



Unveiling the Cracks in Virtualization, Mastering the Host System

— VMware Workstation Escape

Speaker: VictorV

About Me : VictorV (@vv474172261)



Top 3 of MSRC
2023 Q3/Q4
Leaderboard

VMware
Workstation
Escape
TianfuCup
2018/2021/2023

Zer0Con 2022
HITB 2020

Bugs in
SQLServer, RDP,
QEMU, DNS,
DHCP, Samba,
ESXi...

Hyper-V Escape
CVE-2019-0887
In 2021

目录 CONTENTS

Virtualization Basic Info 01

Historic Bugs In UHCI 02

Exploit for TianfuCup 2023 03

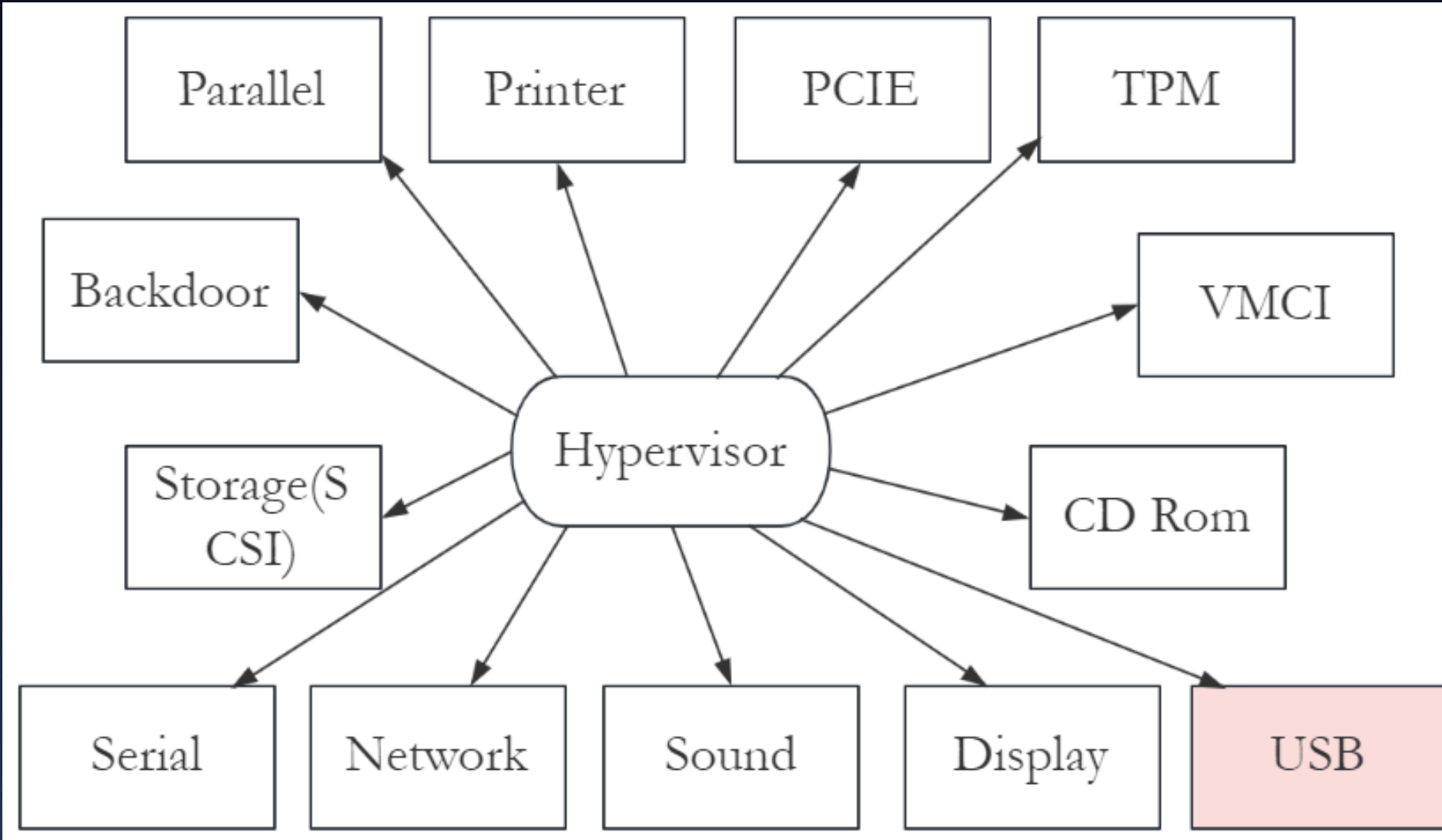
Summary 04

PART ONE

Virtualization Basic Info

Virtualization Basic Info

VMware Workstation Architecture



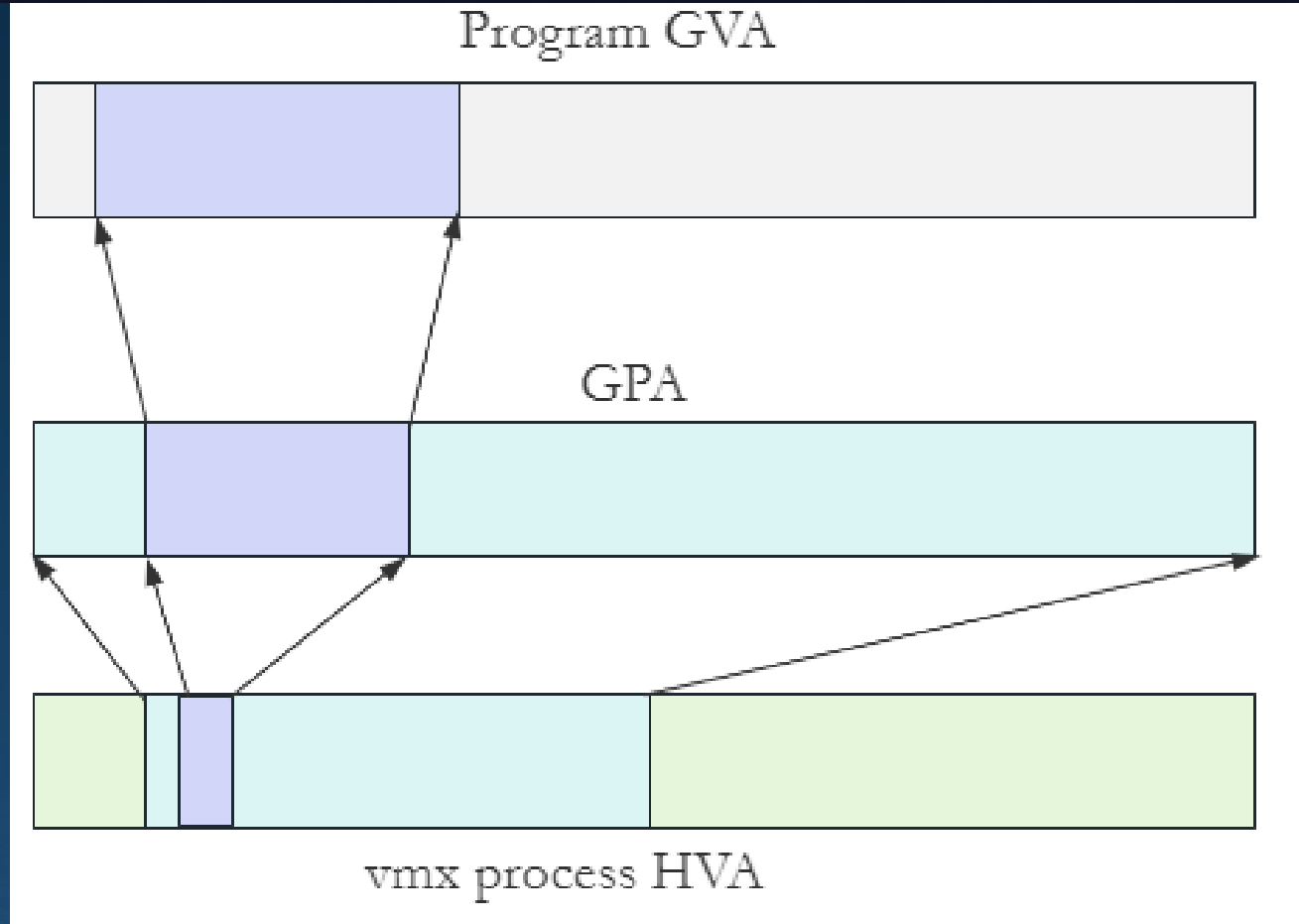
Virtualization Basic Info

Virtual Process Address and Guest Physical Address

Guest Virtual Address(GVA)
Guest Physical Address(GPA)
Host process Virtual Address(HVA)

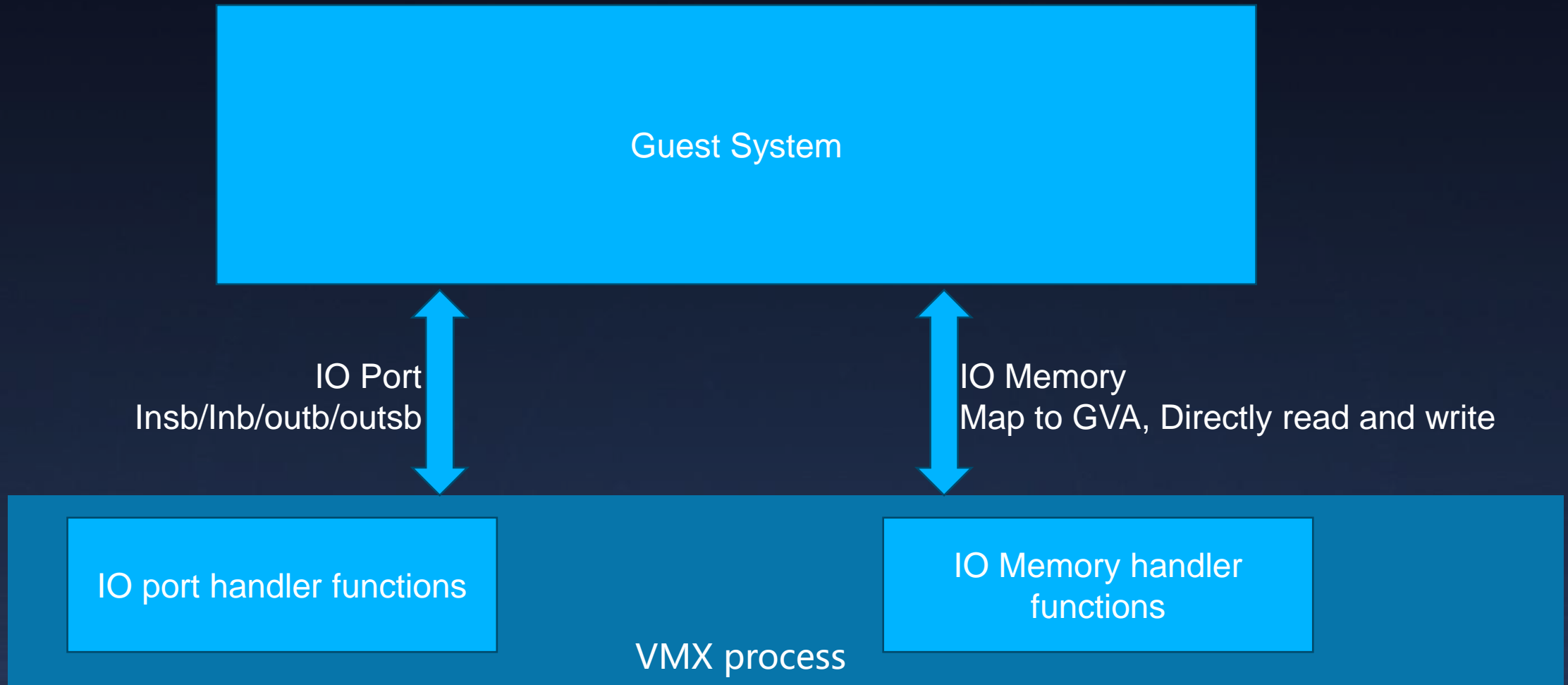
In Guest, use GVA access its
physical memory

In Host vmx, use HVA of GPA access
Guest memory



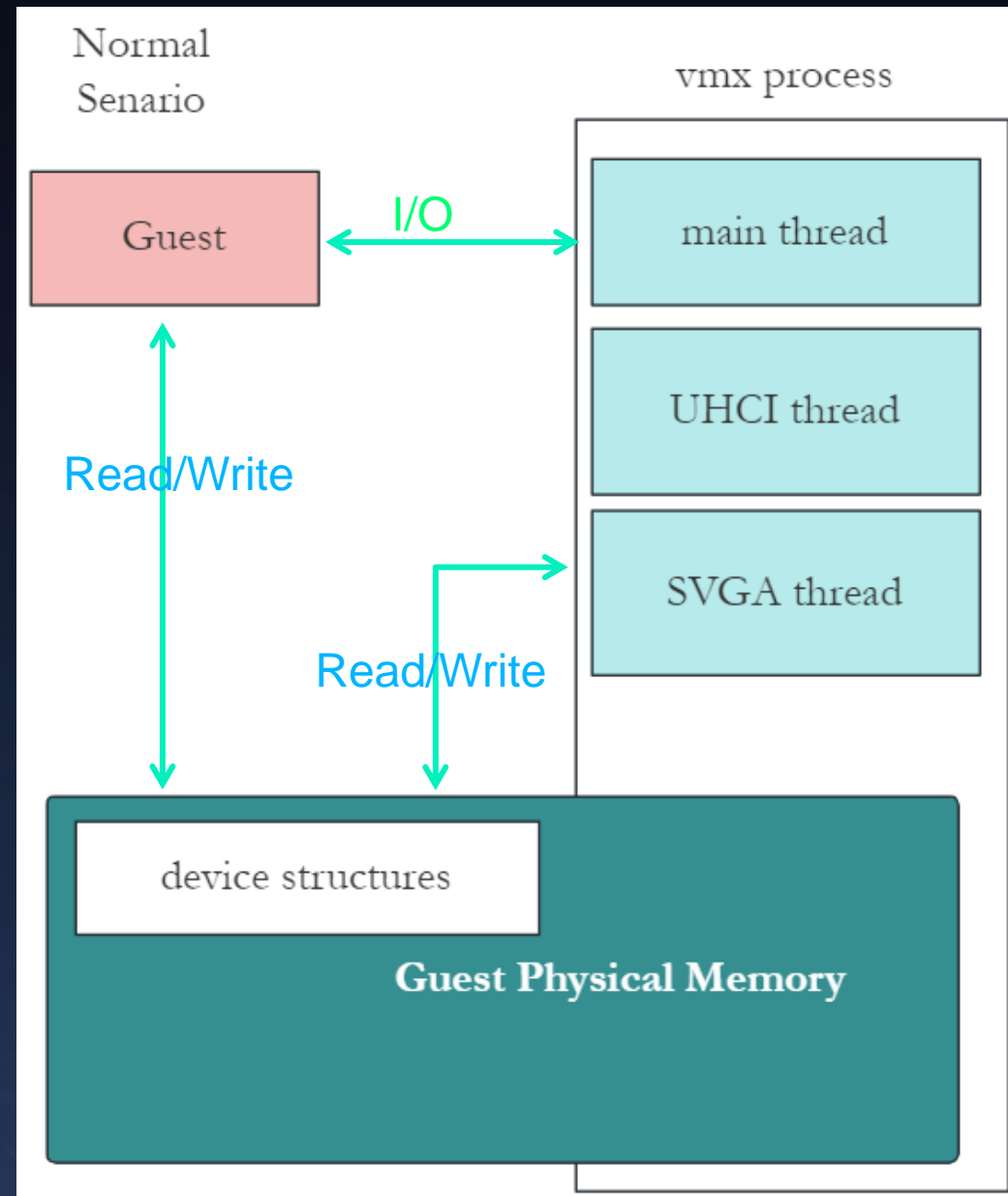
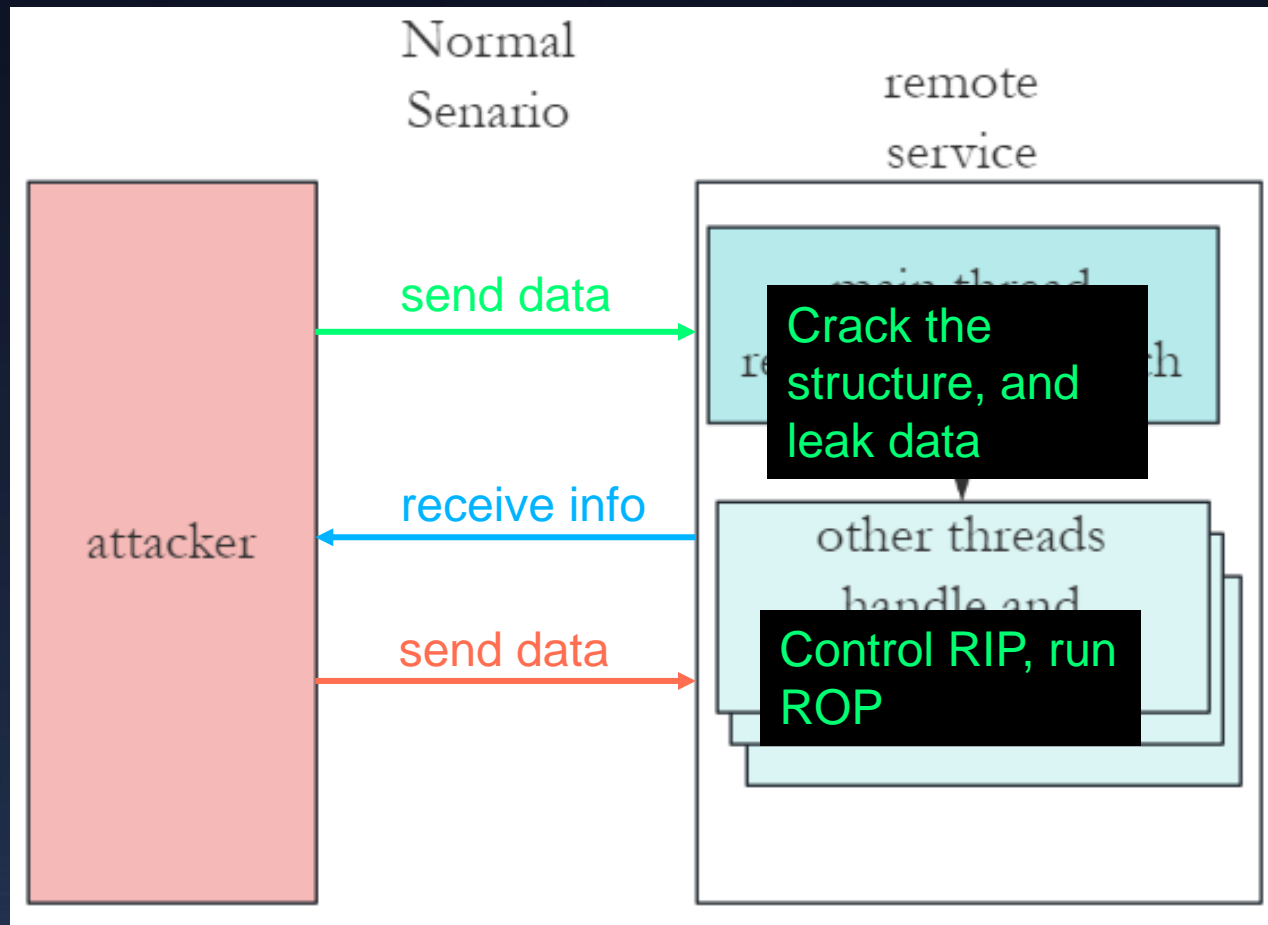
Virtualization Basic Info

Virtual Device and Guest Driver Interaction



Virtualization Basic Info

VM Escape and RCE exploit



Virtualization Basic Info

USB Controller



USB 1.x
UHCI

CVE-2021-22041
CVE-2019-5519
CVE-2019-5518
CVE-2023-20870 ...



USB 2.0
EHCI

CVE-2022-31705 ...



USB 3.x
XHCI

CVE-2024-22252
CVE-2021-22040
CVE-2020-4004
CVE-2020-3968
CVE-2017-4904 ...



USB 4.0
Future

#BHASIA @BlackHatEvents

Virtualization Basic Info

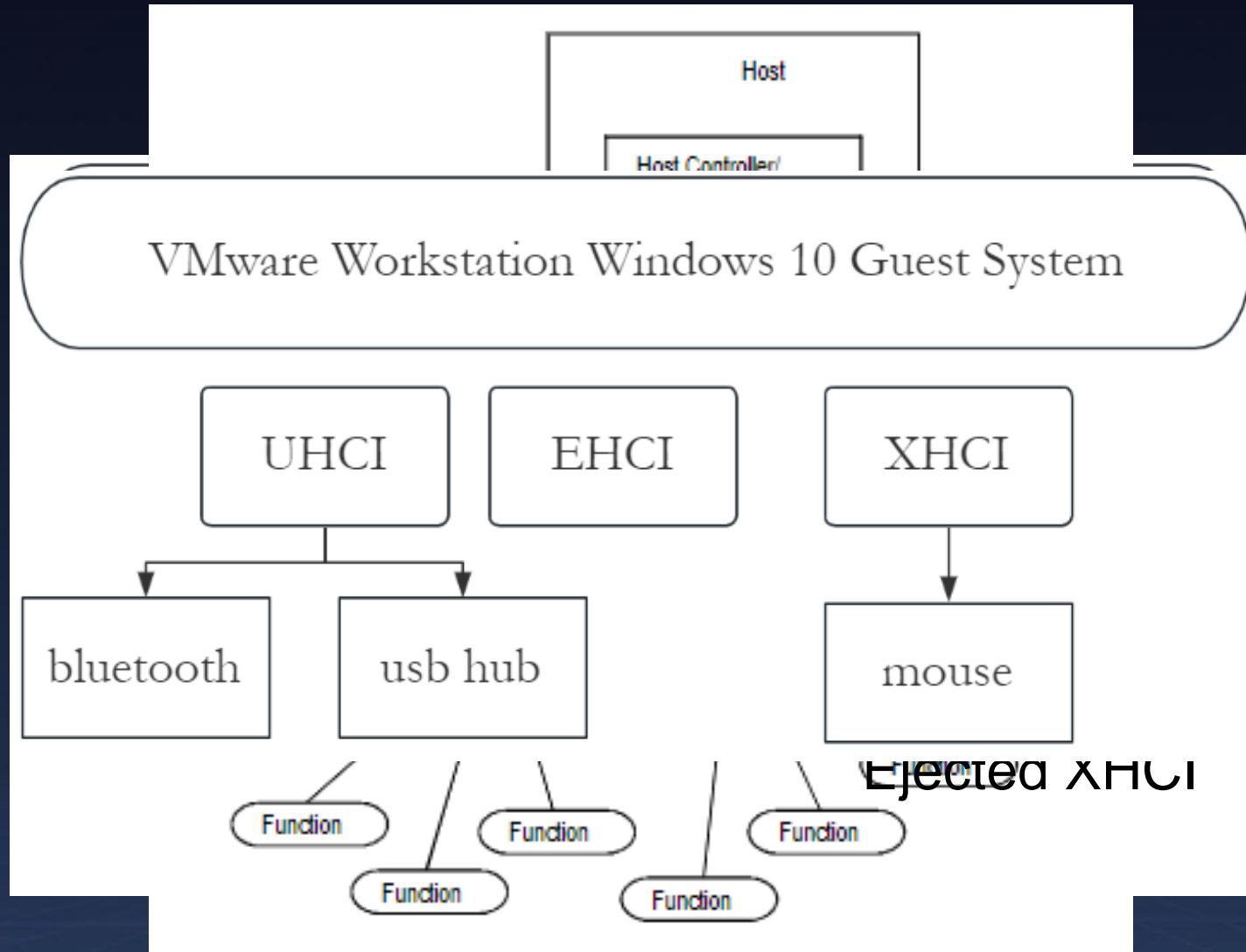
Virtual USB Controller Device Info

```
[root@localhost vv]# lspci -v -s 03:00.0
03:00.0 USB controller: VMware USB3 xHCI 1.0 Controller (prog-if 30 [XHCI])
  Subsystem: VMware USB3 xHCI 1.0 Controller
  Physical Slot: 160
  Flags: bus master, fast devsel, latency 64, IRQ 18
  Memory at fd400000 (64-bit, non-prefetchable) [size=128K]
  Capabilities: [64] Power Management version 3
  Capabilities: [6c] Express Endpoint, MSI 00
  Capabilities: [a8] MSI: Enable- Count=1/1 Maskable+ 64bit+
  Capabilities: [c0] MSI-X: Enable+ Count=31 Masked-
  Kernel driver in use: xhci_hcd

[root@localhost vv]# lspci -v -s 02:00.0
02:00.0 USB controller: VMware USB1.1 UHCI Controller (prog-if 00 [UHCI])
  Subsystem: VMware Device 1976
  Physical Slot: 32
  Flags: bus master, medium devsel, latency 64, IRQ 18
  I/O ports at 20c0 [size=32]
  Capabilities: [40] PCI Advanced Features
  Kernel driver in use: uhci_hcd
```

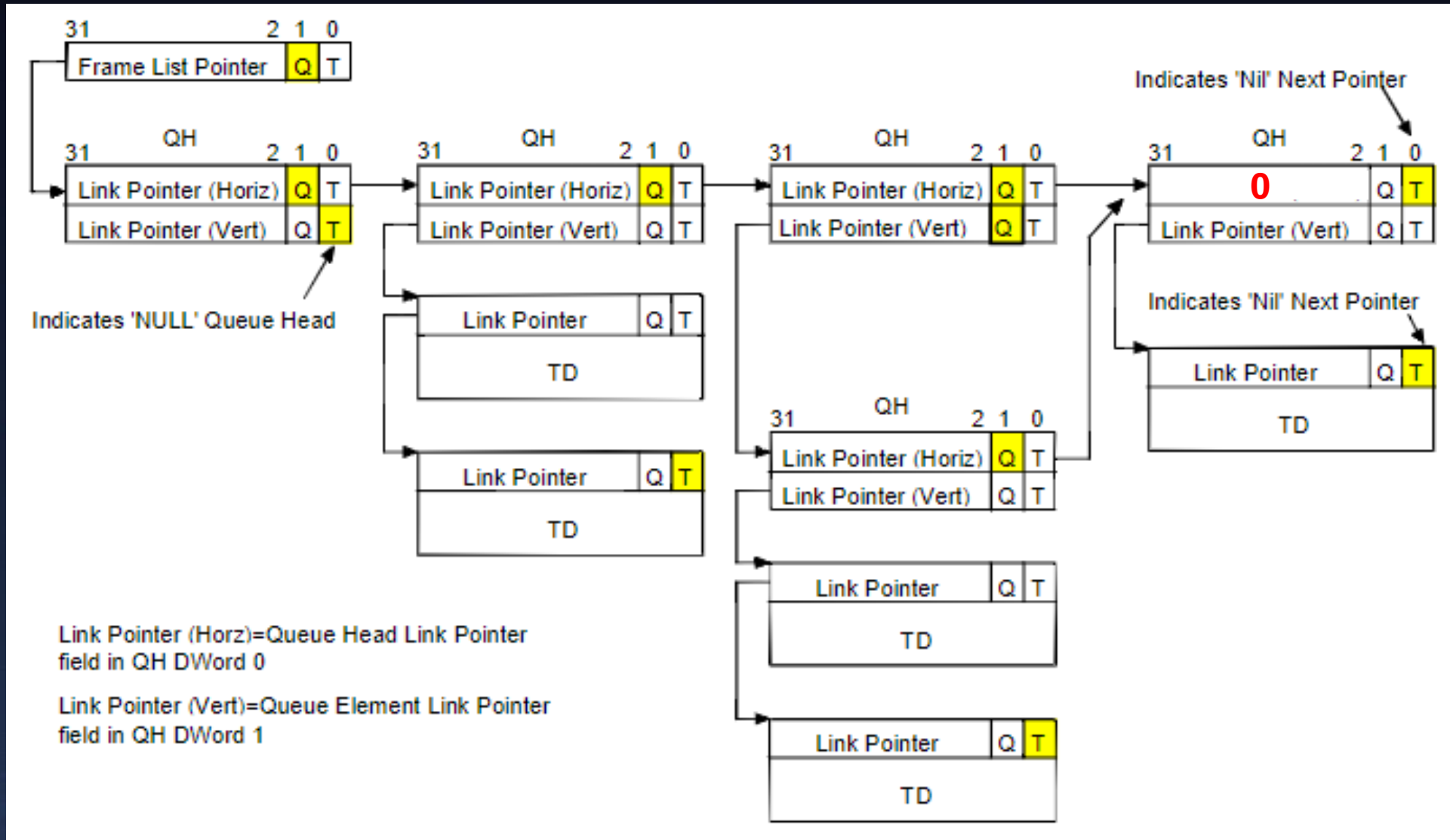
Virtualization Basic Info

UHCI Controller



Virtualization Basic Info

UHCI Controller



Virtualization Basic Info

UHCI Controller

```
u32 * TD = dmaAlloc(0x10, &TD_GPA);  
buffer = dmaAlloc(0x10, &buffer_GPA);
```

```
frame_list[0] = TD_GPA | 1;
```

```
TD[0] = 1; // end
```

```
TD[1] = 1 << 23; // active
```

```
TD[2] = (2 << 8) | (0 << 15) | (7 << 21) | 0x2d;
```

```
//dev_id: 2, ep_id: 0, length: 8(7+1), type: setup(0x2d)
```

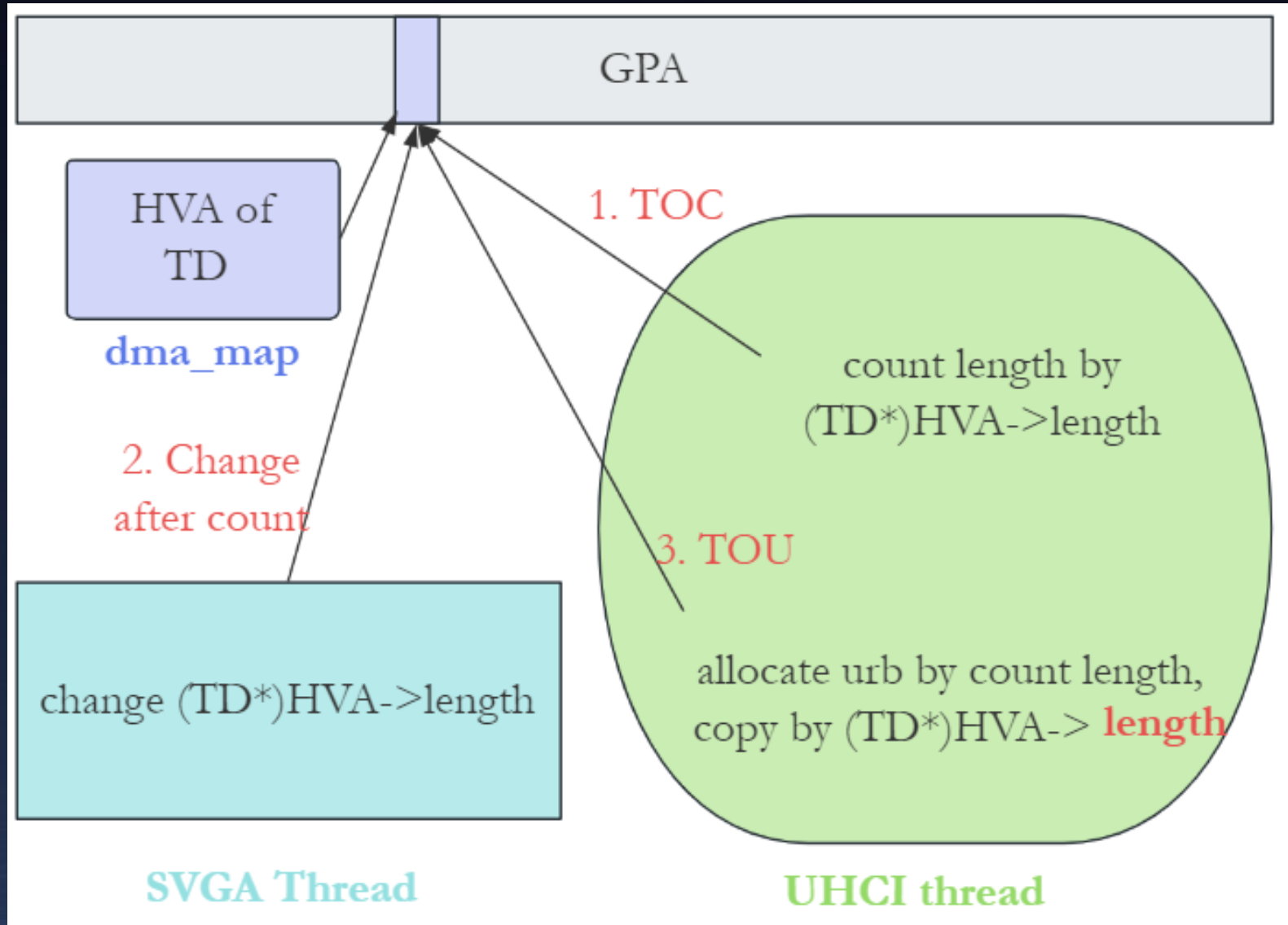
```
TD[3] = buffer_GPA;
```

```
buffer[0] = XXX;
```

PART TWO

Historic Bugs In UHCI

CVE-2019-5519 TOCTOU



CVE-2021-22041 TOCTOU

```
1  do{
2      while(uhci_prepare_qh_td(i++));
3  }while(i < 0x400)
4  uhci_prepare_qh_td(int index){
5      frame_index = (uhci_main->frame_start + index)&0x3ff;
6      frame_dma = uhci_main->frame_dma_addr + frame_index*4;
7      qh_td_dma = dma_map_read(frame_dma, 4);
8      ret = !frame_index || frame_index == 0x3ff;
9      while(...){
10 next:
11     if(no next TD or QH){
12         return ret;
13     }
14     qh_td_map = dma_map(uhci_main, qh_td_dma);
15     if(qh_td_map->transfer_tag == uhci_main->transfer_tag){
16         goto next;
17     }
18     qh_td_map->transfer_tag = uhci_main->transfer_tag;
19     ep = uhci_main->usb_dev[dev_id][ep_id];
20     if(ep->type == 1){
21         if(!ep->map_array){
22             ep->map_array = calloc(0x400, sizeof(TD));
23         }
24         ep->map_array[index] = *(TD*)qh_td_map->real_addr;
25         if(index < 0x400) ret = 1;
26     }
```

If frame_start = 0x3ff,
i=0x400,

frame_index =
(0x400+0x3ff)&0x3ff = 0x3ff;

ret = 1

frame[(0+0x3ff)&0x3ff] ==
frame[(0x400+0x3ff)&0x3ff]

transfer_tag will match

CVE-2021-22041 TOCTOU

```
1  do{
2      while(uhci_prepare_qh_td(i++));
3  }while(i < 0x400)
4  uhci_prepare_qh_td(int index){
5      frame_index = (uhci_main->frame_start + index)&0x3ff;
6      frame_dma = uhci_main->frame_dma_addr + frame_index*4;
7      qh_td_dma = dma_map_read(frame_dma, 4);
8      ret = !frame_index || frame_index == 0x3ff;
9      while(...){
10 next:
11     if(no next TD or QH){
12         return ret;
13     }
14     qh_td_map = dma_map(uhci_main, qh_td_dma);
15     if(qh_td_map->transfer_tag == uhci_main->transfer_tag){
16         goto next;
17     }
18     qh_td_map->transfer_tag == uhci_main->transfer_tag;
19     ep = uhci_main->usb_dev[dev_id][ep_id];
20     if(ep->type == 1){
21         if(!ep->map_array){
22             ep->map_array = calloc(0x400, sizeof(TD));
23         }
24         ep->map_array[index] = *(TD*)qh_td_map->real_addr;
25         if(index < 0x400) ret = 1;
26     }
```

1. Access frame[0x3ff]

2. Change frame[0x3ff] in SVGA thread

3. Access frame[(0x400+0x3ff)&0x3ff]
again
Get a new GPA

CVE-2023-20870 Uninitialize Leak

```
1 uhci_handle_type0_td(){
2     while(i < ep->map_max){
3         TD = ep->map_array[i];
4         length = TD->length;
5         if(TD->type != 0x69 && (TD->length&0x7ff) != 0){
6             td_buff = dma_map(TD->buff_dma);
7         }
8         if(TD->type == 0x2d){// setup
9             if(urb){
10                uhci_handle_urb(urb)
11                goto next_handle;
12            }
13            if(length != 8) goto fail;
14            size = td_buff->real_addr[0]+length;
15            urb = allocate_urb(size);// malloc
16            urb->count_size = size;
17            memcpy(urb->cur_buff, td_buff, length);
18            urb->cur_buff += length;
19        }else if(TD->type == 0xe1){// OUT
20            if(length > size) goto fail;
21            memcpy(urb->cur_buff, td_buff, length);
22            urb->cur_buff += length;
23            size -= length;
24        }else if(TD->type == 0x69){// IN
25            size -= length;
26        }else {goto fail;}
27    }
28    if(urb) uhci_handle_urb(urb)
```

```
struct urb{
    +0h reference;
    +4h buffer size;
    +8h count size;
    +Ch size can read to vm; default 0
    ...
    +18h endpoint;
    ...
    +78h buffer start;
    +80h cur_buff;
    char buffer[xxx]; size is determined by input size
}
```

```
18 ep = urb->usb_endpoint_18h;
19 len = urb->count_transaction_len_8h;
20 urb_about = *(bluetooth_urb_buff **) &urb[-1].packe
21 pkt = urb->p_self_packet_78h;
22 dev = ep->usb_dev_20h;
23 f_260h = (usb_bluetooth_main *)dev->f_230h.f_260h;
24 urb->_ret_status_de_ne1_58h = 0;
25 urb->size_for_vm_to_read_Ch = len; // u
```

CVE-2023-20870 Uninitialize Leak

```
1 uhci_handle_type0_td(){
2     while(i < ep->map_max){
3         TD = ep->map_array[i];
4         length = TD->length;
5         if(TD->type != 0x69 && (TD->length&0x7ff) != 0){
6             td_buff = dma_map(TD->buff_dma);
7         }
8         if(TD->type == 0x2d){// setup
9             if(urb){
10                uhci_handle_urb(urb)
11                goto next_handle;
12            }
13            if(length != 8) goto fail;
14            size = td_buff->real_addr[0]+length;
15            urb = allocate_urb(size);// malloc
16            urb->count_size = size;
17            memcpy(urb->cur_buff, td_buff, length);
18            urb->cur_buff += length;
19        }else if(TD->type == 0xe1){// OUT
20            if(length > size) goto fail;
21            memcpy(urb->cur_buff, td_buff, length);
22            urb->cur_buff += length;
23            size -= length;
24        }else if(TD->type == 0x69){// IN
25            size -= length;
26        }else {goto fail;}
27    }
28    if(urb) uhci_handle_urb(urb)
```

Fix:

Set urb->Ch = 8 in Bluetooth handler

```
18 ep = urb->usb_endpoint_18h;
19 len = urb->count_transaction_len_8h;
20 urb_about = *(bluetooth_urb_buff **) &urb[-1].packe
21 pkt = urb->p_self_packet_78h;
22 dev = ep->usb_dev_20h;
23 f_260h = (usb_bluetooth_main *)dev->f_230h.f_260h;
24 urb->_ret_status_de_ne1_58h = 0;
25 urb->size_for_vm_to_read_Ch = len; // u
```

CVE-2024-22255 Uninitialize Leak

```
58 case 9:
59     if ( v7 < 0 || (v7 & 0x1F) != 1 )
60         goto LABEL_26;
61     v9 = *(void (__fastcall **)(_QWORD, _QWORD, _QWORD, _BYTE *, int
62     if ( v9 )
63         v9(
64             *(unsigned __int16 *)(v5 + 4),
65             HIBYTE(*(unsigned __int16 *)(v5 + 2)),
66             (unsigned __int8)*(_WORD *)(v5 + 2),
67             v8,
68             *(_DWORD *)(a1 + 8) - 8);
69     break;
```

```
24     v6 = *(_DWORD *) (a1 + 8) - 8;
```

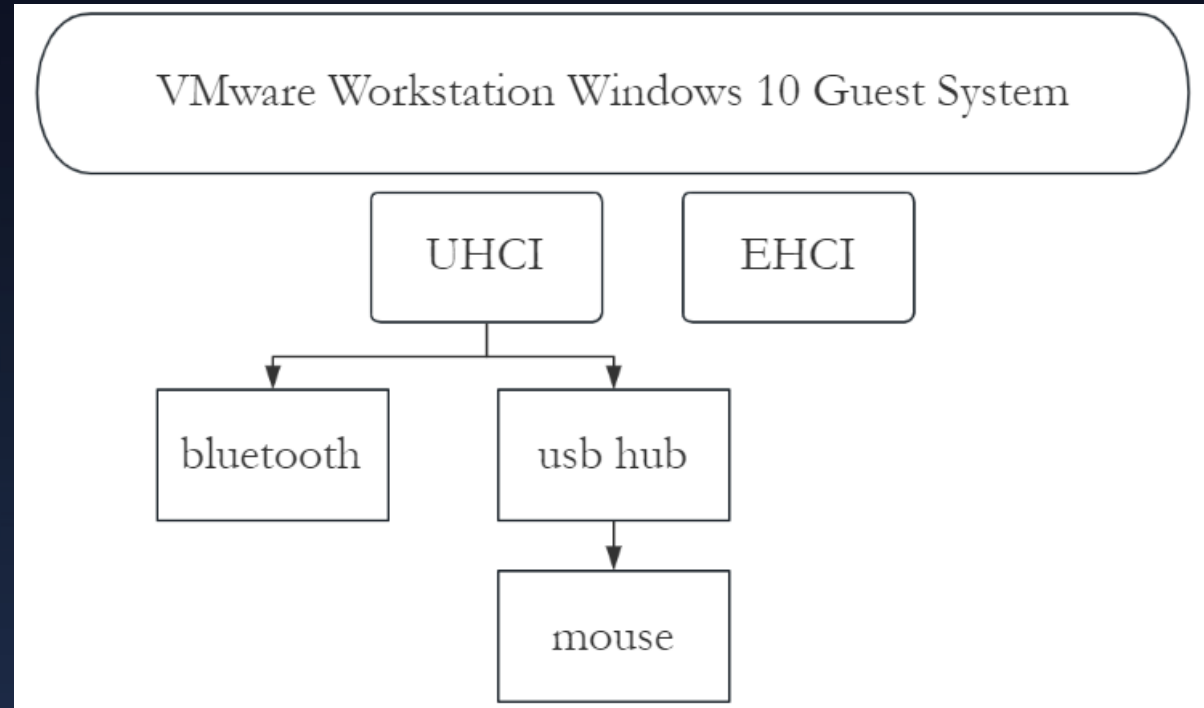
```
79     if ( v6 >= 0 )
80     {
81         *(_DWORD *) (a1 + 88) = 0;
82         *(_DWORD *) (a1 + 12) = v6 + 8;
```

```
struct urb{
    +0h reference;
    +4h buffer size;
    +8h count size;
    +Ch size can read to vm; default 0
    ...
    +18h endpoint;
    ...
    +78h buffer start;
    +80h cur_buff;
    char buffer[xxx]
}
```

U8(buffer, 0) = 0x21;
U8(buffer, 1) = 9; // CASE
U16(buffer, 6) = buffer size - 8;

CVE-2024-22253 UAF

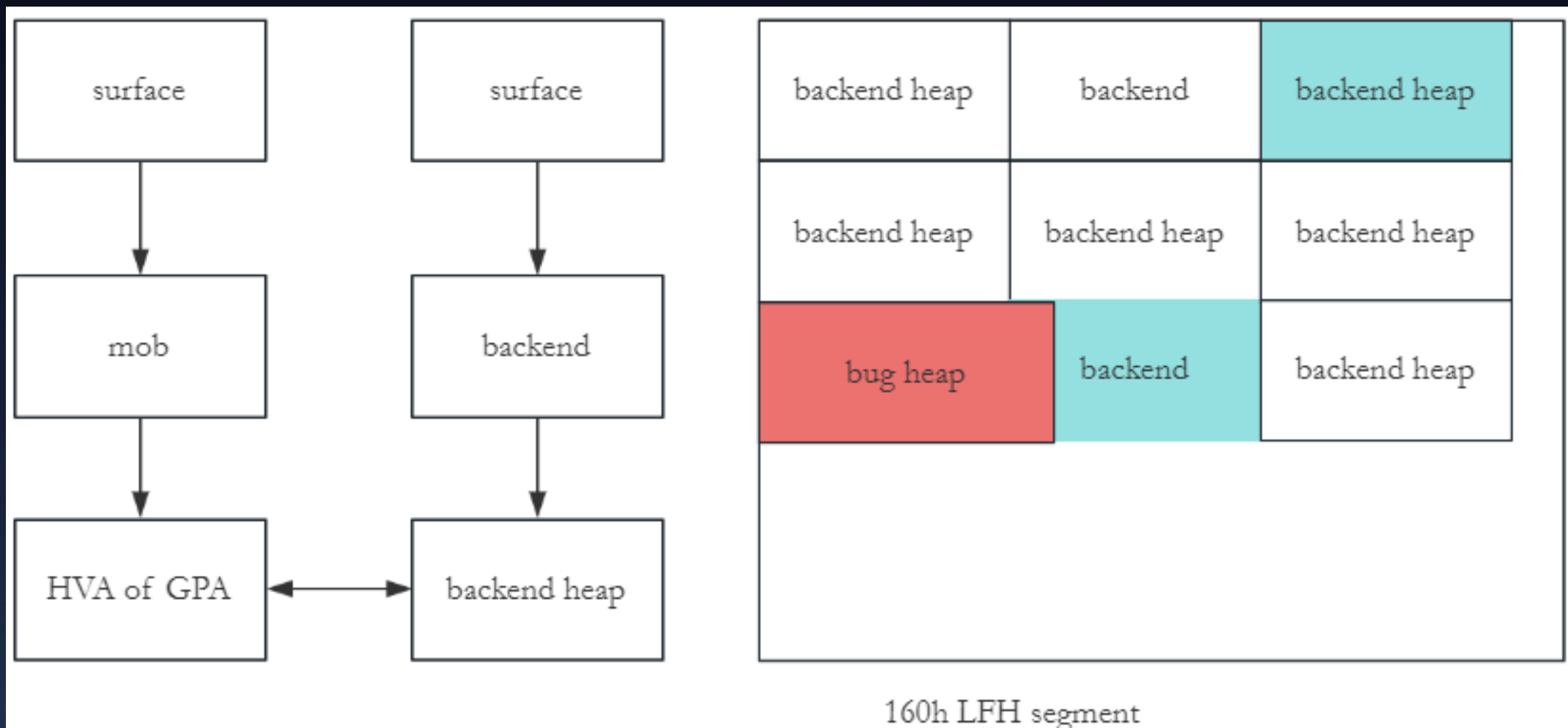
```
1  uhci_handle_TDs(){
2      next = uhci_main->ep0_link.next;
3      while(next != &uhci_main->ep0_link){
4          ep = next-18;
5          next = ep->link_uhci_main.next;
6          if(uhci_handle_in_urb(ep))
7              break
8          uhci_handle_type0_td(ep)
9          if(uhci_handle_in_urb(ep))
10             }
11     }
12     usb_hub_reset_sub_port(sub_dev){
13         reset_usb_dev(sub_dev){
14             free_all_endpoints(dev){
15                 for(;ep = dev->eps[i]; i++){
16                     free_pending_urbs(ep);
17                     unlink ep->link_uhci_main;
18                     dev->eps[i] = 0;
19                     free(ep);
20                 }
21             }
22             dev->ep[0] = new_ep();
23         }
24     }
```



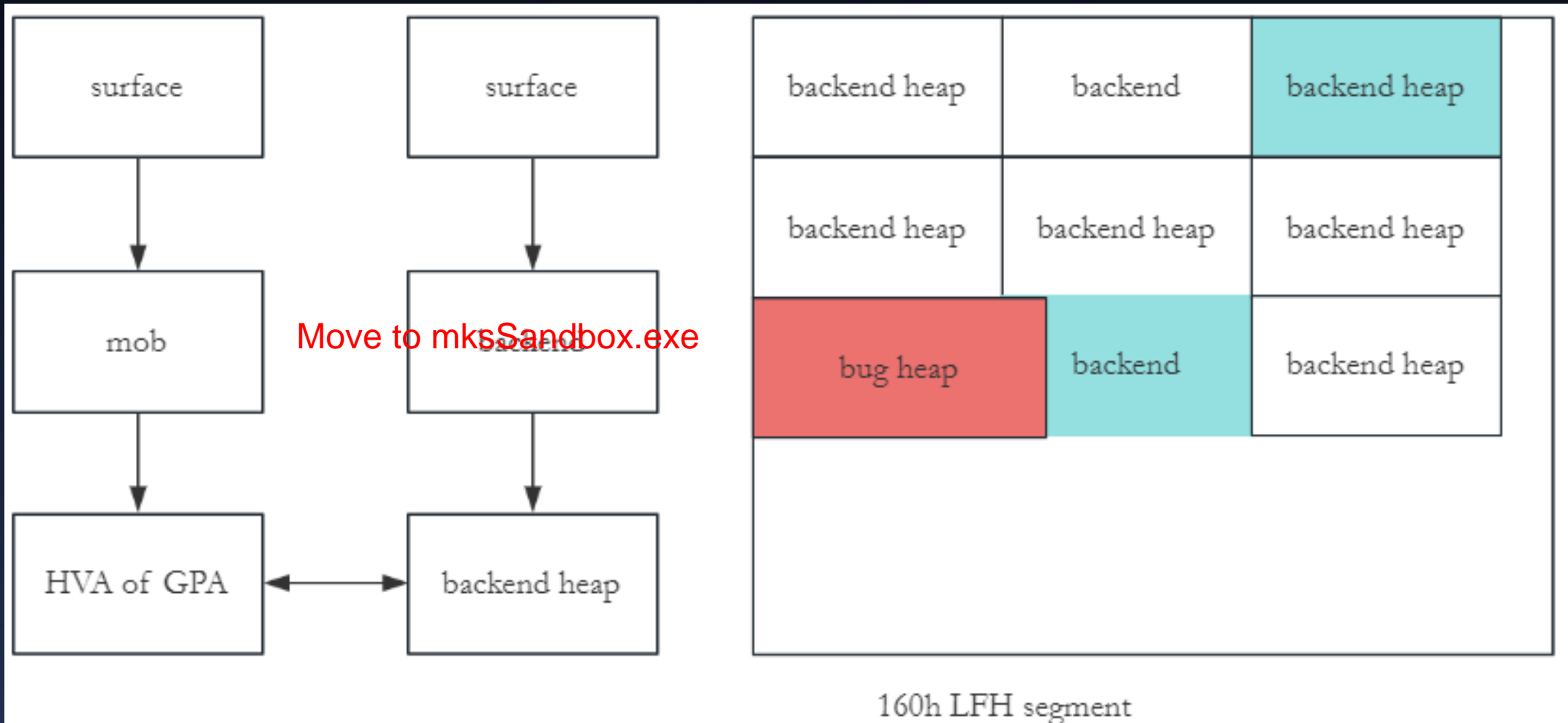
PART THREE

Exploit for TianfuCup 2023

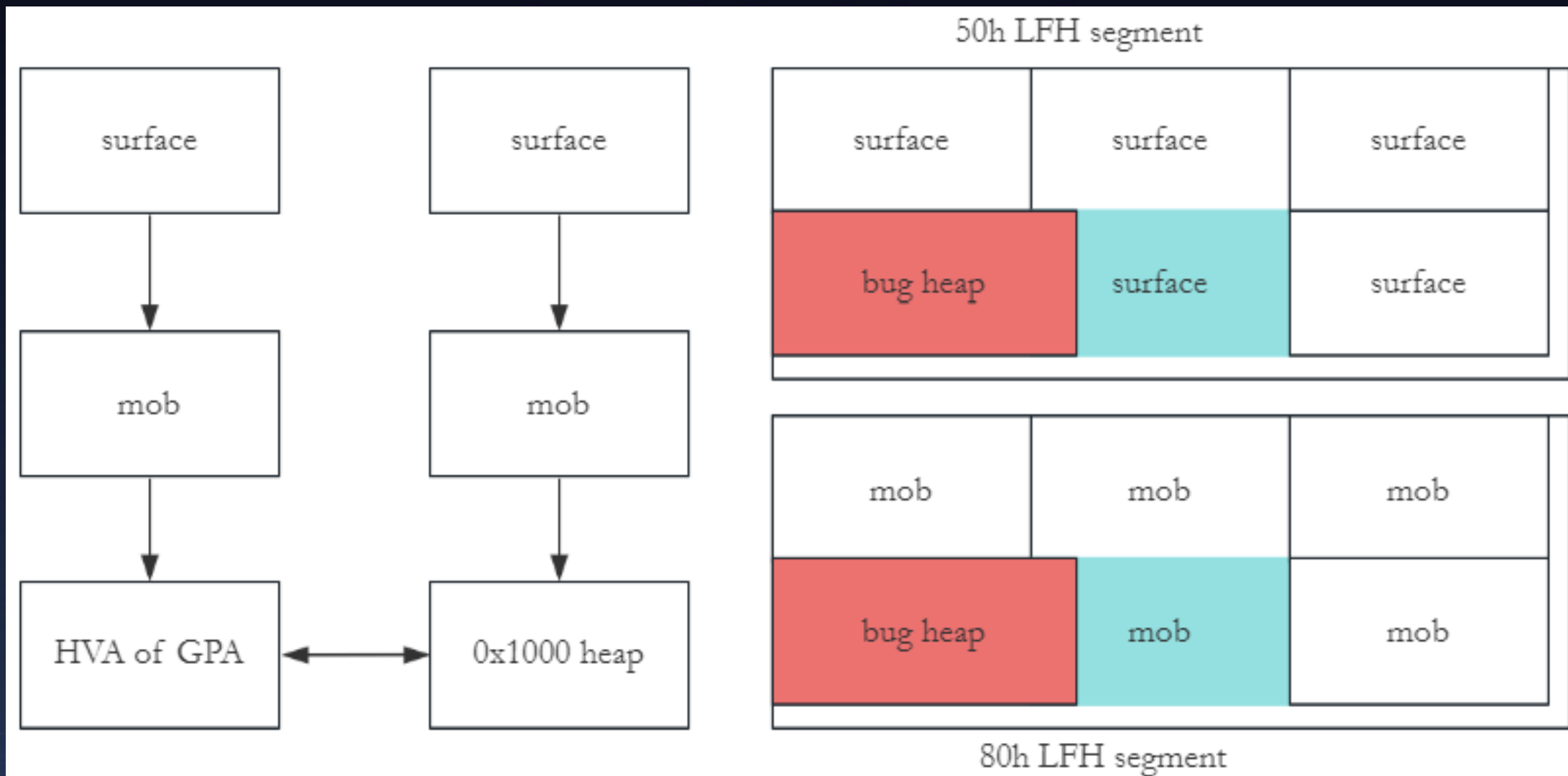
Old Exploit primitives-Straight outta VMware



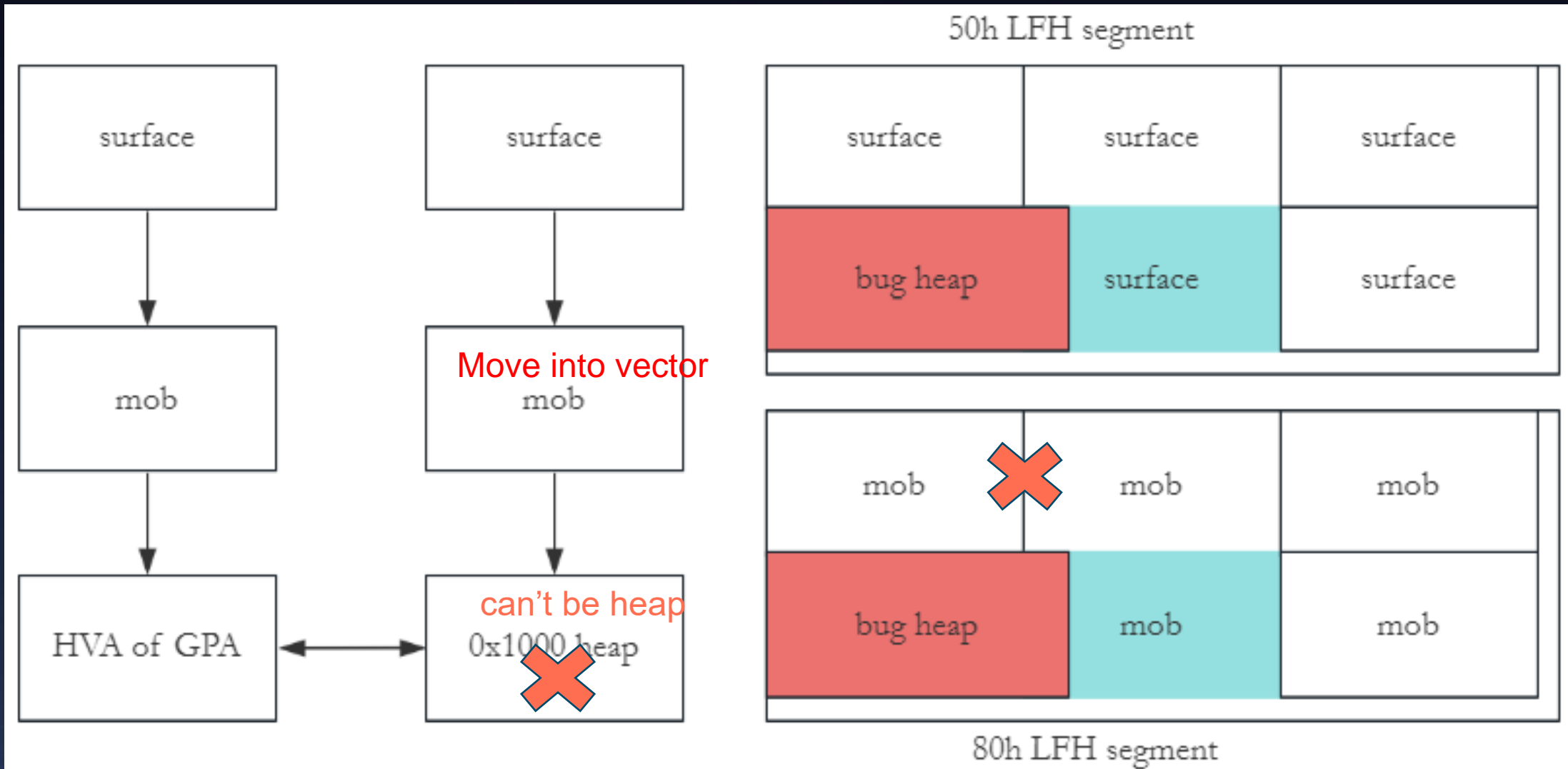
Old Exploit primitives-Straight outta VMware



Old Exploit primitives-Breakout Script of the Westworld



Old Exploit primitives-Breakout Script of the Westworld

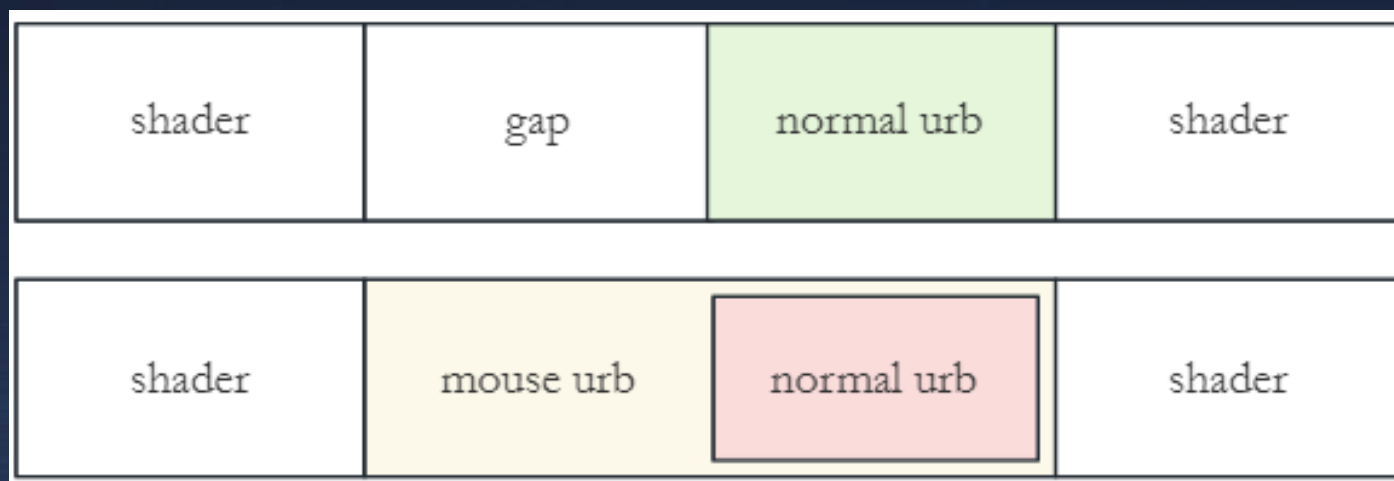


Exploit primitives-UHCI Endpoint

```
1  {
2  |   urb_link = (a2 + 64);
3  |   do{
4  |       urb = *urb_link - 40;
5  |       ep = *(urb + 24);
6  |       while(1){
7  |           v8 = v7 + *(ep + 104);
8  |           v9 = *(v8 + 8);
9  |           if(!v9 || (*(v9 + 1) & 0xffffffff) != **v8)
10 |              goto next;
11 |           *(*v8 + 8) + 4 = **(*v8 + 8);
12 |       }
13 |   }while(xx)
14 }
```

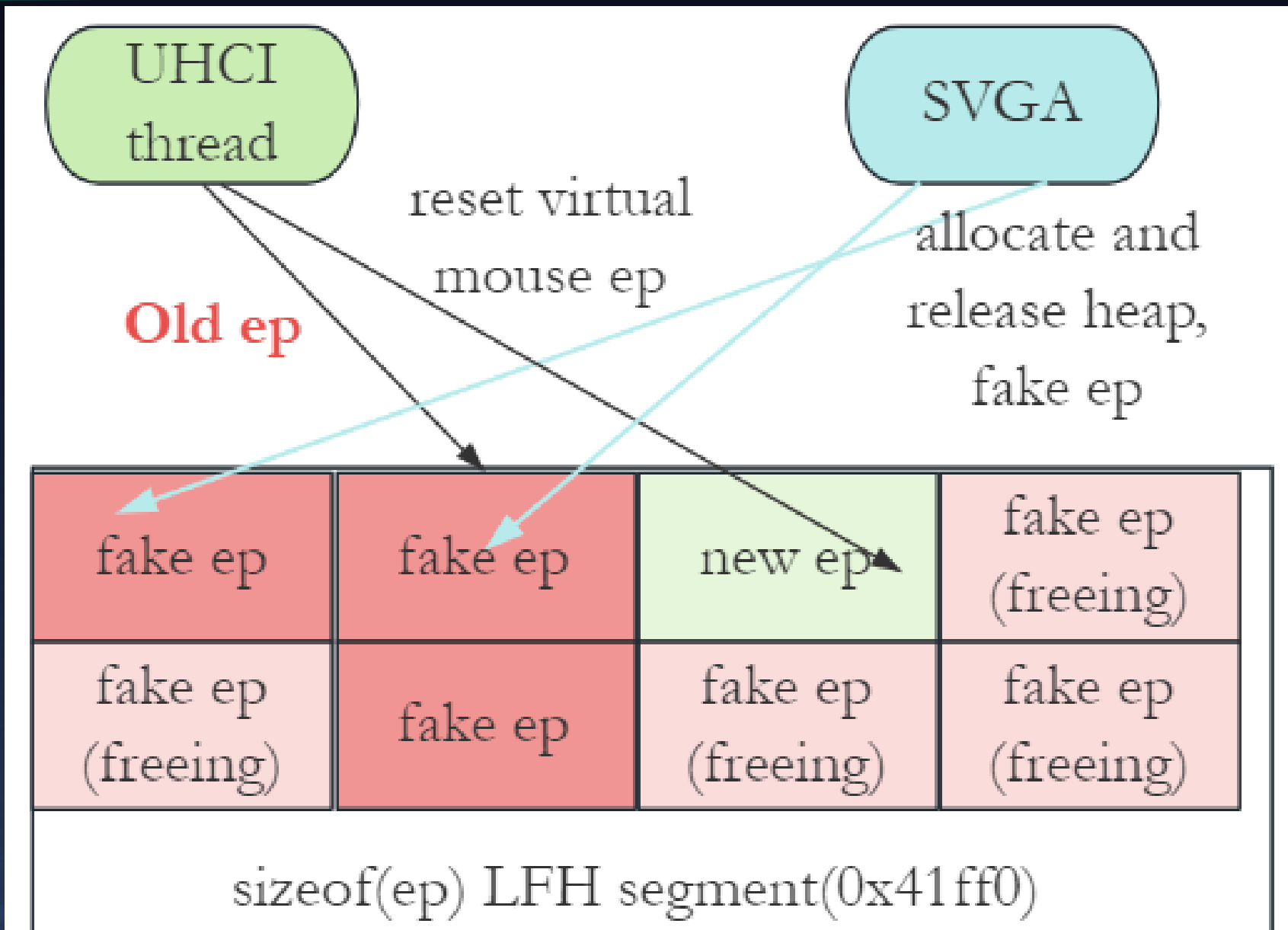
Exploit primitives-Leak address by Urb bug

```
struct urb{  
    ...  
    +70h vmx related process address  
    +78h buffer start;  
    +80h cur_buff;  
    char buffer[xxx]; size is determined by input  
    size  
}
```

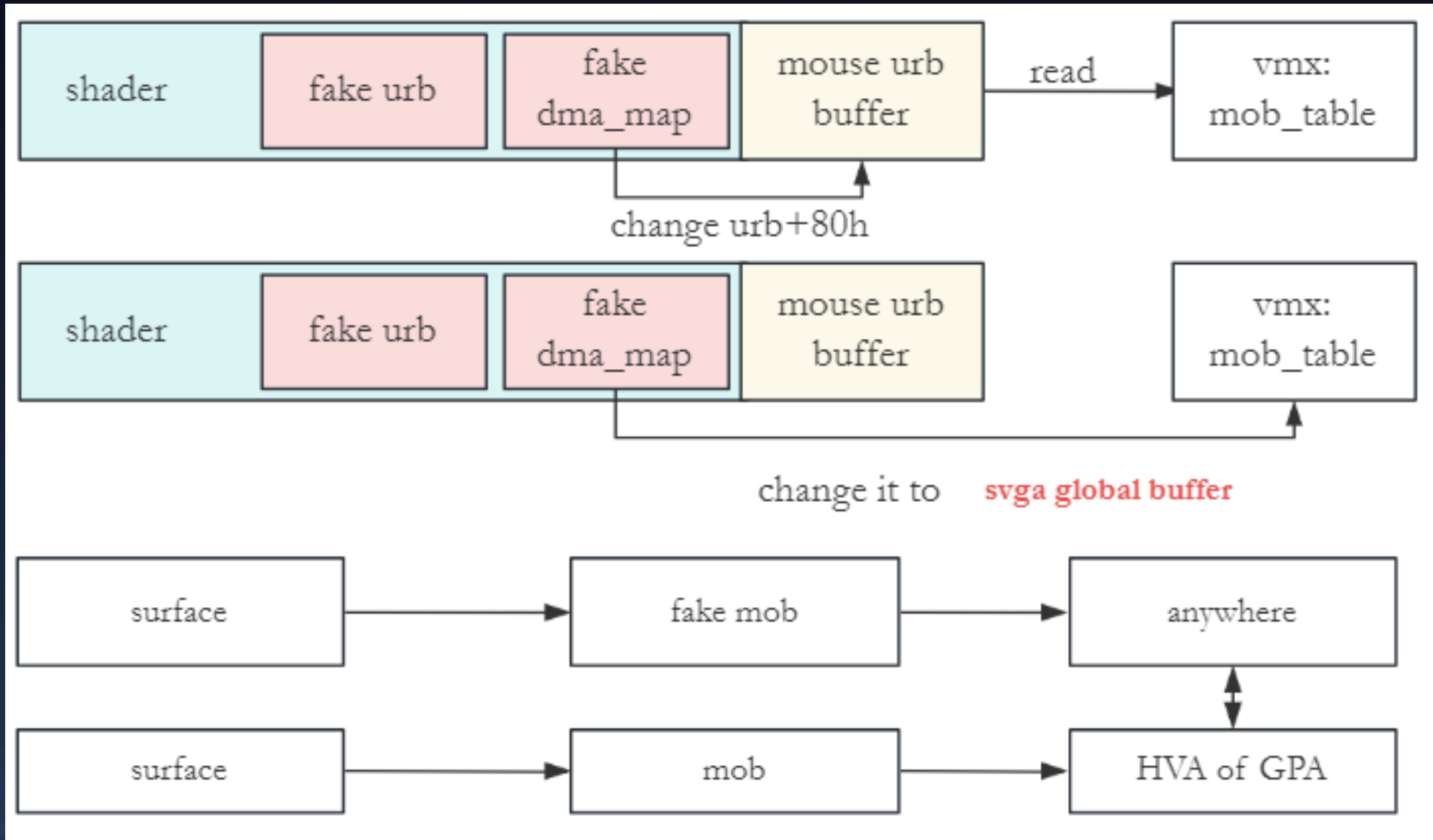


Get a urb heap address
Get VMX related address

Exploit primitives-R/W Everywhere by Endpoint primitive and urb



Exploit primitives-R/W Everywhere by Endpoint primitive and urb



Exploit Demo



PART FOUR

Summary

Black Hat Sound Bytes

Bug



Bug Research Tips

- TOCTOU, data of HVA can complete
- UAF, Notice reset operation, similar bug: cve-2020-4004

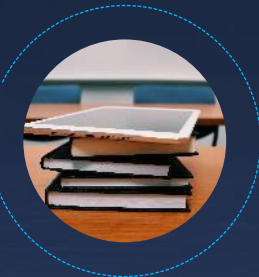
Exp



Exploit Tips

- Urb to leak data
- Endpoint to write arbitrary anywhere

Defense



Defense Escape Attack

- Remove unnecessary virtual devices: Usb, Sound, CDrom
- Disable SVGA 3D
- Keep your software newest

<https://census-labs.com/media/straightouttavmware-wp.pdf>

[Zero Day Initiative — Taking Control of VMware Through the Universal Host Control Interface: Part 2](#)

<https://github.com/474172261/slides/blob/main/Breakout%20Script%20of%20the%20Westworld-new%5B1088%5D.pdf>

[Universal Host Controller Interface \(UHCI\) Design Guide](#)

Q&A

THANKS