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

- 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

- 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

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 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.
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

Here is a code snippet explaining the process:

```c
PORT_HOOKER HookRtl( void *TargetPtr, void *Hook, void *Context )
{
    PORT_HOOKER hooker;
    int i;
    unsigned char *TargetBytePtr = (unsigned char*)TargetPtr;

    if (!IsSystemInitialized)
    {
        HookInit();
    }

    for ( i = 0; i < MAX_HOOK_RATE; i++ )
    {
        hooker = &hooks[i];
        if ( hooker->TransferSize ) break;
    }

    if ( i == MAX_HOOK_RATE ) return NULL;

    // test if our transient pool size isn't too big
    if (TransferSize > MAX_TRANSIENT_INSTRUCTION) {
        if (IsAssemblyJump( (hooker->CodeData, HookedPtr) ) {
            return NULL;
        }
    }
}
```

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!
Exception Handler

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

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.
• 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

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

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

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!