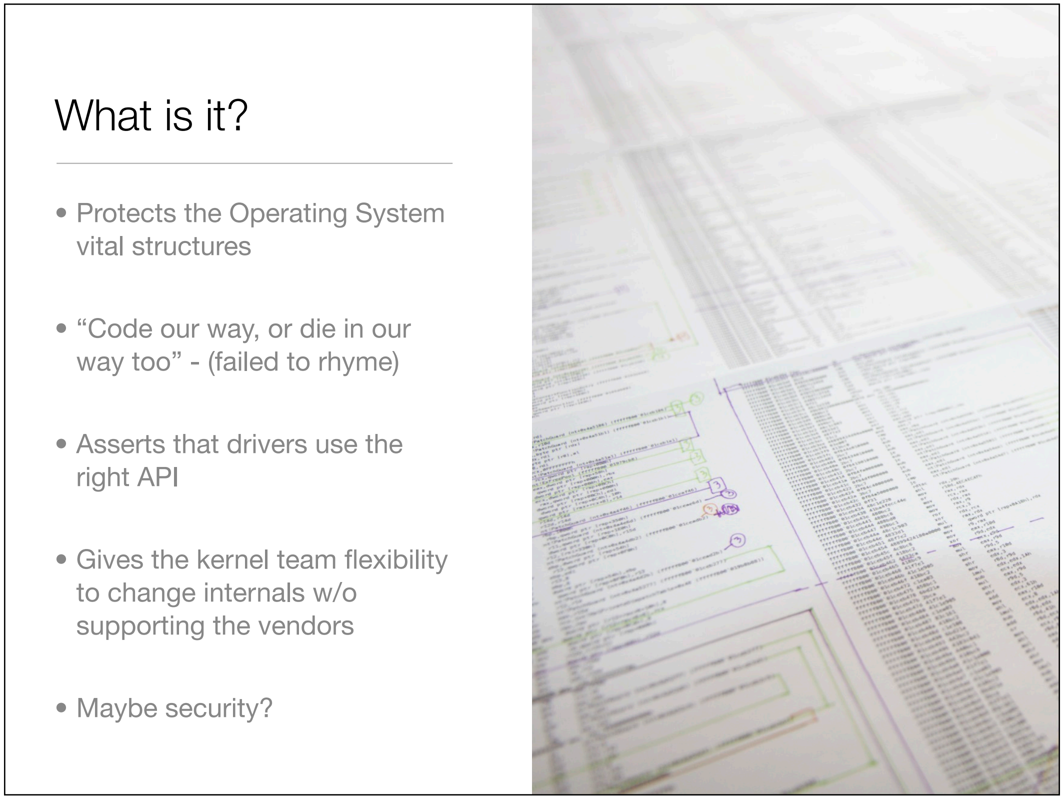
Using Exception - In Action

You can have a pause after the video to a quick demonstration of a different approach for exception handling in x64. You can reach new levels of exploitation by dealing with it. More slides on this?



The PatchGuard

“Opening the PatchGuard” – that’s the silly joke for this one.



What is it?

- Protects the Operating System vital structures
- “Code our way, or die in our way too” - (failed to rhyme)
- Asserts that drivers use the right API
- Gives the kernel team flexibility to change internals w/o supporting the vendors
- Maybe security?

Use the link for that silly joke about the debit card, so DRM-based codecs use the Patchguard to ensure their code is not getting tempered. Please concentrate this slide on flaming Microsoft for convincing the world that Patchguard is good for anything but to protect against malicious code.

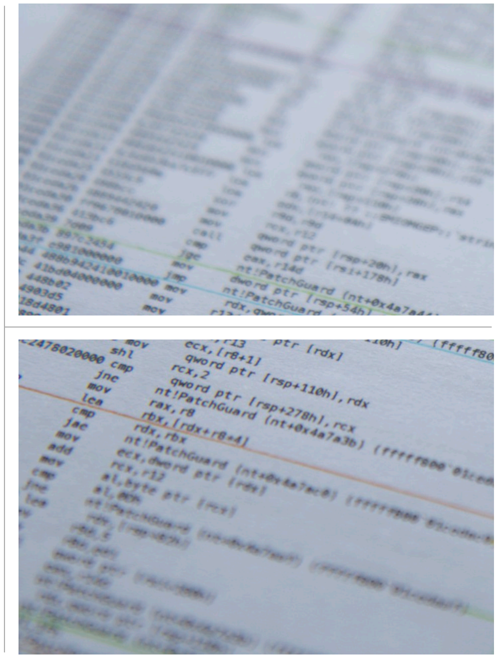
Obfuscation++

- PoolTag randomly chosen
- Random “Fat” allocation space
- Random split the Fat before and after real data
- Fat filled with garbage
- Uses Timer + DPC, using random valid DPC dispatchers, with invalid context data (which will throw an exception)

Break this thing up, show some code, show some graphics on how it’s done.

Obfuscation++

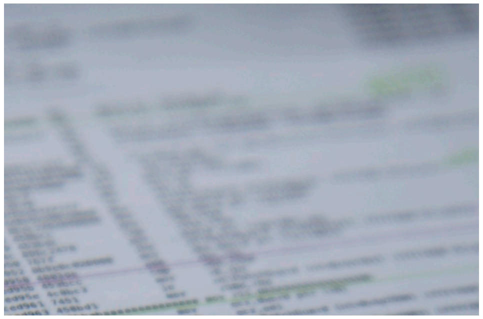
- 2 PgContexts are injected into the Kernel Memory
- One is close to the processor context
- 3/13 chances to have custom DPC dispatcher, to prevent public PatchGuard deactivators
- PgContext is encrypted
- Checks performed inside trap interruption



This is a cool slide. Put some picture of the code here.

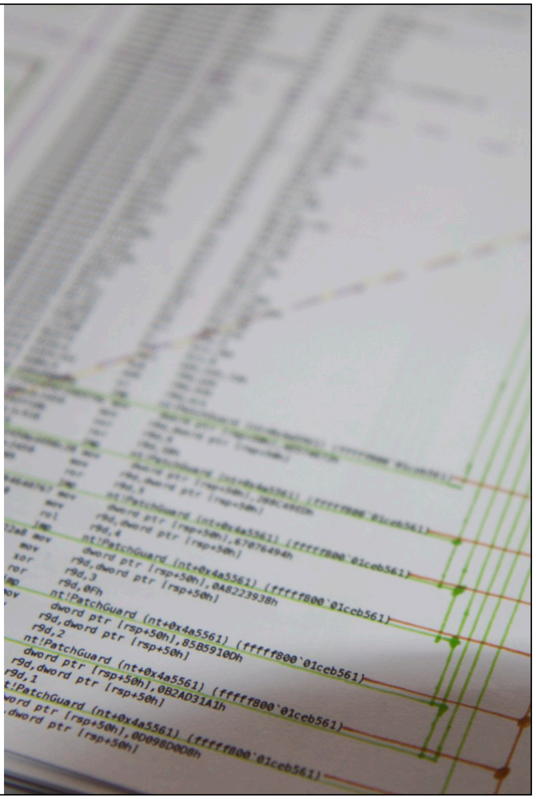
Obfuscation++

- Copy kernel vital functions like KeBugCheckEx, KeBugCheck2, KiBugCheckDebugBreak, etc...
- About 20 debugger_is_attached checks, leading to infinite loop with interrupts disabled



PatchGuard - review

- uninformed.org has published several papers on how to deactivate
- Some proposed paths to block is already patched by Microsoft in latest builds.
- Windows 7 follows the same PG code from Vista, even encryption constants are the same
- X86 can run PatchGuard



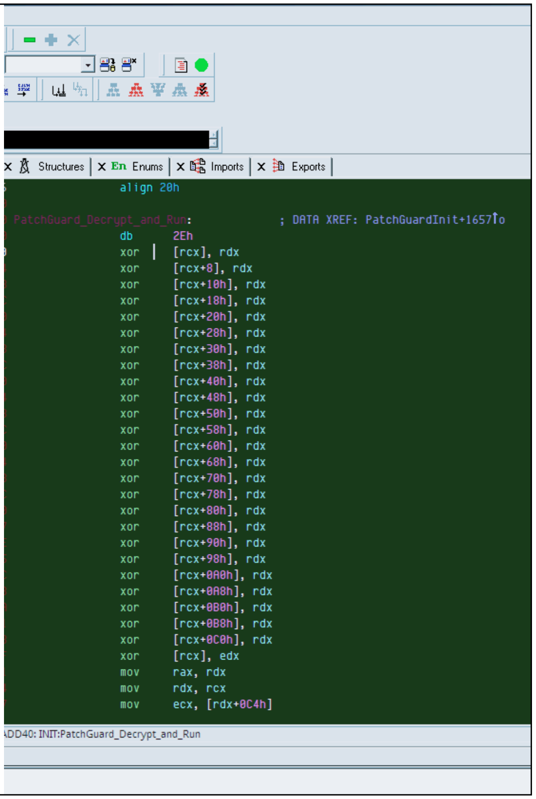
```
nt!PatchGuard (nt+0x4a5561) (fffff800`01ceb561)
dword ptr [rsp+50h],67076494h
r9d,dword ptr [rsp+50h]
r9d,4
nt!PatchGuard (nt+0x4a5561) (fffff800`01ceb561)
dword ptr [rsp+50h],0A8223938h
r9d,dword ptr [rsp+50h]
r9d,3
r9d,0Fh
nt!PatchGuard (nt+0x4a5561) (fffff800`01ceb561)
dword ptr [rsp+50h],85B5910Dh
r9d,dword ptr [rsp+50h]
r9d,2
nt!PatchGuard (nt+0x4a5561) (fffff800`01ceb561)
dword ptr [rsp+50h],0B2AD31A1h
r9d,dword ptr [rsp+50h]
r9d,1
t!PatchGuard (nt+0x4a5561) (fffff800`01ceb561)
word ptr [rsp+50h],0D098D0D8h
,dword ptr [rsp+50h]
```

Deactivating it

- All encryption are based on RDTSC instruction, which can be deactivated by CR4.2

- DPC for timers are encrypted, but decryption is trivial

- Seek and destroy timers



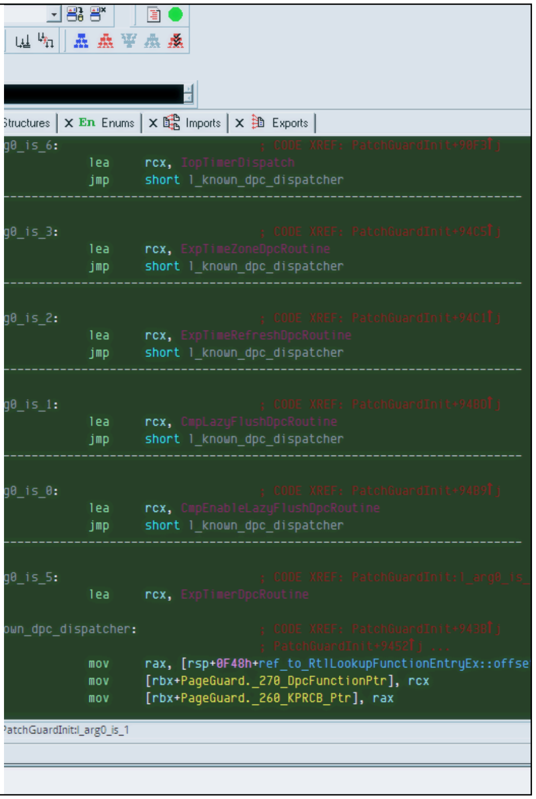
[redacted]

```
                align 20h

PatchGuard_Decrypt_and_Run:             ; DATA XREF: PatchGuardInit+1657↑o
                db      2Eh
                xor     [rcx], rdx
                xor     [rcx+8], rdx
                xor     [rcx+10h], rdx
                xor     [rcx+18h], rdx
                xor     [rcx+20h], rdx
                xor     [rcx+28h], rdx
                xor     [rcx+30h], rdx
                xor     [rcx+38h], rdx
                xor     [rcx+40h], rdx
                xor     [rcx+48h], rdx
                xor     [rcx+50h], rdx
                xor     [rcx+58h], rdx
                xor     [rcx+60h], rdx
                xor     [rcx+68h], rdx
                xor     [rcx+70h], rdx
                xor     [rcx+78h], rdx
                xor     [rcx+80h], rdx
                xor     [rcx+88h], rdx
                xor     [rcx+90h], rdx
                xor     [rcx+98h], rdx
                xor     [rcx+0A0h], rdx
                xor     [rcx+0A8h], rdx
                xor     [rcx+0B0h], rdx
                xor     [rcx+0B8h], rdx
                xor     [rcx+0C0h], rdx
                xor     [rcx], edx
                mov     rax, rdx
                mov     rdx, rcx
                mov     ecx, [rdx+0C4h]
```

ADD40: INIT:PatchGuard_Decrypt_and_Run

Deactivating it

- You can change the IDT, but get ready for behind the scenes dirty job for exception handling
- Use the rewind info to construct the call backtrace
- Find if this is a Dpc in the list of PatchGuard borrowed Dispatcher routines (9/13 chances).
- If custom Dpc, figure the structures, go, go & go!

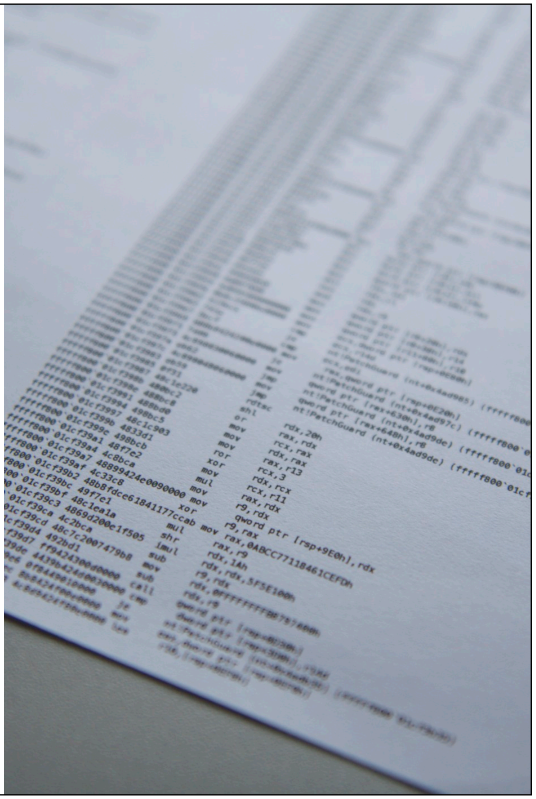


[redacted]

```
g0_is_6:                                        ; CODE XREF: PatchGuardInit+90F3↑j
                lea     rcx, IopTimerDispatch
                jmp     short l_known_dpc_dispatcher
-----------------------------------------------------------------------------------

g0_is_3:                                        ; CODE XREF: PatchGuardInit+94C5↑j
                lea     rcx, ExpTimeZoneDpcRoutine
                jmp     short l_known_dpc_dispatcher
-----------------------------------------------------------------------------------

g0_is_2:                                        ; CODE XREF: PatchGuardInit+94C1↑j
                lea     rcx, ExpTimeRefreshDpcRoutine
                jmp     short l_known_dpc_dispatcher
-----------------------------------------------------------------------------------

g0_is_1:                                        ; CODE XREF: PatchGuardInit+94BD↑j
                lea     rcx, CmpLazyFlushDpcRoutine
                jmp     short l_known_dpc_dispatcher
-----------------------------------------------------------------------------------

g0_is_0:                                        ; CODE XREF: PatchGuardInit+94B9↑j
                lea     rcx, CmpEnableLazyFlushDpcRoutine
                jmp     short l_known_dpc_dispatcher
-----------------------------------------------------------------------------------

g0_is_5:                                        ; CODE XREF: PatchGuardInit:l_arg0_is_
                lea     rcx, ExpTimerDpcRoutine

own_dpc_dispatcher:                             ; CODE XREF: PatchGuardInit+943B↑j
                                                ; PatchGuardInit+9452↑j ...
                mov     rax, [rsp+0F48h+ref_to_RtlLookupFunctionEntryEx::offse
                mov     [rbx+PageGuard._270_DpcFunctionPtr], rcx
                mov     [rbx+PageGuard._260_KPRCB_Ptr], rax
```

Deactivating it

- Get ready to the cat and mouse race! This obfuscation and Patchguard techniques WILL change, and eventually a KeBugCheck issued



Cerca de 38 mil clientes seguem sem luz no RS após temporais

Agência bancária é assaltada na Capital

Encapuzados atacaram o Banrisul da Avenida Protásio Alves

Metrô de Porto Alegre pode começar com outro traçado mais racional, diz ministro

Duda: "Grêmio jamais vai entrar para perder"

• Meira viaja a SC, mas nega contato com Silas

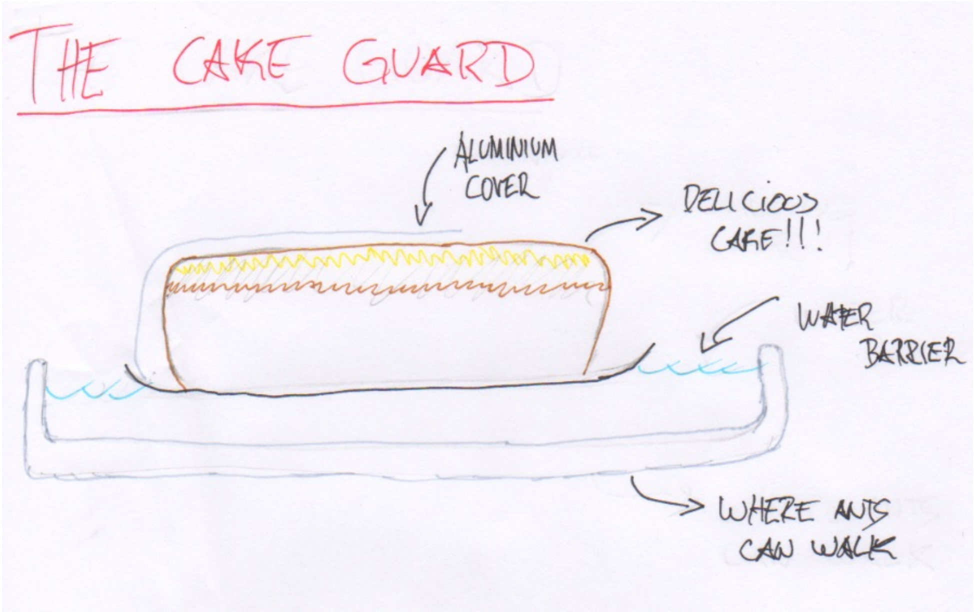
Inter volta aos treinos no Beira-Rio. Acompanhe

• Centenário rende prêmio de marketing ao Inter

Vídeo: Jogadores da Dupla lançam coleção de material esportivo

McLaren confirma Button para 2010

Let's get it all

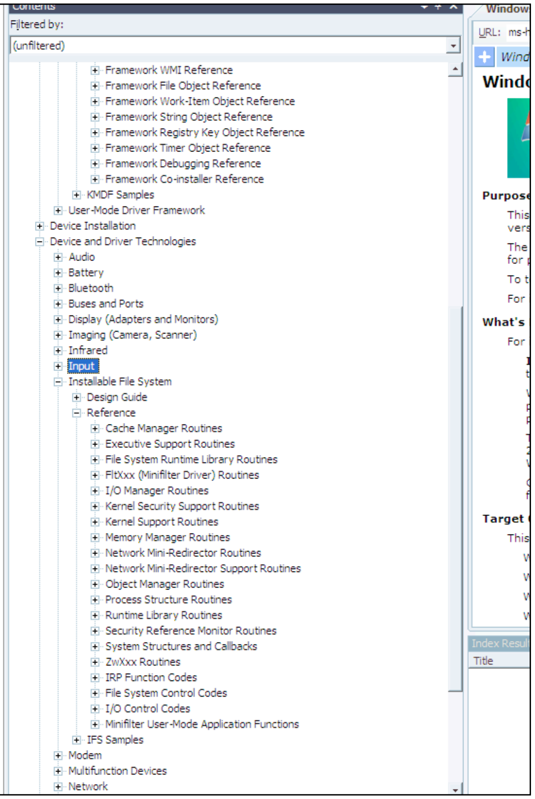


The Two-Stages

The Two-Stages

Natural root kits

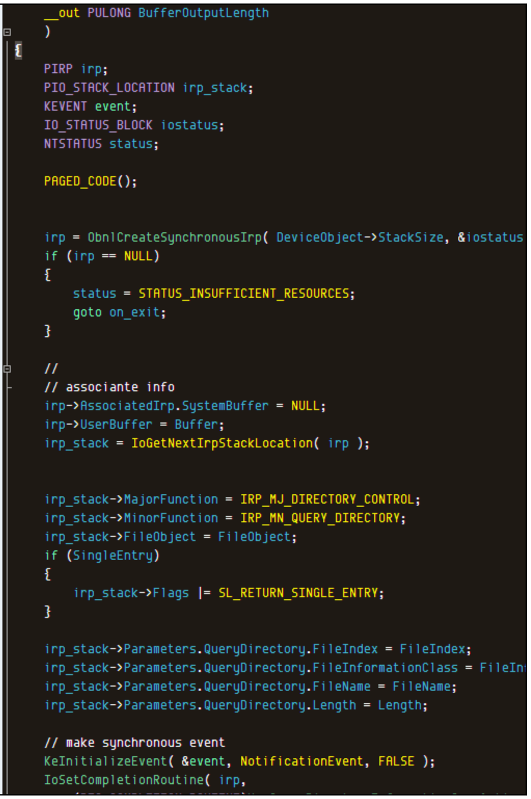
- You can hide files and folders
- Hide registry information
- Change process information
- Open network connections
- A new TCP/IP stack can be built, and raw packets sent through ndis.sys
- A key logger still can be coded



- Framework WMI Reference
- Framework File Object Reference
- Framework Work-Item Object Reference
- Framework String Object Reference
- Framework Registry Key Object Reference
- Framework Timer Object Reference
- Framework Debugging Reference
- Framework Co-installer Reference
- KMDF Samples
- User-Mode Driver Framework
- Device Installation
- Device and Driver Technologies
  - Audio
  - Battery
  - Bluetooth
  - Buses and Ports
  - Display (Adapters and Monitors)
  - Imaging (Camera, Scanner)
  - Infrared
  - Input
  - Installable File System
    - Design Guide
    - Reference
      - Cache Manager Routines
      - Executive Support Routines
      - File System Runtime Library Routines
      - FltXxx (Minifilter Driver) Routines
      - I/O Manager Routines
      - Kernel Security Support Routines
      - Kernel Support Routines
      - Memory Manager Routines
      - Network Mini-Redirector Routines
      - Network Mini-Redirector Support Routines
      - Object Manager Routines
      - Process Structure Routines
      - Runtime Library Routines
      - Security Reference Monitor Routines
      - System Structures and Callbacks
      - ZwXxx Routines
      - IRP Function Codes
      - File System Control Codes
      - I/O Control Codes
      - Minifilter User-Mode Application Functions
    - IFS Samples
  - Modem
  - Multifunction Devices
  - Network

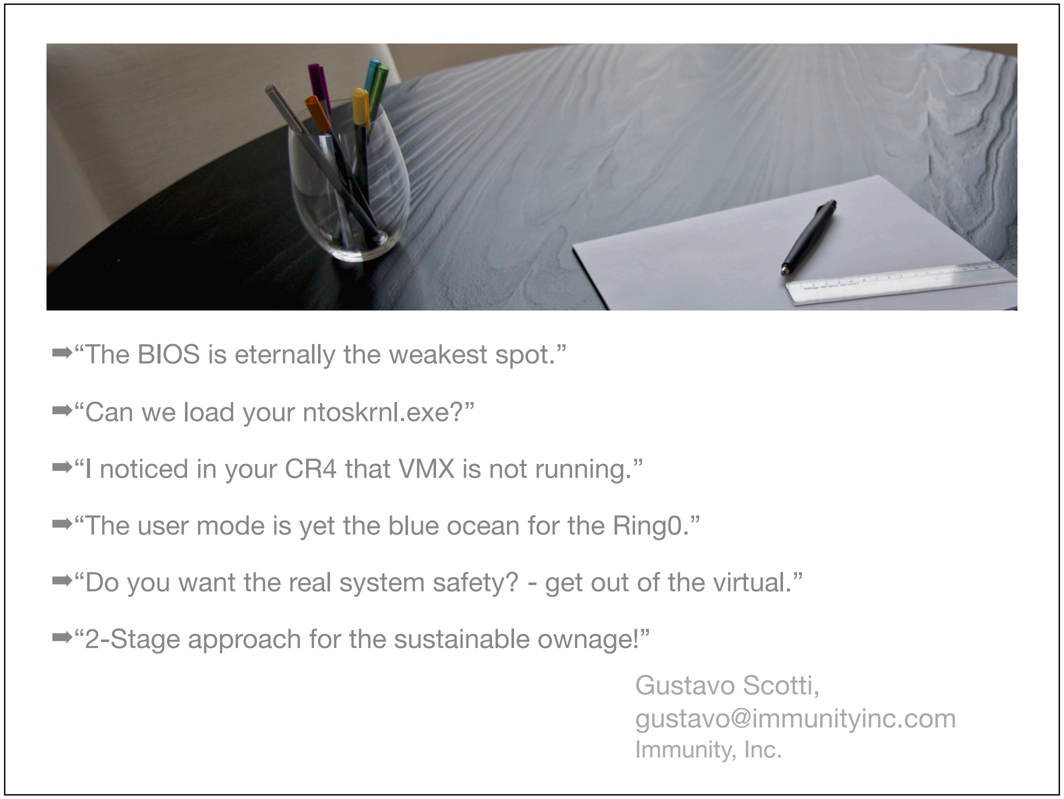
Dialing The Patcher

- Is this kernel PG-removable?
- If not, tcp connect to TP server.
- Send me your kernel details
- If no patching code available, just leave
- Will operate in stage-1
- Once patched, go to stage-2



```
    __out PULONG BufferOutputLength
    )
{
    PIRP irp;
    PIO_STACK_LOCATION irp_stack;
    KEVENT event;
    IO_STATUS_BLOCK iostatus;
    NTSTATUS status;

    PAGED_CODE();


    irp = ObnlCreateSynchronousIrp( DeviceObject->StackSize, &iostatus
    if (irp == NULL)
    {
        status = STATUS_INSUFFICIENT_RESOURCES;
        goto on_exit;
    }

    //
    // associante info
    irp->AssociatedIrp.SystemBuffer = NULL;
    irp->UserBuffer = Buffer;
    irp_stack = IoGetNextIrpStackLocation( irp );


    irp_stack->MajorFunction = IRP_MJ_DIRECTORY_CONTROL;
    irp_stack->MinorFunction = IRP_MN_QUERY_DIRECTORY;
    irp_stack->FileObject = FileObject;
    if (SingleEntry)
    {
        irp_stack->Flags |= SL_RETURN_SINGLE_ENTRY;
    }

    irp_stack->Parameters.QueryDirectory.FileIndex = FileIndex;
    irp_stack->Parameters.QueryDirectory.FileInformationClass = FileIn
    irp_stack->Parameters.QueryDirectory.FileName = FileName;
    irp_stack->Parameters.QueryDirectory.Length = Length;

    // make synchronous event
    KeInitializeEvent( &event, NotificationEvent, FALSE );
    IoSetCompletionRoutine( irp,
```

This is the last subject-based slide. May finish it better, though.



➡“The BIOS is eternally the weakest spot.”

➡“Can we load your ntoskrnl.exe?”

➡“I noticed in your CR4 that VMX is not running.”

➡“The user mode is yet the blue ocean for the Ring0.”

➡“Do you want the real system safety? - get out of the virtual.”

➡“2-Stage approach for the sustainable ownage!”

Gustavo Scotti,
gustavo@immunityinc.com
Immunity, Inc.

The Thank You Final Act!
You may incite the audience to flame you about the virtual environment.
Either send them to hell, tell them to check for your next presentations, or both!