Windows Kernel Driver Challenge

r3kapig

Challenge Design

● Object GM

- 0x30 session paged pool(including pool header)
- Manage object Ghost and object Area

```
typedef struct _FLAPPY_PIG_GM {
    PVOID Ghost_Handle_Entry;    //Singly linked list header for object Ghost
    PVOID Area_Handle_Entry;     //Singly linked list header for object Area
    PVOID GhostHandleTable;      // Handle table which store Ghost object kernel address
    PVOID AreaHandleTable;       // Handle table which store Area object kernel address
}FLAPPY_PIG_GM, *PFLAPPY_PIG_GM;
```

Challenge Design

●Object Ghost

- session paged pool(including pool header) , pool size depends on user input
- Manage object Area
- Can only create up to 8 objects

```
typedef struct _FLAPPY_PIG_GHOST {
    PVOID Ghost_Link;    //Singly linked list for object Ghost
    DWORD64 Ghost_ID;   //Ghost ID (index of Ghost handle table)
    DWORD64 Ghost_Lock;  //Decided whether Area can be managed by user
    DWORD64 Ghost_Data1;  //Ghost Data
    DWORD64 Ghost_Data2; //Ghost Data
    DWORD64 Ghost_Info_Data; //Ghost info data which can be edit by user
    CHAR Ghost_Info[512];  // Ghost Info
}FLAPPY_PIG_GHOST, *PFLAPPY_PIG_GHOST;
```

Challenge Design

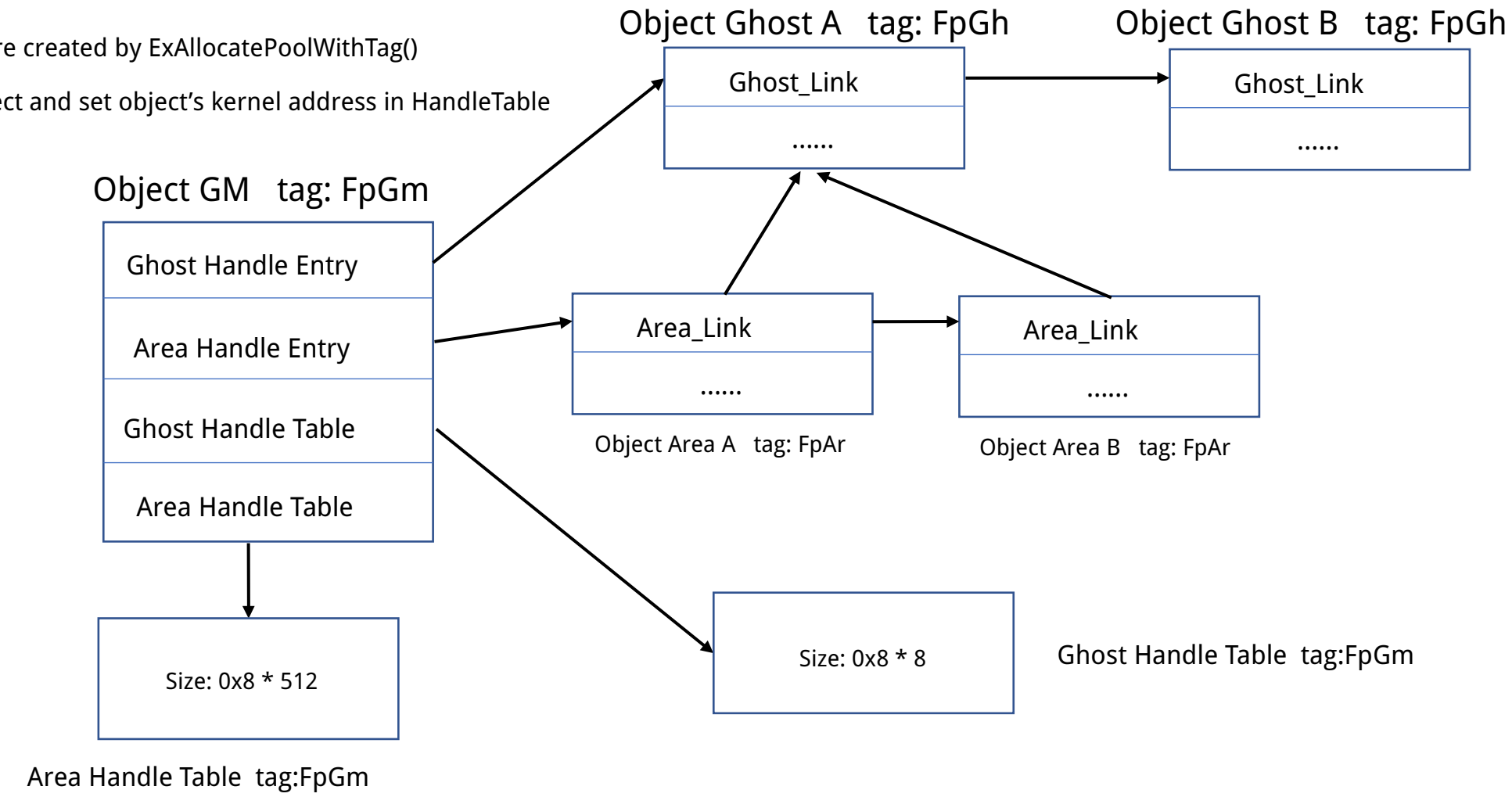
•Object Area

- session paged pool(including pool header) , pool size depends on user input
- Can only create up to 512 objects

```
typedef struct _FLAPPY_PIG_AREA {
    PVOID Area_Link; //Singly linked list for object Area
    DWORD64 Area_Location; // Decided which chunk can be modified by user in Area_Data(just offset!)
    DWORD32 Area_size; // Area size
    DWORD64 Area_index; // Area index(index of Area handle table)
    DWORD64 Area_Ghost_ID; // Area belongs to which Ghost
    PVOID Area_Entry; // Point to entry of Area Data
    CHAR Area_Data[0x1000]; // Area data,and it’s can be edit if Ghost_Lock is set
}FLAPPY_PIG_AREA, *PFLAPPY_PIG_AREA;
```

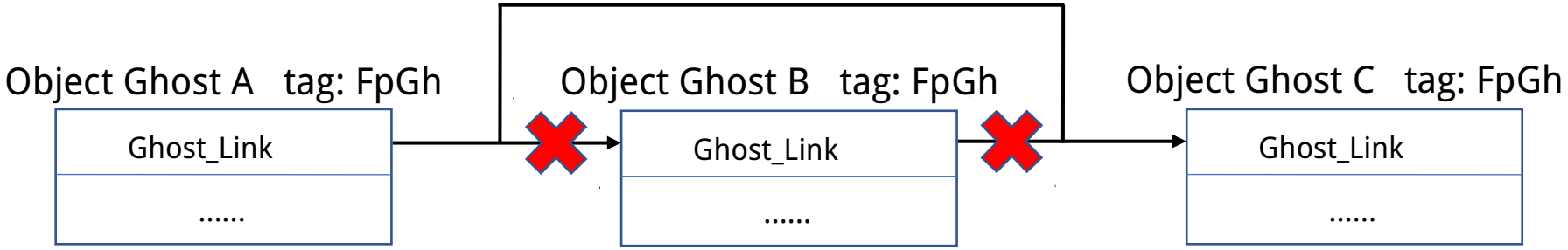
Challenge Design

- Create
  - All object are created by ExAllocatePoolWithTag()
  - Create Object and set object's kernel address in HandleTable



Challenge Design

- Delete
- Unlink just like fastbin
- Clear object kernel address in HandleTable (set NULL!)



Challenge Design

- Edit
- Edit Ghost(only Ghost_Lock and Ghost_Info_Data), it depends on Ghost_ID

```
if (temp_EGLinkLink->Ghost_ID == m_GhostEditBuffer->User_Ghost_ID) {
    temp_EGLinkLink->Ghost_Lock = m_GhostEditBuffer->User_Ghost_Lock;
    temp_EGLinkLink->Ghost_Info_Data = m_GhostEditBuffer->User_Ghost_Data;
}
```

- Edit Area

- It depends on Ghost_ID
- Ghost_Lock must be set to NULL
- Only Area_Data

```
m_TargetLocationData = (PVOID)((DWORD64)temp_EALinkLink->Area_Entry + (DWORD64)temp_EALinkLink->Area_Location * 0x8);
RtlCopyMemory(m_TargetLocationData, &m_AreaEditBuffer->Area_Data, sizeof(DWORD64));
```

Challenge Design

- Show

- It depends on Ghost_ID or Area_index

```
RtlCopyMemory(OutputBuffer, (PVOID)((DWORD64)g_FlappypigGM->GhostHandleTable +
(DWORD64)m_HandleTableLocation*0x8), sizeof(DWORD64));
```

```
RtlCopyMemory(OutputBuffer, (PVOID)((DWORD64)g_FlappypigGM->AreaHandleTable +
(DWORD64)m_HandleTableLocation*0x8), sizeof(DWORD64));
```

Vulnerabilities

● Info leak

- It’s necessary to leak kernel object address in this chellenge
- Windows 10 build 1803...so GdiSharedHandleTable, gSharedinfo or tagWND is useless...
  (https://github.com/sam-b/windows_kernel_address_leaks)
- Some 0day:) (if you have)?

✓ Object Ghost and Object Area’s kernel address is wrote in their HandleTable in Driver

```
kd> dq ffffde0f`442cde50
ffffde0f`442cde50  ffffde0f`443531e0 ffffde0f`442b53d0
ffffde0f`442cde60  ffffde0f`454e8fe0 ffffde0f`454e9fe0
ffffde0f`442cde70  ffffde0f`4a4a2fe0 ffffde0f`4a4a4fe0
ffffde0f`442cde80  ffffde0f`4a4a6fe0 ffffde0f`4a4a8fe0
```

Vulnerabilities

- Integer Overflow

- Object pool size depends on user input
- Size variable must less than or equal sizeof(Ghost_Info_Data) (0x200) or sizeof(Area_Data) (0x1000)
- But the type of size variable in CreateGhost is signed!

```
if ((signed)mGhostUserBuffer > 0x200 )
{
    Status = 0xC000000D;
    return Status;
}
```

- It will create a small pool and cause oob write in EditGhost

Exploitation

- Pool FengShui

- To make a 0x30 pool hole(including pool header)
- Too many objects to complete Pool FengShui
- Out-of-bound write’s offset is 0x30 (GhostObject->Ghost_Info_Data)

  and the Ghost object minimum pool size is 0x30(including pool header)
- GDI object, tagWND and some other objects which are useful in Arbitrary Write is useless
- Area Object seems a nice target

Exploitation

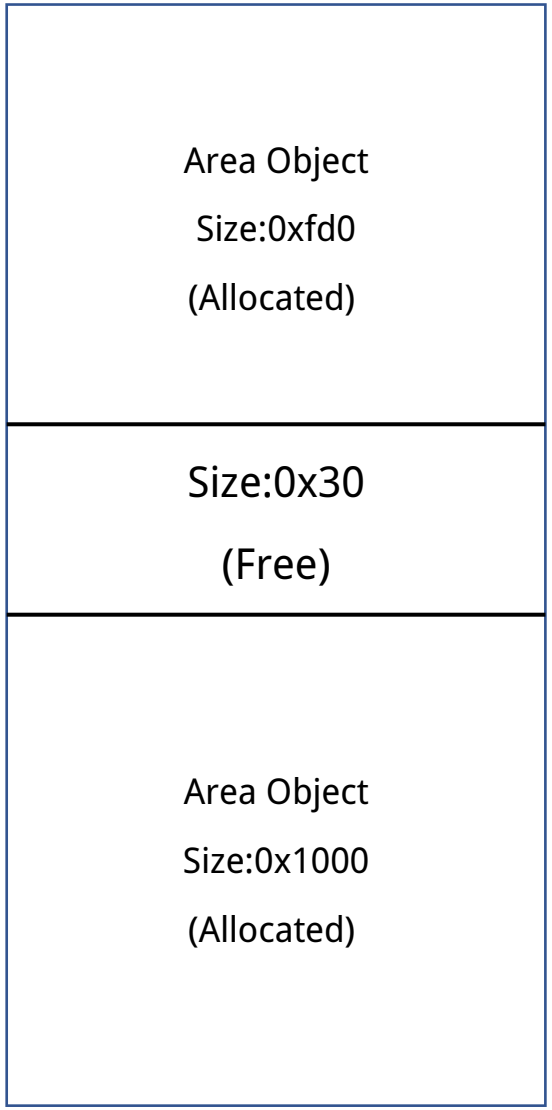
- Step1

✓ Create a lot of 0x1000 Area Objects,and they are continuous

✓ Release some Area Objects to make a 0x1000 hole

✓ Create some 0xfe0 Area Objects it will be create at pool hole

✓ Make some 0x30 pool holes for Ghost Object



Exploitation

- Step2

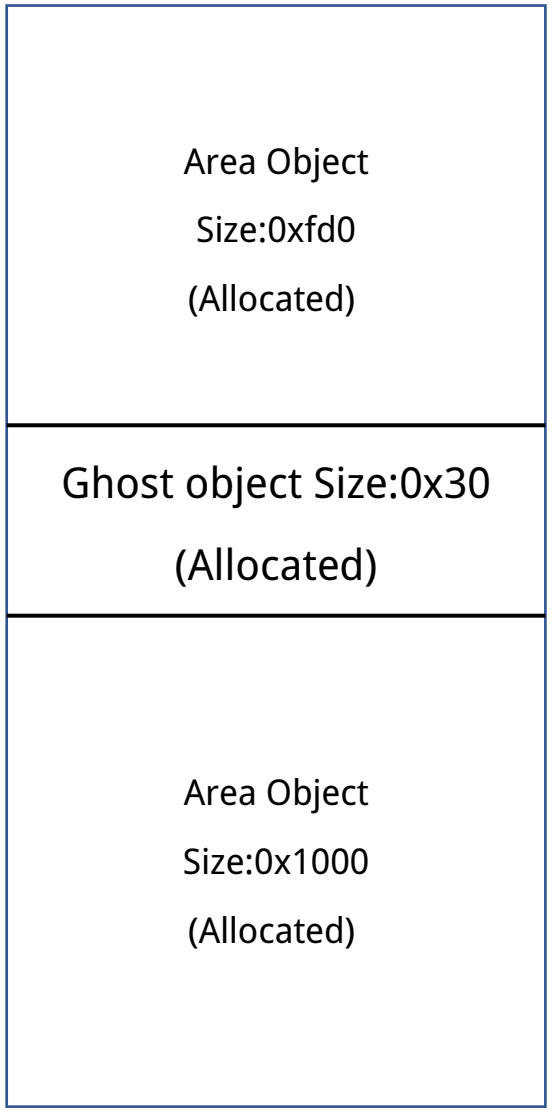
✓ Create some 0x30 session paged pool to fill lookaside list

✓ Trigger Integer Overflow to occupy one of 0x30 pool hole

```
kd> !pool ffffde0f`4a4a8fe0
Pool page ffffde0f4a4a8fe0 region is Unknown
 ffffde0f4a4a8000 size:  fd0 previous size:    0  (Allocated)  FpAr
*ffffde0f4a4a8fd0 size:   30 previous size:  fd0  (Allocated) *FpGh
                Owning component : Unknown (update pooltag.txt)
```

Area Object

Ghost Object



Exploitation

Step3

- We want R/W,so we need arbitrary write first
- EditArea seems a nice function

```
m_TargetLocationData = (PVOID)((DWORD64)temp_EALinkLink->Area_Entry + (DWORD64)temp_EALinkLink->Area_Location * 0x8);
RtlCopyMemory(m_TargetLocationData, &m_AreaEditBuffer->Area_Data, sizeof(DWORD64));
```

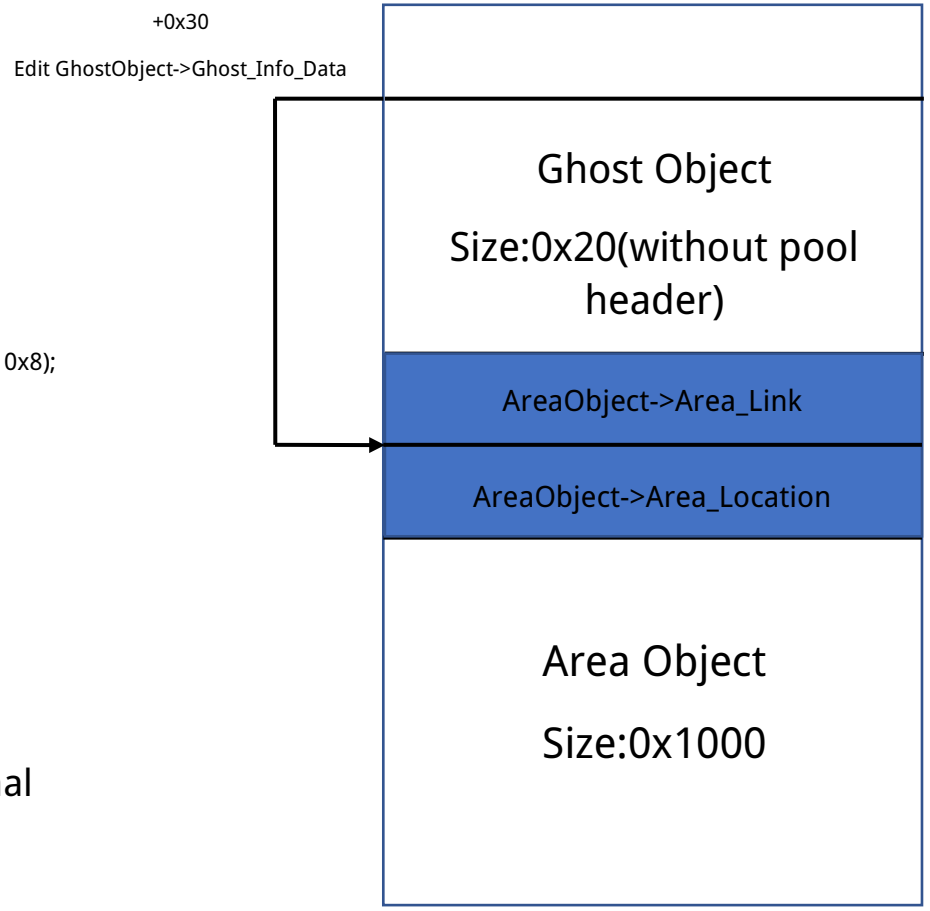
- Area_Location is kept with in bounds

```
if (m_AreaEditBuffer->Area_Location > temp_EALinkLink->Area_size / 0x8) {
    Status = 0xC000000D;
    return Status;
}
```

- But it depends on user input, if user set input Location to NULL, it will use original Location Value, or it will update to user input Area_Location Value

```
if (m_AreaEditBuffer->Area_Location) {
    temp_EALinkLink->Area_Location = m_AreaEditBuffer->Area_Location;
}
```

- So Trigger oob write to modify AreaObject->Area_Location, and EditArea with Area_Location input NULL!!(bypass!)




```
kd> dq ffffde0f4a4a9000
ffffde0f`4a4a9000  ffffde0f`4a4aa010 00000000`00000000
ffffde0f`4a4a9010  00000000`00000fc0 00000000`00000008
ffffde0f`4a4a9020  00000000`00000000 ffffde0f`4a4a9030
ffffde0f`4a4a9030  000
ffffde0f`4a4a9040  007
ffffde0f`4a4a9050  005
ffffde0f`4a4a9060  005
ffffde0f`4a4a9070  003
```

Area_Location

```
kd> dq ffffde0f4a4a9000
ffffde0f`4a4a9000  ffffde0f`4a4aa010 00000000`000003ff
ffffde0f`4a4a9010  00000000`00000fc0 00000000`00000008
ffffde0f`4a4a9020  00000000`00000000 ffffde0f`4a4a9030
ffffde0f`4a4a9030  00000000`00000000 80001244`434115ed
ffffde0f`4a4a9040  00720065`00760069 00610074`00610044
ffffde0f`4a4a9050  005c003a`0043003d 8000136e`434315eb
ffffde0f`4a4a9060  005c0073`0077006f 00740073`00790053
ffffde0f`4a4a9070  00320033`006d0065 8c001472`434515e9
```

After trigger oob write

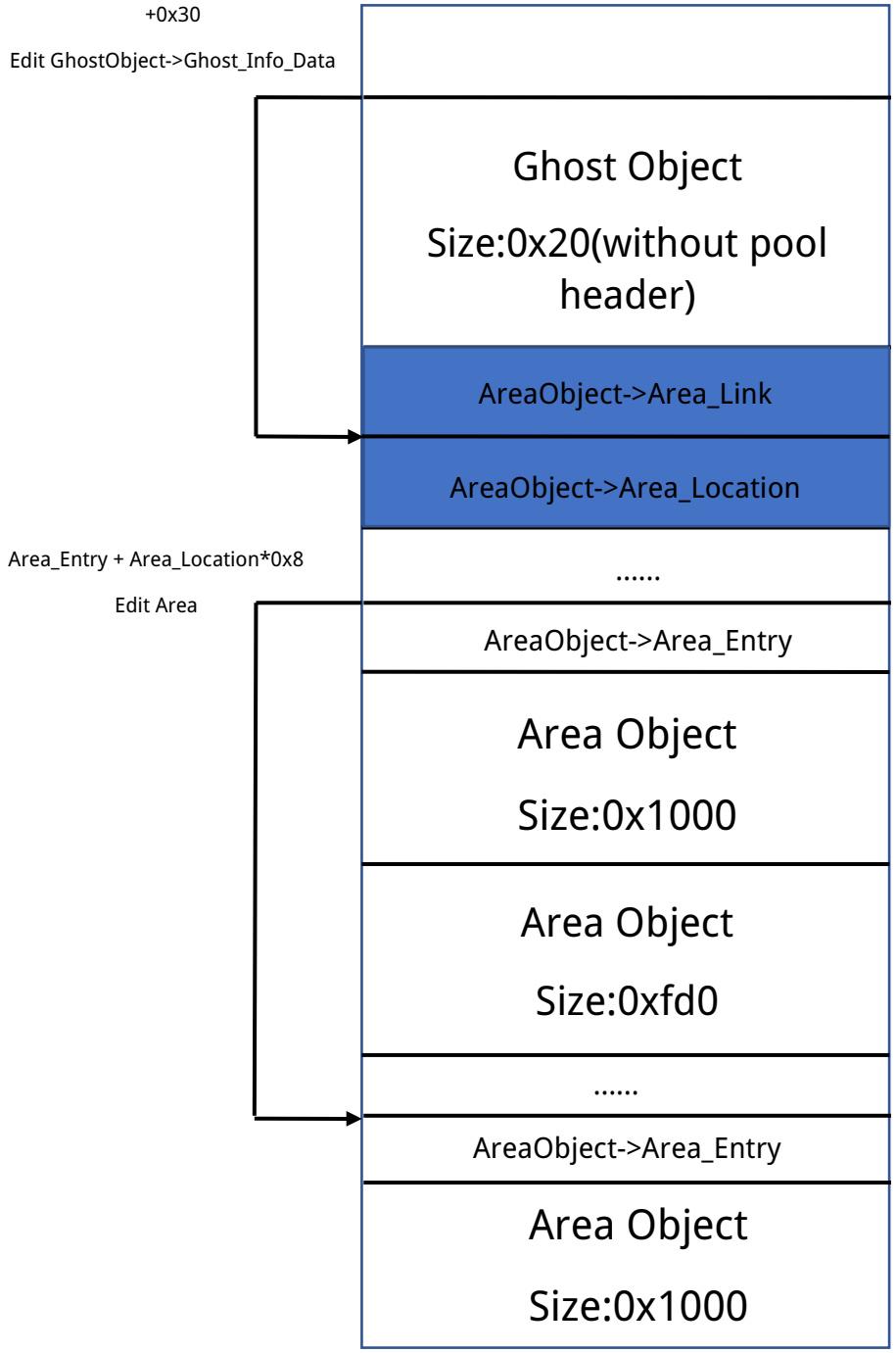
Area_Location

Exploitation

Step4

- We can trigger another oob write by Area_Location,

  to modify another AreaObject->Area_Entry to arbitrary address
- We get one chance to Arbitrary Write!



Exploitation

Step5

- We have one chance to arbitrary write,what can we do?
- Occupy pool header,but we can’t leak pool cookie
- Occupy process token or ACL or some other important variable,but we can’t leak token or some address soon
- We decide to R/W by GDI object!
- There is a Windows Kernel mitigation named Win32k typeisolation, and it isolate GDI header and data.

  (https://blog.quarkslab.com/reverse-engineering-the-win32k-type-isolation-mitigation.html)

- But 360Vulcan Team bypass it!

  (http://blogs.360.cn/blog/save-and-reborn-gdi-data-only-attack-from-win32k-typeisolation-2/)

Exploitation

Step6

- We can leak Ghost Object and Area Object kernel address
- Make a stable pool hole for typeisolation->CSectionEntry and leak it by Ghost Object (https://www.coresecurity.com/system/files/publications/2016/10/Abusing-GDI-Reloaded-ekoparty-2016_0.pdf)
- Create fake view and fake bitmap_allocator
- Trigger Arbitrary Write by EditArea() to occupy CSectionEntry->view and CSectionEntry->bitmap_allocator
- R/W!!
- Fix up(unlink, and clear win32k!gpHandleManager)

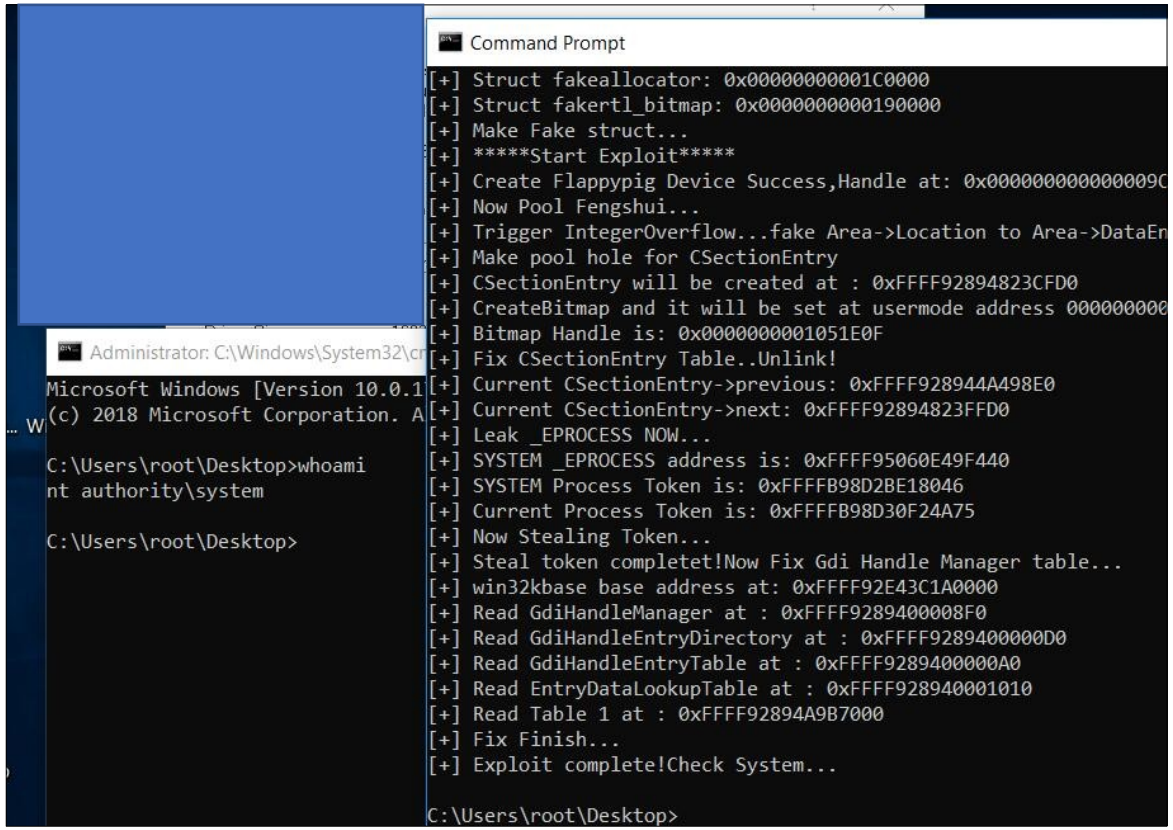
```
kd> dq ffffde0f4a4ab000
ffffde0f`4a4ab000  ffffde0f`4a4ac010 00000000`00000000
ffffde0f`4a4ab010  00000000`00000fc0 00000000`0000000a
ffffde0f`4a4ab020  00000000`00000000 ffffde0f`484c3fd0
ffffde0f`4a4ab030  00000000`00000000 80001244`434115ed
ffffde0f`4a4ab040  00720065`00760069 00610074`00610044
ffffde0f`4a4ab050  005c003a`0043003d 8000136e`434315eb
ffffde0f`4a4ab060  005c0073`0077006f 00740073`00790053
ffffde0f`4a4ab070  00320033`006d0065 8c001472`434515e9
kd> !pool ffffde0f`484c3fd0
Pool page ffffde0f484c3fd0 region is Unknown
 ffffde0f484c3000 size:  fc0 previous size:    0  (Allocated)  Uscb
*ffffde0f484c3fc0 size:   40 previous size:  fc0  (Allocated) *Uiso
                Pooltag Uiso : USERTAG_ISOHEAP, Binary : win32k!TypeIsolation::Create
```

Modify Area_Entry to CSectionEntry

CSectionEntry created by Win32kTypeisolation

Final

- Got System!



Q&A

Thanks!