Understanding File System Minifilter and Legacy Filter Load Order

ntdebug 25 Mar 2013 4:39 PM **2**

Hello, my name is Fred Jeng from the Global Escalation Services team. For today's post, I want to go over how Windows 7 and Windows Server 2008 R2 load file system mini-filters in a mixed environment when legacy filters are also present. I recently came across an issue where the filters were being loaded out of order based on their altitudes. This can cause all sorts of problems with a filter driver's functionality if they are incorrectly positioned on the stack. Take for example the following filter stack, obtained using the fltmc command from the cmd prompt:

```
C:\Windows\system32>fltmc

Filter Name                     Num Instances    Altitude    Frame
------------------------------  -------------  ------------ -----
AVLegacy                                       389998.99   <Legacy>
EncryptionLegacy                               149998.99   <Legacy>
AVMiniFilter                            3      328000         0
luafv                                   1      135000         0
FileInfo                                13     45000          0
```

At first glance it looks like there is a problem causing the legacy encryption filter to be loaded above the antivirus minifilter, which has a higher altitude. This may cause issues with AVMiniFilter as the IOs that it receives are still encrypted. Due to limitations in how the filter drivers attach to the driver stack, this is actually the intended behavior. However, there is a solution to manipulate the load order to load the legacy filters correctly based on their altitude.

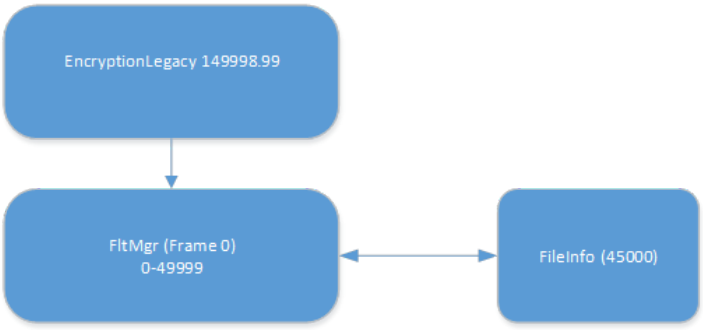
First some background information regarding legacy filters and minifilters.
In the old days before minifilters, legacy drivers can only attach at the top of the driver stack so the load order also controlled the attachment order. The earlier a legacy driver loads, the lower it can attach on the file system stack. Minifilters on the other hand can load at any time, but their positions relative to other minifilters are controlled by their altitude. When a minifilter loads, it needs to register with an appropriate frame created by fltmgr. Each frame is a fltmgr device object and represents a range of altitudes. There can be more than one frame on the file system stack but the range of altitudes that each frame represents cannot overlap with the altitude range of another frame. For interoperability with legacy drivers, minifilters must still maintain a load order group. The frames are created and managed by fltmgr, which itself is a legacy driver. The ramification of this is that fltmgr must follow the old legacy filter driver rules and attach only at the top of the stack.

From the above example, let's walk through how the legacy and minifilters above are loaded to get us into the state such that the altitude appears to be out of order.
First, here are the details for the 5 drivers.

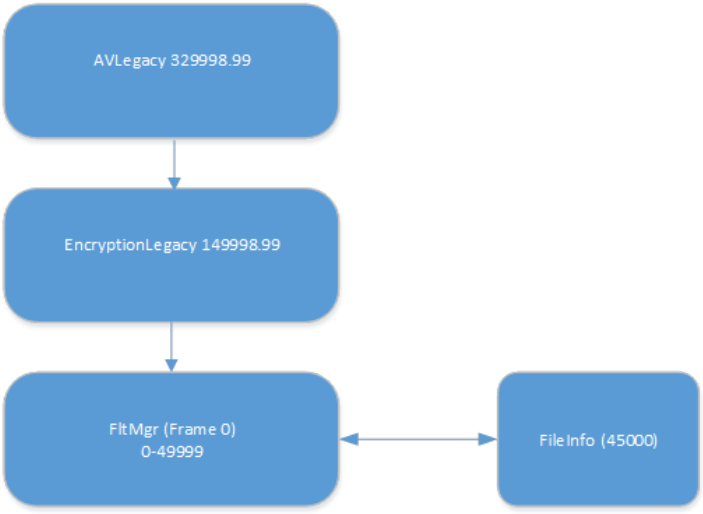
Driver Name	Type	Load Order	Start Type	Altitude
AVLegacy	Legacy	FsFilter Anti-Virus	SERVICE_BOOT_START	329998.99
AVMiniFilter	Minifilter	FsFilter Anti-Virus	SERVICE_BOOT_START	328000
EncryptionLegacy	Legacy	FsFilter Encryption	SERVICE_BOOT_START	149998.99
Luafv	Minifilter	FsFilter Virtualization	SERVICE_AUTO_START	135000
FileInfo	Minifilter	FSFilter Bottom	SERVICE_BOOT_START	45000

MSDN has an article that describes load order groups and altitudes for minifilters: **http://msdn.microsoft.com/en-us/library/windows/hardware/ff549689%28v=vs.85%29.aspx**.
Referencing this article regarding load order groups and altitudes for minifilter drivers, we can determine that our filters will load in the following order.
FileInfo
EncryptionLegacy
AVLegacy
AVMiniFilter
luafv

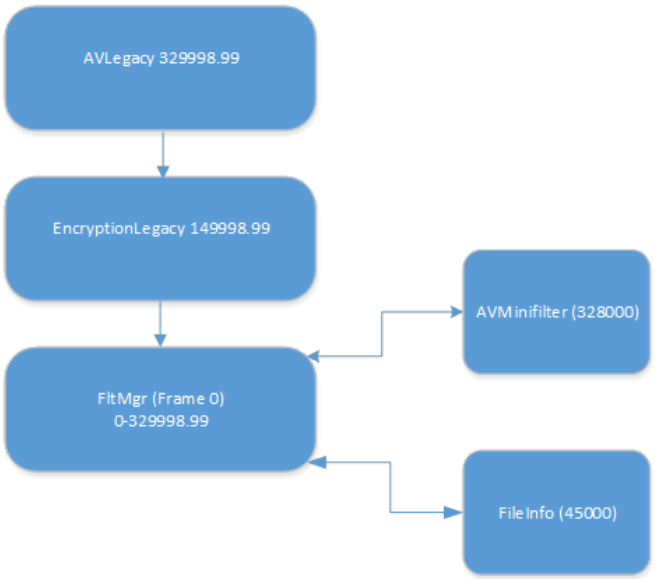
On system bootup, when fltmgr.sys loads it will create Frame 0 with a default altitude range of 0 to 49999. When FileInfo loads with an altitude of 45000, it will fit into the default Frame 0. Next to load is EncryptionLegacy. Since this is a legacy driver, it will attach on top of the legacy driver fltmgr.sys. So this is how our file system stack looks right now.



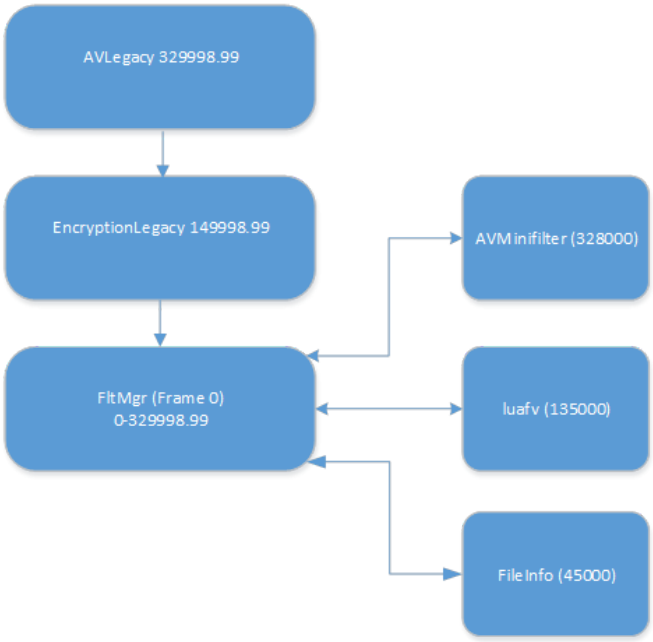
Next up is the AVLegacy driver. This is a legacy driver, so it has to attach above EncryptionLegacy.



Now the AVMinifilter will load with an altitude of 328000. The OS will check if it will fit in the Frame 0 Fltmgr, but this frame only supports an altitude of 0-45000. Before deciding to create a new fltmgr Frame instance, it will check if there are any legacy filters attached above Frame 0 and adjust Frame 0's altitude if there are. So in our case, we do have legacy filters on the file system stack at this point and so we go up the list of legacy drivers. First we see EncryptionLegacy with an altitude of 149998.99 so we adjust Frame 0 to cover from 0 to 149998.99. We continue up the list and see AVLegacy with an altitude of 329998.99 so we again adjust the altitude of Frame 0 to now cover 0-329998.99. The reason we do this is because Frame 0 now must handle all minifilters below 329998.99. Since we can only attach legacy filters to the top of the stack, if we add an additional FltMgr frame instance, it has to sit above AVLegacy and can only support minifilters with altitude of 329998.99 or higher. Now that Frame 0 supports 0-329998.99, we can register AVMinifilter with Frame 0.



At this point, you can already see that AVMinifilter which has a higher altitude then EncryptionLegacy will be loaded below EncryptionLegacy. The last driver to load is the luafv minifilter, and it will fit into Frame 0.



A couple of things to point out.
Why can't we insert a Frame between AVLegacy and EncryptionLegacy when ACMinifilter loads?
This is due to how the file system stack is constructed with legacy drivers only being able to attach to the top of the stack. Since FltMgr is a legacy driver, it has to conform to these rules.

Why do we adjust the altitude in Frame 0 to cover 0-329998.99? Why not stop at 149998.99?
If Frame 0 only adjusts its altitude to the legacy filter directly attached above it and not all the way to the highest attached legacy filter, we won't be able to handle some range of mini-filters. For example, assume we only adjust Frame 0 to cover 0-149998.99, then when the AVMinifilter with an altitude of 328000 comes along, it won't fit in Frame 0, and we're unable to insert a Frame between AVLegacy (329998.99) and EncryptionLegacy(149998.99) so we would either be unable to load AVMinifilter, or we would have to create Frame 1 above AVLegacy and load AVMinifilter there. In which case we would again be faced with the altitude disordering issue.

If this is the expected behavior, how do we resolve the problem of EncryptionLegacy being loaded above AVMinifilter? The solution is to inject a dummy minifilter that loads at the appropriate time to force fltmgr to create a Frame between the legacy filters. For our case above, I used the DDK to create the NullFilter minifilter driver and changed the load order to FSFilter Compression and gave it an altitude of 160030 which is within the assigned altitude for FSFilterCompression and set the start type to SERVICE_BOOT_START. Please note that I only used this driver in a test environment, production minfilter drivers must use an **altitude assigned by Microsoft**.

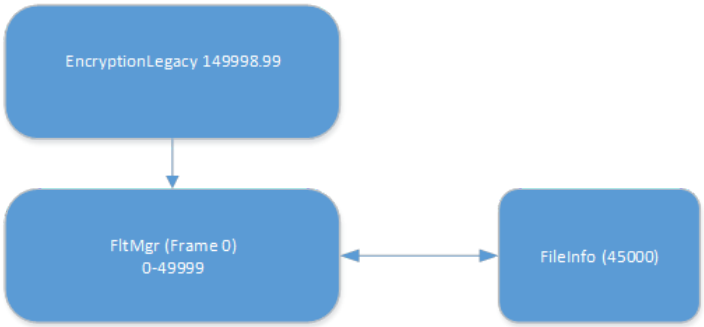
For information on minifilter load order groups and altitude, reference **http://msdn.microsoft.com/en-us/windows/hardware/gg462963.aspx**.

Our list of filters are as follows:

Driver Name	Type	Load Order	Start Type	Altitude
AVLegacy	Legacy	FsFilter Anti-Virus	SERVICE_BOOT_START	329998.99
AVMiniFilter	Minifilter	FsFilter Anti-Virus	SERVICE_BOOT_START	328000
EncryptionLegacy	Legacy	FsFilter Encryption	SERVICE_BOOT_START	149998.99
Luafv	Minifilter	FsFilter Virtualization	SERVICE_AUTO_START	135000
FileInfo	Minifilter	FSFilter Bottom	SERVICE_BOOT_START	45000
NullFilter	Minifilter	FSFilter Compression	SERVICE_BOOT_START	160030

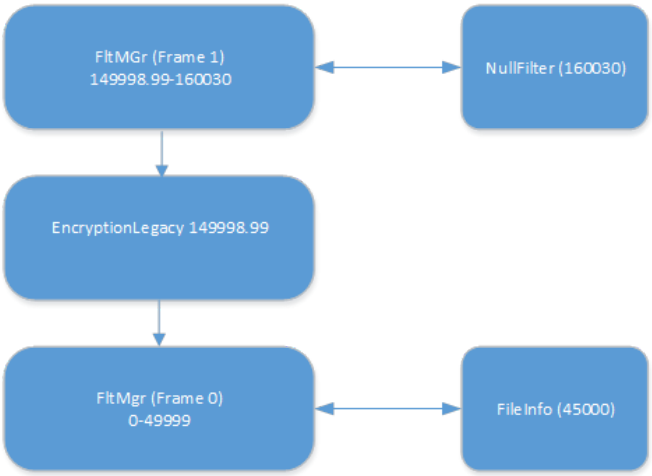
So with the new NullFilter dummy driver, our filter load order should be as follows:
FileInfo
EncryptionLegacy
NullFilter
AVLegacy
AVMiniFilter
luafv

After FileInfo and EncryptionLegacy loads, the stack is the same as what we had earlier.

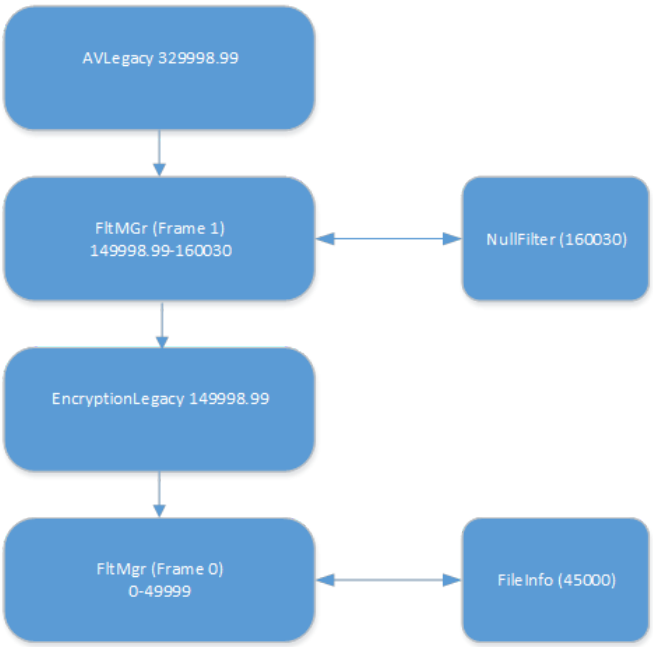


Now when the NullFilter minifilter loads with an altitude of 160030, we see that it doesn't fit in Frame 0. As before, we check for any attached legacy filter drivers and see EncryptionLegacy so we adjust Frame 0 to cover 0-149998.99. Since NullFilter still does not fit in Frame 0, we will create a new Frame and attach it above the EncryptionLegacy

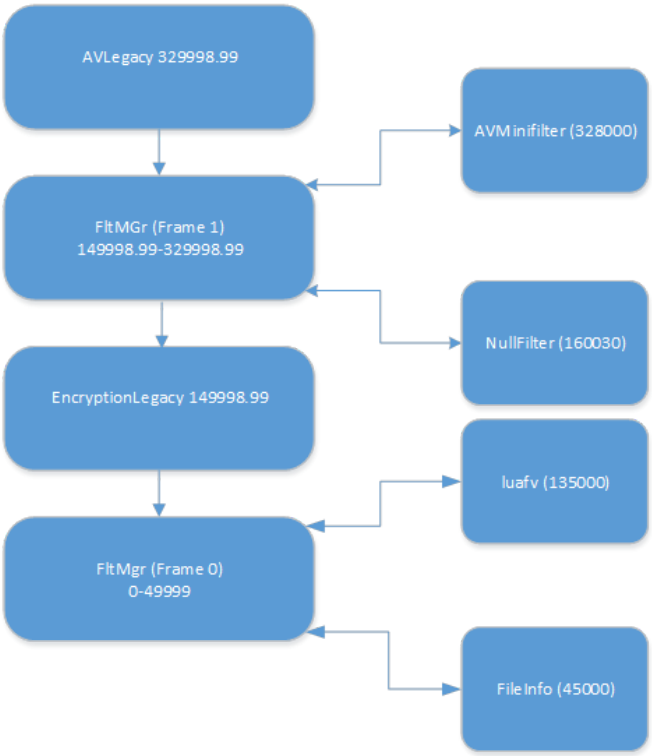
driver.



The AVLegacy driver will load next, and since it is a legacy driver, it will attach above the Frame 1 instance of FltMgr.



The last two minifilters to load are AVMiniFilter and luafv. When AVMinifilter loads into Frame 1 with an altitude of 328000, it will see that Frame 1 at the time only supports 149998.99-160030. It follows the same algorithm to check if there are any legacy filters attached above the frame. In this case, we have AVLegacy attached above Frame 1 so we adjust Frame 1 to cover 149998.99-329998.99 before inserting AVMinifilter into Frame 1.



By strategically injecting a dummy minifilter driver, we can get the legacy and minifilter drivers to all load at the correct altitude.

Comments

Thierry Franzetti
27 Mar 2013 2:15 AM

When I run the fltmc command on a system (Windows 7, 32 bits) with a legacy file system filter driver, the 'altitude' colmun is empty.

How is it possible that a legacy filter be assigned an altitude? I thought it was only for mini-filters.

[Altitudes for legacy filters are OS assigned when the legacy filter loads based on its Load Order Group.]

Miroslav Reisinger
27 Mar 2013 2:42 AM

Hi Fred, thanks for the article. When I run fltmc on my Win7 laptop, I'm getting this:

Filter Name	Num Instances	Altitude	Frame
BackupMinifilter	1	280500	1
luafv	1	135000	1
AVLegacy		<Legacy>	
FileInfo	3	45000	0

If I understood this correctly, because AVLegacy is missing Altitude, fltmgr will create Frame 0 with a default altitude range of 0 to 49999 and AVLgeacy (SERVICE_BOOT_START) will attach at the top of the driver stack. FileInfo fits into frame 0. So for for luafv and BackupMinifilter (both SERVICE_AUTO_START), 2 new fltmgr frame instances will be created and placed above AVLegacy?

[AVLegacy should have an implicit altitude based on its load order group that is not shown in the output so AVLegacy filter will load before luafv and BackMinifilter based on its start type and load order group. When AVLegacy loads, it will cap Frame 0 so Fltmgr will create Frame 1 above AVLegacy for luafv and BackupMinifilter.]