ASRock RGB Driver VR

Jun 20, 2020

374 Words

ASRock Driver Reverse Engineering X86 Windows

Recently my friend built a new Threadripper desktop. Although being three years late to the game, it is still a huge upgrade from laptop. The only annoying thing is that the ASRock X399 motherboard we use has some RGB lights that are quite annoying to say the least. The only way to control them is through this ASRock application under windows.

Early Investigation

We first probed around the userspace application to see what it does to communicate with the LEDs. It loads some ASRockIODriver dll that exports a function called setRGB. This seems very simple, before we dive deep into reversing this DLL, we decided to do a quick sanity check by setting a break point on this function see if it actually gets called. To our suprise, the break point is never hit. This is a dead end then. (Glad we didn't spend hours working on this dead code path)

Then we probed futher, we see the application opens a interesting handle to `AsDrv` device. This looks like some sort of handle to a kernel driver. Then looking at the files it loaded, we see that it loads this file called `AsrDrv103.sys`. It seems to be a very simple ioctl driver.

AsrDrv103.sys

A quick look at this file provides some very interesting details. The code seems to read and write all the CR registers. My first instinct is why would it ever need to touch any of the CRx registers? At the beginning of this very large switch statement that controls what memory, register and IO ports to read/write we found some ASE decryption code used to pre-process the argument.

It turns out that all the operation logic is actually encoded in the user-space application, and it passes the value and and address it want's to write through an AES encrypted buffer to the kernel space so this driver can execute it. However, using AES as the only means of protection is obviously insecure. For a proof of concept we were able to triple-fault the computer by sending an request to zero CR3.

Update 2022

Never got time around to finish this post, however, my friend actually did a video series on this exploit, you can find it here on this github https://github.com/stong/CVE-2020-15368