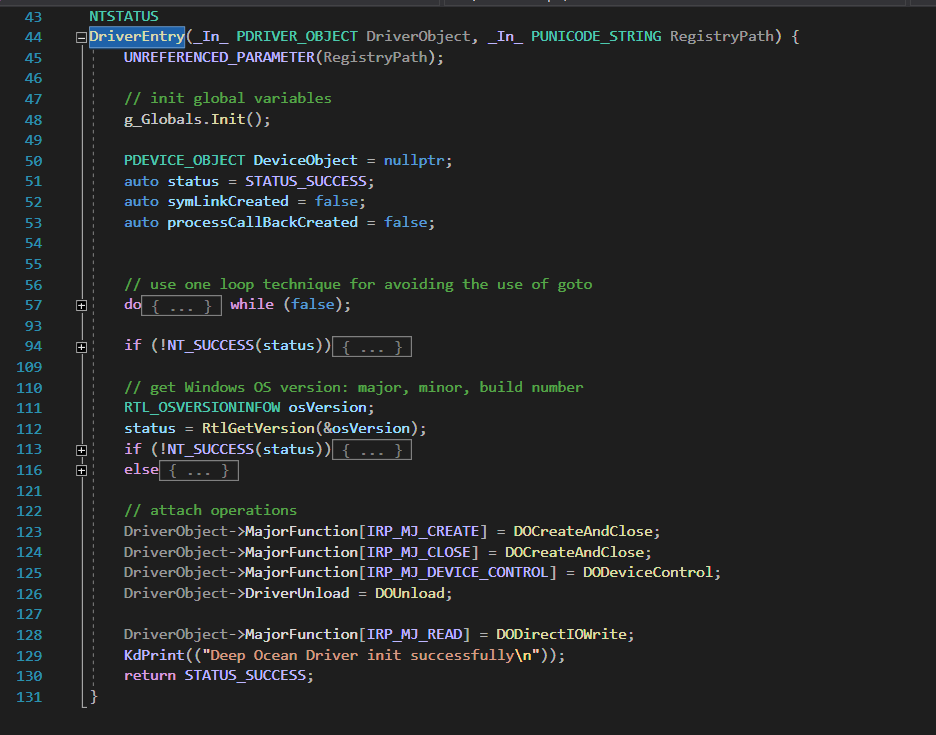
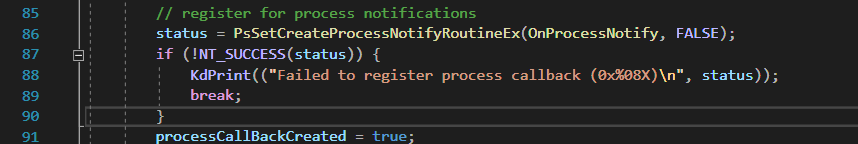
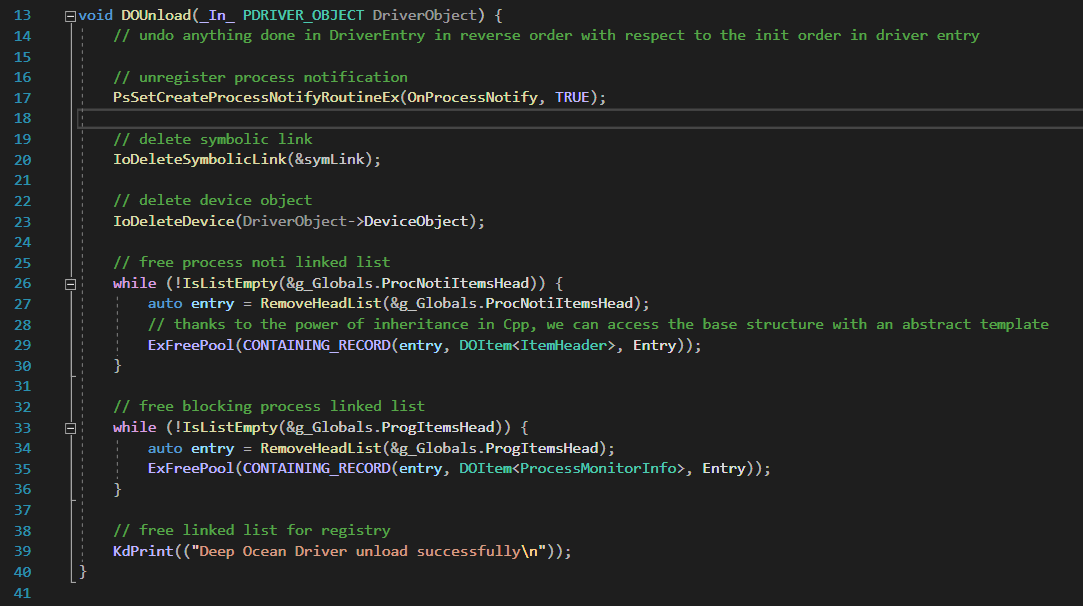
1. Code analysis
   1. DODriver Module

* Driver’s running flow starts from main.cpp, which contains “DriverEntry” (similar to main() in user mode) and “DOUnload”
  + DriverEntry entry point is considered as the main of the driver, called by system thread at IRLQL\_PASSIVE\_LEVEL (0). This entry point is used to regist the driver with the system along with needed routines. In our case is “PsSetCreateProcessNotifyRoutineEx”

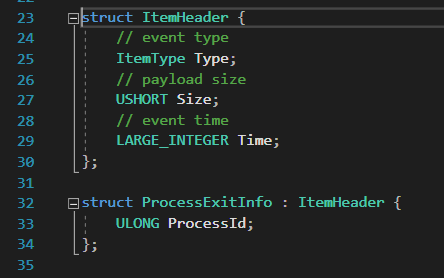




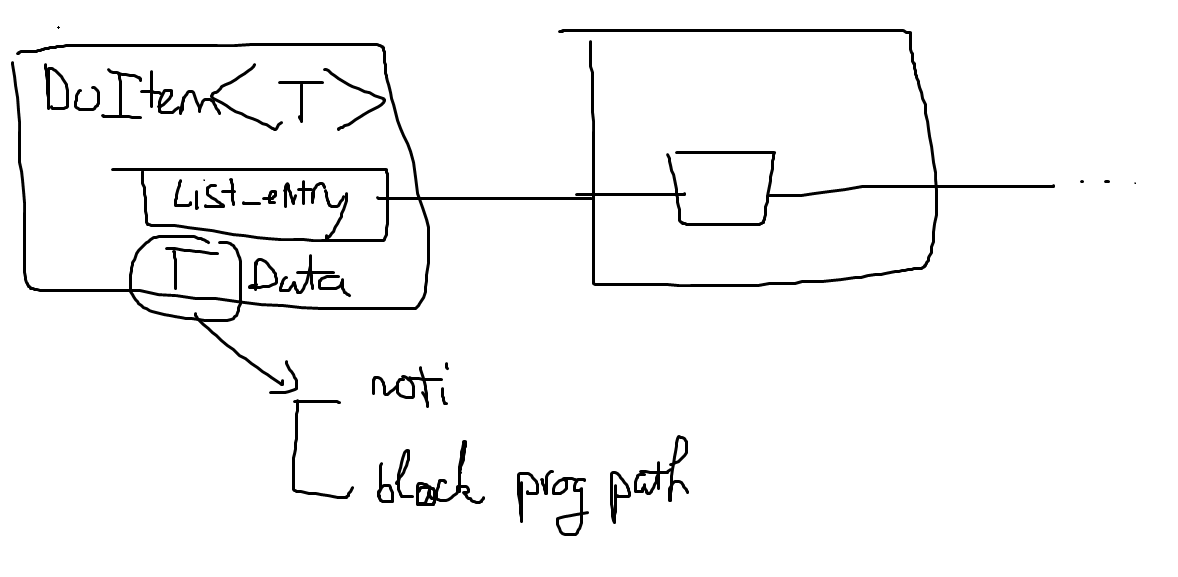
* + DOUnload is used to undone any thing registered in “DriverEntry”. In our case, it unregist OnProcessNotify and release the memory cost by linnked lists



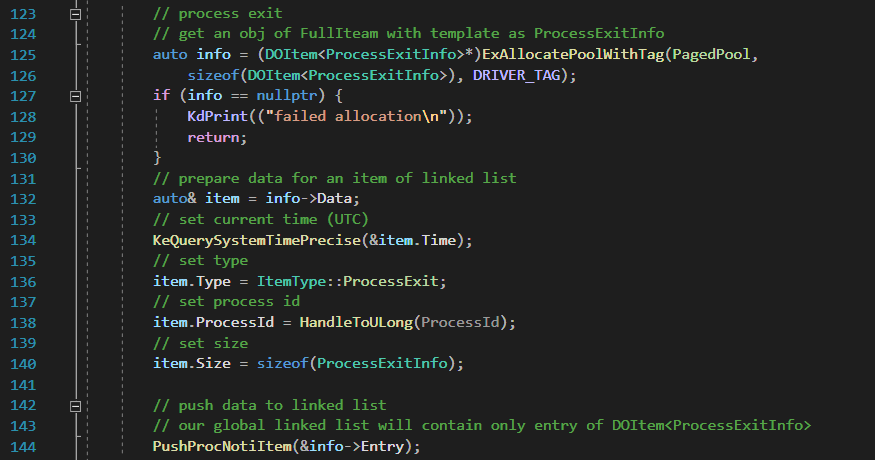
* + To handshake with the user mode client, in “dispatchroutine.cpp”, driver uses “DOCreateAndClose” for support creating and closing device to kernel driver. On the other hand, driver also supports writing straightly to memory that user mode client provide. This feature is expressed via “DODirectIOWrite" in the same file. Nevertheless, to receive requrests from client, driver explicitizes IOCTL\_DO\_PROGRAM\_BLOCK and IOCTL\_DO\_PROGRAM\_UNBLOCK via “DODeviceControl”
    - IOCTL\_DO\_PROGRAM\_BLOCK is used for adding program path to blacklist
    - IOCTL\_DO\_PROGRAM\_UNBLOCK is used for removing program path from backlist
  + DODriver has 2 main doubly linked lists with head node as ` ProcNotiItemsHead ` and ` ProgItemsHead ` for storing list of process creation/destruction notifications and list of blocking program paths, respectively.
    - ` ProcNotiItemsHead ` will implicitly accept other nodes with the type of either ProcessCreateInfo or ProcessExitInfo
    - ` ProgItemsHead ` implicity accept other nodes with the type as ProcessMonitorInfo
    - Both three struct ProcessCreateInfo, ProcessExitInfo and ProcessMonitorInfo inherits the ItemHeader to avoid duplicate code



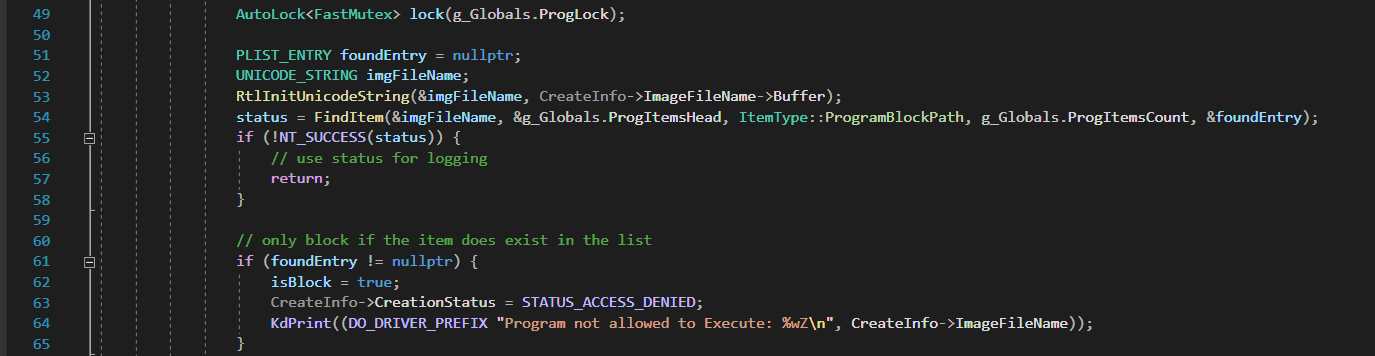
* + - To integrate our data structure with doubly linked list in Windows in a nice loose-coupled way, we decide to make a wrap up:
      * DOItem is a struct wraps up List Entry (which connects other nodes) and a data with template T. Therefore, we can use DOItem with both linked lists.



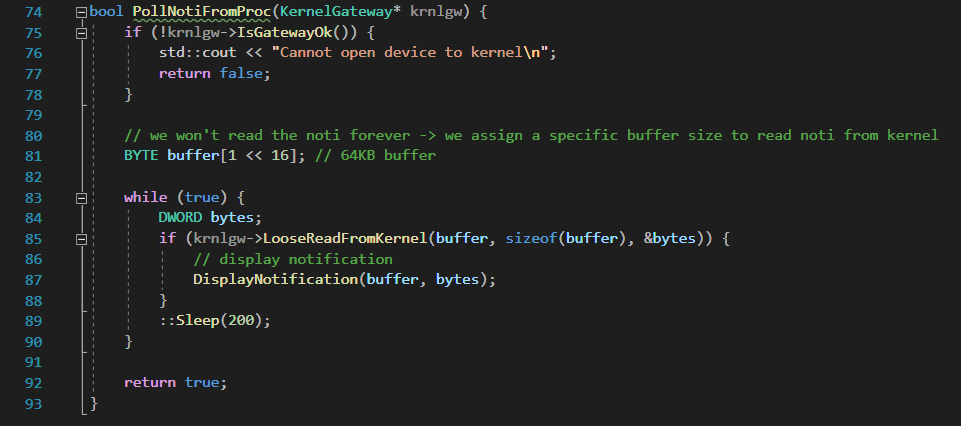
* + - Because there are many asynchronous routines accessing these linked lists, driver must take advantage of mutex whenever linked list is requested. The mutex libraries has been implemented in `autolock.h` and `fastmutex.h`
  + Inside “OnProcessNotify” routine which is registered to receive process creation/destruction notifications, we handled 2 cases corresponds to process creation and destruction notifications.
    - With process destruction notification, we simply added the notification to the notification linked list (the head is `ProcNotiItemsHead`), which later be sent to client mode.



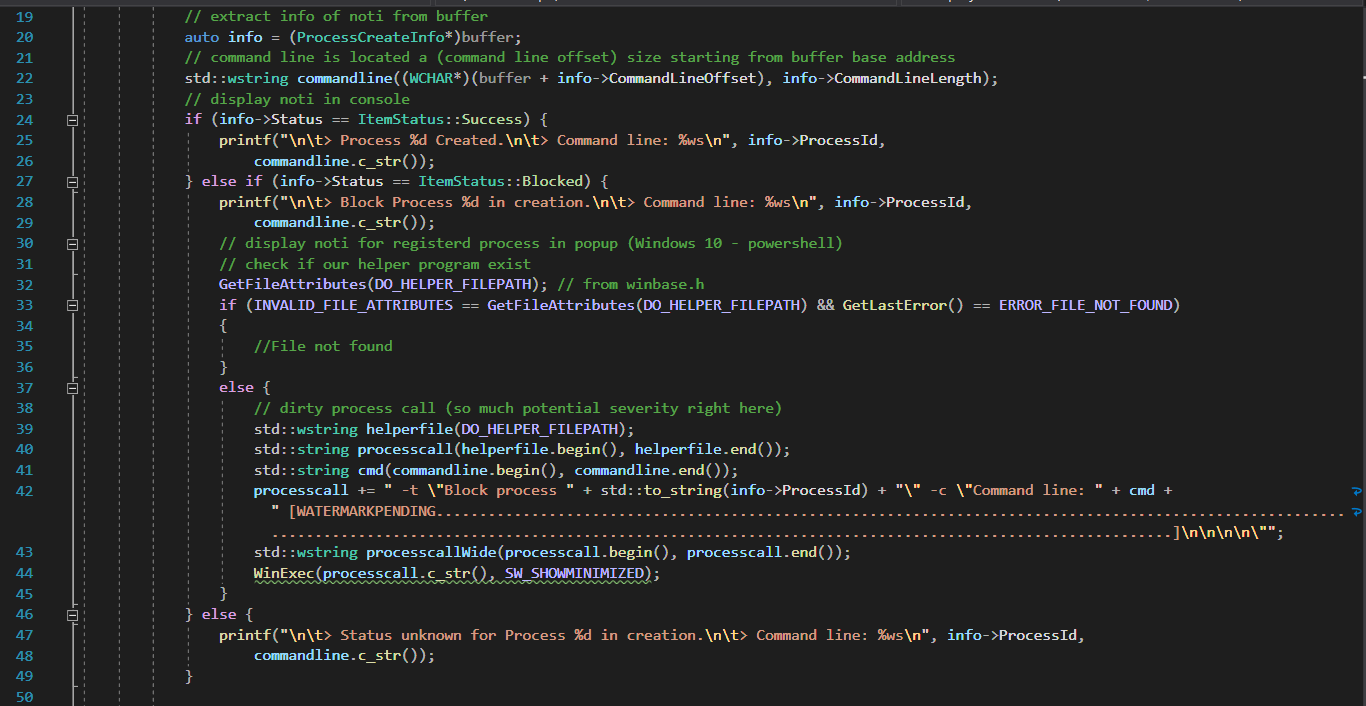
* + - With process creation notification, we compared the process path (if there is) with all process paths in blacklist. If the process path exists in the blacklist linked list (the head is `ProgItemsHead`), the driver simply adds the notification to the linked list (the head is `ProcNotiItemsHead` and the node type is `ProcessCreateInfo`) and marks this notification as blocked



* + - * Note that the autolock will automatically release the lock when the code moves out of range
  + Global data are stored in `datastore.h` and other helper functions locate in “commonhelpers.cpp”, “linkedlisthelpers.cpp” and “csstring.cpp”
  + To avoid duplicate code, driver shares essential data structures and IOCTL with user mode client via “drivercommon.h”
  1. DOClient Module
* There are 3 important files in client: main.cpp, kernelgateway.cpp and dofeature.cpp
  + Main.cpp shows up the arguments parser and control the program flow
  + KernelGatway class shows how to interact with kernel driver. Specifically, it sends command to driver using SendDataToKernel and read notifications from driver using LooseReadFromKernel
  + Dofeature.cpp contains DisplayNotification and PollNotification
    - PollNotification will assign a fixed buffer and announce it with driver. Client will polling new notifications from kernel driver via this buffer.



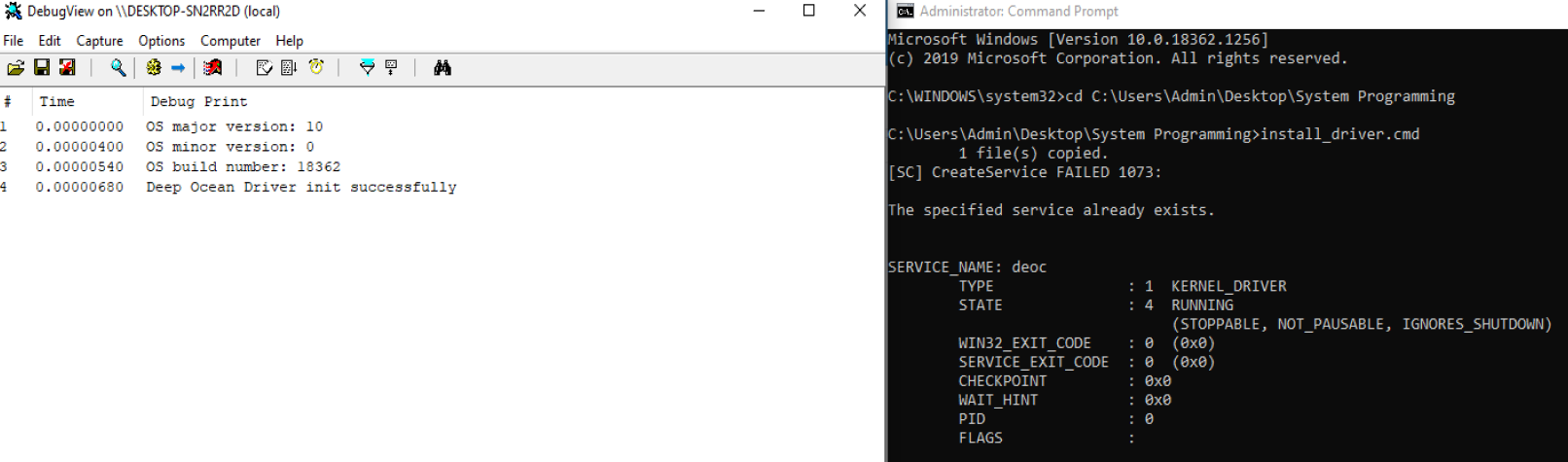
* + - * There are some severities with this method. For instance, client does not have authentication way with driver. Hence, attacker can replace DODriver with another driver and send faulty notifications to user mode client
    - DisplayNotification will be called by PollNotification whenever there’s process creation/destruction notification
      * Specially when there’s process blocked by driver, the client will call DOClientHelper for posting toast notification.



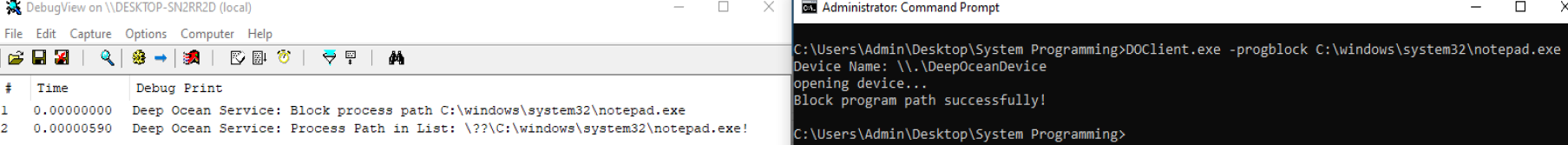
* Other functions such as displaying instructions and time are implemented in helpers.cpp
  1. DOClientHelper Module
* This module only receive arguments and pass it to toast notification. Every thing is wrapped up nicely in single main.go.

1. Demo

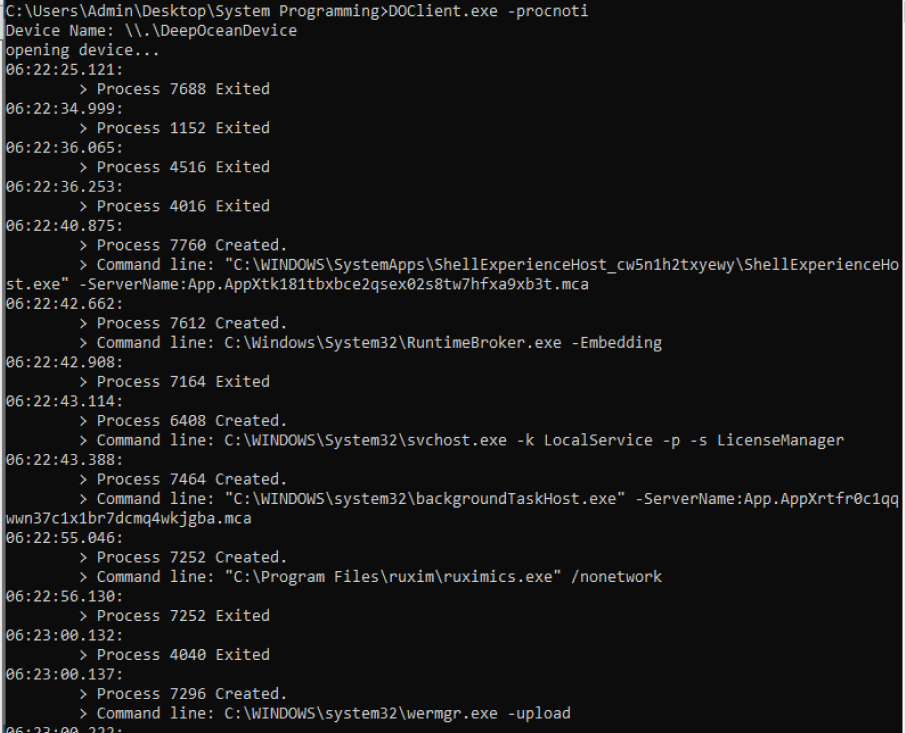
* Viewing Kernel Driver installation using DbgView.exe



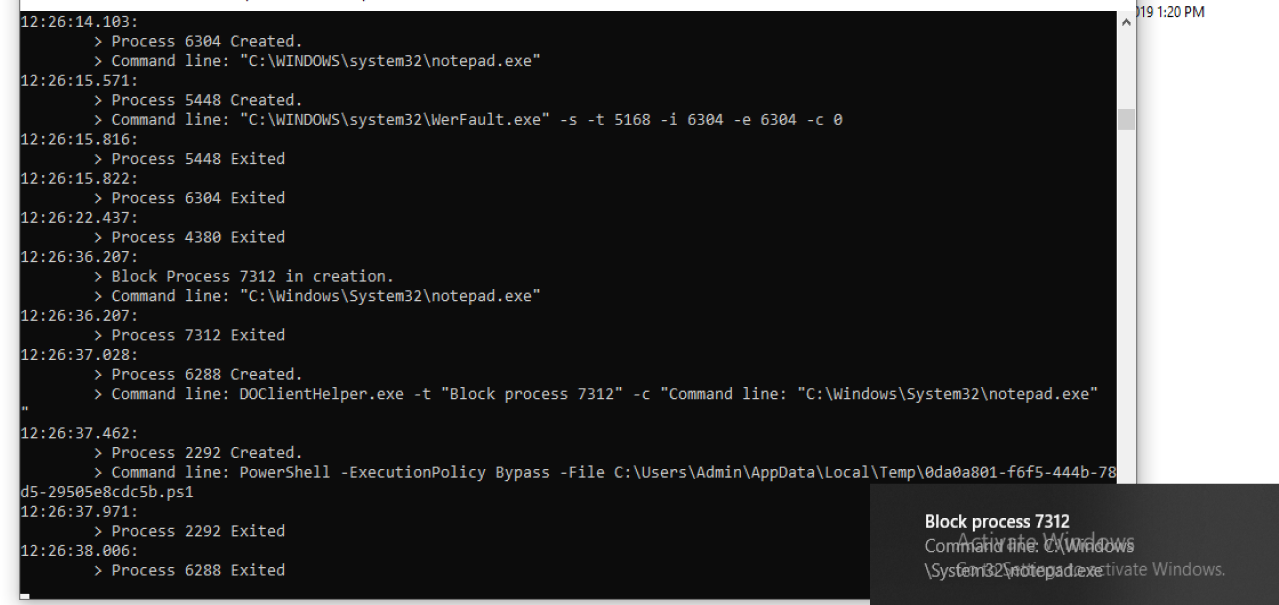
* Adding program path for blocking



* Viewing process creation/destruction notifications in console



* View toast notification when a process is blocked by DODriver



* Uninstall DODriver

