Both IOCTLs (0x22C004, 0x22C008) need to have 8 bytes for the output buffer.

In both of them there is a main line of operations:

1. (a1 + 64) – a1 = PDEVICE\_OBJECT, offset 64 = DeviceExtension (type: PVOID)
2. First 8 bytes from after a1 = VALUE of DeviceExtension
3. Depending on the IOCTL, the PVOID value inside DeviceExtension gets added with 8/32. Meaning of value in DeviceExtension: DeviceExtension is a driver defined structure that gets allocated in IoCreateDevice. In this case the structure is 0x38 bytes. Because the extension gets played with inside DriverEntry before the dispatch function, I can guess that both of the functions called are function pointers that exist inside the device extension in offset 8/32. Both of these functions probably have something to do with the gpu energy consumption information or something similar.
4. After 3), the value (which should be an executable address) gets casted to a function pointer that receives no parameters and returns an 8 byte value. This is the value that gets saved into v6 and is the one that its output is saved in the output buffer

In the DriverEntry we can see which functions get called:

* If IsEnergySupportedCoreProcessor() returns TRUE: DeviceExtension[1] = ComputeGpuEnergyICore, DeviceExtension[4] = ComputePackageEnergyICore
* If IsEnergySupportedCoreProcessor() returns FALSE and IsEnergySupportedAtomProcessor() returns TRUE: DeviceExtension[1] = ComputeGpuEnergyAtom, DeviceExtension[4] = ComputePackageEnergyAtom
* If both return FALSE: DeviceExtension[1] = ComputeEnergyDefault, DeviceExtension[4] = ComputeEnergyDefault (same function)

Energy support for atom/core processor are probably GPU settings that can be set on/off. Depending on if one/neither are applied on the machine, these functions will get called in respond to my IOCTLs, so I cannot inject/change/execute my chosen memory, a specific chosen function will get called and I will get its result.

These functions are probably in response to some calculation of energy consumption of the GPU (??) and return the calculated amount of energy consumed (??)

ComputeEnergyDefault – returns 0

Other dispatches calculate some unclear value that is probably what I mentioned, and this value gets returned to me.

So to sum up – this driver is probably not vulnerable as there is no primitive for any type of vulnerability here, only if the GPU energy consumption value is considered disclosed information.