SugiotAV - Daniel Ayzenshteyn

SugiotAV is a kernel mode antivirus. It works as a minifilter driver with the ability to communicate with userspace client applications (SugiotAVConsole.exe) with control device object (CDO). It has multiple key features:

1. **Init** - Initialize the AV, clear blacklist/whitelist/block settings, if exist. For debugging and demonstration purposes.
2. **Kill** - Terminate process by <PID>.
3. **Dump** - Dump first <n> bytes of process <PID> memory, starting at its baseaddress. Save the dump in <dumpfile>.
4. **Block** - If block is specified: block any access (opening file) to file name <filename>.
5. **Blacklist** - If blacklist is specified: block any process with filename.exe from starting (with any path).
6. **Whitelist** - If whitelist is specified: allow only processes with filename.exe to be started (with any path), and block all others.

Console application usage:

```
.\SugiotAVConsole.exe -block <filename>
.\SugiotAVConsole.exe -blacklist <processname>
.\SugiotAVConsole.exe -whitelist <processname>
.\SugiotAVConsole.exe -init
.\SugiotAVConsole.exe -kill <pid>
.\SugiotAVConsole.exe -dump <pid> -size <n (bytes)> -file <DOS
file path>
```

*Example for DOS path: \??\C:\Windows\Temp\lsass.dmp

Implementation

The block functionality is implemented by minifilter driver, it registers a callback function ‘PreOperationCallback’ which is called on each file system access.

The blacklist and whitelist could be implemented in the same minifilter driver, because processes startup requires file access, and we as a file system minifilter driver can block that access as in ‘block filename’ functionality. I implemented the blacklist and whitelist functionality with ‘CreateProcessNotifyRoutine’ which is registered with the PsSetCreateProcessNotifyRoutine call. It allows our callback to be activated on each CreateProcess call, and filters it accordingly.

Kill process is tied up to the blacklist and whitelist functionality, all three of those are tied up to a function I implemented ‘TerminateProcessByPid’ - as the name says, it terminates a process by its PID.

The init is pretty trivial, only resetting a couple of global variables.
Dumping process memory has been the most challenging one. It was implemented with the help of multiple resources referenced in the code. The final solution involves calling undocumented MmCopyVirtualMemory and undocumented PsGetProcessSectionBaseAddress. The NtReadVirtualMemory could be used here, however it requires scanning the memory to find it (or searching it in KeServiceDescriptorTable). I preferred the usage of MmGetSystemRoutineAddress to resolve the addresses of the two undocumented functions at runtime.
Writing to a file from the driver also required some hassling, but was quickly resolved by the 'stackoverflow.com' solution (referenced in the code).

Installation
To install the driver we need the .inf file which is included in the provided code. We need to right click it and press 'install' and accept the installation.
After the driver is installed we can use the regular SCM to start and stop the driver:

```
sc start SugiotAV
sc stop SugiotAV
```

We also can observe it appearing in the list of minifilter drivers with 'fltmc' windows tool.
Note: we need to run windows in TestMode to allow self signed driver installation.

Compilation and linking
Compilation done in visual studio, we need to adjust manually the next 3:

1. Setup driver entry point to the "DriverEntry" (in linker advanced)
2. Add "C:\Program Files (x86)\Windows Kits\10\Include\10.0.26100.0\km" to additional include directories in C/C++ General.
3. Minifilter: Add in Linker Input additional dependency - "$(DDK_LIB_PATH)\fltMgr.lib"