Webcam Interception and Protection in Kernel Mode in Windows

Michael Maltsev
VB2019 London
October 3, 2019
About Us

The Importance of Webcam Security
Multimedia Frameworks
Attack Strategies
Protection Driver Development
Existing Protection Solutions
• Founded in 2012 by Andrew Newman, a leading security expert.

• Focused on privacy protection.

• Products:
  • Reason Antivirus (formerly known as Reason Core Security)
  • Should I Remove It?
  • Unchecky

• The Reason Antivirus engine scans over 1B files in 180 countries a day.
Outline

• The importance of webcam security
• Multimedia frameworks
  • DShow Bridge
  • Frame Server
• Attack strategies
• Protection driver development
• Existing protection solutions
The Importance of Webcam Security
The Importance of Webcam Security

Global Surveillance
NSA, Hacking Team, etc.

Stalkerware
e.g. FlexiSPY

Webcam Ransom
The Importance of Webcam Security
“Since being founded in 2003, the Italian spyware vendor Hacking Team gained notoriety for selling surveillance tools to governments and their agencies across the world. [...] The capabilities of its flagship product, the Remote Control System (RCS), include [...] remotely activating a device’s webcam and microphone.”

Filip Kafka, WeLiveSecurity (ESET)

“Mark Zuckerberg masks Mac webcam and microphone [...] FBI director James Comey has previously said he also covers his laptop's webcam to prevent hackers spying on him.”

Kim Zetter, WIRED

“According to The Intercept, the NSA uses a plug-in called GUMFISH to take over cameras on infected machines and snap photos.”

BBC News
Multimedia Frameworks
Multimedia Frameworks

- VfW: 1992
  - Obsolete
- DirectShow: 1996
- Media Foundation: 2006
Multimedia Frameworks

**VfW (Video for Windows)**

- A multimedia framework that enables applications to process video data.
- Replaced by DirectShow in 1996.
- Nowadays implemented by the VFWWDM32 driver as a backward compatibility layer.
Direct Show

- An architecture for streaming media based on the Component Object Model (COM).
- Announced in 1996, originally named ActiveMovie.
- Became a standard component of all Windows operating systems starting with Windows 98.
- Probably the most popular API for interacting with the camera on Windows today.

WebcamCapture.exe
ksproxy.ax
WDM Streaming
ActiveMovie Proxy
ks.sys
Kernel CSA (Connection and Streaming Architecture) Library

User mode
Kernel mode
Multimedia Frameworks

Media Foundation

• A multimedia platform based on the Component Object Model (COM), intended to replace DirectShow (but apparently is not there yet).

• Introduced in Windows Vista (2006), enhanced in Windows 7 and further enhanced in Windows 8.

• “Media Foundation has a couple features that DirectShow did not. The playback experience should be better (fewer glitches), and it also has much more robust support for content protection systems.”
  Becky Weiss, Microsoft
Multimedia Frameworks

Media Foundation

WebcamCapture.exe
mfcore.dll
Media Foundation Core DLL
ks.sys
Kernel CSA (Connection and Streaming Architecture) Library

User mode
Kernel mode
**DShow Bridge**

- A proxy to allow applications designed for DirectShow to use Media Foundation.
- Introduced in Windows 10 Anniversary Update (Version 1607, August 2016).
- Implemented in mfksproxy.dll.
DShow Bridge

Enabled if:

• DShow Bridge is enabled for the camera device.
  • HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Enum\USB\<DeviceVID&PID>\<DeviceInstance>\Device Parameters
  • Enabled if the EnableDshowRedirection DWORD value has the 0x00000001 bit set.

• Capturing application is not blacklisted from using DShow Bridge:
  • HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\OEM\DshowBridge\<number>
  • HKEY_LOCAL_MACHINE\SOFTWARE\WOW6432Node\Microsoft\OEM\DshowBridge\<number>
  • There’s an existing entry for “Launcher_Main.exe” from Logitech in the above branches.
Frame Server

- Introduced in Windows 10 Anniversary Update together with DShow Bridge.

- A service virtualizing a camera device, allowing the device to be shared between multiple applications.

- In practice, camera device sharing is reserved for the system.

“It was important for us to enable concurrent camera access, so Windows Hello, Microsoft Hololens and other products and features could reliably assume that the camera would be available at any given time”

Mike M, Windows Camera Team
Frame Server

Enabled if:

• Application is using Media Foundation (or DirectShow with DShow Bridge enabled).
• Frame Server is not disabled in the registry.
  • The EnableFrameServerMode DWORD value in the following registry branch:
    HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows Media Foundation\Platform
  • Enabled by default if the value is absent.
• The application path doesn’t contain “RealSenseDCM”.

Attack Strategies
Attack Strategies

Simple
Just record

Stealth
Use an existing session

God Mode
Just record, LED off
Attack Strategies (1/3)

Simple

• Just start recording.

• Advantages:
  • Just works, no questions are asked by Windows.
    Note: Since Windows 10 Fall Creators Update, the user is prompted for access to the camera device, but it only applies to Store apps.
  • Many code examples and available programs that can be used.

• Disadvantages:
  • Camera usage will turn on the indicator LED, giving the victim an indication that somebody is watching him.
Attack Strategies (2/3)

**Stealth**

- Use an existing recording session. For example, inject into Skype and intercept the camera traffic.
- **Advantages:**
  - No unusual indicator LED activity, more difficult for the victim to detect.
- **Disadvantages:**
  - Not as easy to implement.
  - The attacker can’t choose when to record. The recording sessions are chosen by the victim, and might be less valuable.
  - Can be intrusive depending on the implementation, increases the chances of being detected by a security software.
Attack Strategies (3/3)

**God Mode**

- Like the first option, but keep the indicator LED turned off.
- Ideally, it shouldn’t be possible to use the camera without turning on the indicator LED. In practice, some models provide this option, or have a design flaw that allows it.
- Examples:
  - “I’ve found info on Logitech Webcams, where you can turn off the LED in the registry keys.”
    psalomonsen, Information Security Stack Exchange
  - “[…] on our Dell laptop, we find the DLL that comes with the RealTek drivers for our webcam. We quickly zero in on the exported function "TurnOnOffLED()".”
    Robert Graham, Errata Security: How to disable webcam light on Windows
God Mode

- Advantages:
  - The attacker can record at any time without a visible indication.

- Disadvantages:
  - Device-specific, hard or impossible to do on well-designed devices.
Protection Driver Development
Protection Driver Development

- Mostly undocumented, not much information on the internet.
- After some research we can come to the conclusion that the way to go is to implement a filter driver.
- Let’s implement a filter driver which logs the IRPs (I/O request packets) that pass through.
Protection Driver Development

WebcamCapture.exe
Via the relevant system DLL

User mode

filter.sys
Camera interception filter driver

Kernel mode

ks.sys
Kernel CSA (Connection and Streaming Architecture) Library
Protection Driver Development

The Interesting IRPs

Start streaming

Read data from stream

major: IRP_MJ_CREATE filename: 5C 00 67 00 6C 00 6F 00 62 00 61 00 6C 00...

major: IRP_MJ_CREATE filename: 7B 00 31 00 34 00 36 00 31 00 [...]

major: IRP_MJ_DEVICE_CONTROL ioctl: IOCTL_KS_PROPERTY request: Connection
KSPROPERTY_CONNECTION_STATE set type: KSSTATE_ACQUIRE

major: IRP_MJ_DEVICE_CONTROL ioctl: IOCTL_KS_PROPERTY request: Connection
KSPROPERTY_CONNECTION_STATE set type: KSSTATE_PAUSE

major: IRP_MJ_DEVICE_CONTROL ioctl: IOCTL_KS_PROPERTY request: Connection
KSPROPERTY_CONNECTION_STATE set type: KSSTATE_RUN

major: IRP_MJ_DEVICE_CONTROL ioctl: IOCTL_KS_READ_STREAM

major: IRP_MJ_DEVICE_CONTROL ioctl: IOCTL_KS_READ_STREAM...

major: IRP_MJ_DEVICE_CONTROL ioctl: IOCTL_KS_PROPERTY request: Connection
KSPROPERTY_CONNECTION_STATE set type: KSSTATE_STOP

major: IRP_MJ_CLEANUP

major: IRP_MJ_CLEANUP

major: IRP_MJ_CLOSE

major: IRP_MJ_CLOSE

“\global” The KS filter

Next slide... The KS pin
Protection Driver Development

The IRP_MJ_CREATE for the KS Pin

major: IRP_MJ_CREATE filename: 7B 00 31 00 34 00 36 00 46 00 31 00 [...] 00 00 00 00

{146F1A80-4791-11D0-A5D6-28DB04C10000}\???

The object name is a GUID followed by binary data

{146F1A80-4791-11D0-A5D6-28DB04C10000} stands for KSNAME_Pin
The **KSPIN_CONNECT** structure describes the connection details, such as the pin ID.

The **KSDATAFORMAT** structure describes the image format.
Protection Driver Development

Blocking Access to the Camera

- The simplest solution is to block `IOCTL_KS_PROPERTY` when the `KSSTATE_ACQUIRE` command is sent.
- Note: The kernel streaming IOCTLs use the 'neither' buffering method. Data must be accessed from the context of the calling process.
- The capturing program will get an error while trying to initiate the capture.

Something went wrong

If you need it, here's the error code: 0xA0F4271(0x80070005)
What if we want to return a fake image instead?
Protection Driver Development

**Replacing the Image Frames**

- We can post-process `IOCTL_KS_READ_STREAM` and replace the frame image before it leaves the kernel mode.

- The frame image buffer is mapped for us by the camera function driver, and can be accessed via `Irп->MdlAddress`.

- `Irп->MdlAddress` can be `NULL` sometimes. The reason - the `ksthunk` filter driver, designed to provide streaming compatibility for 32-bit programs on a 64-bit system.
Protection Driver Development

Replacing the Image Frames

WebcamCapture.exe
Via the relevant system DLL

User mode

Kernel mode

filter.sys
MdlAddress is NULL

ksthunk.sys
Kernel Streaming WOW Thunk Service

ks.sys
Kernel CSA (Connection and Streaming Architecture) Library

MdlAddress is NULL

MdlAddress points to the frame
Protection Driver Development

Replacing the Image Frames

• The solution - place our filter driver before *ksthunk* on the stack.
Protection Driver Development

Replacing the Image Frames

- WebcamCapture.exe
  Via the relevant system DLL

- ksthunk.sys
  Kernel Streaming WOW Thunk Service

- filter.sys
  MdIAddress is valid

- ks.sys
  Kernel CSA (Connection and Streaming Architecture) Library

MdIAddress points to the frame
Typical design - maintain a list of allowed programs such as Skype and the web browser while blocking other, unknown programs.

Used to be straightforward before the introduction of the Frame Server - The process accessing the camera device is the one to be checked against the list.

Since the introduction of the Frame Server, `svchost.exe` (hosting the Frame Server) will be accessing the camera device every time Media Framework is used.
Protection Driver Development

Per-process Selective Blocking

• We need an alternative way to determine which process is accessing the camera.

• A possible solution - intercept access to \texttt{mskssrv.sys}, the Kernel Streaming Server.
Things to note:

- If intercepting access to `mskssrv.sys`, the process hosting Frame Server needs to be whitelisted.

- You need to be careful about trusting an allowed program, e.g. in case of a code injection scenario.
We checked 6 well known security products with camera protection functionality.

<table>
<thead>
<tr>
<th>Product</th>
<th>Frame Server support</th>
<th>Upper filter registered (*)</th>
<th>Default action on first run</th>
</tr>
</thead>
<tbody>
<tr>
<td>Product 1</td>
<td>✓</td>
<td>✓</td>
<td>Block and notify</td>
</tr>
<tr>
<td>Product 2</td>
<td>✗</td>
<td>✗</td>
<td>Block and notify if untrusted</td>
</tr>
<tr>
<td>Product 3</td>
<td>✓</td>
<td>Only for the Frame Server</td>
<td>Always notify, block if untrusted</td>
</tr>
<tr>
<td>Product 4</td>
<td>✓</td>
<td>✗</td>
<td>Block and notify if untrusted</td>
</tr>
<tr>
<td>Product 5</td>
<td>✗</td>
<td>✓</td>
<td>Block and notify if untrusted</td>
</tr>
<tr>
<td>Product 6</td>
<td>✗</td>
<td>✗</td>
<td>Notify, no blocking functionality</td>
</tr>
</tbody>
</table>

(*) If not, an interception device is being attached to the camera driver stack at runtime.
After some experimentation we can see that Frame Server is supported by intercepting and blocking access to `mskssrv.sys`, the Kernel Streaming Server.

In that case, the Frame Server user mode process (`svchost.exe`, hosting the Frame Server) is probably whitelisted.

What if we inject into it and try to capture?

<table>
<thead>
<tr>
<th>Product 1</th>
<th>Frame Server support</th>
<th>Upper filter registered</th>
<th>Default action on first run</th>
</tr>
</thead>
<tbody>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>Block and notify</td>
</tr>
</tbody>
</table>
Existing Protection Solutions

Product 1, Bypass Attempt 1

- WebcamCapture.exe
- svchost.exe
- fsclient.dll
- mfc core.dll
- mskssrv.sys
- ks.sys

Intercept and block
Allow for the Frame Server

User mode
Kernel mode
WebcamCapture.exe
mfcore.dll
Media Foundation Core DLL
User mode
Kernel mode
mskssrv.sys
Microsoft Kernel Streaming Server
fsclient.dll
Frame Server Client DLL
svchost.exe
Windows Camera Frame Server

Intercept and block
Allow for the Frame Server
ks.sys
Kernel CSA (Connection and Streaming Architecture) Library
Existing Protection Solutions

**Product 1, Bypass Attempt 1**

- WebcamCapture.dll
- svchost.exe
  - Windows Camera Frame Server
- mfc.exe
  - Media Foundation Core DLL
- ks.sys
  - Kernel CSA (Connection and Streaming Architecture) Library

**User mode**

**Kernel mode**
Existing Protection Solutions

Product 1, Bypass Attempt 1

https://youtube.com/watch?v=-bnpcIzlXsA
Existing Protection Solutions

Product 1, Bypass Attempt 1

- WebcamCapture.dll
- mfcore.dll (Media Foundation Core DLL)
- svchost.exe (Windows Camera Frame Server)
- ks.sys (Kernel CSA (Connection and Streaming Architecture) Library)

User mode: DLL injection
Kernel mode:

Allow for the Frame Server

BYPASSED
Existing Protection Solutions

Product 1, Bypass Attempt 2
Now let’s try something simpler: What happens if we try to remove the registered upper filter driver from the registry?

Surprisingly, nothing stops us from doing it.

Now it’s enough to disable and re-enable the camera device to get the security filter driver removed from the driver stack.

<table>
<thead>
<tr>
<th>Existence Protection Solutions</th>
<th>Product 1, Bypass Attempt 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frame Server support</td>
<td>✓</td>
</tr>
<tr>
<td>Upper filter registered</td>
<td>✓</td>
</tr>
</tbody>
</table>
Existing Protection Solutions

Product 1, Bypass Attempt 2

https://youtube.com/watch?v=JKzoqGk3vTk
Existing Protection Solutions

Product 2
No Frame Server support - every program accessing the camera via the Frame Server is being reported as `svchost.exe`.

`svchost.exe`, being a system process, is trusted by default.

It’s enough to have our capturing program use Media Foundation to bypass the protection.

<table>
<thead>
<tr>
<th>Product 2</th>
<th>Frame Server support</th>
<th>Upper filter registered</th>
<th>Default action on first run</th>
</tr>
</thead>
<tbody>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>Block and notify if untrusted</td>
</tr>
</tbody>
</table>
Existing Protection Solutions

Product 2

https://youtube.com/watch?v=Z-64HGY8HtM
No Frame Server support - every program accessing the camera via the Frame Server is being reported as `svchost.exe`.

`svchost.exe`, being a system process, is trusted by default.

It’s enough to have our capturing program use Media Foundation to bypass the protection.

<table>
<thead>
<tr>
<th>Product 2</th>
<th>Frame Server support</th>
<th>Upper filter registered</th>
<th>Default action on first run</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>X</td>
<td>Block and notify if untrusted</td>
<td></td>
</tr>
</tbody>
</table>
### Existing Protection Solutions

#### Summary

<table>
<thead>
<tr>
<th></th>
<th>Frame Server injection</th>
<th>Media Foundation usage</th>
<th>Upper filter removal</th>
</tr>
</thead>
<tbody>
<tr>
<td>Product 1</td>
<td>Bypassed</td>
<td></td>
<td>Bypassed</td>
</tr>
<tr>
<td>Product 2</td>
<td></td>
<td>Bypassed</td>
<td></td>
</tr>
<tr>
<td>Product 3</td>
<td>Bypassed</td>
<td></td>
<td>Bypassed</td>
</tr>
<tr>
<td>Product 4</td>
<td>Bypassed (*)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Product 5</td>
<td></td>
<td>Bypassed</td>
<td>Bypassed</td>
</tr>
<tr>
<td>Product 6</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(*) Frame Server injection was blocked, but injecting into any other trusted program works.
Lessons

- If you trust a system service, make sure it can’t be tampered with.
- Don’t forget to apply self-protection for features that can be otherwise just disabled.
- Follow and adapt to the OS architecture changes - in our case that’s the introduction of the Frame Server.
Conclusions

A significant privacy threat

Webcam-related Windows internals

Software protection is not perfect
Sharing is Caring

- A technical paper on the subject: [Through the looking glass: webcam interception and protection in kernel mode](https://github.com/ReasonSoftware/webcam-interception-driver)
- A sample driver for blocking access to the camera: [https://github.com/ReasonSoftware/webcam-interception-driver](https://github.com/ReasonSoftware/webcam-interception-driver)
- Contact me for any additional information.  
  - michael.maltsev@reasonsecurity.com  
  - @m417z
Thank You

Questions?