black hat USA 2022

The Journey Of Hunting In-The-Wild Windows LPE 0day

Quan Jin

DBAPPSecurity

Who am I

- Quan Jin (@jq0904)
  - Security Researcher at DBAPPSecurity
  - Member of
    - *DBAPPSecurity Lieying Lab*
    - *DBAPPSecurity WeBin Lab*
  - Interested in
    - *Vulnerability discovery and exploiting*
    - *In-the-wild 0day hunting*
  - Presented at
    - *Bluehat Shanghai 2019*
    - *HITB2021AMS*
  - 37 CVE acknowledgments from Microsoft
  - 2020~2022 MSRC Most Valuable Researcher

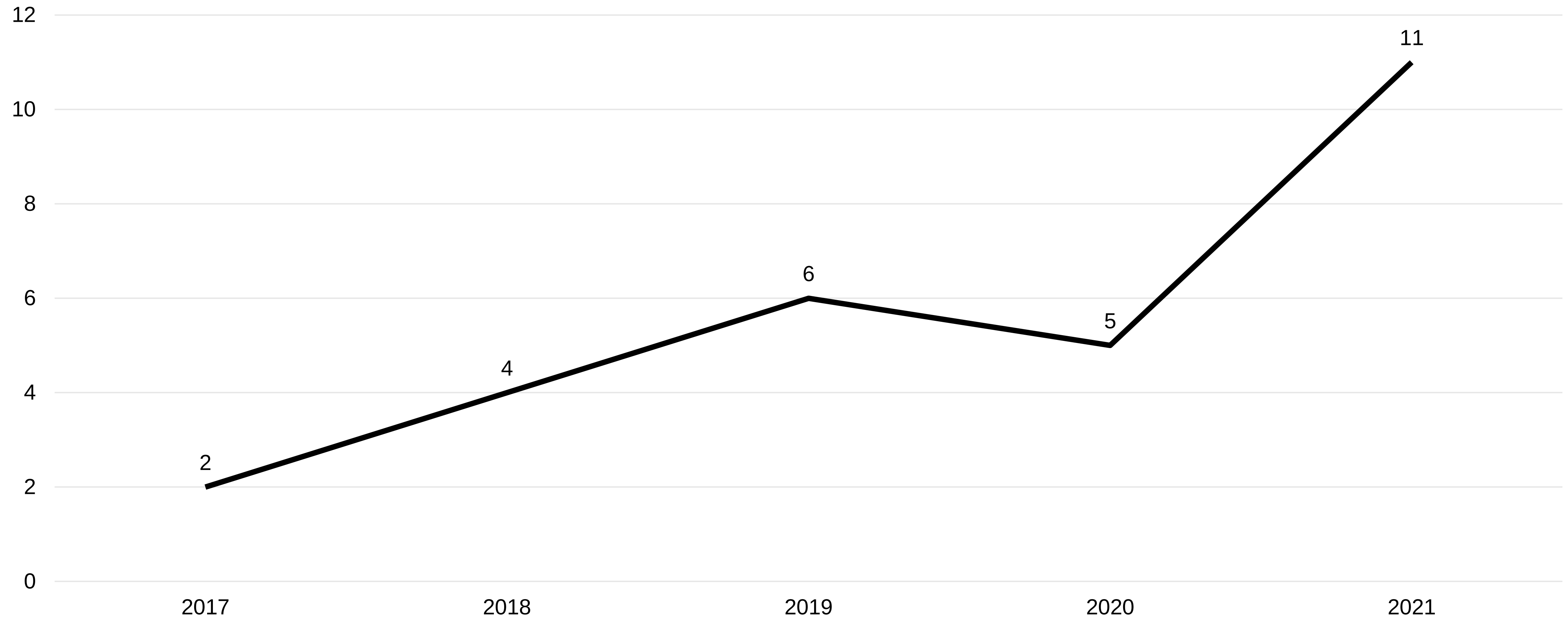
Agenda

- Motivation
- Learn from history (and now)
- One road leads to Rome
- Results
  - The Story of CVE-2021-1732
  - The Story of CVE-2021-33739
  - The Story of a “Patched” 1day
- Takeaways

Agenda

- **Motivation**
- Learn from history (and now)
- One road leads to Rome
- Results
  - The Story of CVE-2021-1732
  - The Story of CVE-2021-33739
  - The Story of a “Patched” 1day
- Takeaways

Motivation

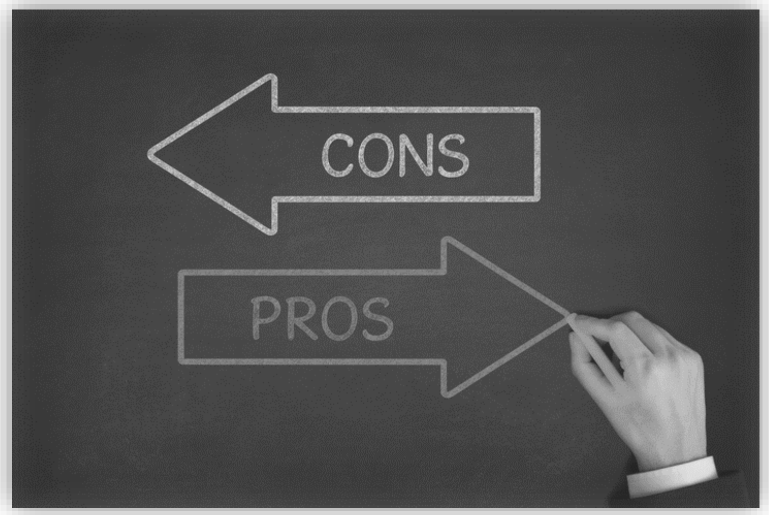


In-The-Wild Windows LPE 0day (2017-2021)

Is it possible to catch an itw LPE 0day?

- **How to obtain valueable datasets (that potentially have 0days)**
  - ✓ Private datasets (Data from our own products)
  - ✓ Public platform datasets (Such as VirusTotal)

- **How to develop an effective detection method**
  - ✓ Dynamic detection (picking out a sample through abnormal behavior)
    - Antivirus, Sandbox, …
  - ✓ Static detection (matching samples with static signatures)
    - YARA, …



Agenda

- Motivation
- **Learn from history (and now)**
- One road leads to Rome
- Results
  - The Story of CVE-2021-1732
  - The Story of CVE-2021-33739
  - The Story of a “Patched” 1day
- Takeaways

Why should we learn from history

1. Some exploit techniques are consistent over time

2. Thinking from the attacker's view allows for better defense

3. Historical cases have been carefully studied by the community

How we study historical cases

- *Discovered vendor*
- *Using organization*
- *Patch cycle*
- *Initial disclosure article*
- *Usage scenario*
- *Targeted system versions*

- *Vulnerability module*
- *Vulnerability type*
- *Exploit techniques*
- *Public analysis blogs*
- *Public exploits*
- *The original sample (if have)*

Usage scenario

- **Whether the sample was used as a standalone component, or as part of a chain**
  - CVE-2021-1732 (Standalone component)
  - CVE-2021-31956 (In conjunction with Chrome vulnerability)

- **Whether the exploit was used in a fileless form, or was just contained in a drop file**
  - CVE-2017-0263 (Dll reflection)
  - CVE-2019-0803 (Contained in a single file)

- ***Affect the selection of different detection methods***

Targeted system versions

- **Many Windows LPE samples will check OS version before exploit**
  - CVE-2018-8611 itw exploit (***Windows 7 ~ Windows 10 1803***)
  - CVE-2019-0797 itw exploit (***Windows 8 ~ Windows 10 1703***)
  - CVE-2021-40449 itw exploit (***Windows Vista ~ Windows 10 1809***)

- ***Useful when making a sandbox or reproduction environment***
  - Which Windows 10 version is best as a sandbox/reproduction environment?
  - Is it necessary to maintain an oldest and newest Windows environment in the long term?

Vulnerability module

- **Which module is most targeted**
  - WIN32K
  - ATMFD
  - NT

- **Which component is most favored by attackers during a specific period of time**
  - Desktop Window Manager (DWM)
  - Common Log File System (CLFS)

- ***Useful when predicting the most likely vulnerability***

Vulnerability type

- **Infer which type of vulnerability is the attacker more favored**
  - ***Integer Overflow***: CVE-2020-17087, CVE-2021-31956, CVE-2021-31979
  - ***Type Confusion***: CVE-2021-1732, CVE-2022-21882
  - ***Race Condition***: CVE-2018-8589, CVE-2018-8611, CVE-2019-0797
  - ***Use After Free***: CVE-2018-8453, CVE-2019-0859, CVE-2021-33771
  - …
- ***Help us config the right reproduction environment***
  - Whether need to config Driver Verifier
- ***Show the popularity of different vulnerability types***

Exploit techniques

- **We count the exploit techniques for most itw Windows LPE 0days (2014-2021)**
  - "***bServerSideWindowProc***" method was popular from 2015 to 2016
    - *CVE-2015-1701, CVE-2015-2360, CVE-2015-2546, CVE-2016-0167*
  - The method of using "***Previous Mode***" to achieve arbitrary address read and write has become more and more popular since 2018
    - *CVE-2018-8611, CVE-2021-28310, CVE-2021-31956, CVE-2021-31979, CVE-2021-33771*
  - The method of using "***HMValidateHandle***" to leak kernel information is popular in the past five years
    - *CVE-2017-0263, CVE-2018-8453, CVE-2019-0859, CVE-2019-1132, CVE-2021-1732*

Public analysis blogs & exploits

- **Standing on the shoulders of giants**
  - "*Hunting for exploits by looking for the author's fingerprints*" by Check Point
  - "*The Story of PlayBit*" by Check Point
  - "*Overview of the latest Windows OS kernel exploits found in the wild*" by Kaspersky
  - "*Retrospective on the latest zero-days found in the wild*" by Kaspersky

The original sample (if have)

- **The first-hand information**
  - Files
  - Hashes
  - Behaviors
  - Exploit techniques

- ***Help us detect similar samples in the future***
  - Some exploit techniques are consistent over time

Why should we learn from now

1. A new disclosed vulnerability may have variants
   - *CVE-2022-21882 is a variant of CVE-2021-1732*
2. A new targeted module will be fuzzed and audited by community
   - *CLFS is heavily fuzzed and audited in the past two years*
3. An attacker may have some similar vulnerabilities in use or wait to use
   - *Kaspersky discovered CVE-2021-28310 based on CVE-2021-1732*
4. A new exploit technique tends to be used by attackers soon
   - *"Pipe Attribute" in "Scoop the Windows 10 pool!" is popular after 2020*

Why should we learn from now

1. A new disclosed vulnerability may have variants
   - *CVE-2022-21882 is a variant of CVE-2021-1732*

2. A new targeted module will be fuzzed and audited by community
   - *CLFS is heavily fuzzed and audited in the past two years*

3. An attacker may have some similar vulnerabilities in use or wait to use
   - *Kaspersky discovered CVE-2021-28310 based on CVE-2021-1732*

4. A new exploit technique tends to be used by attackers soon
   - *"Pipe Attribute" in "Scoop the Windows 10 pool!" is popular after 2020*

Why should we learn from now

1. A new disclosed vulnerability may have variants
   - *CVE-2022-21882 is a variant of CVE-2021-1732*

2. A new targeted module will be fuzzed and audited by community
   - *CLFS is heavily fuzzed and audited in the past two years*

3. An attacker may have some similar vulnerabilities in use or wait to use
   - *Kaspersky discovered CVE-2021-28310 based on CVE-2021-1732*

4. A new exploit technique tends to be used by attackers soon
   - *"Pipe Attribute" in "Scoop the Windows 10 pool!" is popular after 2020*

Why should we learn from now

1. A new disclosed vulnerability may have variants
   - *CVE-2022-21882 is a variant of CVE-2021-1732*
2. A new targeted module will be fuzzed and audited by community
   - *CLFS is heavily fuzzed and audited in the past two years*
3. An attacker may have some similar vulnerabilities in use or wait to use
   - *Kaspersky discovered CVE-2021-28310 based on CVE-2021-1732*
4. A new exploit technique tends to be used by attackers soon
   - *"Pipe Attribute" in "Scoop the Windows 10 pool!" is popular after 2020*

Agenda

- Motivation
- Learn from history (and now)
- **One road leads to Rome**
- Results
  - The Story of CVE-2021-1732
  - The Story of CVE-2021-33739
  - The Story of a “Patched” 1day
- Takeaways

Choose the right tool

Antivirus **Sandbox** **YARA**

Antivirus

- **The most powerful tool**
  - Kaspersky have caught some itw Windows LPE 0days with their antivirus in the past few years
- **Pros**
  - Deployed in large-scale ***real-world environments***
  - Opportunity to extract encrypted LPE components
- **Cons**
  - There are ***strong technical barriers***
  - It is likely to be bypassed or detected

Sandbox

- **Another power tool**
  - There are some successful experience on itw Office 0day hunting with the help of sandbox

- **Pros**
  - The environment is ***highly controllable*** and can be freely configured
  - Behavior-based detection makes it accuracy

- **Cons**
  - Easy to miss a Windows LPE sample (OS version check)
  - ***Huge resource overhead*** (More environments means more time and money)

Sandbox

- **Other disadvantages of sandbox on Windows LPE hunting**
  - Some samples require parameters but the sandbox ***cannot provide valid parameters*** by default
    - *For example, a process id*
  - Some samples only lead to ***BSOD without subsequent behavior***, which is difficult to detect
    - *Such as a proof of concept (poc), or a sample running in a wrong system version*
  - There is ***a cycle between sandbox development and deployment***
    - *Which will lead to missing the best detection cycle for some latest exploits*

YARA

- **Powerful static method**
  - Widely used for malware hunting
- **Pros**
  - Almost ***no technical barriers***
  - ***No fear of various checks***
  - ***Flexible*** in development and deployment
  - ***Low cost***
- **Cons**
  - It can easily lead to ***false positives*** and ***false negatives***

Choose the right tool

Antivirus

Sandbox

YARA

Build the right rule

1. Write rules according to the signatures of each stage of exploitation

2. Write rules for latest exploit techniques

3. Write rules for the most likey vulnerability

Write rules according to stages

- **Normally, a Windows kernel LPE exploit has these stages**
  - Vulnerability Triggering
  - Heap Feng Shui
  - ***Kernel Information Leak***
  - ***Arbitrary Address Read and Write***
  - Control Flow Hijacking
  - Privilege Escalation

- ***Write rules based on the common features of each stage***

Kernel information leak

- **Common kernel information leak techniques**
  - NtQuerySystemInformation
    - ***SystemBigPoolInformation***
    - *SystemModuleInformation*
    - …
  - Win32k Shared Info User Handle Table
  - Descriptor Tables
  - ***HMValidateHandle***
  - GdiSharedHandleTable

Arbitrary address read/write

- **Common arbitrary address read/write techniques**
  - SetWindowLong / SetWindowLongPtr
  - SetWindowText / InternalGetWindowText / NtUserDefSetText
  - GetMenuItemRect / SetMenuItemInfo / GetMenuBarInfo
  - ***NtUpdateWnfStateData / NtQueryWnfStateData***
  - GetBitmapBits / SetBitmapBits
  - GetPaletteEntries / SetPaletteEntries
  - ***CreatePipe / NtFsControlFile***
  - ***Previous Mode + NtReadVirtualMemory / WriteVirtualMemory***

- ***Note: Not all techniques are suitable for YARA rules***

Build the right rule

1. Write rules according to the signatures of each stage of exploitation

2. Write rules for latest exploit techniques

3. Write rules for the most likey vulnerability

Information Classification: General

Write rules for latest exploit techniques

- **Arbitrary address read with the help of *Pipe Attribute***
  - *July 2020, " Scoop the Windows 10 pool! "* by Paul Fariello and Corentin Bayet of Synacktiv

- **Arbitrary address read and write via *Windows Notification Facility (WNF)***
  - *June 2021, " PuzzleMaker attacks with Chrome zero-day exploit chain "* by Kaspersky
  - *July 2021 " CVE-2021-31956 Exploiting the Windows Kernel (NTFS with WNF) "* by Alex Plaskett
  - *July 2021 " Windows Pool OverFlow Exploit "* by YanZiShuang(@YanZiShuang)

- ***These two exploit techniques are universal, we wrote rules for them and caught some high-value samples***

Build the right rule

1. Write rules according to the signatures of each stage of exploitation
2. Write rules for latest exploit techniques
3. **Write rules for the most likey vulnerability**

black hat USA 2022

Write rules for the most likey vulnerability

- **The case of *Desktop Windows Manager (DWM)* vulnerability**
  - ***April 13, 2021****, Kaspersky wrote a blog and disclosed CVE-2021-28310, which is an itw 0day in the Windows DWM component*
    - *" Zero-day vulnerability in Desktop Window Manager (CVE-2021-28310) used in the wild "*
  - ***May 03, 2021****, ZDI published another blog, disclosing another vulnerability CVE-2021-26900, which is also a vulnerability in Windows DWM component*
    - *" CVE-2021-26900: Privilege escalation via a use after free vulnerability in win32k "*
  - ***May 22, 2021****, we caught a new itw DWM 0day CVE-2021-33739*

Build a workable system

- **When an exploit is matched by a rule, how to notify us in time?**
  - VirusTotal: ***noreply@vt-community.com***
  - Own Products: ***Email***

- **When we get an exploit, how to quickly reproduce and classify it?**
  - Prepare three types of reproduce environments: Nday, 1day, 0day
  - VMs of Windows 7, Windows 10, Windows 11, covered both x86 and x64

- **What skills should we master to debug different Windows LPE exploits?**
  - ***Driver Verifier***
  - Windows Remote Debugging (when debugging dwm.exe)

Build a workable system

- **When an exploit is matched by a rule, how to notify us in time?**
  - VirusTotal: ***noreply@vt-community.com***
  - Own Products: Email (Slack is also a good choice)

- **When we get an exploit, how to quickly reproduce and classify it?**
  - Prepare ***three types of reproduce environments***: Nday, 1day, 0day
  - ***VMs*** of Windows 7, Windows 10, Windows 11, covered both x86 and x64

- **What skills should we master to debug different Windows LPE exploits?**
  - ***Driver Verifier***
  - Windows Remote Debugging (when debugging dwm.exe)

Build a workable system

- **When an exploit is matched by a rule, how to notify us in time?**
  - VirusTotal: ***noreply@vt-community.com***
  - Own Products: Email (Slack is also a good choice)

- **When we get an exploit, how to quickly reproduce and classify it?**
  - Prepare three types of reproduce environments: Nday, 1day, 0day
  - VMs of Windows 7, Windows 10, Windows 11, covered both x86 and x64

- **What skills should we master to debug different Windows LPE exploits?**
  - ***Driver Verifier***
  - Windows Remote Debugging (when debugging dwm.exe)

Test and improve the system

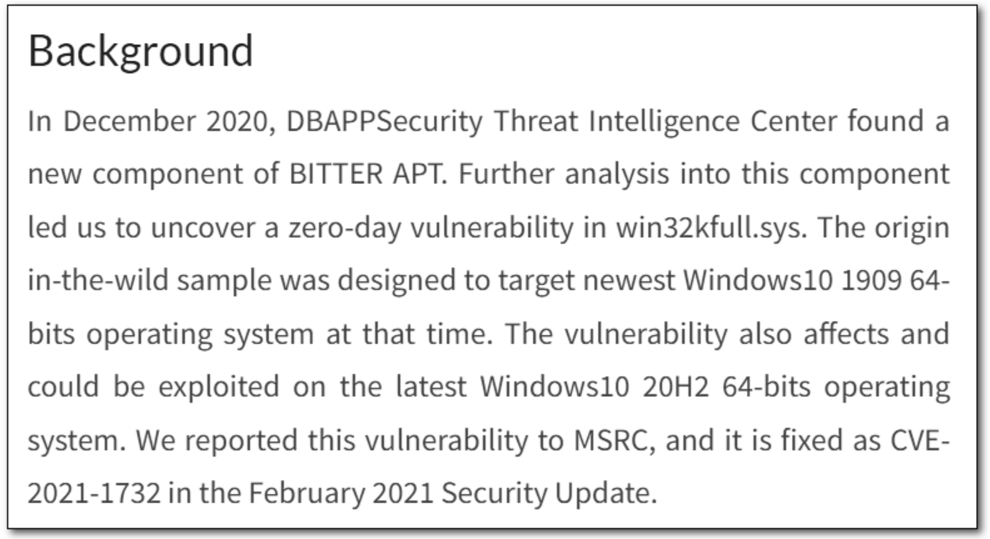
- **Eliminate false positives and false negatives**
  1. Use ***historical itw samples*** to test the rules
  2. Use ***public pocs/exploits*** to test the rules
  3. ***Write pocs/exploits*** and test the rules (when public pocs/exploits are unavailable)
  4. Apply the rules to a large number of samples for ***stress testing***
  5. (Continuously) ***Convert the latest exploit*** techniques into rules and test them

Agenda

- Motivation
- Learn from history (and now)
- One road leads to Rome
- **Results**
  - **The Story of CVE-2021-1732**
  - **The Story of CVE-2021-33739**
  - **The Story of a “Patched” 1day**
- Takeaways

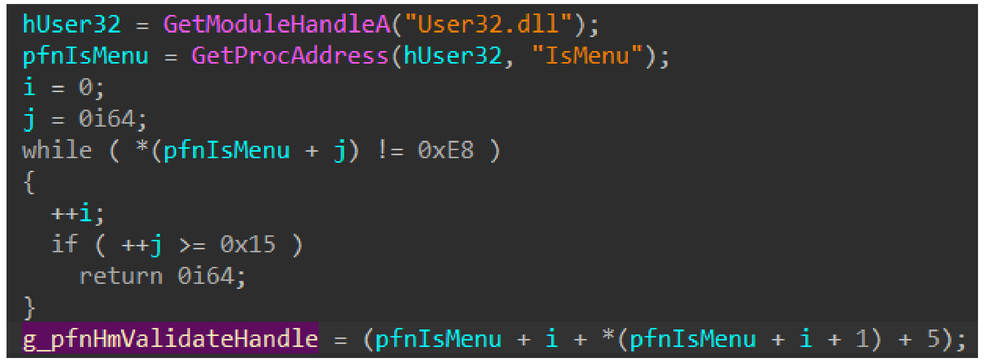
The Story of CVE-2021-1732

- **Data source**
  - The itw sample was from our ***private dataset***
- **Why it caught our attention**
  - It used ***HMValidateHandle*** to leak kernel address
- **Usage scenario**
  - The sample was used as a ***standalone component***
  - ***Need a parameter*** (process id)



Background

In December 2020, DBAPPSecurity Threat Intelligence Center found a new component of BITTER APT. Further analysis into this component led us to uncover a zero-day vulnerability in win32kfull.sys. The origin in-the-wild sample was designed to target newest Windows10 1909 64-bits operating system at that time. The vulnerability also affects and could be exploited on the latest Windows10 20H2 64-bits operating system. We reported this vulnerability to MSRC, and it is fixed as CVE-2021-1732 in the February 2021 Security Update.



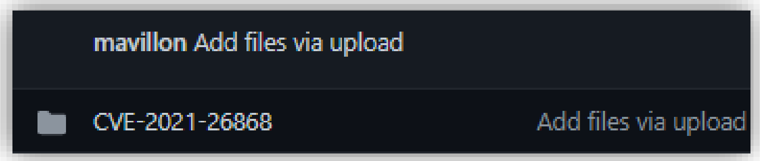
```
hUser32 = GetModuleHandleA("User32.dll");
pfnIsMenu = GetProcAddress(hUser32, "IsMenu");
i = 0;
j = 0i64;
while ( *(pfnIsMenu + j) != 0xE8 )
{
  ++i;
  if ( ++j >= 0x15 )
    return 0i64;
}
g_pfnHmValidateHandle = (pfnIsMenu + i + *(pfnIsMenu + i + 1) + 5);
```

The Story of CVE-2021-1732

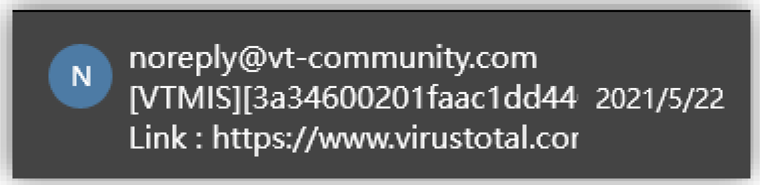
- **Some highlights of the itw sample**
  - It ***targeted the latest version of Windows*** operating system (*Windows 10 1909 x64*)
    - *The sample was compiled in May 2020*
  - It used ***GetMenuBarInfo*** to built arbitrary address read primitive
    - *Which is a previously undisclosed exploit technique*
  - Before exploit, it ***detected specific antivirus and performed system version check***
    - *Targeted Windows 10 1709 x64 ~ Windows 10 1909 x64*
- More details can refer to our **blog**

The Story of CVE-2021-33739

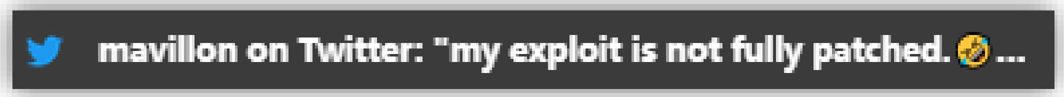
- **Data source**
  - The itw sample was from ***VirusTotal*** (compiled from ***GitHub***)


mavillon Add files via upload
CVE-2021-26868 Add files via upload

- **Why it caught our attention**
  - It hit a rule we wrote for ***the most likely vulnerability*** (***Desktop Window Manager***)


noreply@vt-community.com
[VTMIS][3a34600201faac1dd44 2021/5/22
Link : https://www.virustotal.cor

- **Side note**
  - The "author" ***accidentally introduced*** this new bug when writing an exploit for CVE-2021-26868
  - It seems that ***the "author" knew this***


mavillon on Twitter: "my exploit is not fully patched. 🤣 ...

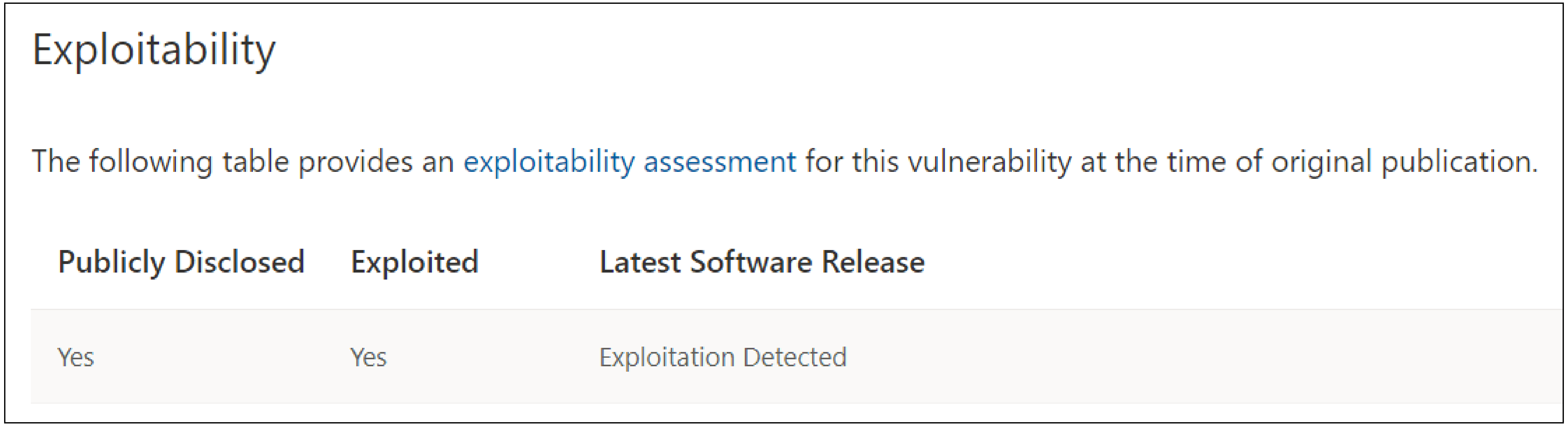
The Story of CVE-2021-33739

- **This is what we sent to MSRC before the bug was fixed**

> The first vulnerability has been fixed by Microsoft in the May 2021 patch, but the second vulnerability is still a zero day.
>
> ***I think the exploit author accidentally included a second vulnerability when attempting to publish the exploit code of a known vulnerability***, and the second vulnerability happened to be discovered by me when I hunting for in-the-wild zero day.
>
> So ***I think the second vulnerability is not strictly a zero-day vulnerability in the wild***. This is just my opinion, the final release definition depends on you.

The Story of CVE-2021-33739

- This is the exploit status finally published by MSRC 😉



Exploitability

The following table provides an exploitability assessment for this vulnerability at the time of original publication.

Publicly Disclosed	Exploited	Latest Software Release
Yes	Yes	Exploitation Detected

Root cause of CVE-2021-33739

- Unbalanced reference count on ***CinteractionTrackerBindingManager*** Object in ***dwmcore.dll***

- Steps to trigger the vulnerability
  1. Create a ***CInteractionTrackerBindingManagerMarshaler*** resource
  2. Create a ***CinteractionTrackerMarshaler*** resource
  3. ***Bind*** the resource created by step 1 twice to the resource created by step 2 (as follows), and ***do not release*** these resources manully

```
DWORD dwDataSize = 12;
DWORD* szBuff = (DWORD*)malloc(4 * 3);
szBuff[0] = 0x02;   // resource1_id is DirectComposition::CInteractionTrackerMarshaler
szBuff[1] = 0x02;   // resource2_id is DirectComposition::CInteractionTrackerMarshaler
szBuff[2] = 0xffff; // new_entry_id
```

Root cause of CVE-2021-33739

- Normally, the CinteractionTrackerBindingManager object will call ***ProcessSetTrackerBindingMode*** twice to ***add reference count by 2***
- Then the code will call *RemoveTrackerBindings* twice to sub reference count, and release the *CinteractionTrackerBindingManager* object normally when reference count is reduced to 0

```
// reference count starts from 0
CResourceFactory::Create +1 .................................... ref_count = 1
CResourceTable::CreateEmptyResource +1 ......................... ref_count = 2
CComposition::Channel_CreateResource -1 ........................ ref_count = 1
CInteractionTrackerBindingManager::ProcessSetTrackerBindingMode +1 ....... ref_count = 2
CInteractionTrackerBindingManager::ProcessSetTrackerBindingMode +1 ....... ref_count = 3
CResourceTable::DeleteHandle -1 ................................ ref_count = 2
CInteractionTrackerBindingManager::RemoveTrackerBindings -1 ............. ref_count = 1
CInteractionTrackerBindingManager::RemoveTrackerBindings -1 ............. ref_count = 0
// release object when reference count is reduced to 0
```

Root cause of CVE-2021-33739

- Normally, the *CinteractionTrackerBindingManager* object will call *ProcessSetTrackerBindingMode* twice to add reference count by 2.

- Then the code will call ***RemoveTrackerBindings*** twice to ***sub reference count by 2***, and release the CinteractionTrackerBindingManager object normally when reference count is reduced to 0

```
// reference count starts from 0
CResourceFactory::Create +1 .............................................. ref_count = 1
CResourceTable::CreateEmptyResource +1 ................................... ref_count = 2
CComposition::Channel_CreateResource -1 .................................. ref_count = 1
CInteractionTrackerBindingManager::ProcessSetTrackerBindingMode +1 ....... ref_count = 2
CInteractionTrackerBindingManager::ProcessSetTrackerBindingMode +1 ....... ref_count = 3
CResourceTable::DeleteHandle -1 .......................................... ref_count
CInteractionTrackerBindingManager::RemoveTrackerBindings -1 .............. ref_count = 1
CInteractionTrackerBindingManager::RemoveTrackerBindings -1 .............. ref_count = 0
// release object when reference count is reduced
```

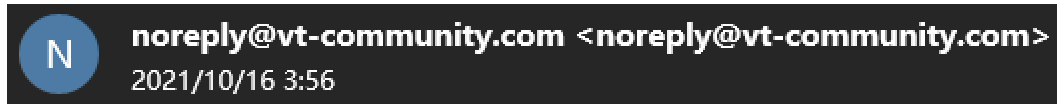
Root cause of CVE-2021-33739

- In the vulnerability scenario, it will call ***ProcessSetTrackerBindingMode*** only once to ***add reference count by 1***
- But the code will still call ***RemoveTrackerBindings*** twice to ***sub reference count by 2***
- ***UAF*** in the second RemoveTrackerBindings call

```
// reference count starts from 0
CResourceFactory::Create +1 ................................... ref_count = 1
CResourceTable::CreateEmptyResource +1 ........................ ref_count = 2
CComposition::Channel_CreateResource -1 ....................... ref_count = 1
CInteractionTrackerBindingManager::ProcessSetTrackerBindingMode +1 ....... ref_count = 2
CResourceTable::DeleteHandle -1 ............................... ref_count = 1
CInteractionTrackerBindingManager::RemoveTrackerBindings -1 ............. ref_count = 0
// release object when reference count is reduced to 0
CInteractionTrackerBindingManager::RemoveTrackerBindings -1 ............. UAF
```

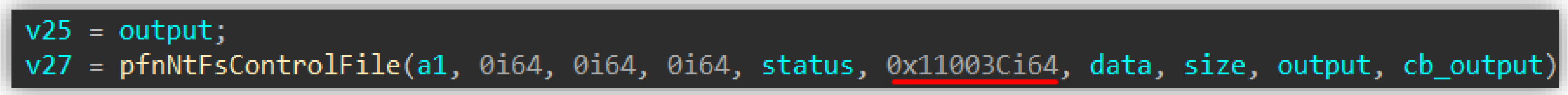
The Story of a “Patched” 1day

- **Data source**
  - The itw sample was from ***VirusTotal***



N noreply@vt-community.com <noreply@vt-community.com>
2021/10/16 3:56

- **Why it caught our attention**
  - It hit a rule we wrote for the latest exploit techniques (***Pipe Attribute***)



```
v25 = output;
v27 = pfnNtFsControlFile(a1, 0i64, 0i64, 0i64, status, 0x11003Ci64, data, size, output, cb_output)
```

- **Usage scenario**
  - The sample was used as a ***standalone component***
  - The exploit code was ***adapted to a variety of Windows*** OS versions

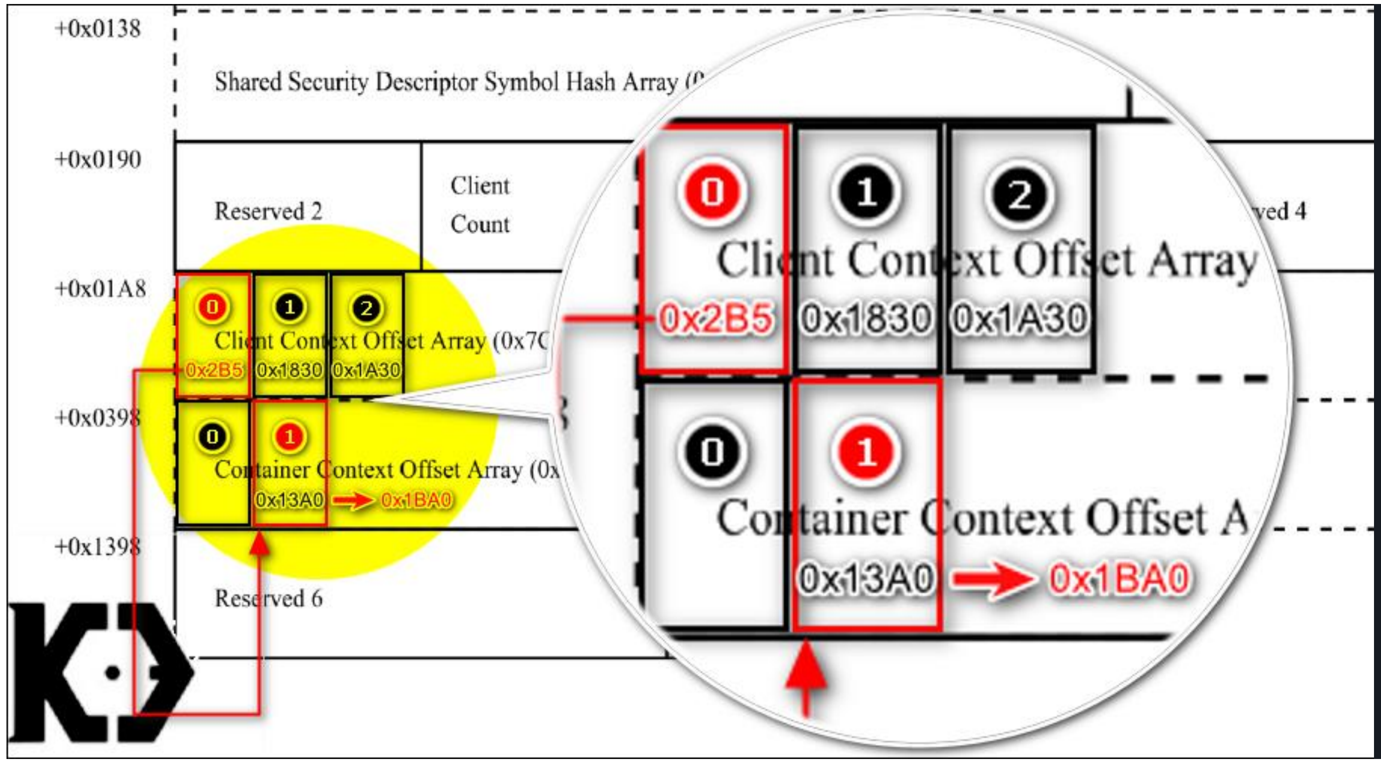
The Story of a "Patched" 1day

- **Basic information**
  - A type confusion vulnerability in clfs.sys
  - ***Caught in October 2021, "Patched" in September 2021***
  - May be one of *CVE-2021-36963, CVE-2021-36955, CVE-2021-38633 or none of them*

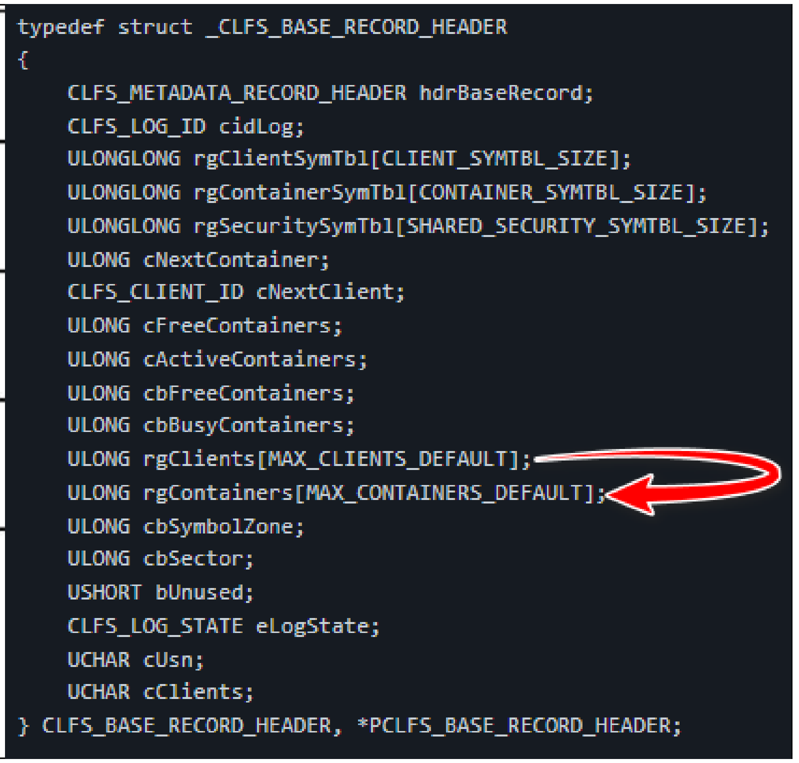
- **Root cause**
  - The clfs.sys ***lacks some checks on Client Context Offset,*** an attacker can take advantage of this to provide an invalid Client Context Offset

The Story of a “Patched” 1day

- **How the itw sample use this vulnerability**
  - The itw sample leveraged this to make the first Client Context Offset(***0x2B5***) point to the second ***Container Context Offset***, then it used an ***1-bit flip*** to change the second Container Context Offset from ***0x13A0*** to ***0x1BA0***


+0x0138
Shared Security Descriptor Symbol Hash Array
+0x0190
Reserved 2
Client Count
+0x01A8
Client Context Offset Array (0x7C
0x2B5
0x1830
0x1A30
+0x0398
Container Context Offset Array (0x
0x13A0
0x1BA0
+0x1398
Reserved 6
Client Context Offset Array
Container Context Offset A
Reserved 4

“DeathNote of Microsoft Windows Kernel”, KeenLab, 2016

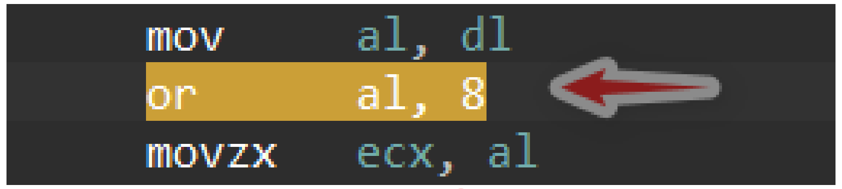


```
typedef struct _CLFS_BASE_RECORD_HEADER
{
    CLFS_METADATA_RECORD_HEADER hdrBaseRecord;
    CLFS_LOG_ID cidLog;
    ULONGLONG rgClientSymTbl[CLIENT_SYMTBL_SIZE];
    ULONGLONG rgContainerSymTbl[CONTAINER_SYMTBL_SIZE];
    ULONGLONG rgSecuritySymTbl[SHARED_SECURITY_SYMTBL_SIZE];
    ULONG cNextContainer;
    CLFS_CLIENT_ID cNextClient;
    ULONG cFreeContainers;
    ULONG cActiveContainers;
    ULONG cbFreeContainers;
    ULONG cbBusyContainers;
    ULONG rgClients[MAX_CLIENTS_DEFAULT];
    ULONG rgContainers[MAX_CONTAINERS_DEFAULT];
    ULONG cbSymbolZone;
    ULONG cbSector;
    USHORT bUnused;
    CLFS_LOG_STATE eLogState;
    UCHAR cUsn;
    UCHAR cClients;
} CLFS_BASE_RECORD_HEADER, *PCLFS_BASE_RECORD_HEADER;
```

“CLFS Internals”, Alex Ionescu, 2021

The Story of a “Patched” 1day

- The *1-bit filp* in *CClfsLogFcbPhysical::FlushMetadata*



```
mov     al, dl
or      al, 8
movzx   ecx, al
```

```
1: kd> .formats 13A0
Evaluate expression:
  Hex:     00000000`000013a0

  Binary:  00000000 00000000 00000000 00000000 00000000 00000000 00010011 10100000

1: kd> ? 13 | 8
Evaluate expression: 27 = 00000000`0000001b

1: kd> .formats 1BA0
Evaluate expression:
  Hex:     00000000`00001ba0

  Binary:  00000000 00000000 00000000 00000000 00000000 00000000 00011011 10100000
```

The Story of a “Patched” 1day

- **The *arbitrary address write* primitive in *CClfsBaseFilePersisted::RemoveContainer***
  - The normal virtual table of a ClfsContainer object

```
1: kd> dps fffff804`2e9354b8
fffff804`2e9354b8  fffff804`2e960c10 CLFS!CClfsContainer::AddRef
fffff804`2e9354c0  fffff804`2e94c060 CLFS!CClfsContainer::Release
fffff804`2e9354c8  fffff804`2e92b570 CLFS!CClfsContainer::GetSListEntry
fffff804`2e9354d0  fffff804`2e9489e0 CLFS!CClfsContainer::Remove
```

  - The fake virtual table of the fake ClfsContainer object

```
0: kd> dps 0000003a`b777f1e8
0000003a`b777f1e8  00000000`00000000
0000003a`b777f1f0  fffff804`2f0cc390 nt!HalpDmaPowerCriticalTransitionCallback
0000003a`b777f1f8  00000000`00000000
0000003a`b777f200  fffff804`2ef95f70 nt!XmXchgOp
```

The Story of a "Patched" 1day

- The *arbitrary address read* primitive
  - "Pipe Attribute" technique described in the "*Scoop the Windows 10 pool!*" (SSTIC2020)

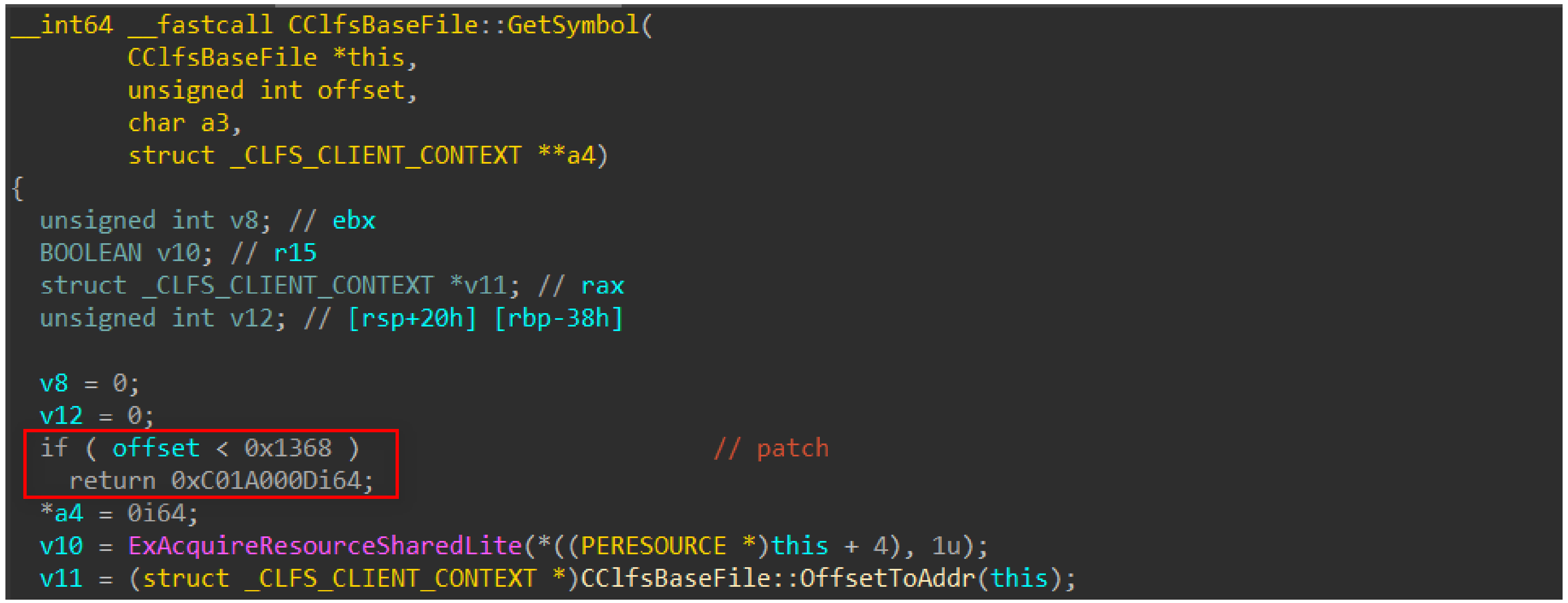
```
struct PipeAttribute {
    LIST_ENTRY list;
    char * AttributeName;
    uint64_t AttributeValueSize;
    char * AttributeValue;
    char data [0];
};
```

  - SystemBigPoolInformation technique described in "*windows_kernel_address_leaks*" (Github)

```
hRes = NtQuerySystemInformation(SystemBigPoolInformation, pBuffer, dwBufSize, &dwOutSize);
```

The Story of a “Patched” 1day

- **The September 2021 patch**
  - The patch only ***checked the value of Client Context Offset*** to make sure it ***couldn't be less than 0x1368***

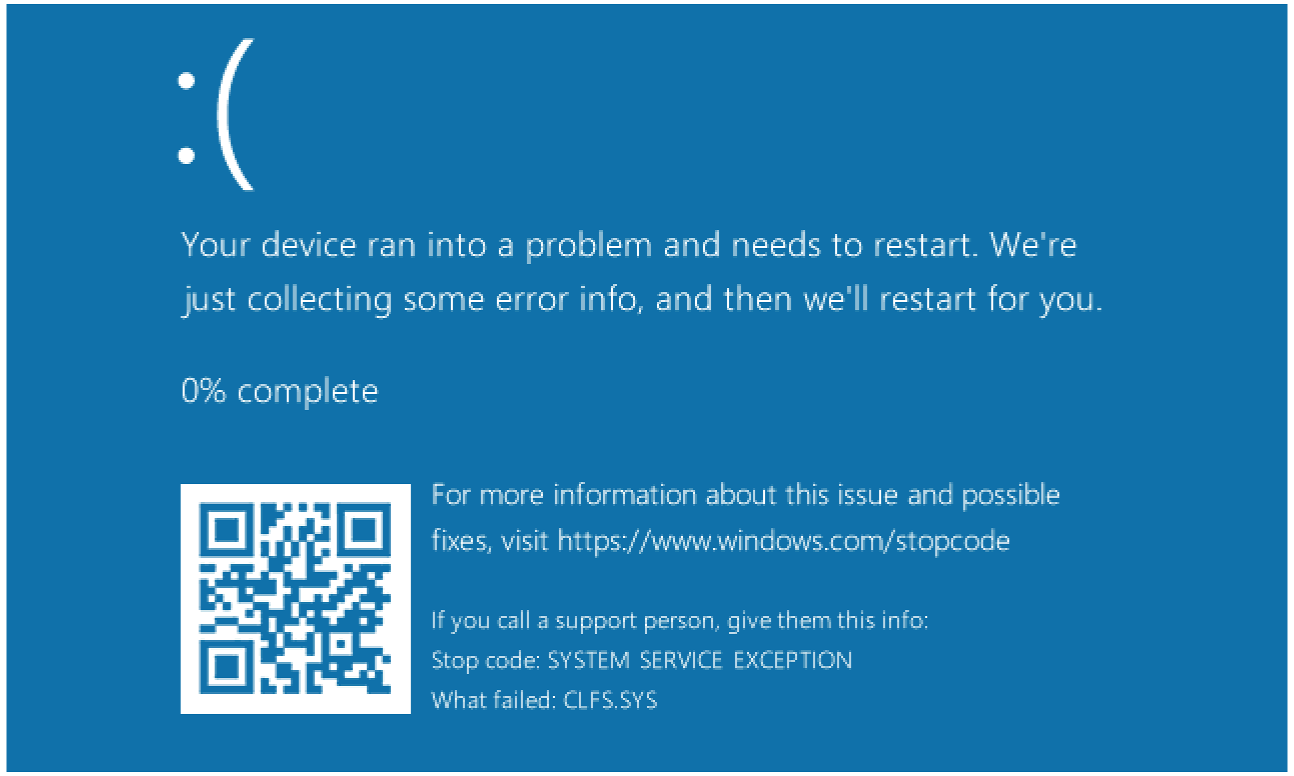


```
__int64 __fastcall CClfsBaseFile::GetSymbol(
        CClfsBaseFile *this,
        unsigned int offset,
        char a3,
        struct _CLFS_CLIENT_CONTEXT **a4)
{
  unsigned int v8; // ebx
  BOOLEAN v10; // r15
  struct _CLFS_CLIENT_CONTEXT *v11; // rax
  unsigned int v12; // [rsp+20h] [rbp-38h]

  v8 = 0;
  v12 = 0;
  if ( offset < 0x1368 )                              // patch
    return 0xC01A000Di64;
  *a4 = 0i64;
  v10 = ExAcquireResourceSharedLite(*((PERESOURCE *)this + 4), 1u);
  v11 = (struct _CLFS_CLIENT_CONTEXT *)CClfsBaseFile::OffsetToAddr(this);
```

The Story of a “Patched” 1day

- **Has it been fixed thoroughly?**
  - What if we construct a Client Context Offset that is ***greater than 0x1368***, and make the Client Context Offset point directly to the CClfsContainer object?
    - BSOD
    - A **variant** of this “patched” 1day
    - Reported to MSRC at December 2021
    - Microsoft fixed this case in April 2022 as ***CVE-2022-24481***



:(

Your device ran into a problem and needs to restart. We're just collecting some error info, and then we'll restart for you.

0% complete

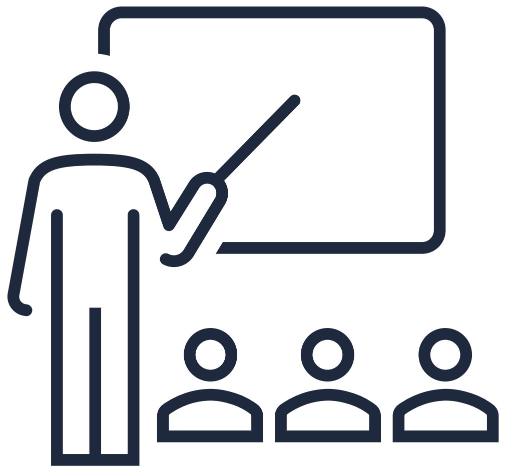
For more information about this issue and possible fixes, visit https://www.windows.com/stopcode

If you call a support person, give them this info:
Stop code: SYSTEM SERVICE EXCEPTION
What failed: CLFS.SYS

Agenda

- Motivation
- Learn from history (and now)
- One road leads to Rome
- Results
  - The Story of CVE-2021-1732
  - The Story of CVE-2021-33739
  - The Story of a “Patched” 1day
- **Takeaways**

Suggestions



1. Choose the most suitable method within your capability
2. Carefully study historical cases is always a good thing
3. Keep an eye out for new variants of a new itw vulnerabillity

Insights

1. **More vulnerabilities in clfs may appear in the future**

2. **“Pipe Attribute” method will be using again in the future**

3. **The following techniques may be popular in the future**
   - *Arbitrary address read/write with the help of WNF, POC2021*
   - *Arbitrary address read/write with the help of ALPC, Blackhat Asia 2022*
   - *Arbitrary address read/write with the help of I/O Ring, TyphoonCon 2022*

Acknowledgements

- Thanks to guys of DBAPPSecurity WeBin Lab
- Thanks to @megabeets_ and @EyalItkin for their inspiring blogs
- Thanks to Xiaoyi Tu and Dong Wu of DBAPPSecurity Lieying Lab
- Thanks to @YanZiShuang and @oct0xor for sharing debugging tips

Thank you!
Questions?