BLUEHAT

IL 2022

Win32k: Smash The Ref

Gil Dabah
CEO, Piiano

Gil Dabah
@_arkon

Guys & girls!
Exactly a year ago I promised over 15 bugs in win32k.
You're welcome to read and find out about my biggest research so far: #win32k #SmashTheRef bug class - github.com/gdabah/win32k-...
Check out the paper and the POCs, there are some crazy stuff going on. Promise!

gdabah/**win32k-bugs**

Dump of win32k POCs for bugs I've found

1 Contributor · 0 Issues · 339 Stars · 83 Forks

github.com
win32k-bugs/SmashTheRef at master · gdabah/win32k-bugs
Dump of win32k POCs for bugs I've found. Contribute to gdabah/win32k-bugs development by creating an account on GitHub.

Agenda

4 parts:

- Introduction – classic win32k bugs
- Zombies – new attack & techniques
- Exploitation – doing the impossible
- Wrap up – the real cool stuff

Part 1

Introduction

Win32K

- GUI done in kernel
- UI objects are managed in kernel
- What can go wrong?

- Good - Performance

  U->K instead of U->U

- Bad - Security

  From K->U for helpers

- Ugly – the code quality

Kernel -> User Callbacks

- Win32k calls back to user for
  - window events
  - hooks
  - notification events
  - helper functions

- Note ‘xxx’ prefix function names for leaving kernel->user
  - (aka ‘usercrit’)
  - xxxSendMessage
  - xxxCreateWindow
  - xxxDestroyWindow
  - & more

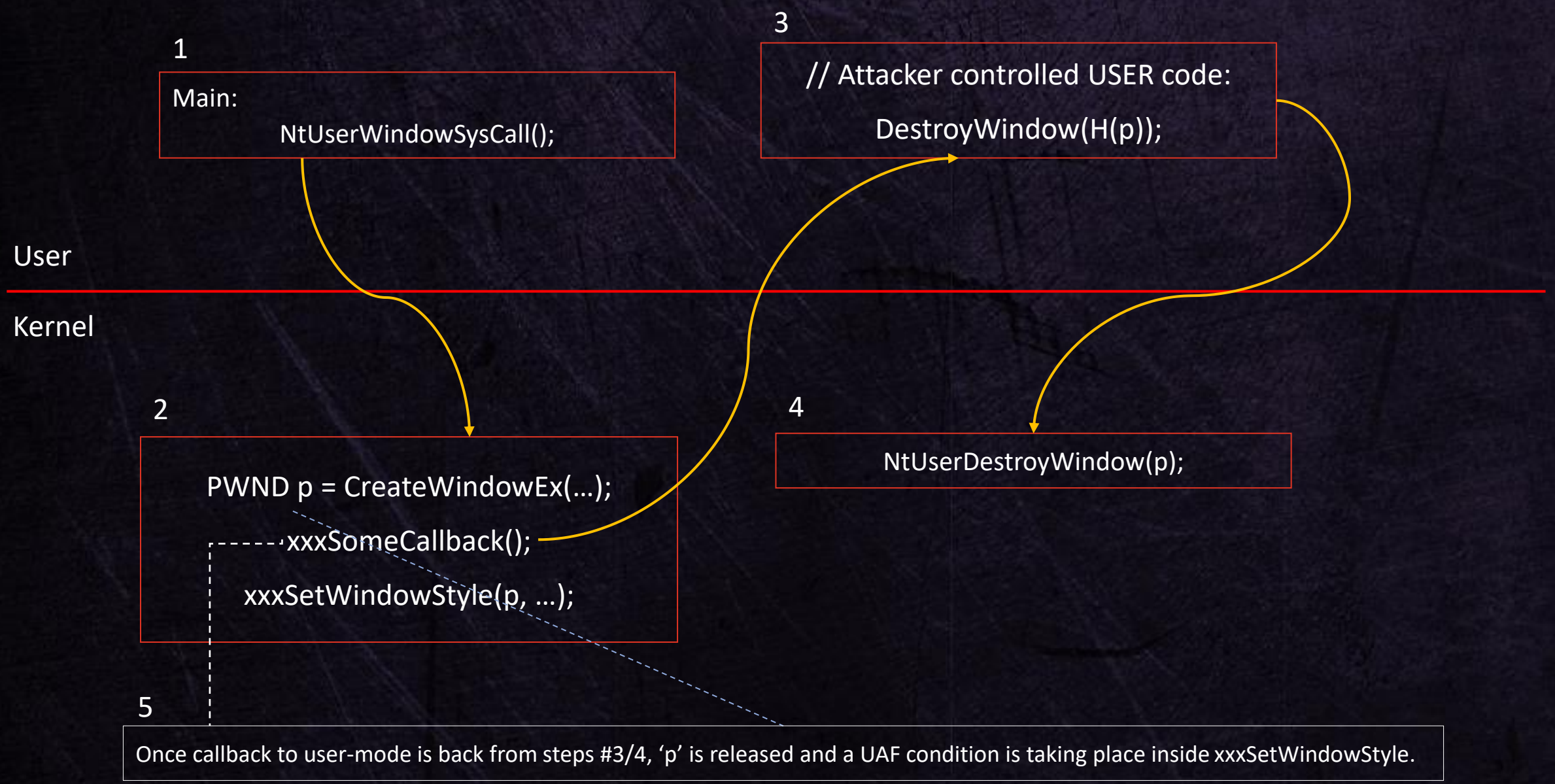
Abusing Callbacks – Introducing use-after-free

Kernel pseudo code snippet:

```
NtUserWindowSysCall()
{
    PWND p = xxxCreateWindowEx(…);
    xxxSomeCallback();
    xxxSetWindowStyle(p);
```

Where's the bug?

Trigger UAF Flow



Kernel UI Objects Locking Mechanism

- Create/Destroy syscalls for objects
- Objects start with 0 ref count
- Destroy functions free objects *only if 0 refs!*
- ThreadLock/ThreadUnlock – local scope for locking
  - Blocks usermode from killing objects (hmmm)

Introducing The Fix – Thread Scope Locking

```
PWND p = xxxCreateWindowEx(…);
ThreadLock(p);
xxxSomeCallback();
// Zombie window is used next, immune!
xxxSetWindowStyle(p);
ThreadUnlock();
```

Part 2

Zombies

Introducing Zombie Objects

Destroying an object with >0 refs???

```
PMENU p = CreateMenu();
ThreadLock(p);
xxxSomeCallback();
//'p' menu is now a zombie!
ThreadUnlock(p);
// Really frees 'p' inside
decrement ref - last ref goes from 1 to 0!
```

Zombie Disorders

Destroyed objects that are still referenced are called zombies!
(otherwise, they're freed)

Zombies still exist (allocated) in kernel (otherwise bad things will happen)

Zombies cannot be accessed from usermode

Each object-destruction function behaves differently

Pretty much cannot be used in this state, only waiting to be freed

DestroyWindow API

- DestroyWindow *releases* all linked objects immediately
  - Menus, carets, child-windows, timers, etc

- xxxDestroyWindow is first called from user mode
- Then again (a second time!) *when* last ref is gone - *side effects?!*

- Remember - like OOP:

  When a destructor is called, it calls the destructors of its members!

Smashable Site - Recipe

1. A dumb pointer that's assigned to a UI object
2. A window object's **last decref** functionality (ThreadUnlock & friends)
3. Control where the last decref happens
4. Use-after-free of #1's pointer
5. A side effect behavior that can be **reloaded** unto a zombie window

xxxDestroyWindow – Releasing Sub-resources

```
void xxxDestroyWindow(PWND pwnd)
{
  …
  xxxFW_DestroyAllChildren(); // Destroy child windows, if exist!
  …
  if (NULL != pwnd->spmenu) // If there's a menu, remove and destroy it
      DestroyMenu(pwnd->spmenu);
  …
  if (pwnd == ptiCurrent->pq->capturedPwnd)
    xxxReleaseCapture();
  …
  DereferenceClass(pwnd);
  …
  if (HMMarkObjectDestroy(pwnd)) // Check for zero refs!
    HmFreeObject(pwnd); // Only now free the object
```

A Real-Life Example: xxxMnOpenHierarchy

```
ThreadLock(pwndParent);
// Notice there's a parent-child relation :)
pwnd = xxxCreateWindow(..., pwndParent, ...);
ThreadUnlock();
if (NULL == pwnd) return ...
// From now on the window exists and can be used.
xxxSendMessage(pwnd...)
```

We got a UAF, give me a yeahhh!

Killing in The Stack Of – The Secret Sauce

xxxDestroyWindow
xxxDestroyWindowIfSupported
HMDestroyUnlockedObject
HMUnlockObjectInternal
HMAssignmentUnlock
xxxFreeWindow
xxxDestroyWindow
xxxDestroyWindowIfSupported
HMDestroyUnlockedObject
ThreadUnlock1
xxxMNOpenHierarchy

Chain-effect:
ThreadUnlock kills parent window first and then kills the child window too!

Reloading a Zombie–woOT??

But if an object is destroyed – how can we link sub-resources to it?

We **stack** the operation *before* it's destroyed,
but (happening) *after* it's destroyed

מבולבלים? גם אנחנו!

Reloading a Zombie–A Simple Example

Part 3

Exploitation

UAF Exploitation Problems

Race Condition

- Destruction doesn’t leave (the *usercrit)* to umode!!!!
- NTGDI
- Pool type allocations
- kLFH

TypeIsolation Objects (allocator pools per object type)

SmartObjectStackRef OOP

Let’s Go Back To The 90’s

By calling back to usermode whenever WE want

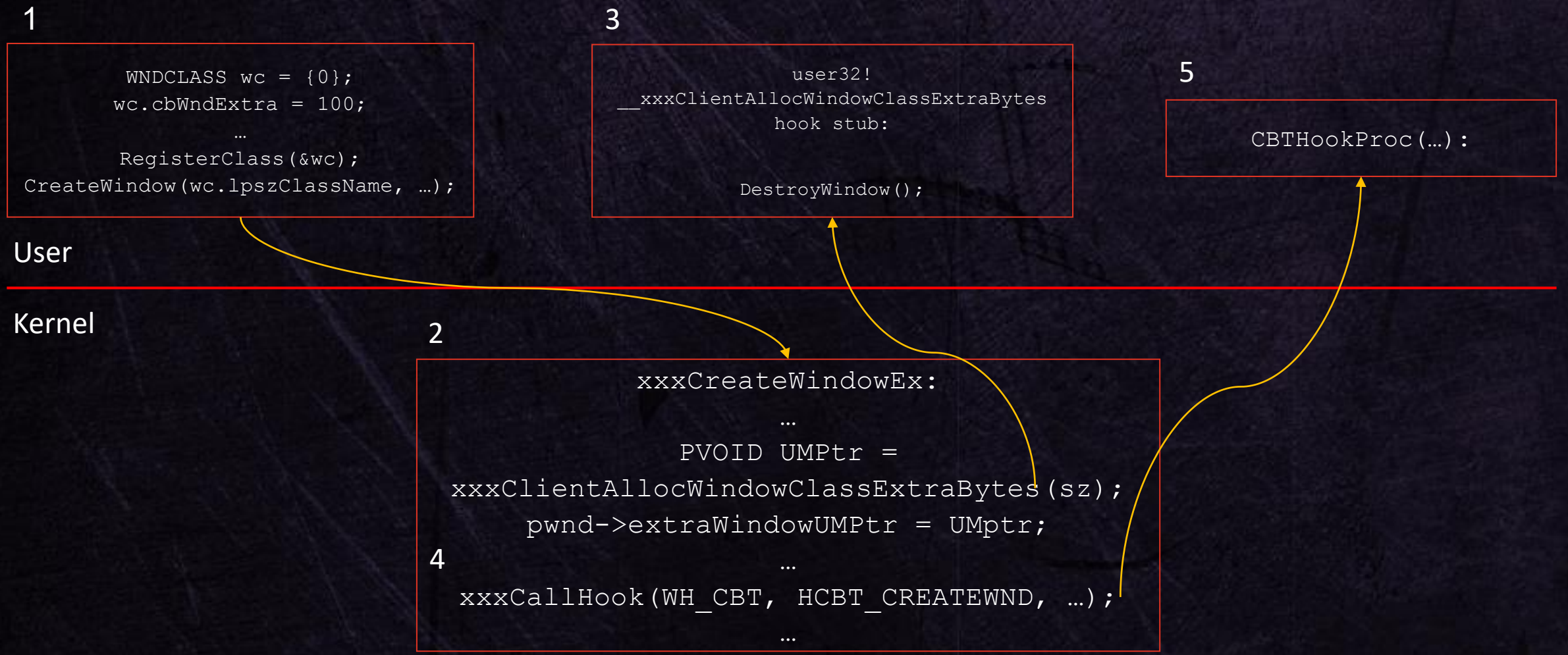
Then we can exploit the freed object without a race

But how???

The Ultimate Reloading!

```
PWND xxxCreateWindowEx(…)
{
   …
   PVOID ptr =
xxxClientAllocWindowClassExtraBytes();
   if (NULL == ptr) // error handling
   pwnd->extraWindowUMPtr = ptr;
   …
   xxxCallHook(WH_CBT, ..);
   …
   if (windowIsDestroyed(pwnd)) return NULL;
}
```

Reload ‘Back To User-Mode’



Reload ‘Back To User-Mode’- OMG

1

```
WNDCLASS wc = {0};
wc.cbWndExtra = 100;
…
RegisterClass(&wc);
CreateWindow(wc.lpszClassName, …);
```

3

```
user32!
__xxxClientAllocWindowClassExtraBytes
hook stub:

DestroyWindow();
```

5

```
CBTHookProc(…):
ExitThread();
```

User

Kernel

2

```
xxxCreateWindowEx:
…
PVOID UMPtr =
xxxClientAllocWindowClassExtraBytes(sz);
pwnd->extraWindowUMPtr = UMptr;
```

4

```
…
xxxCallHook(WH_CBT, HCBT_CREATEWND, …);
…
```

New Opportunities

Sometimes you cannot chain objects –
but you can go back to usermode and then kill'em all

```
void UnlockNotifyWindow(pmenu)
{
  pitem = pmenu->items;
  for (; pitem != &pmenu->items[pmenu->cItems]; pitem++)
  {
    if (NULL != pitem->psubmenu)
    {
      UnlockNotifyWindow(pitem->psubmenu);
    }
  }
  HMAssignmentUnlock(&pmenu->notificationWnd);
}
```

Part 4

Wrap Up

Smash The Ref - Summary

Code that calls some destructors

- like DestroyMenu, FreeTimer, DestroySMWP and many more

expects (primary) objects to be destroyed

However – it doesn't expect *other (secondary)* objects to be freed

- with the 'zombie chain' and 'reloading a zombie'

- Primary vs secondary objects, no leaving usercrit
- 'Ultimate reloading' - > xxx and sync ~~race~~ exploitation

- This attack works for multiple types of objects
  - Some don't even have refcount, oi oi oi

MSFT’s Mitigation

- Class IdentifyPrimaryDestroyTarget that is used in non-xxx destructors

- Whenever a second object is destroyed during this scope,

  it is queued instead of being really destroyed

- Global queue is emptied when it’s safe - upon return to user-mode

He has made us all very busy, but we are enjoying the work tremendously, and he his findings are helping us to improve the security of win32k*.sys quite significantly 😊

How It All Started – rare.c!xxxSendMinRectMessages

```
pdeskinfo = GETDESKINFO(pti);
   ...
   while (pwndShellHook = VWPLNext(pdeskinfo->pvwplShellHook)
   {
       ThreadLock(pwndShellHook, &tlpwnd);
       xxxSendMessageTimeout(pwndShellHook, …);
       ThreadUnlock(&tlpwnd);
   }
```

POCs:

1. xxxMnOpenHierarchy window UAF
2. FreeTimer timer UAF
3. xxxCreateCaret PQ UAF
4. Ultimate reloading - exit to user-mode from xxxFreeWindow of a zombie window
5. FreeSPB window UAF
6. xxxCapture window UAF
7. xxxCapture PQ UAF
8. zzzAttachThreadInput PQ UAF
9. xxxSendMinRectMessages DesktopInfo UAF (POC tries to catch free block before kLFH)
10. UnlockNotifyWindow menu UAF via user-mode callback
11. CSRSS Arbitrary user-mode heap-free

Spotted More Instances

1. xxxMnKeyDown
2. xxxQueryDropObject
3. xxxSetFocus
4. xxxMouseActivate
5. xxxDragObject
6. xxxCapture
7. SetMenuItemInfo
8. xxxQueryDropObject
9. SendMsgCleanup
10. MnFreePopup
11. Un/LockPopupMenu
12. zzzDestroyQueue
13. LockMFMWFPWindow
14. MNFreeItem
15. xxxFreeWindow

MSRC top 30 for 2020

$300K bounty award

Thanks to Tomer Schwartz and Saar Amar of MSRC-IL ☺

Thank You

https://github.com/gdabah/win32k-bugs/

distorm@gmail.com
@_arkon

Gil Dabah, CEO, Piiano