

Reverse Engineering RGB Keyboard Backlights

Rishit Bansal

Press Space for next page →

ctrl

option

command



Hi, I'm Rishit Bansal!

- From Bangalore, India
- Software Developer @ DYTE (<https://dyte.io>)
- I play CTFs with csictf, dytesec
- 2nd time at Nullcon, participant last year
 - Placed 1st along with dytesec at hardware CTF

<https://www.linkedin.com/in/bansal-rishit/>





What will we cover in this talk?

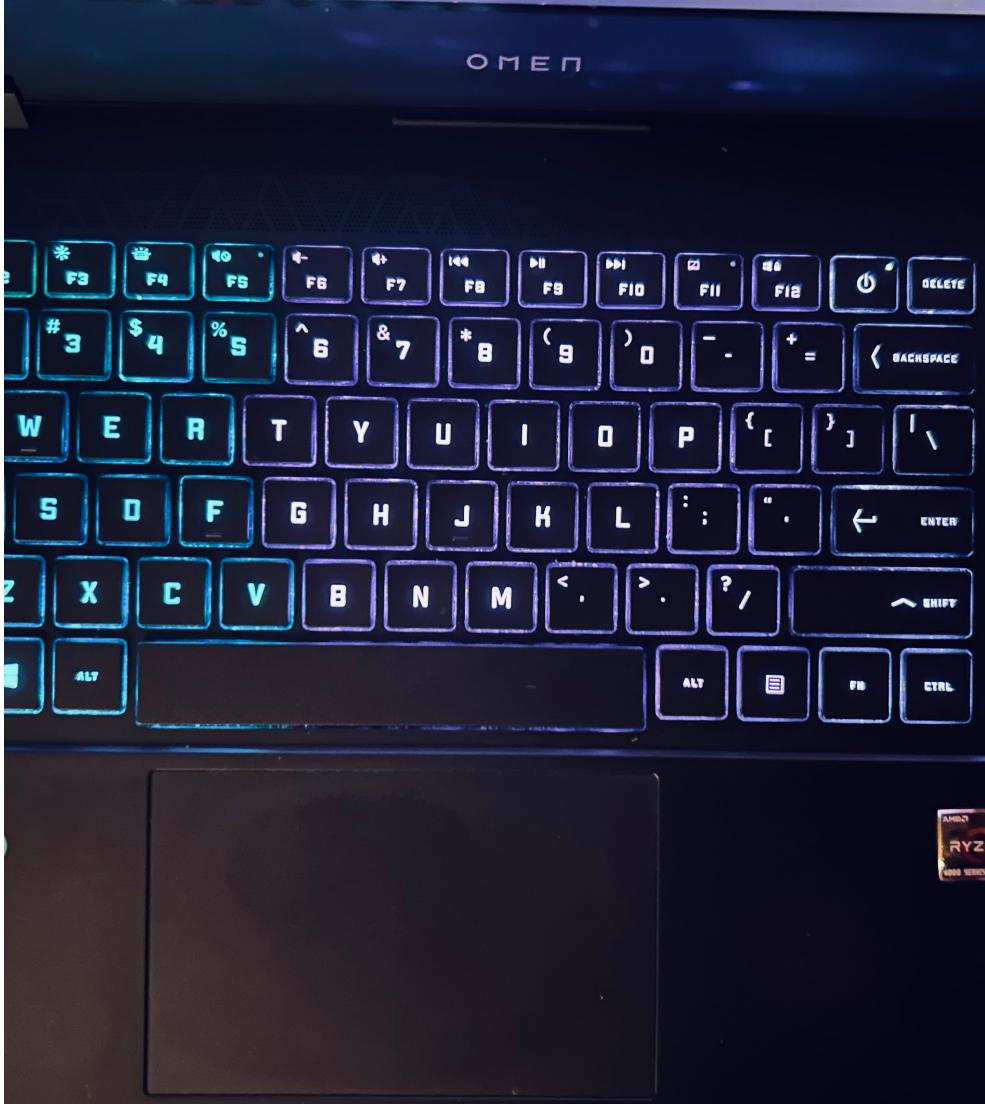
- Chapter 1: Reverse Engineer a Windows Service which interacts with keyboard backlight firmware
- Chapter 2: Understand the system protocols involved in interacting with device firmware
- Chapter 3: Re-implement the same functionality on the Linux Kernel
- Bonus: Discover new functionality possible in hardware!

Chapter 1

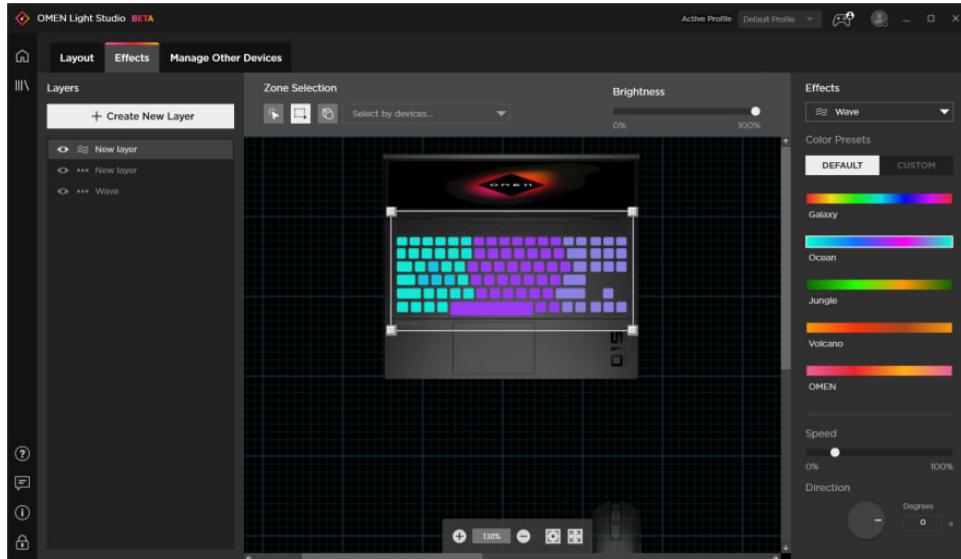
Reverse Engineering the HP Light Studio Application

The Laptop

- Laptop: HP Omen 15
- Four zones of configurable backlights
- Can set colors at a really fast rate to simulate animations.
- Hardware Key to toggle the backlight on and off.

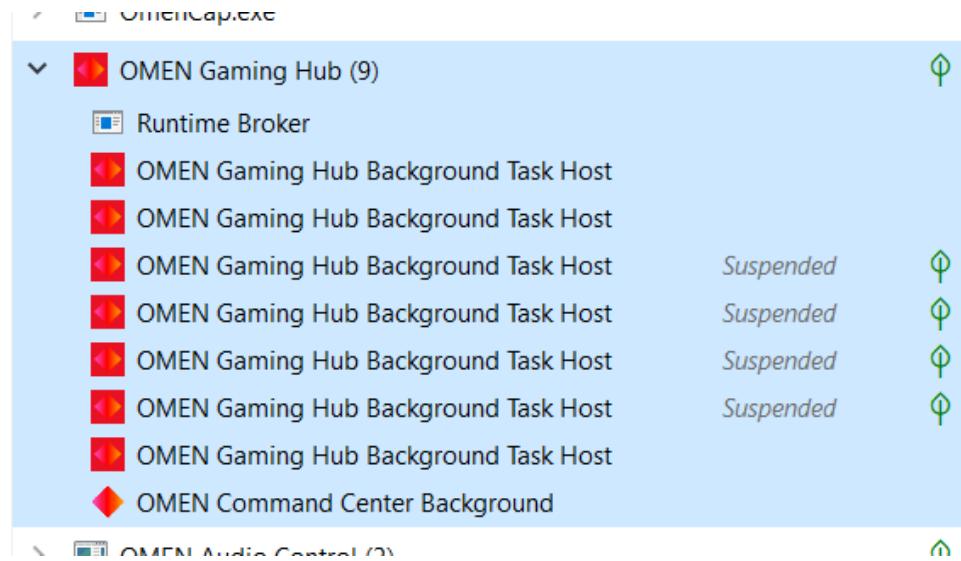


HP Omen Light Studio Application



- Windows Desktop Application
- Allows you to choose between template/custom animations
- Closed source, 0 documentation
- Animations only work when Windows boots, otherwise just static colors

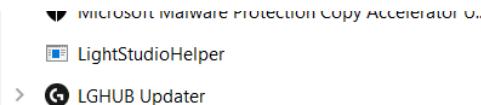
Where is the service?



Omen Gaming Hub on Task Manager

C:\Program

Files\WindowsApps\AD2F1837.OMENLightStudio_1.0.37.0_x64__v10z8vjag6ke6



Light Studio Service on Task Manager

C:\Program Files\HP\LightStudioHelper

Name	Size	Modified
CommonLib.dll	65.1 kB	17 Sep 2022
CommonLib.dll.config	432 bytes	18 Nov 2021
LightStudioHelper.exe	28.2 kB	17 Sep 2022
LightStudioHelper.exe.config	546 bytes	18 Nov 2021
Microsoft.Win32.TaskScheduler.dll	333.8 kB	18 Nov 2021
Microsoft.Win32.TaskScheduler.xml	563.0 kB	18 Nov 2021
netstandard.dll	98.6 kB	19 Jan 2022
Newtonsoft.Json.dll	702.0 kB	22 Jan 2022
Newtonsoft.Json.xml	710.2 kB	24 Oct 2021
NLog.config	1.3 kB	18 Nov 2021
NLog.dll	875.5 kB	18 Nov 2021
NLog.xml	1.7 MB	18 Nov 2021
OmenFourZoneLighting.dll	17.4 kB	17 Sep 2022
OmenShareCommonLib.dll	1.5 MB	17 Sep 2022
OmenShareCommonLib.dll.config	446 bytes	18 Nov 2021
Prism.dll	76.7 kB	22 Jan 2022
Prism.xml	151.0 kB	9 Apr 2022

Decompiling the Executable and DLL Files (Ilspy Demo)

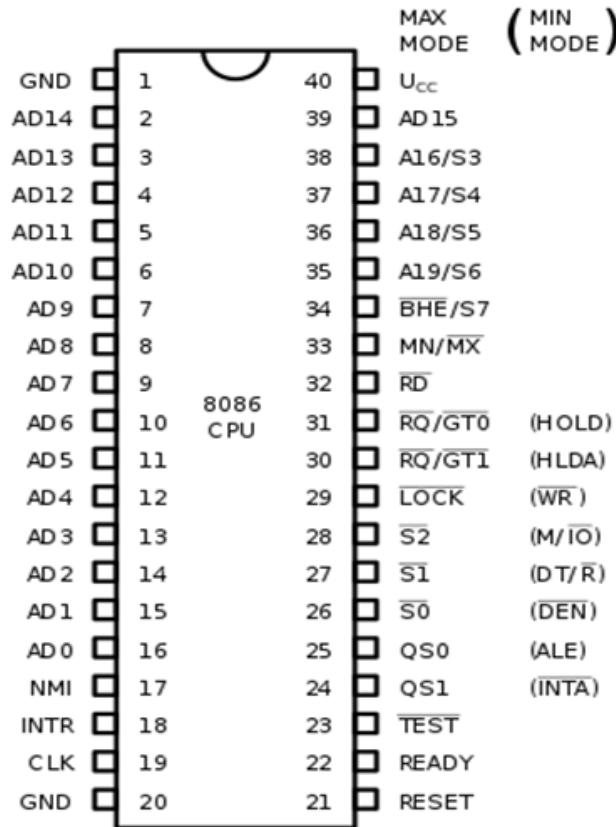
```
1 private static int Execute(int command, int commandType, int inputAxisSize, byte[] inputData, out byte[] returnData)
2 {
3     returnData = new byte[0];
4     try
5     {
6         ManagementObject val = new ManagementObject(
7             "root\\wmi", "hpqBIntM.InstanceName='ACPI\\PNP0C14\\0_0'", (ObjectGetOptions)null
8         );
9         ManagementObject val2 = (ManagementObject)new ManagementClass("root\\wmi:hpqBDataIn");
10        ManagementBaseObject methodParameters = val.GetMethodParameters("hpqBIOSInt128");
11        ManagementBaseObject val3 = (ManagementBaseObject)new ManagementClass("root\\wmi:hpqBDataOut128");
12        ((ManagementBaseObject)val2).set_Item("Sign", (object)Sign);
13        ((ManagementBaseObject)val2).set_Item("Command", (object)command);
14        ((ManagementBaseObject)val2).set_Item("CommandType", (object)commandType);
15        ((ManagementBaseObject)val2).set_Item("Size", (object)inputAxisSize);
16        ((ManagementBaseObject)val2).set_Item("hpqBData", (object)inputData);
17        methodParameters.set_Item("InData", (object)val2);
18        InvokeMethodOptions val4 = new InvokeMethodOptions();
19        ((ManagementOptions)val4).set_Timeout(TimeSpan.MaxValue);
20        InvokeMethodOptions val5 = val4;
21        object obj = val.InvokeMethod("hpqBIOSInt128", methodParameters, val5).get_Item("OutData");
22        val3 = (ManagementBaseObject)((obj is ManagementBaseObject) ? obj : null);
23        returnData = val3.get_Item("Data") as byte[];
24        return Convert.ToInt32(val3.get_Item("rwReturnCode"));
25    }
26    catch (Exception ex)
27    {
28        Console.WriteLine("OMEN Four zone lighting - WmiCommand.Execute occurs exception: " + ex);
29    }
30}
```

Chapter 1 Summary

- Used ILSpy to decompile a .NET Windows Service, identified how the Light Studio Helper Service Works
- Mysterious References to other protocols and Windows APIs
 - Windows C# API: `ManagementObject`
 - References to `root\\wmi`
 - `ACPI\\PNP0C14\\0_0`
- "Command" parameter is passed `131081`
 - Type `1` : Checks if lighting is supported
 - Type `2` : Getting the colors of each zone on the keyboard
 - Type `3` : Setting the colors of each zone on the keyboard
 - Type `4` : Checks if backlight is on or off
 - Type `5` : Sets brightness for each of the 4 zones

Chapter 2: Understanding ACPI and WMI

History of I/O device interop



- On the 8086, I/O devices sent interrupts are sent by:
 - Sending a pulse on `INTR` pin
 - Receives interrupt vector on Data pins, and jumps to interrupt execution

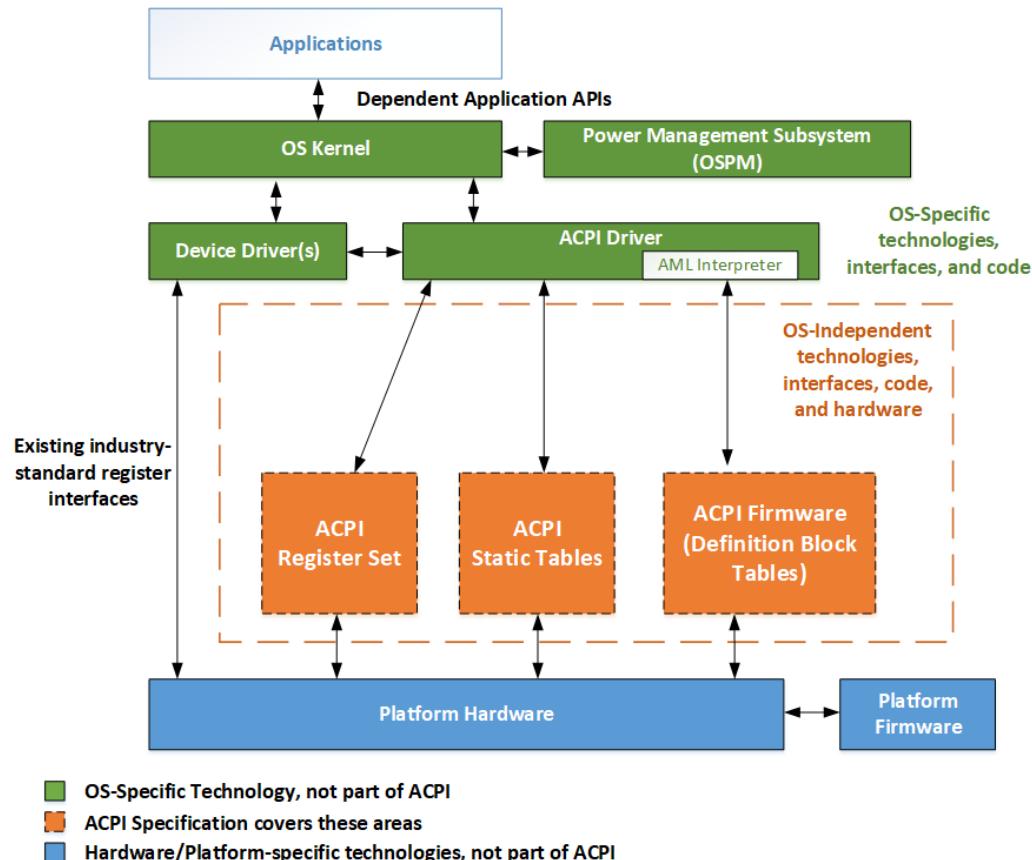
Disadvantages:

- No standardization on interrupt numbers, vectors
- I/O Device API handlers were hardcoded in Bios Firmware
- OS/Kernel space software has no direct I/O access, functionality hardcoded in Bios
 - Needed in modern systems (read temp sensors, power management, etc.)

ACPI (Advanced Configuration and Power Interface)

- Introduced to move I/O API interfaces out of firmware to operating system
- A new language to write I/O related code
 - AML (ACPI Machine Language Bytecode)
 - Interpreted and executed on the operating system!
- ASL code is stored in firmware on "ACPI tables"
 - Loaded into main memory during boot time for OS access

ACPI Architecture



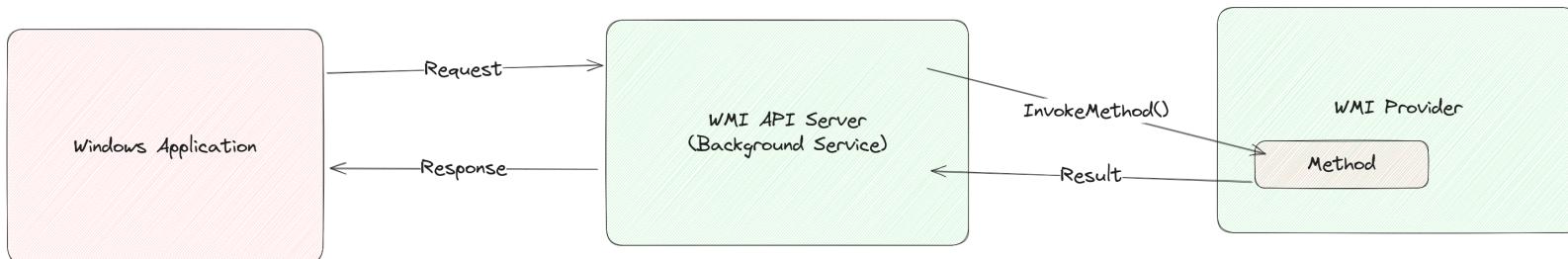
Reading/Decompiling AML code on Linux

- AML Code is loaded into main memory on boot by firmware
 - Linux ACPI Driver mounts this at `/sys/firmware/acpi/tables/DSDT`
- `iasl -d <dsdt_dump>` CLI tool can be used to decompile AML bytecode

```
1  Scope (_SB)
2  {
3      Device (ACAD)
4      {
5          Name (_HID, "ACPI0003" /* Power Source Device */) // _HID: Hardware ID
6          Name (_PCL, Package (0x01) // _PCL: Power Consumer List
7          {
8              _SB
9          })
10         Name (XX00, Buffer (0x03) {})
11         Name (ACSB, One)
12         Name (ACDC, 0xFF)
13         Method (_STA, 0, NotSerialized) // _STA: Status
14         {
15             Return (0x0F)
16         }
17     ...
}
```

WMI (Windows Management Instrumentation)

- A protocol to help sysadmins manage distributed network of windows machines.
 - Execute management scripts/retrieve information from machines.
- Runs a simple server which accepts requests from clients, and executes them on "WMI Providers"
 - Providers expose classes and methods, each class has a unique "GUID"
- Simply put, just a way to do RPC between applications on windows machines.



WMI Explorer

WMI Explorer 2.0.0.2

File Launch Help

Computer: RISHIT Connect Mode: Asynchronous Class Enumeration Options Filter: % Include System Classes Include Perf Classes Include CIM Classes Include MSFT Classes Refresh

Namespaces: \\RISHIT\ROOT

- ROOT\aspnet
- ROOT\CIMV2
- ROOT\Cli
- ROOT\DEFAULT
- ROOT\directory
- ROOT\Hardware
- ROOT\HP
- ROOT\Interop
- ROOT\Microsoft
- ROOT\msdtc
- ROOT\PEH
- ROOT\RSOP
- ROOT\SECURITY
- ROOT\SecurityCenter
- ROOT\SecurityCenter
- ROOT\ServiceModel
- ROOT\StandardCimv2
- ROOT\subscription
- ROOT\WMI
 - ROOT\WMI\ms_40
 - ROOT\WMI\ms_80

Classes (1643) Search Quick Filter: int

Instances (0) Properties (2) Methods (5) Query Script Logging

Name	Lazy...	Description
_IntervalTimerInstruct...	False	
BcdIntegerElement	False	This repre...
BcdIntegerListElement	False	This repre...
DebugPrint_Event	False	DebugPrintE...
hpqBlnM	False	
KernelThermalConstr...	False	Kernel Therm...
MS_WmiInternal	False	
MsftDiscFormat2Strea...	False	IMAPI2 Inter...
MSLSA_Lookupsolat...	False	Lookupsolat...
MSLSA_Lookupsolat...	False	Lookupsolat...
MSNdis_InterruptMod...	False	
MSNdis_QueryInterru...	False	NDIS Query...
MSNdis_SetInterruptM...	False	NDIS Set Inte...
SampledProfileInterv...	False	Sampled Pro...
SampledProfileInterv...	False	Sampled Pro...
SystemConfig_V2_Co...	False	System Conf...
TSRdpEndPointTrace	False	TS Rdp Sou...

Class Properties

Property Name	Type	Enumeration Available	Lazy	Description
Active	Boolean	False	False	
InstanceName	String	False	False	

hpqBlnM Class
[Class Description Not Available]

Class Qualifiers:

- dynamic - True
- guid - {5FB7F034-2C63-45e9-BE91-3D44E2C707E4}
- provider - WmiProv
- WMI - True

Properties:

How it all links together: WMI-ACPI!

- Microsoft's proprietary extension to the ACPI specification
- Allows you to expose ACPI Methods, as WMI Methods on Windows.
- Developer must create a custom ACPI device with ID `PNP0C14` in AML.
 - Must have a field called `_WDG`
 - `_WDG` stores metadata for links between WMI Class GUIDs and ACPI Functions (Wmxx)

```
Device (WMID)
{
    Name (_HID, "PNP0C14" /* Windows Management Instrumentation Device */) // _HID: Hardware ID
    Name (_WDG, Buffer (0x0118)
    {
        /* 0000 */ 0x34, 0xF0, 0xB7, 0x5F, 0x63, 0x2C, 0xE9, 0x45, // 4..._c,.E
        /* 0008 */ 0xBE, 0x91, 0x3D, 0x44, 0xE2, 0xC7, 0x07, 0xE4, // ..=D....
        /* 0010 */ 0x41, 0x41, 0x01, 0x02, 0x79, 0x42, 0xF2, 0x95, // AA..yB..
        /* 0018 */ 0x7B, 0x4D, 0x34, 0x43, 0x93, 0x87, 0xAC, 0xCD, // {M4C....
        /* 0020 */ 0xC6, 0x7E, 0xF6, 0x1C, 0x80, 0x00, 0x01, 0x08, // .~....
```

Using wmidump to read WDG buffers

- ` `\$ wmidump < file_with_wdg_buffer`
- Extracts out the WMI method GUIDs and ACPI function mappings

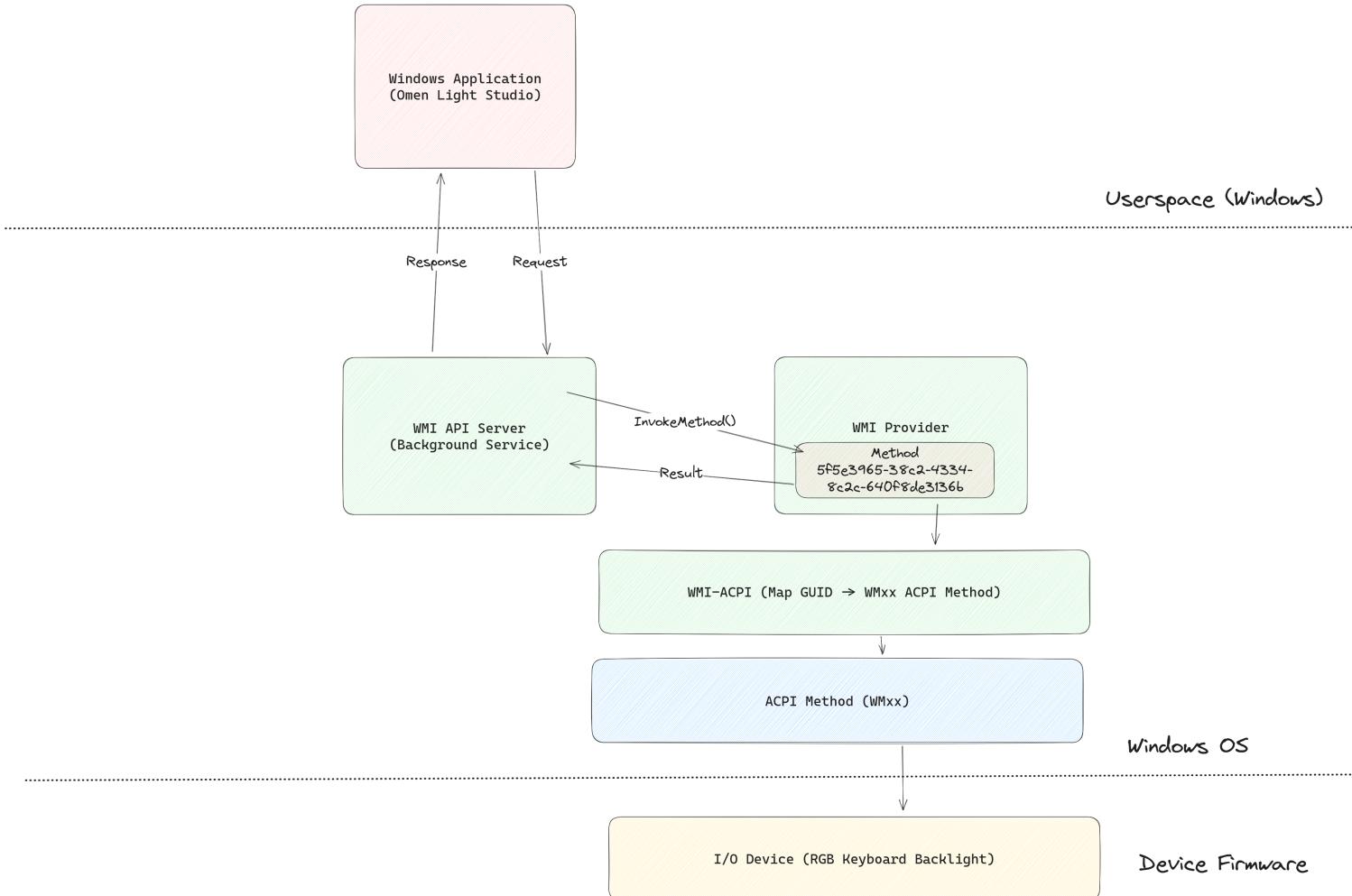
5FB7F034-2C63-45E9-BE91-3D44E2C707E4:

```
object_id: AA
instance_count: 1
flags: 0x2 ACPI_WMI_METHOD
95F24279-4D7B-4334-9387-ACCDC67EF61C:
notify_id: 0x80
instance_count: 1
flags: 0x8 ACPI_WMI_EVENT
....
```

- WMAA Method in AML Code:

```
Method (WMAA, 3, Serialized)
{
    Acquire (MUTZ, 0xFFFF)
    Local0 = HWMC (Arg1, Arg2)
    Release (MUTZ)
    Return (Local0)
}
```

WMI-ACPI Flow for Omen Light Studio Application



Chapter 3: Developing WMI Drivers on the Linux Kernel

acpi.h in the Linux Kernel

- Implementation of the ACPI specification / AML interpreter
 - `acpi_boot_init()` is called on boot to parse AML from ACPI tables in system-memory
 - Provides helper functions to invoke WMI-ACPI methods using their GUID directly from a kernel driver:

Interfacing kernel APIs from userspace

- `sysfs` allows you to create custom files in the `/sys/` to represent device driver APIs
 - You can write/read to these files from userspace, to trigger handlers in the kernel
- In our case, `/sys/class/leds` is most relevant to represent the keyboard backlight device
 - Has special handlers we have to implement for brightness control
 - Added a custom file called `zone_colors` to represent the RGB backlights

```
static DEVICE_ATTR_RW(zone_colors);
static struct attribute *omen_kbd_led_attrs[] = {
    &dev_attr_zone_colors.attr,
    NULL,
};

ATTRIBUTE_GROUPS(omen_kbd_led);

static struct led_classdev omen_kbd_led = {
    .name = "hp_omen::kbd_backlight",
    .brightness_set = set_omen_backlight_brightness,
    .brightness_get = get_omen_backlight_brightness,
    .max_brightness = 1,
    .groups = omen_kbd_led_groups,
};
```

What the new device driver file tree looks like

```
rishit@OMEN-laptop:~/Documents/kernels/staging$ ls /sys/class/leds/hp_omen\:\kbd_backlight/
brightness  max_brightness  subsystem  uevent
device      power          trigger     zone_colors
```

- We need to write handlers to read/write `brightness` and `zone_colors`

```
#define HPMI_READ_ZONE 0x02
#define HPMI_WRITE_ZONE 0x03
#define OMEN_ZONE_COLOR_LEN 0x0c // 12 bytes (3 components (R,G,B) * 4 zones)
#define OMEN_ZONE_COLOR_OFFSET 0x19 // 25
#define HPMI_KB 0x20009 // 131081

static ssize_t zone_colors_store(struct device *dev, struct device_attribute *attr, const char *buf, size_t count)
{
    u8 val[128];
    int ret;

    ret = hp_wmi_perform_query(HPMI_READ_ZONE, HPMI_KB, &val, zero_if_sup(val), sizeof(val));

    if (ret)
        return ret;

    if (count != OMEN_ZONE_COLOR_LEN)
        return -1;

    memcpy(&val[OMEN_ZONE_COLOR_OFFSET], buf, count);

    ret = hp_wmi_perform_query(HPMI_WRITE_ZONE, HPMI_KB, &val, sizeof(val), 0);

    if (ret)
        return ret;

    return OMEN_ZONE_COLOR_LEN;
}
```

Handlers for brightness control

```
#define HPWMI_WRITE_BRIGHTNESS 0x05

#define HPWMI_KB 0x20009 // 131081

static void set_omen_backlight_brightness(struct led_classdev *cdev, enum led_brightness value)
{
    char buffer[4] = { (value == LED_OFF) ? 0x64 : 0xe4, 0, 0, 0 };

    hp_wmi_perform_query(HPWMI_WRITE_BRIGHTNESS, HPWMI_KB, &buffer,
                         sizeof(buffer), 0);
}
```

- **Bonus:** Even though this LED only supports ON and OFF state, we can *simulate* brightness using a trick
- We can use the previous zone_colors_store/read methods and "scale" the RGB components by the brightness multiplier
- $R_{eff}(zone) = R(zone) * (brightness/100)$

Demo

Thank you!

Questions?

