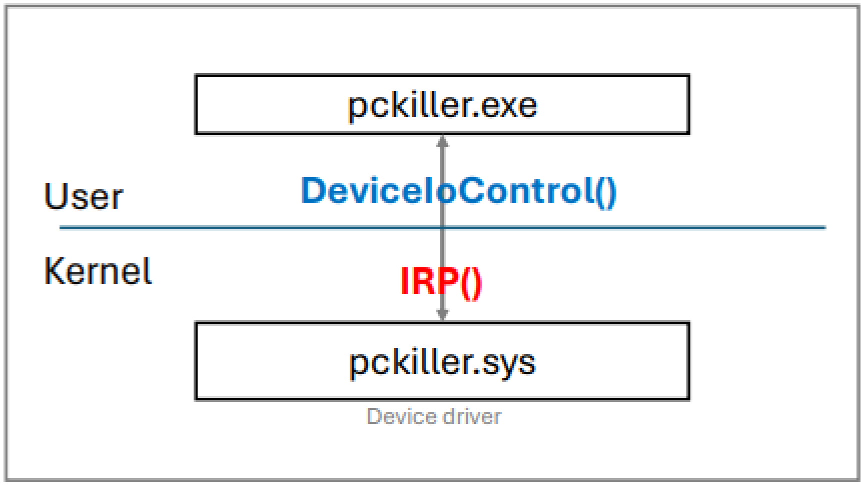
PCKiller - Daniel Ayzenshteyn

Summary

PCKiller is my first cute kernel software driver. It can shutdown the system without any problems with a delayed timer.
The driver is accompanied by a cure client which can communicate with the driver via IRP (DeviceIoControl).



PCKiller architecture

Usage:

```
sc create PCKiller binPath= "PATH_TO_PCKILLER_SYS_FILE" type= kernel
sc start PCKiller
sc stop PCKiller
sc delete PCKiller

.\PCKillerClient.exe -shutdown <n (seconds)>
.\PCKillerClient.exe -remaining
.\PCKillerClient.exe -cancel
```

The first 4 commands are basic sc.exe commands which install the kernel driver, start, stop and delete it if needed.
The last 3 commands are the custom made PCKillerClient which communicates with the PCKiller driver. It can issue a shutdown of the system and specify a delay. During the delayed shutdown we can query the driver on how much time is left until the shutdown initiated or even cancel the shutdown.

Interesting parts:

The most challenging part was to shutdown the system without a blue screen. The main function which can issue the shutdown is undocumented - NtShutdownSystem.
I researched and found a couple of helpful resources for this task which I will reference at

the end of this report.

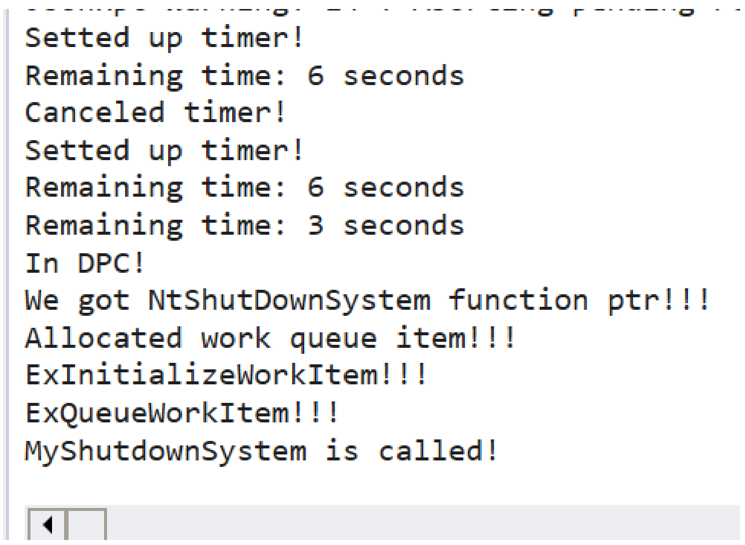
The timer is created and managed by KeInitializeTimer, KeSetTimer, KeCancelTimer and it's binded to a DPC (TimerDpcRoutine in the code) - which is basically a callback function which is executed when the timer "goes off".

The following page contains the interesting parts from the undocumented **NtShutdownSytem** undocumented function.

The code snippet is mainly shows the following 3 tricks:
- We have a declaration of `_SHUTDOWN_ACTION` and `NtShutdownSystem` from NTDOC.
- The NtShutdownSystem cannot be called from DPC - the solution uses `ExQueueWorkItem` which does the trick.
- `MmGetSystemRoutineAddress` is used to retrieve NtShutdownSystem function address - as learned in lecture, similar to GetProcAddress in userland.

```
typedef enum _SHUTDOWN_ACTION
{
    ShutdownNoReboot,
    ShutdownReboot,
    ShutdownPowerOff,
    ShutdownRebootForRecovery // since WIN11
} SHUTDOWN_ACTION;

typedef
NTSTATUS
(*NtShutdownSystem)(
    _In_ SHUTDOWN_ACTION Action
);

NtShutdownSystem NtShutDownSystem = NULL;


VOID NTAPI MyShutdownSystem(PVOID) {
    NtShutDownSystem(ShutdownNoReboot);
}


VOID TimerDpcRoutine(KDPC* Dpc, PVOID DeferredContext, PVOID
SystemArgument1, PVOID SystemArgument2) {
    ...
    UNICODE_STRING functionName = { 0 };
    RtlInitUnicodeString(&functionName, L"NtShutdownSystem");
    NtShutDownSystem =
(NtShutdownSystem)MmGetSystemRoutineAddress(&functionName);

    if (NtShutDownSystem != NULL) {
        PWORK_QUEUE_ITEM pWorkItem =
            (PWORK_QUEUE_ITEM)ExAllocatePool(NonPagedPool,
sizeof(WORK_QUEUE_ITEM));

        if (pWorkItem != NULL) {
            ExInitializeWorkItem(pWorkItem, &MyShutdownSystem, NULL);
            ExQueueWorkItem(pWorkItem, DelayedWorkQueue);
        }
    }
}
```

Added to this report is a video demonstration. Here is a small capture of debug messages captured by WinDbg:



```
Setted up timer!
Remaining time: 6 seconds
Canceled timer!
Setted up timer!
Remaining time: 6 seconds
Remaining time: 3 seconds
In DPC!
We got NtShutDownSystem function ptr!!!
Allocated work queue item!!!
ExInitializeWorkItem!!!
ExQueueWorkItem!!!
MyShutdownSystem is called!
```

The timer executes the DPC, which retrieves the NtShutDownSystem address using MmGetSystemRoutineAddress, we then use ExAllocatePool, ExInitializeWorkItem and ExQueueWorkItem (those are deprecated by I used #pragma warning(disable : 4996) to use them either way…) - At the end MyShutDownSystem function is called which just calls the undocumented ShutDownSystem outside of the DPC by the pointer we retrieved at the beginning.

<u>References:</u>

[Shutting down Windows from kernel mode?](#)

[windbg does not show dbgprint messages - Microsoft Q&A](#)

[Windows Kernel Debugging Introduction](#)

[NtShutdownSystem - NtDoc](#)

SHUTDOWN_ACTION - NtDoc

MmGetSystemRoutineAddress

KeSetTimer function (wdm.h) - Windows drivers

Compiler Warning (level 3) C4996