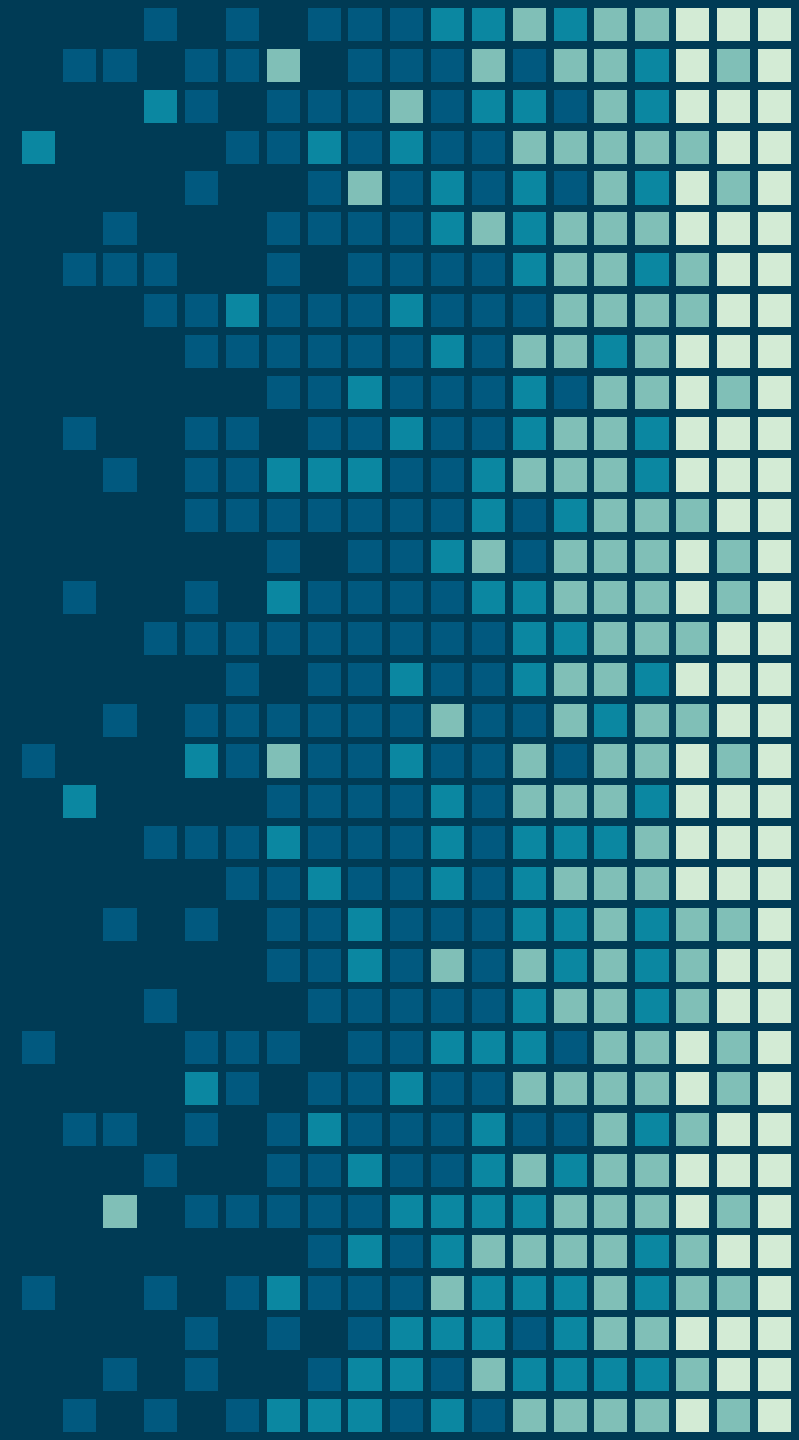


BYOD: Bring Your Own Defender

Turning Windows Defender into
Your own Rootkit



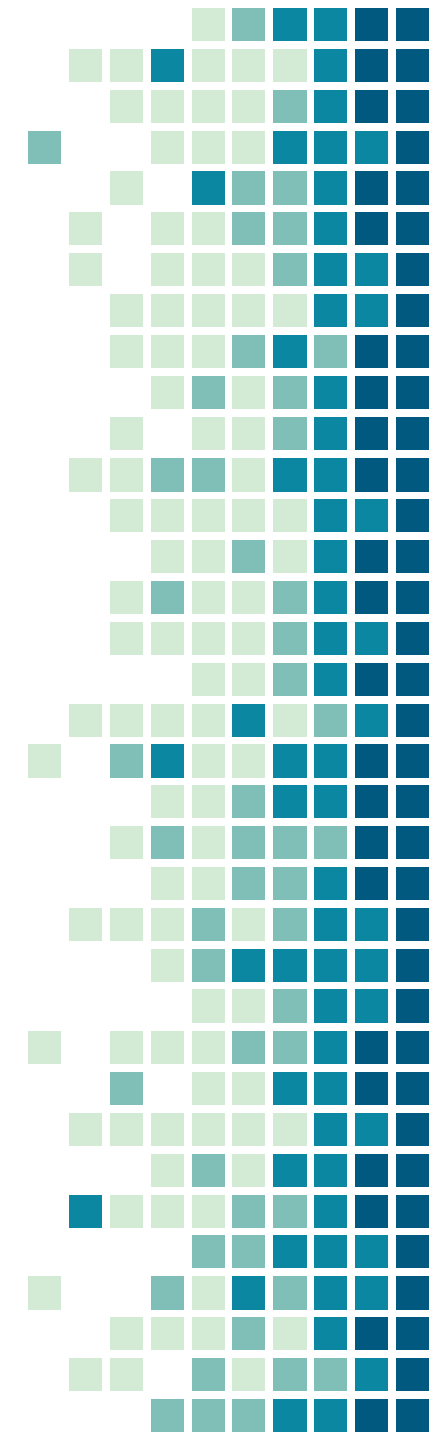
About Me

- Senior Software Engineer at CrowdStrike
 - Previously Security Researcher at SentinelOne
- Instructor of worldwide Windows internals classes
- Circus artist – teaching and performing aerial acrobatics
- Author of articles and tools at www.windows-internals.com
 - CET Internals, Extension Host Hooking, Kernel Exploit Mitigations
 - Heap-Backed Pool Visualizer: PoolViewer
- Former pastry chef



So You Loaded a Rootkit – Now What?

- Rootkits are EDRs with less unittests
 - Both use the same OS mechanisms for similar goals
 - EDRs can also do some very questionable things
- What do they need?
 - Process monitoring
 - Filesystem monitoring
 - Registry monitoring
 - Network communication
 - Not to get caught by EDRs – this is the hard part!

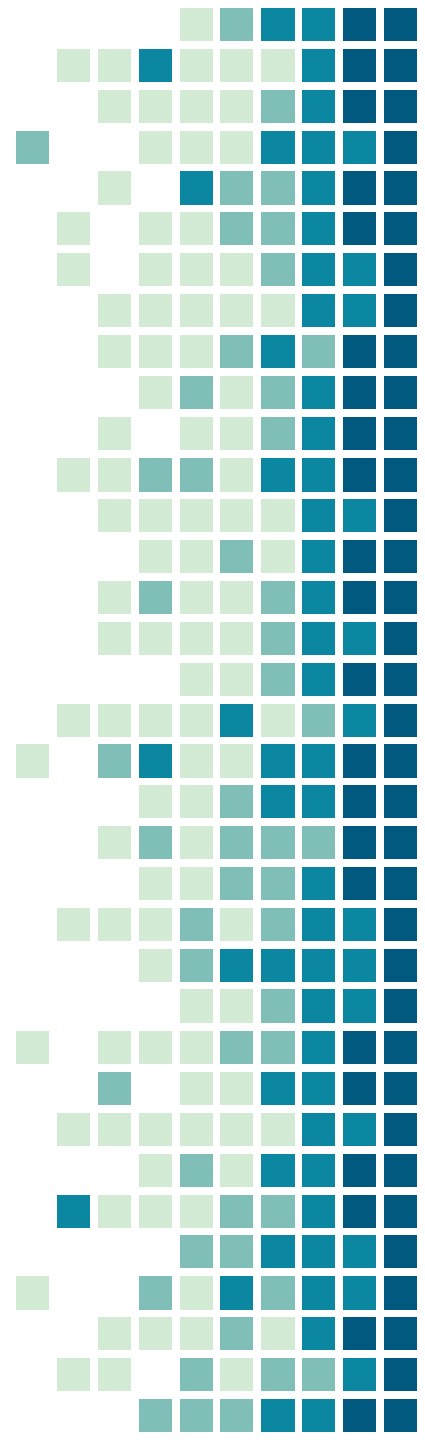




Florian Roth ⚡
@cyb3rops

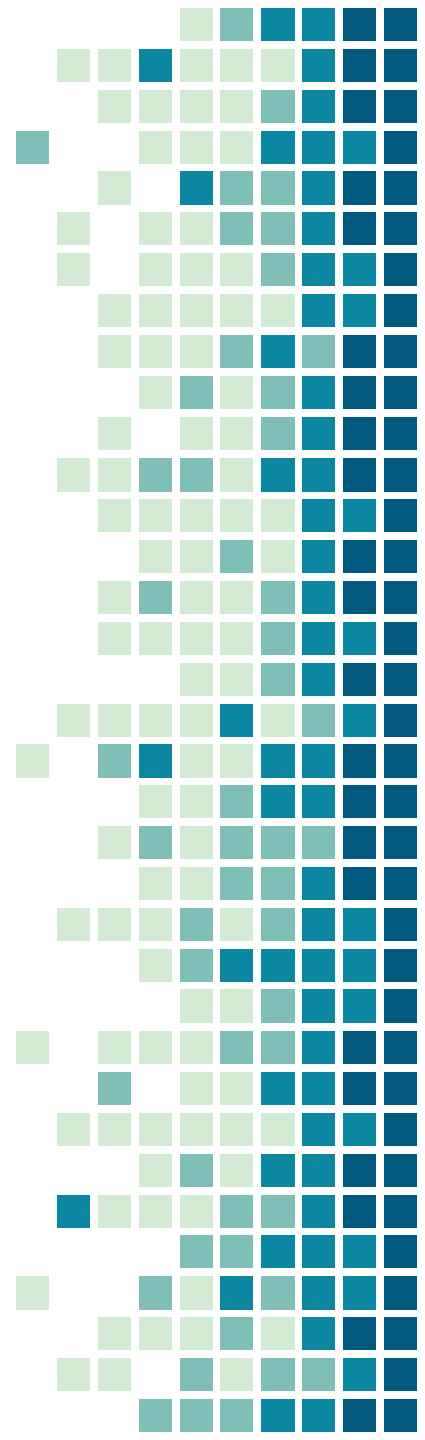
...

OT : Oh - a rule match on an obfuscated PowerShell script written by Matt Graeber that bypasses AMSI ... this must be a threat or a penetration ... nope - just a leading EDR that runs it and may use it to ... well, I have no idea and think they should rewrite their crap

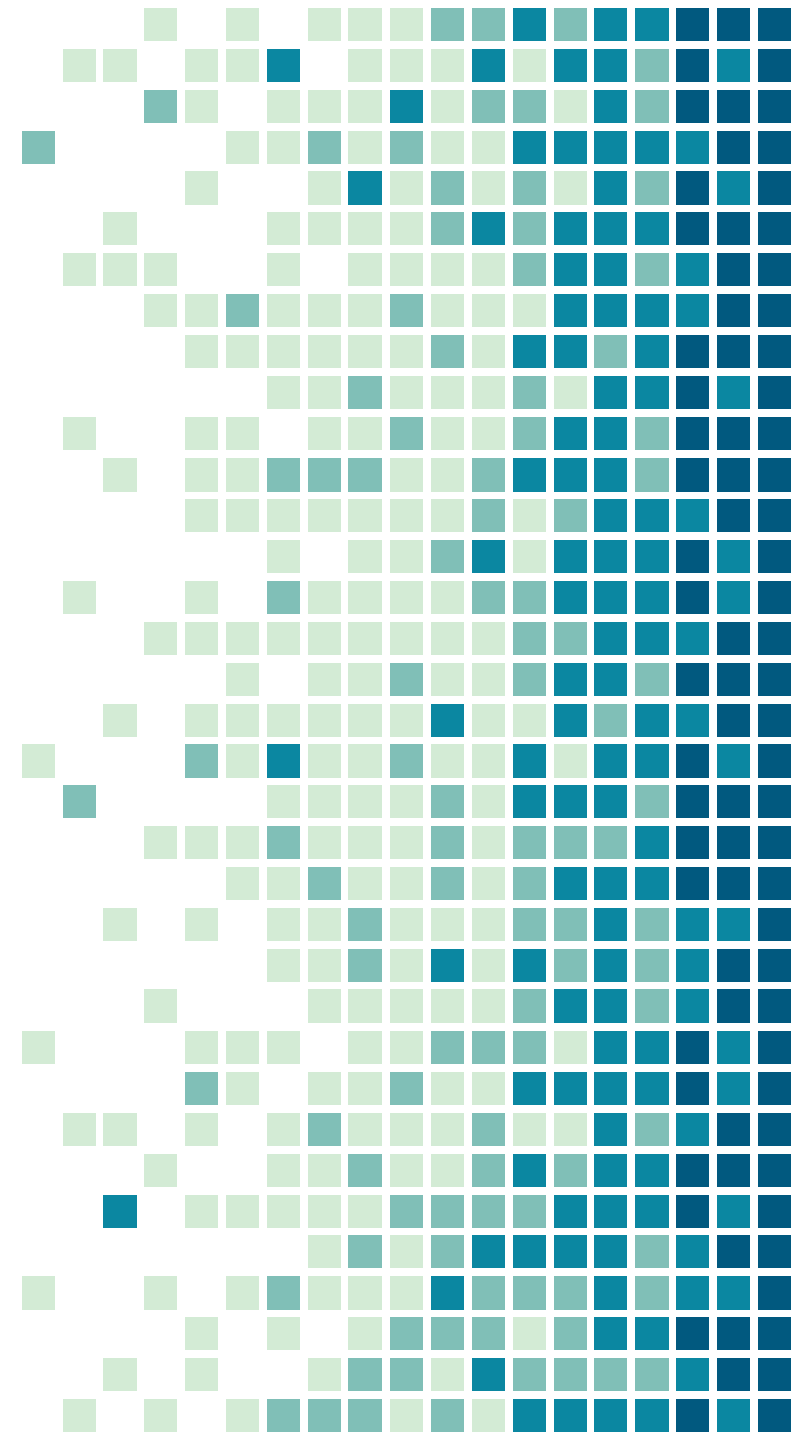


What if someone else did the work for you?

- Windows Defender is running on every Windows 10/11 machine by default
- Has multiple drivers that implement most functionality a rootkit would need
- If you're in the kernel you can access it too
 - Bypasses EDR monitoring – including Windows Defender
 - More limited than implementing it yourself – but less visible too

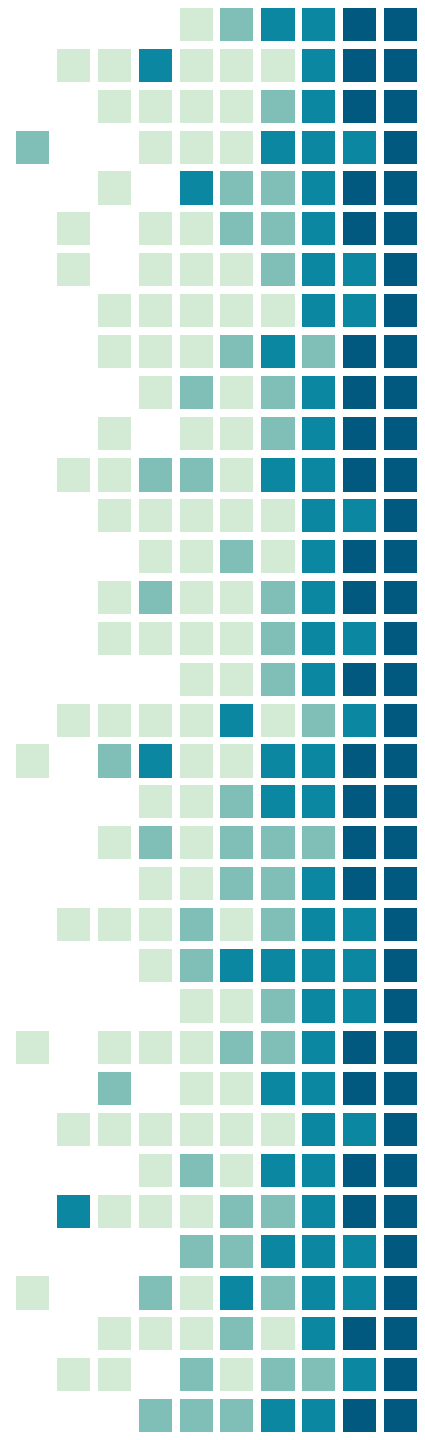


System Callbacks			
Routine Address	Module	Additional Infor...	
✓ CreateProcess			
0xFFFFF80222405980	\SystemRoot\System32\drivers\cng.sys		
0xFFFFF80223125E00	\SystemRoot\system32\drivers\wd\WdFilter.sys		
0xFFFFF8021C95B0C0	\SystemRoot\System32\drivers\ksecdd.sys		
0xFFFFF802237EC980	\SystemRoot\System32\drivers\tcpip.sys		
0xFFFFF80223D4D990	\SystemRoot\system32\drivers\iorate.sys		
0xFFFFF8021CF0C8D0	\SystemRoot\system32\CI.dll		
0xFFFFF802242B63B0	\SystemRoot\System32\drivers\dxgkrnl.sys		
0xFFFFF8022512BCF0	\SystemRoot\system32\drivers\peauth.sys		
0xFFFFF8022BED3AC0	\SystemRoot\System32\drivers\wtd.sys		
0xFFFFF8022BED4FA0	\SystemRoot\System32\drivers\wtd.sys		
✓ CreateThread			
0xFFFFF80223127540	\SystemRoot\system32\drivers\wd\WdFilter.sys		
0xFFFFF802231272A0	\SystemRoot\system32\drivers\wd\WdFilter.sys		
0xFFFFF8021E911010	\SystemRoot\system32\drivers\mmcsc.sys		
✓ LoadImage			
0xFFFFF802231267B0	\SystemRoot\system32\drivers\wd\WdFilter.sys		
0xFFFFF80224DBE580	\SystemRoot\system32\DRIVERS\ahcache.sys		
0xFFFFF8022BED52A0	\SystemRoot\System32\drivers\wtd.sys		
> KeBugCheck			
> KeBugCheckReason			
✓ CmRegistry			
0xFFFFF802214893F0	\SystemRoot\system32\ntoskrnl.exe		
0xFFFFF802231176A0	\SystemRoot\system32\drivers\wd\WdFilter.sys		
0xFFFFF80221264F30	\SystemRoot\system32\ntoskrnl.exe		
> Shutdown			
> LastChanceShutdown			
✓ ObProcess			
0xFFFFF80223123C30	\SystemRoot\system32\drivers\wd\WdFilter.sys	PreCallback	
ObThread			
✓ ObDesktop			
0xFFFFF80223123C30	\SystemRoot\system32\drivers\wd\WdFilter.sys	PreCallback	
SeFileSystem			
> SeFileSystemEx			
PowerSettings			
Total listed callbacks:			

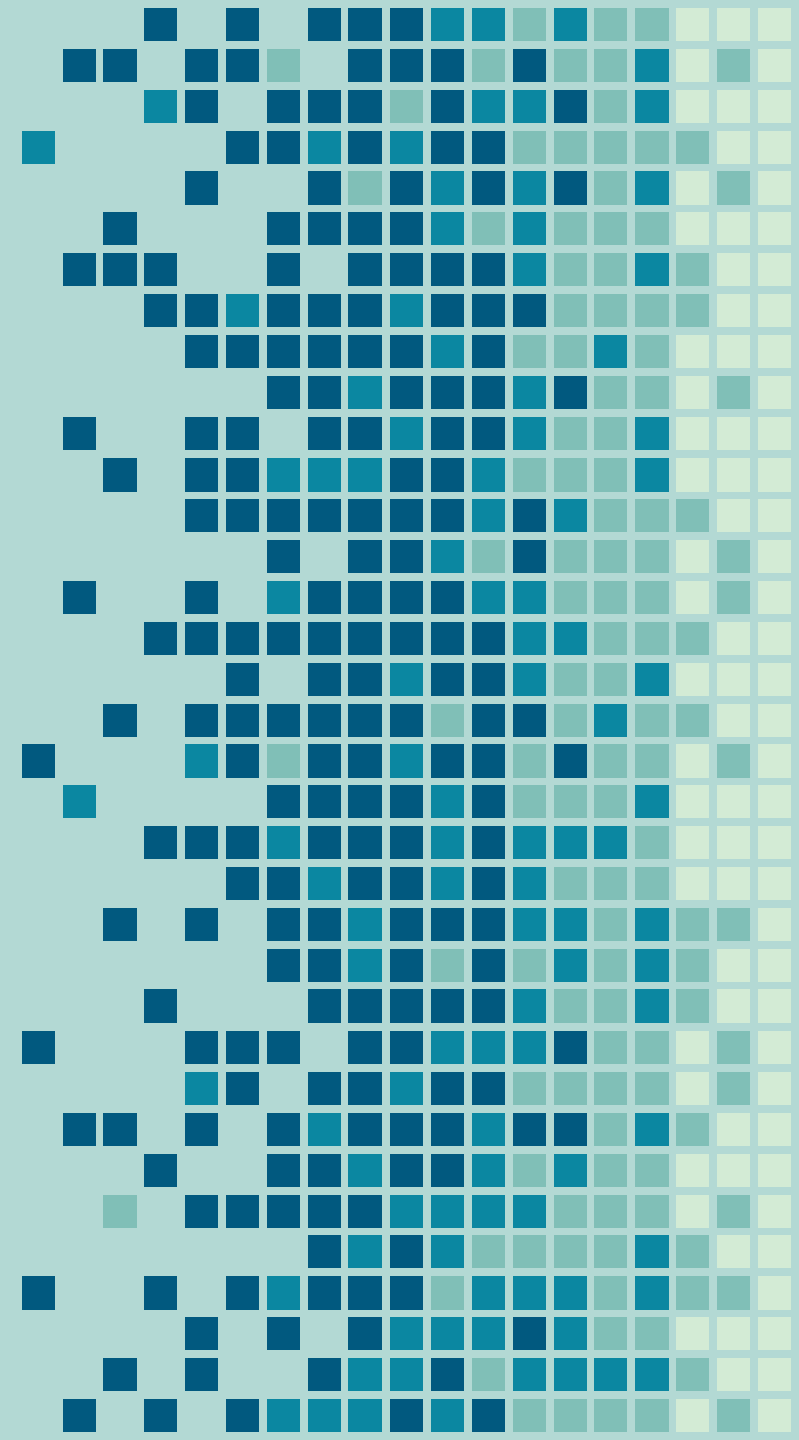


The Lego Pieces: Windows Defender Drivers

- WdBoot.sys – ELAM driver
- WdFilter.sys – File System filter driver
- WdNisDrv.sys – Network driver
- WdDevFlt.sys – Filter driver
- WdFilter.sys is the largest and most interesting driver

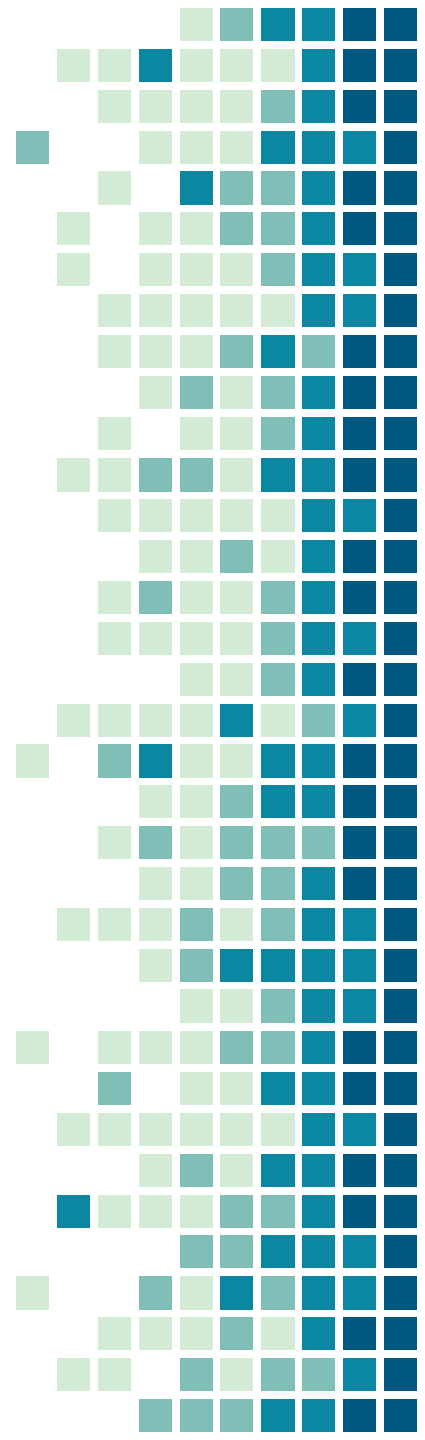


Collect Data with Callback Objects

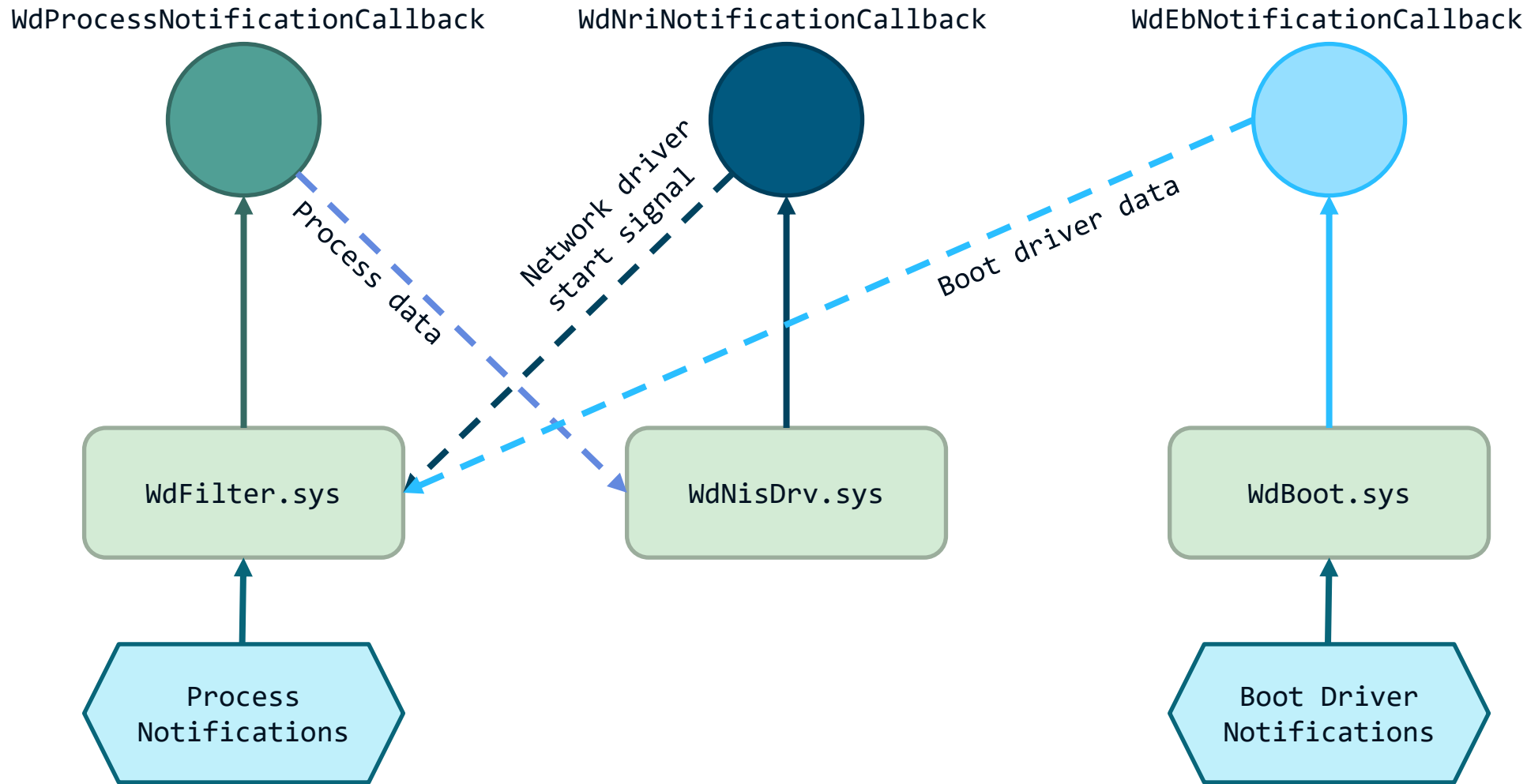


Process Monitoring Through Callback Objects

- Callback objects allow inter-driver communication
 - Named objects
 - Any driver can register to a callback or notify one
- Windows Defender creates 3 callback objects:
 - WdEbNotificationCallback
 - WdNriNotificationCallback
 - WdProcessNotificationCallback
- Used to transfer information between the three drivers

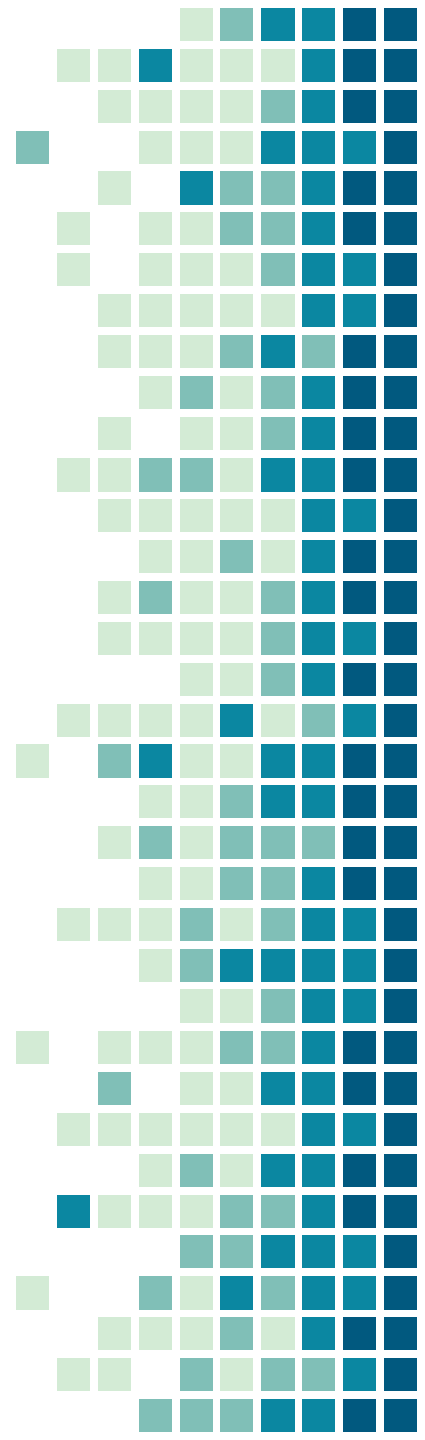


Process Monitoring Through Callback Objects

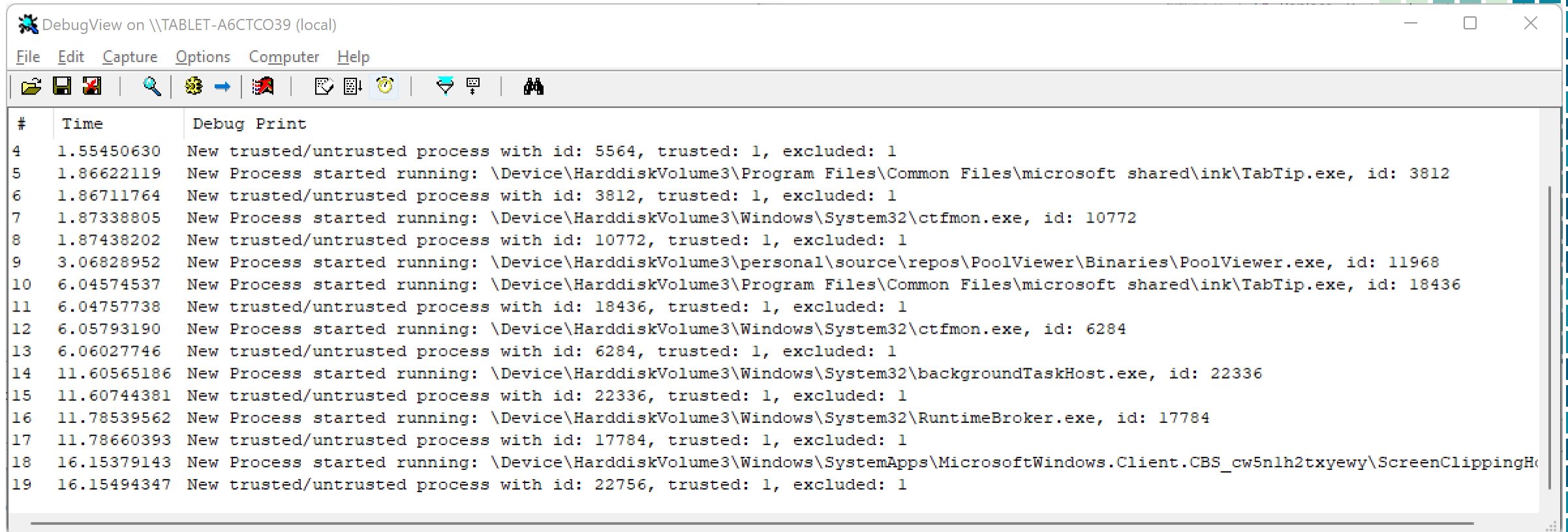


Process Monitoring Through Callback Objects

- WdProcessNotificationCallback
- Gets notified with data for every new process
 - And every new trusted or untrusted process
- PID, Parent PID, Image Name, Status – Created/Terminated
- Register to the callback and get all process data
 - No need for process notifications – monitored by EDRs
 - To get all process data for processes already running: notify WdNriNotificationCallback
- Or notify the callback with false data to lie to Defender



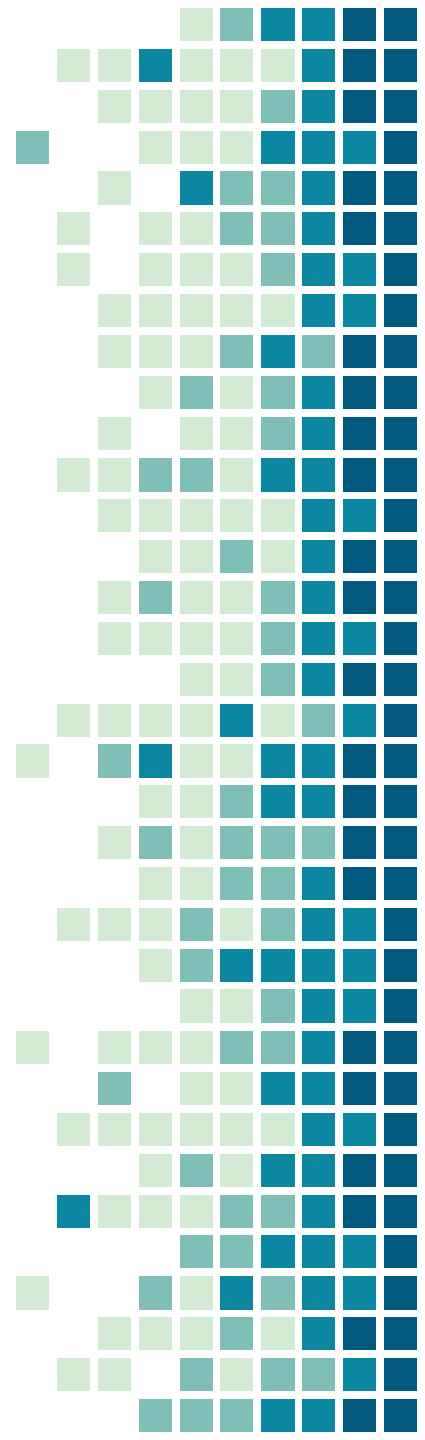
Live Process Monitoring with WD Callback Object



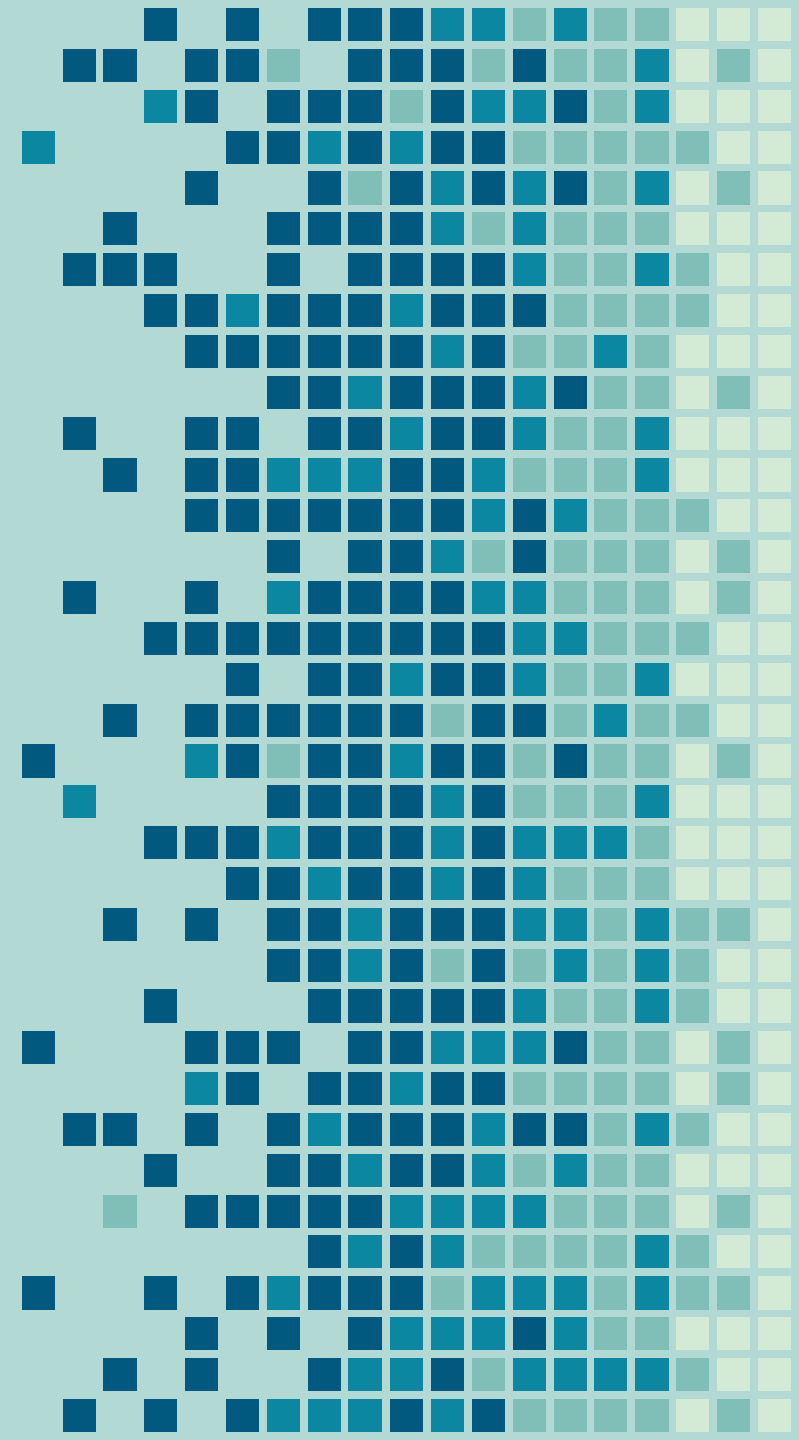
#	Time	Debug Print
4	1.55450630	New trusted/untrusted process with id: 5564, trusted: 1, excluded: 1
5	1.86622119	New Process started running: \Device\HarddiskVolume3\Program Files\Common Files\microsoft shared\ink\TabTip.exe, id: 3812
6	1.86711764	New trusted/untrusted process with id: 3812, trusted: 1, excluded: 1
7	1.87338805	New Process started running: \Device\HarddiskVolume3\Windows\System32\ctfmon.exe, id: 10772
8	1.87438202	New trusted/untrusted process with id: 10772, trusted: 1, excluded: 1
9	3.06828952	New Process started running: \Device\HarddiskVolume3\personal\source\repos\PoolViewer\Binaries\PoolViewer.exe, id: 11968
10	6.04574537	New Process started running: \Device\HarddiskVolume3\Program Files\Common Files\microsoft shared\ink\TabTip.exe, id: 18436
11	6.04757738	New trusted/untrusted process with id: 18436, trusted: 1, excluded: 1
12	6.05793190	New Process started running: \Device\HarddiskVolume3\Windows\System32\ctfmon.exe, id: 6284
13	6.06027746	New trusted/untrusted process with id: 6284, trusted: 1, excluded: 1
14	11.60565186	New Process started running: \Device\HarddiskVolume3\Windows\System32\backgroundTaskHost.exe, id: 22336
15	11.60744381	New trusted/untrusted process with id: 22336, trusted: 1, excluded: 1
16	11.78539562	New Process started running: \Device\HarddiskVolume3\Windows\System32\RuntimeBroker.exe, id: 17784
17	11.78660393	New trusted/untrusted process with id: 17784, trusted: 1, excluded: 1
18	16.15379143	New Process started running: \Device\HarddiskVolume3\Windows\SystemApps\MicrosoftWindows.Client.CBS_cw5nlh2txyewy\ScreenClippingHost.exe, id: 22756
19	16.15494347	New trusted/untrusted process with id: 22756, trusted: 1, excluded: 1

Boot Driver Monitoring through Boot Callback

- WdEbNotificationCallback
- Gets notified with data for all loaded boot drivers
 - Notified by Defender ELAM driver: wdboot.sys
- Contains data such as:
 - Image name
 - Hash
 - Certificate

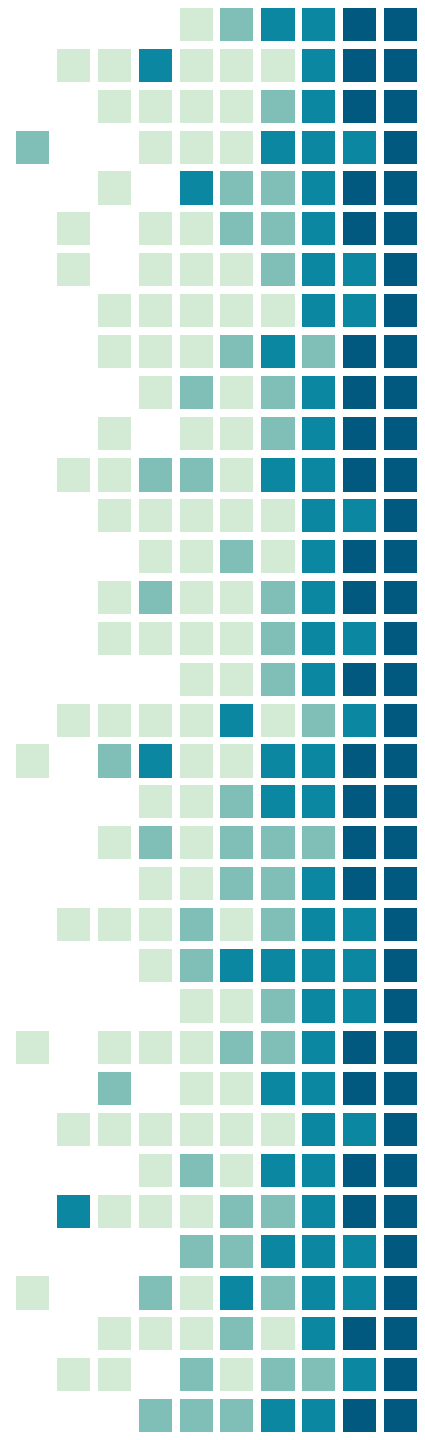


User-Kernel communication



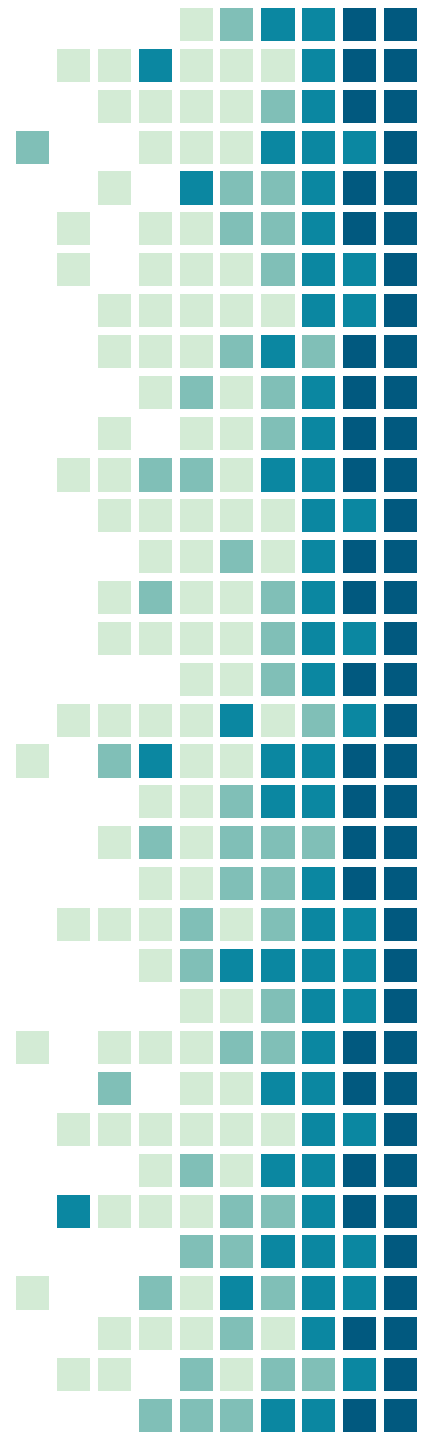
Communication Ports

- Used for communication between WD user-mode process and the WdFilter.sys driver
- Driver initializes 5 ports:
 - MicrosoftMalwareProtectionControlPortWD
 - MicrosoftMalwareProtectionPortWD
 - MicrosoftMalwareProtectionVeryLowIoPortWD
 - MicrosoftMalwareProtectionRemoteIoPortWD
 - MicrosoftMalwareProtectionAsyncPortWD



Communication Ports – Limitations

- Only a process running with the Defender SID can register for the communication ports
 - But you have a driver: can give a process the correct SID or edit the security descriptor of the port
- Only the Windows Defender process allowed to connect
 - Identified by EPROCESS and PID
 - Both are saved in global data structure
 - Data is in the pool and can be edited by a kernel driver
 - Set the first time a process connects to any of the ports
 - Process also gets exempted from detection

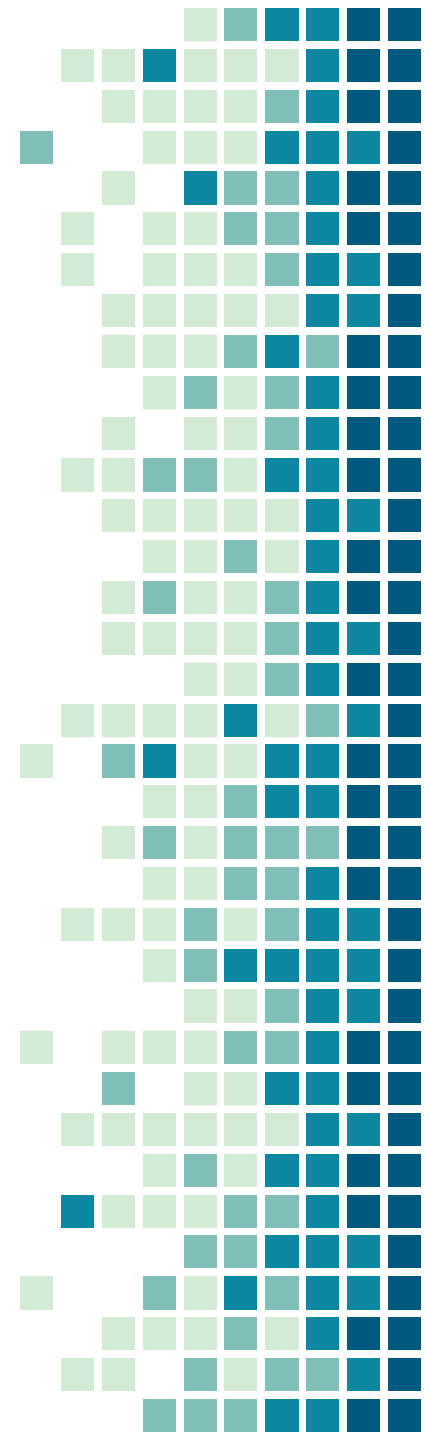


Async Port Security Descriptor

```
lkd> dx -s @$hdr = ((nt!_OBJECT_HEADER*)0xfffffaf8c36fcb670)
lkd> dx new {Name = @$hdr->ObjectName, Type = @$hdr->ObjectType, SD = @$hdr->SecurityDescriptor & ~0xf}
new {Name = @$hdr->ObjectName, Type = @$hdr->ObjectType, SD = @$hdr->SecurityDescriptor & ~0xf}
    Name      : "MicrosoftMalwareProtectionAsyncPortWD"
    Type      : FilterConnectionPort
    SD        : 0xffff850f308cc720
lkd> !sd 0xffff850f308cc720 1
->Revision: 0x1
->Sbz1      : 0x0
->Control    : 0x8004
              SE_DACL_PRESENT
              SE_SELF_RELATIVE
->Owner      : S-1-5-32-544 (Alias: BUILTIN\Administrators)
->Group      : S-1-5-18 (Well Known Group: NT AUTHORITY\SYSTEM)
->Dacl       :
->Dacl       : ->AclRevision: 0x2
->Dacl       : ->Sbz1        : 0x0
->Dacl       : ->AclSize     : 0x30
->Dacl       : ->AceCount    : 0x1
->Dacl       : ->Sbz2        : 0x0
->Dacl       : ->Ace[0]: ->AceType: ACCESS_ALLOWED_ACE_TYPE
->Dacl       : ->Ace[0]: ->AceFlags: 0x0
->Dacl       : ->Ace[0]: ->AceSize: 0x28
->Dacl       : ->Ace[0]: ->Mask : 0x001f0001
->Dacl       : ->Ace[0]: ->SID: S-1-5-80-1913148863-3492339771-4165695881-2087618961-4109116736 (Well Known Group: NT SERVICE\WinDefend)
```

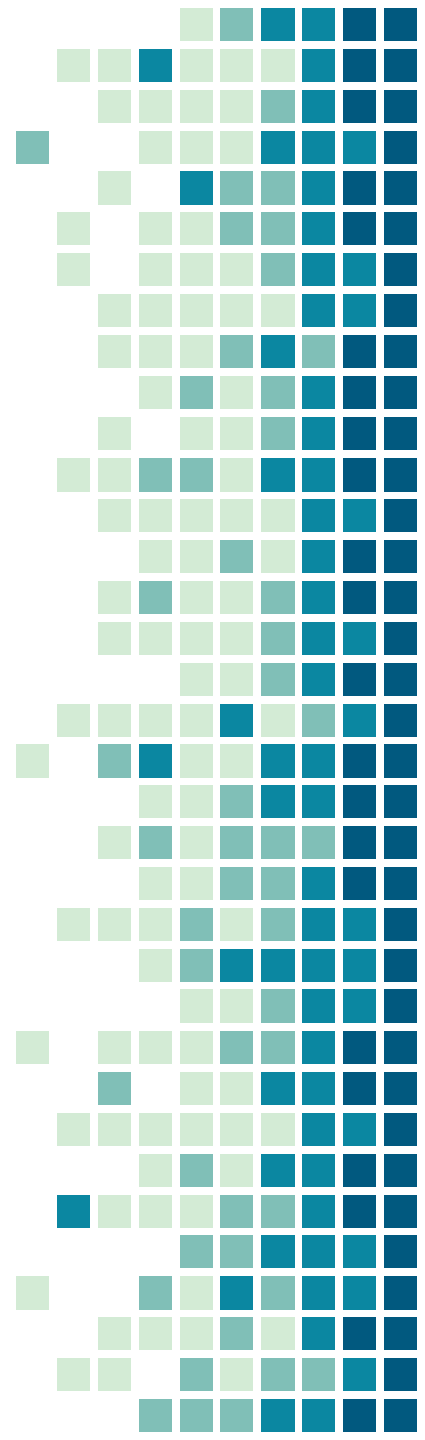
Communication Ports

MicrosoftMalwareProtectionControlPortWD	Receives requests from User Mode process – can return saved data or perform required action
MicrosoftMalwareProtectionPortWD	Notified by driver when file or boot sector is scanned, new process is created or image is loaded
MicrosoftMalwareProtectionVeryLowIoPortWD	Notified by driver when file is scanned
MicrosoftMalwareProtectionRemoteloPortWD	Notified by driver when file is scanned
MicrosoftMalwareProtectionAsyncPortWD	Used by driver to send async notifications to user mode process: process, object access, thread creation, file and volume operations...



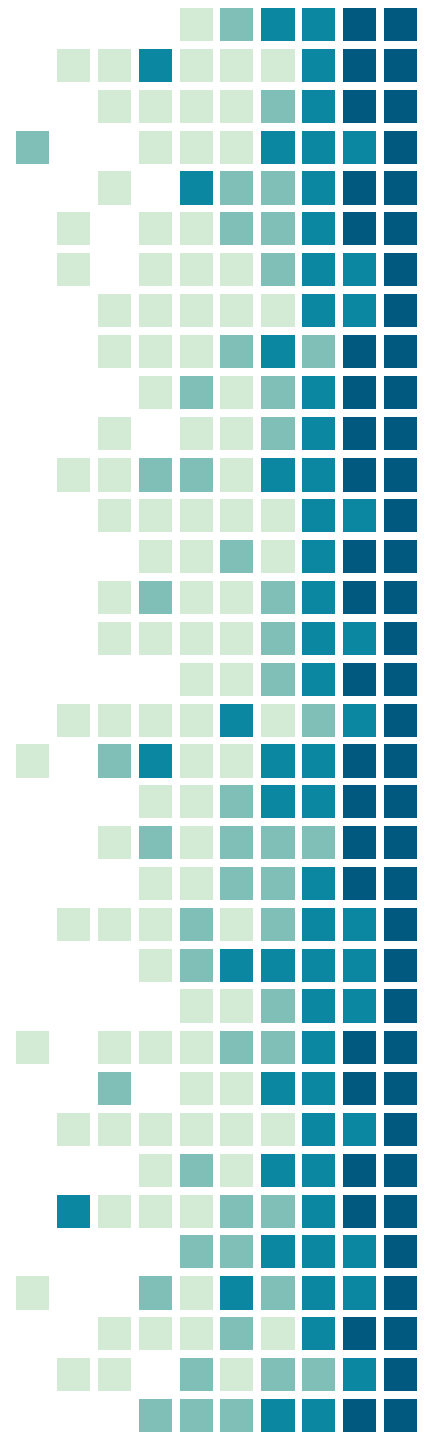
Control Port

- Can retrieve data from the driver or initiate actions
- Some examples:
 - Request scan of file or volume
 - Exclude file, folder or volume from scans
 - Write to boot sector
 - Query file stream data



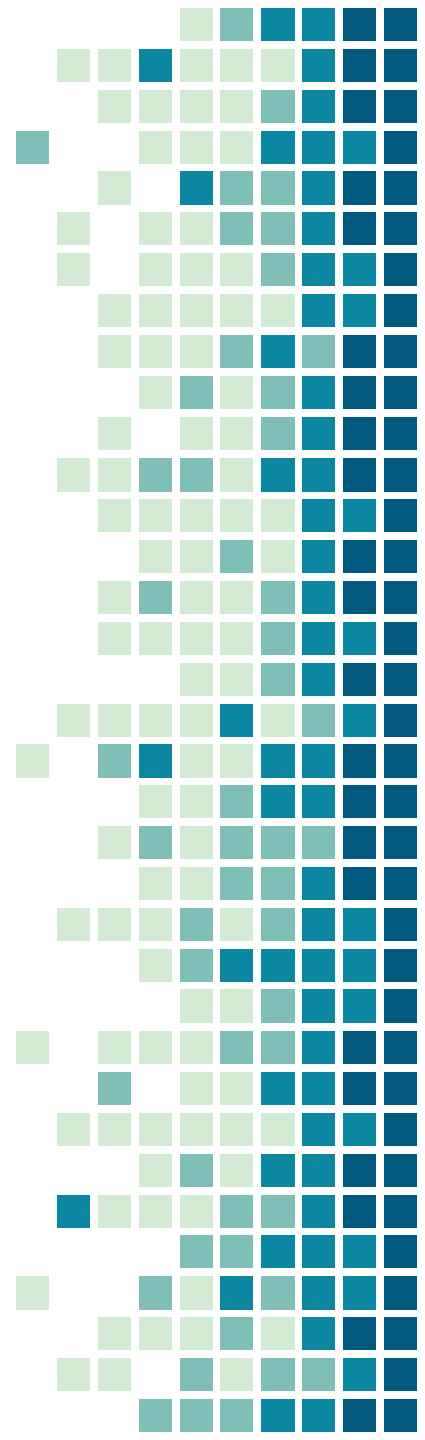
Async Port

- Received data from the driver about various events in the System:
 - Registry operations
 - Process or thread creation
 - Module load
 - Process or desktop open
 - File and volume operations
- Operations are filtered so not all data is sent in the port
 - Filtering done based on settings saved in global data structures – can be edited by a kernel driver

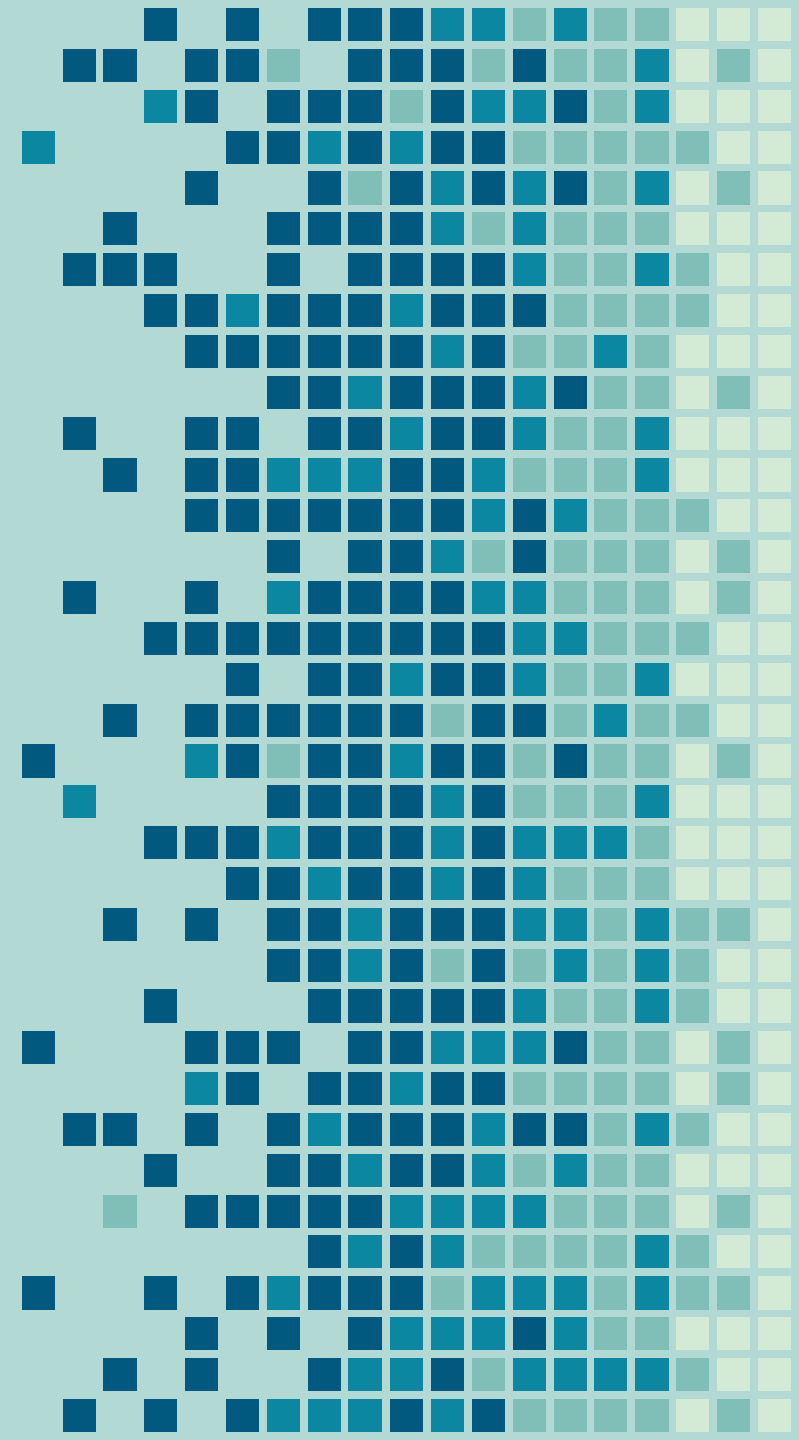


IOCTLs

- WdNisDrv.sys uses IOCTLs to communicate with user-mode process
 - Standard mechanism used by many drivers – driver exposes a device object for UM process to interact with
 - Device has a security descriptor only allowing system or WdNisSvc
- Rootkit driver can give any UM process the right token
 - Or change the security descriptor of the device
- WdNisDrv.sys IOCTLs allow processes to request network communication
 - Can specify target address and data to be delivered

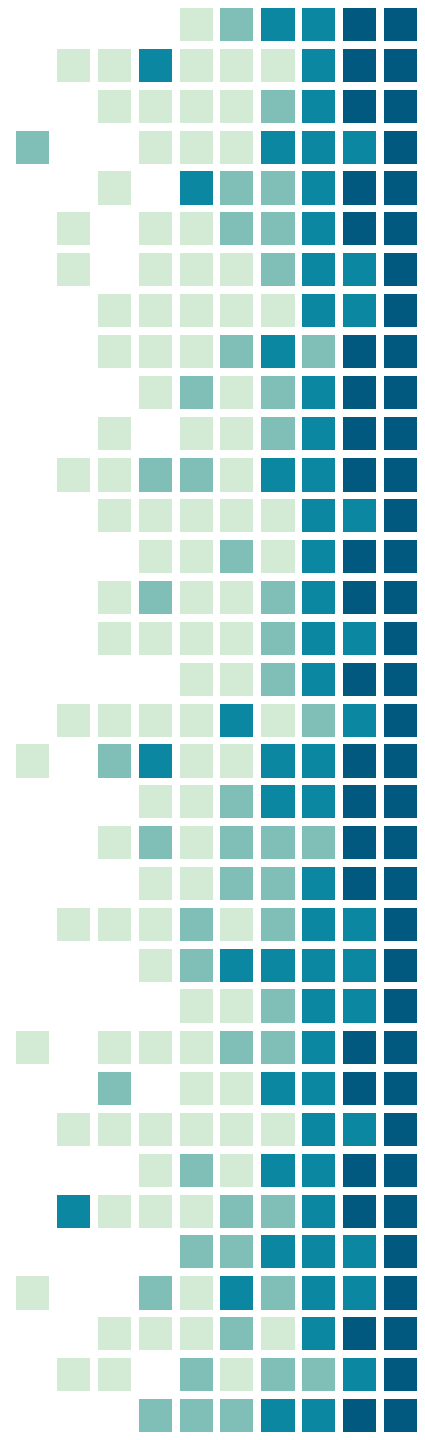


Protecting Yourself



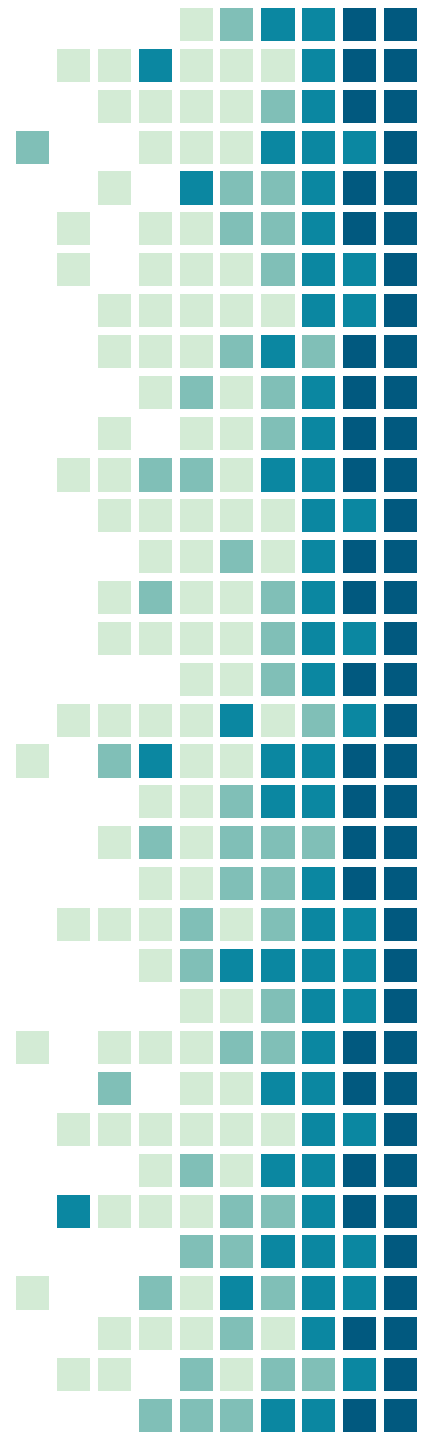
Protecting Your Process

- The system allows drivers to register OB callbacks
 - Will get notified when object of certain type is opened
 - Allows driver to restrict access to target object
 - Used by drivers to protect their processes or block malicious operations against OS processes
- OB callbacks are often monitored
 - Registering a callback can draw attention to a malicious driver



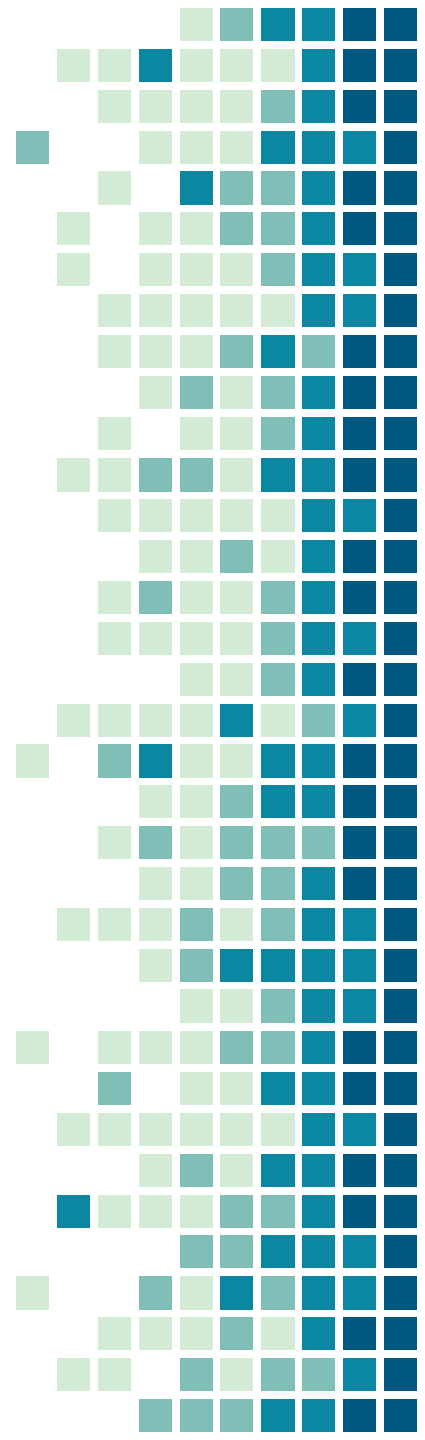
Protecting Your Process with Windows Defender

- WdFilter.sys registers an OB callback to restrict access to certain processes
 - Removes access bits `PROCESS_CREATE_THREAD`, `PROCESS_VM_OPERATION`, `PROCESS_VM_WRITE`
- Decision to restrict access done based on process flags
 - WD has a table of all running processes: `MpProcessTable`
 - Each process has a context created by the driver
 - Contains flags and hardening flags
 - If target process contains hardening flags, access to it is restricted
 - Unless source process has flags indicating that it should be exempt from restriction



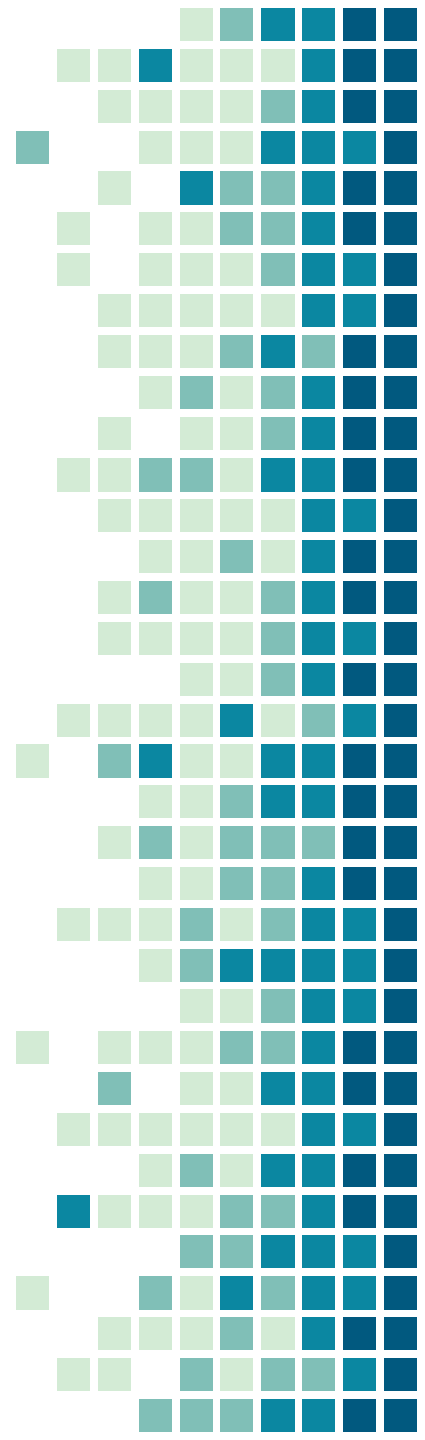
File System Protection

- WdFilter.sys is a filter driver so it can monitor and filter filesystem activity
 - This allows it to protect files from being accessed
- Windows Defender hard-codes protected paths
 - Any file in one of the Windows Defender directories
 - No way to dynamically protect different file paths
- Driver only allows read access to protected files
 - Unless requesting process has flags allowing it access to any file
 - These flags are set in the Defender process context

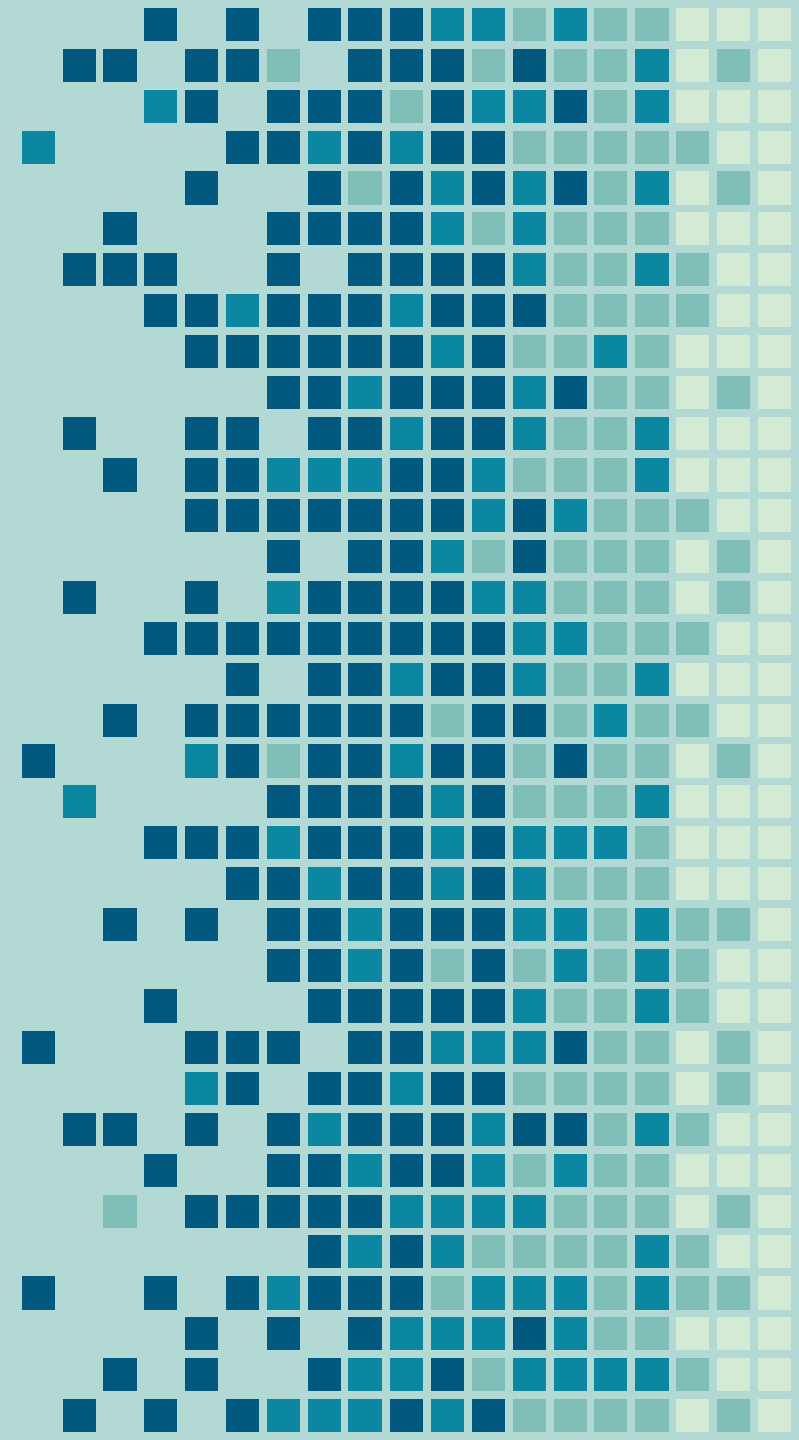


Registry Protection

- WdFilter.sys keeps a list of protected registry keys
 - Can be found in global variable MpRegData
 - If target registry key is in the list, driver will return STATUS_ACCESS_DENIED
 - Some processes are allowed access to all keys – same as with process and filesystem protections
 - Currently contains only keys for Defender services:
 - WdNisDrv, WdNisSvc, WdBoot, WinDefend, WdFilter, MpHardCodedBlockHive
 - New paths can be added easily by adding an entry



Implementation



So About Those Data Structures...

- MpData: largest global data structure in WdFilter.sys
 - Size in latest build: 0xEF0
 - Contains function pointers, Defender user-mode process identity, communication ports, Defender SIDs, callbacks...
 - Contains lots of lookaside lists that make it easy to find
 - Allocated in NonPaged pool with tag MPfd

```
lkd> !poolview -tag MPfd -nonpaged
```

Address	Size	(Status)	Tag	Type
0xfffffaf8c37aed060	0xef0	(Allocated)	MPfd	Vs

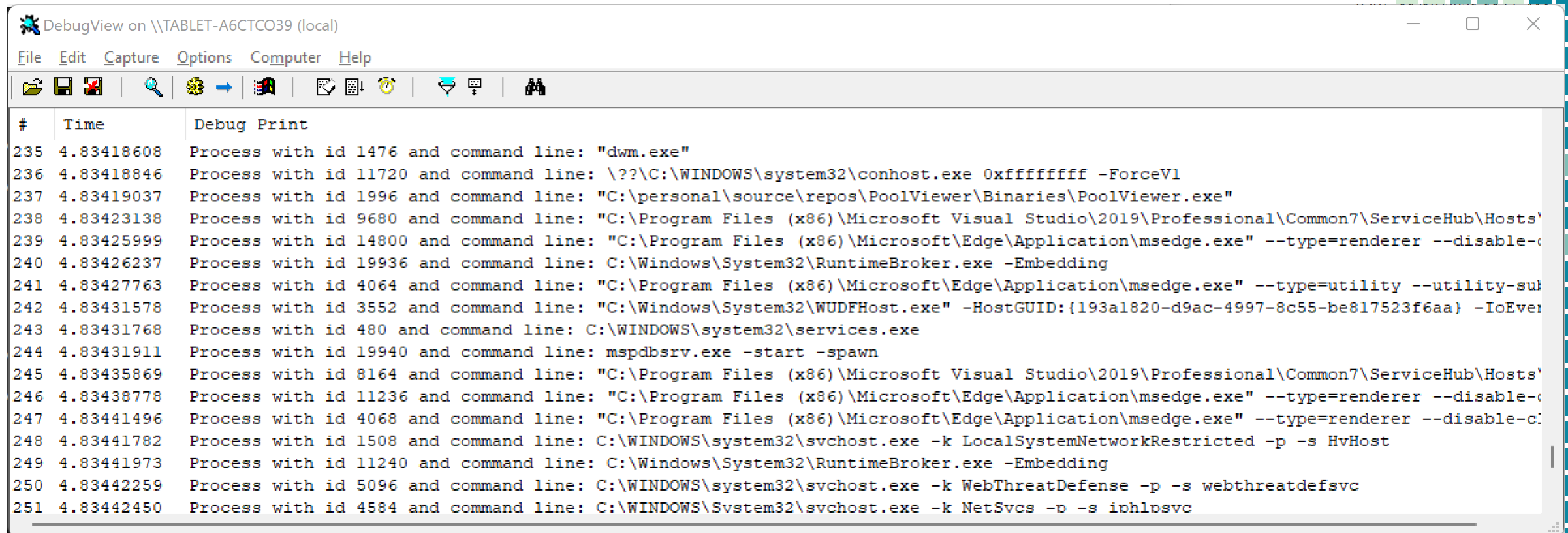
So About Those Data Structures...

- MpProcessTable: Table of all process contexts
 - Has a list linking all process context structures
 - Also has lookaside lists so structure is easy to find
 - Like the rest – all structures are private and can change
 - Process contexts have more data than what is sent in notifications

```
lkd> !poolview -tag MPpT -nonpaged
```

Address	Size	(Status)	Tag	Type
0xfffffaf8c24a090e0	0x1d0	(Allocated)	MPpT	Lfh

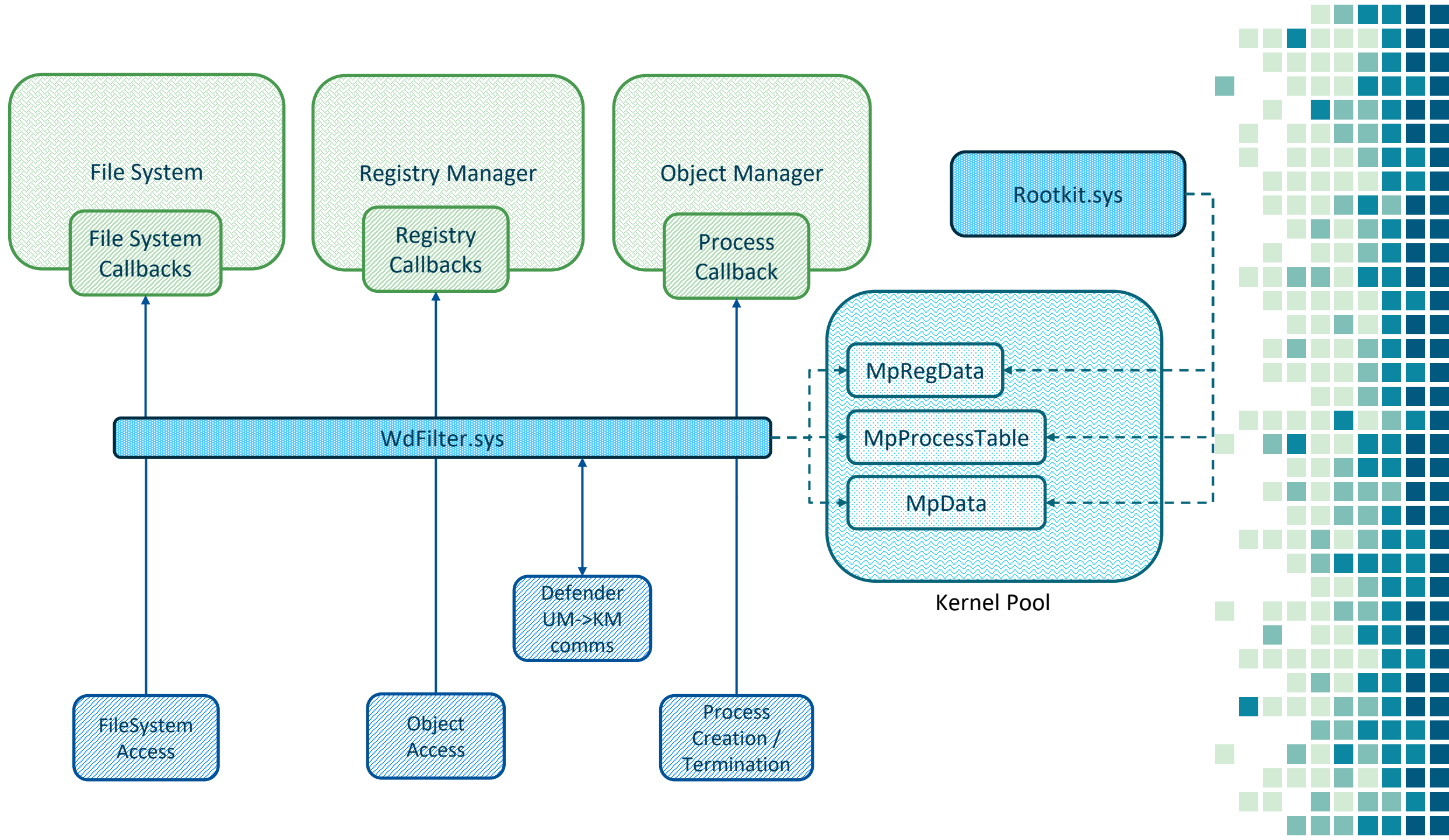
Getting Process Data from MpProcessTable



DebugView on \\TABLET-A6CTCO39 (local)

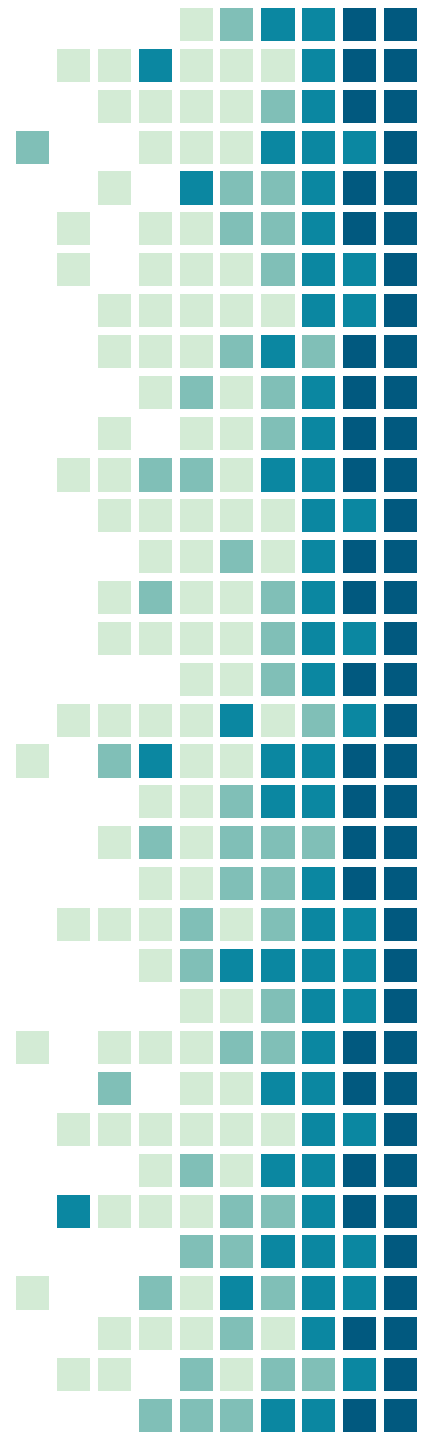
File Edit Capture Options Computer Help

#	Time	Debug Print
235	4.83418608	Process with id 1476 and command line: "dwm.exe"
236	4.83418846	Process with id 11720 and command line: \\??C:\WINDOWS\system32\conhost.exe 0xffffffff -ForceVl
237	4.83419037	Process with id 1996 and command line: "C:\personal\source\repos\PoolViewer\Binaries\PoolViewer.exe"
238	4.83423138	Process with id 9680 and command line: "C:\Program Files (x86)\Microsoft Visual Studio\2019\Professional\Common7\ServiceHub\Hosts\
239	4.83425999	Process with id 14800 and command line: "C:\Program Files (x86)\Microsoft\Edge\Application\msedge.exe" --type=renderer --disable-c
240	4.83426237	Process with id 19936 and command line: C:\Windows\System32\RuntimeBroker.exe -Embedding
241	4.83427763	Process with id 4064 and command line: "C:\Program Files (x86)\Microsoft\Edge\Application\msedge.exe" --type=utility --utility-sul
242	4.83431578	Process with id 3552 and command line: "C:\Windows\System32\WUDFHost.exe" -HostGUID:{193a1820-d9ac-4997-8c55-be817523f6aa} -IoEver
243	4.83431768	Process with id 480 and command line: C:\WINDOWS\system32\services.exe
244	4.83431911	Process with id 19940 and command line: mspdbsrv.exe -start -spawn
245	4.83435869	Process with id 8164 and command line: "C:\Program Files (x86)\Microsoft Visual Studio\2019\Professional\Common7\ServiceHub\Hosts\
246	4.83438778	Process with id 11236 and command line: "C:\Program Files (x86)\Microsoft\Edge\Application\msedge.exe" --type=renderer --disable-c
247	4.83441496	Process with id 4068 and command line: "C:\Program Files (x86)\Microsoft\Edge\Application\msedge.exe" --type=renderer --disable-c
248	4.83441782	Process with id 1508 and command line: C:\WINDOWS\system32\svchost.exe -k LocalSystemNetworkRestricted -p -s HvHost
249	4.83441973	Process with id 11240 and command line: C:\Windows\System32\RuntimeBroker.exe -Embedding
250	4.83442259	Process with id 5096 and command line: C:\WINDOWS\system32\svchost.exe -k WebThreatDefense -p -s webthreatdefsvc
251	4.83442450	Process with id 4584 and command line: C:\WINDOWS\System32\svchost.exe -k NetSvcs -p -s iphlpsvc



What am I Getting From All This Work?

- No registered callbacks or filter driver
 - Easily identified and flagged as suspicious
 - Will leave no forensic traces visible to any tool
- Driver only interacts with anonymous pool data
 - Minimizes API calls and side effects
 - All actions are done by Windows Defender – and suspicious actions from an EDR tend to get ignored or excluded
- Hard to debug and analyze
 - Requires familiarity with Defender mechanisms and data structures



Questions?

Or comments, suggestions, ideas, jokes...

