
ISC 2020
第八届互联网安全大会
360互联网安全中心
Qemu-kvm和ESXi虚拟机
逃逸实例分享
分享人：肖伟
ISC 2020
第八届互联网安全大会
INTERNET SECURITY CONFERENCE 2020
数字孪生时代下的新安全
New Security in the Digital Twin Era

个人简介

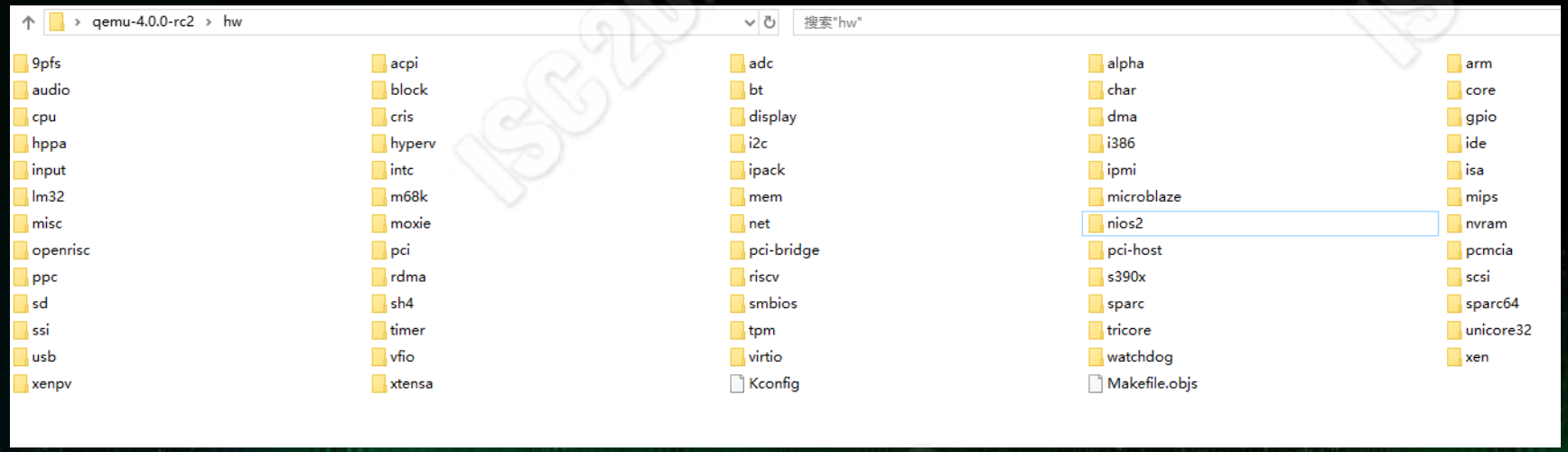
肖伟

360 VulcanTeam 漏洞挖掘与利用高级专家.

虚拟化安全研究员,多次完成vmware,qemu,virtualbox虚拟机逃逸.

Qemu简介

Qemu是一款开源的虚拟化软件,用代码模拟了许多常用的硬件设备,如网卡、显卡、声卡等。



漏洞介绍

```
//qemu-4.0.0-rc2\hw\usb\core.c
static void do_token_setup(USBDevice *s, USBPacket *p)
{
    int request, value, index;

    if (p->iov.size != 8) {
        p->status = USB_RET_STALL;
        return;
    }

    usb_packet_copy(p, s->setup_buf, p->iov.size);
    s->setup_index = 0;
    p->actual_length = 0;
    s->setup_len   = (s->setup_buf[7] << 8) | s->setup_buf[6];
    if (s->setup_len > sizeof(s->data_buf)) {
        fprintf(stderr,
                "usb_generic_handle_packet: ctrl buffer too small (%d > %zu)\n",
                s->setup_len, sizeof(s->data_buf));
        p->status = USB_RET_STALL;
        return;
    }

    request = (s->setup_buf[0] << 8) | s->setup_buf[1];
    value   = (s->setup_buf[3] << 8) | s->setup_buf[2];
    index   = (s->setup_buf[5] << 8) | s->setup_buf[4];
```

此处用户传进来的数据被拷贝到s->setup_buf数组,用户的数据在检查之前赋值给s->setup_len

此处开始检查s->setup_len是否大于s->data_buf数组的size,因为s->setup_len已经被赋值了,此处检查没有意义

实现越界读

```
//qemu-4.0.0-rc2\hw\usb\core.c
static void do_token_in(USBDevice *s, USBPacket *p)
{
    int request, value, index;

    assert(p->ep->nr == 0);

    request = (s->setup_buf[0] << 8) | s->setup_buf[1];
    value   = (s->setup_buf[3] << 8) | s->setup_buf[2];
    index   = (s->setup_buf[5] << 8) | s->setup_buf[4];

    switch(s->setup_state) {
    case SETUP_STATE_ACK:
        if (!(s->setup_buf[0] & USB_DIR_IN)) {
            usb_device_handle_control(s, p, request, value, index,
                                      s->setup_len, s->data_buf);
            if (p->status == USB_RET_ASYNC) {
                return;
            }
            s->setup_state = SETUP_STATE_IDLE;
            p->actual_length = 0;
        }
        break;

    case SETUP_STATE_DATA:
        if (s->setup_buf[0] & USB_DIR_IN) {
            int len = s->setup_len - s->setup_index;
            if (len > p->iov.size) {
                len = p->iov.size;
            }
            usb_packet_copy(p, s->data_buf + s->setup_index, len);
            s->setup_index += len;
            if (s->setup_index >= s->setup_len) {
                s->setup_state = SETUP_STATE_ACK;
```

这个bug导致s->setup_len可以大于s->data_buf数组的size

此处s->data_buf数组可以越界读,且直接读回虚拟机

实现越界写

```
//qemu-4.0.0-rc2\hw\usb\core.c
static void do_token_out(USBDevice *s, USBPacket *p)
{
    assert(p->ep->nr == 0);

    switch(s->setup_state) {
    case SETUP_STATE_ACK:
        if (s->setup_buf[0] & USB_DIR_IN) {
            s->setup_state = SETUP_STATE_IDLE;
            /* transfer OK */
        } else {
            /* ignore additional output */
        }
        break;

    case SETUP_STATE_DATA:
        if (!(s->setup_buf[0] & USB_DIR_IN)) {
            int len = s->setup_len - s->setup_index;
            if (len > p->iov.size) {
                len = p->iov.size;
            }
            usb_packet_copy(p, s->data_buf + s->setup_index, len);
            s->setup_index += len;
            if (s->setup_index >= s->setup_len) {
                s->setup_state = SETUP_STATE_ACK;
            }
            return;
        }
        s->setup_state = SETUP_STATE_IDLE;
        p->status = USB_RET_STALL;
        break;
```

此漏洞导致s->setup_len可以大于s->data_buf数组的size

s->data_buf数组可以越界写,而且写的内容完全可控

实现相对偏移越界写

```
/* definition of a USB device */
struct USBDevice {
    DeviceState qdev;
    USBPort *port;
    char *port_path;
    char *serial;
    void *opaque;
    uint32_t flags;

    /* Actual connected speed */
    int speed;
    /* Supported speeds, not in info because it may be variable (hostdevs) */
    int speedmask;
    uint8_t addr;
    char product_desc[32];
    int auto_attach;
    bool attached;

    int32_t state;
    uint8_t setup_buf[8];
    uint8_t data_buf[4096];
    int32_t remote_wakeup;
    int32_t setup_state;
    int32_t setup_len;
    int32_t setup_index;

    USBEndpoint ep_ctl;
    USBEndpoint ep_in[USB_MAX_ENDPOINTS];
    USBEndpoint ep_out[USB_MAX_ENDPOINTS];
```

这个数组是前面提到的data_buf,可以越界读写

每次读写data_buf时用setup_index作为偏移,所以利用越界写把setup_index设置成想要的值,可以实现相对偏移写

信息泄露和rip control

通常情况下需要利用堆风水,在越界读的结构体后面放一个有函数指针的结构体.
读到一个函数指针来泄露进程的基址.

Qemu的环境比较特殊,data_buf在qemu进程启动的时候申请,直到进程要结束
的时候才释放.这就导致不能够堆风水

其实只要利用越界读不断的往后读,能够读取到一个固定的qemu data节或text节
指针就能够完成信息泄露.

信息泄露和rip control

在data_buf之后,总能找到qxl设备对应的PCIDevice结构体,但是他们之间的偏移不是固定的.

只要利用越界读往后搜索字符串'qxl-vga',就能定位到qxl设备的PCIDevice结构体.

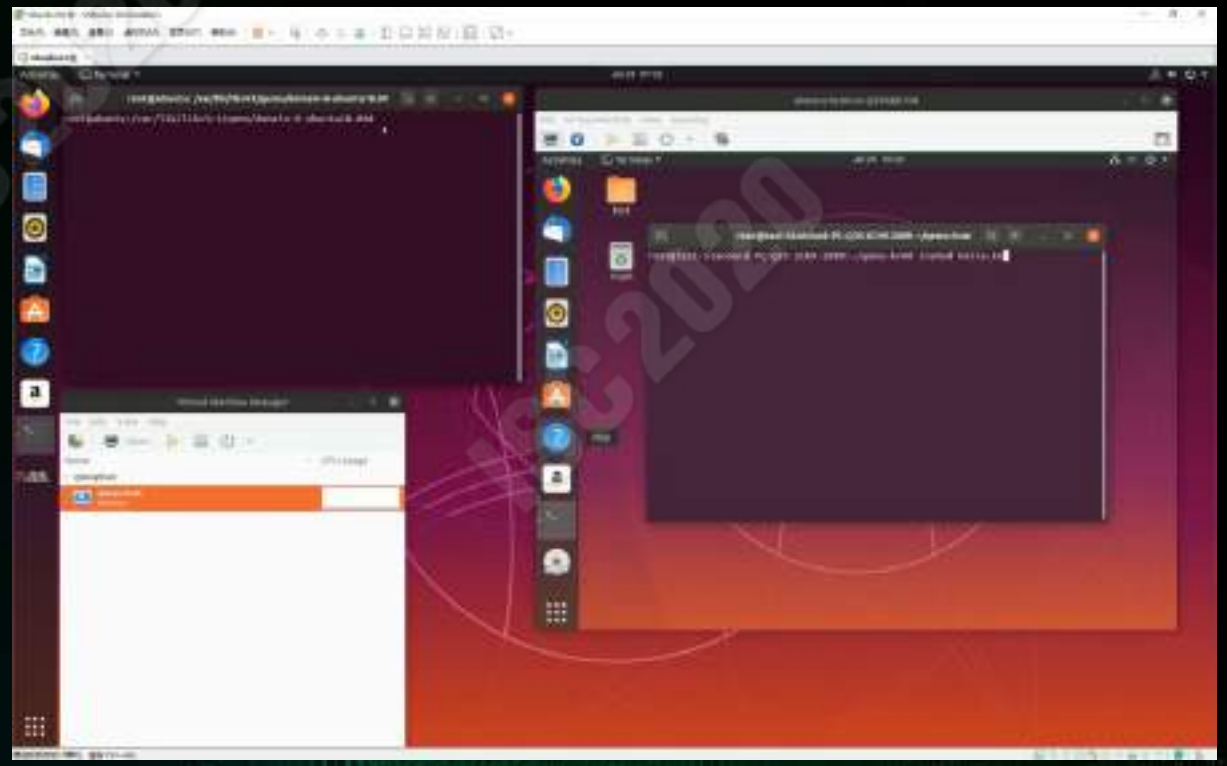
读取config_read就能完成信息泄露,改写config_read就能控制rip

```
struct PCIDevice {
    ...
    PCIReqIDCache requester_id_cache;
    char name[64];//qxl-vga
    PCIIORegion io_regions[PCI_NUM_REGIONS];
    AddressSpace bus_master_as;
    MemoryRegion bus_master_container_region;
    MemoryRegion bus_master_enable_region;

    /* do not access the following fields */
    PCIConfigReadFunc *config_read;
    PCIConfigWriteFunc *config_write;

    /* Legacy PCI VGA regions */
    MemoryRegion *vga_regions[QEMU_PCI_VGA_NUM_REGIONS];
    bool has_vga;
```

保存着设备的名字,qxl-vga

是一个函数指针,在虚拟机里读取pci配置寄存器可以触发config_read的调用

Demo



ESXi简介

ESXi是Vmware Vsphere产品的hypervisor组件。和qemu不同的是ESXi是一个类似linux的系统,直接安装在裸机上。ESXi里面默认运行着很多个服务,例如web服务、slpd服务。

SIpd服务是Open SLP（Service Location Protocol）协议的具体实现,默认在427端口监听,在虚拟机里可以访问到。接下来要介绍的漏洞cve-2019-5544就在SIpd服务里面。

漏洞成因

```
static int v2ParseUrlEntry(SLPBuffer buffer, SLPUrlEntry * urlentry)
{// openslp-2.0.0\common\slp_v2message.c
...
   /* Enforce SLPv2 URL entry size limits. */
   if (buffer->end - buffer->curpos < 6)
      return SLP_ERROR_PARSE_ERROR;

   /* Save pointer to opaque URL entry block. */
   urlentry->opaque = buffer->curpos;

   /* Parse individual URL entry fields. */
   urlentry->reserved = *buffer->curpos++;
   urlentry->lifetime = GetUINT16(&buffer->curpos);
   urlentry->urllen = GetUINT16(&buffer->curpos);
   urlentry->url = GetStrPtr(&buffer->curpos, urlentry->urllen);
   if (buffer->curpos > buffer->end)
      return SLP_ERROR_PARSE_ERROR;

   /* Parse authentication block. */
   urlentry->authcount = *buffer->curpos++;
   if (urlentry->authcount)
   {
      int i;
      urlentry->autharray = xmalloc(urlentry->authcount
            * sizeof(SLPAuthBlock));
      if (urlentry->autharray == 0)
         return SLP_ERROR_INTERNAL_ERROR;
      memset(urlentry->autharray, 0, urlentry->authcount
            * sizeof(SLPAuthBlock));
      for (i = 0; i < urlentry->authcount; i++)
      {
         int result = v2ParseAuthBlock(buffer, &urlentry->autharray[i]);
         if (result != 0)
            return result;
      }
   }
   urlentry->opaquelen = buffer->curpos - urlentry->opaque;
```

这个变量是可以控制的，可以不等于0

如果urlentry->authcount不等于0，那urlentry->opaquelen=urlentry->urllen+6+一个值，也就是说opaquelen可以大于urlen+6

漏洞成因

```
static int ProcessSrvRqst(SLPMessage * message, SLPBuffer * sendbuf,
      int errorcode)
{// openslp-2.0.0\xlpd\slpd_process.c
   ...
   size = message->header.langtaglen + 14;/* 14 bytes for header     */
   if (db && errorcode == 0)
   {
      for (i = 0; i < db->urlcount; i++)
      {
         /* urlentry is the url from the db result */
         urlentry = db->urlarray[i];

         size += urlentry->urllen + 3; /*  1 byte for reserved  */
#ifdef ENABLE_SLPv2_SECURITY
         /* make room to include the authblock that was asked for */
         if (G_SlpdProperty.securityEnabled
               && message->body.srvrqst.spistrlen)
         {
            for (j = 0; j < urlentry->authcount; j++)
            {
               if (SLPCompareString(urlentry->autharray[j].spistrlen,
                     urlentry->autharray[j].spistr,
                     message->body.srvrqst.spistrlen,
                     message->body.srvrqst.spistr) == 0)
               {
                  authblock = &(urlentry->autharray[j]);
                  size += authblock->length;
                  break;
               }
            }
         }
#endif
   ...
   for (i = 0; i < db->urlcount; i++)
   {  ...
      memcpy(result->curpos, urlentry->opaque, urlentry->opaquelen);
```

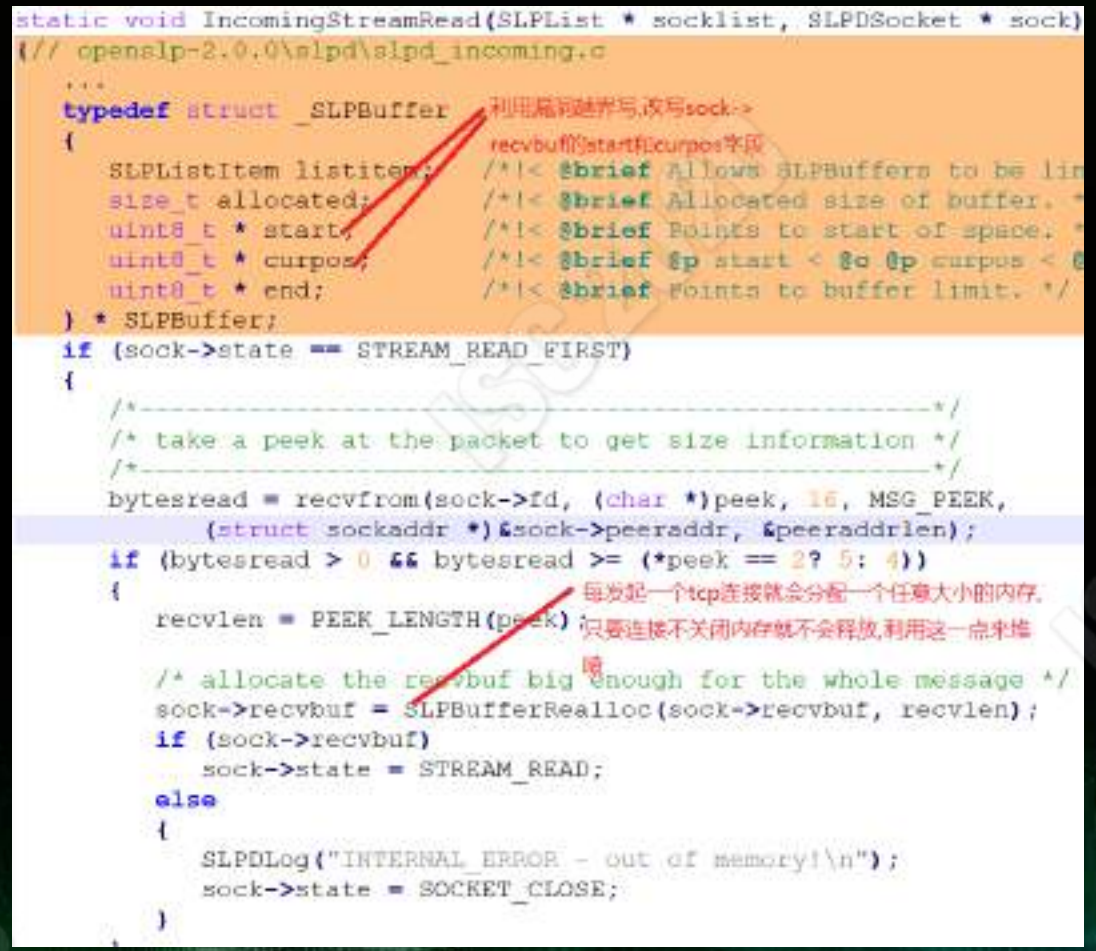
对于每一个urlentry,size只累加了urllen+6

这个特性默认是没有开启的,导致size少加了一个authblock->length

opequelen是可以大于urllen+6的,导致此处的内存拷贝越界写,boom!!!

任意地址写

通过发起多个tcp连接,并且保持连接不关闭可以实现堆喷。
利用堆风水溢出一个SLPBuffer结构体,改写它的start和curpos字段,下一次使用该结构体的时候可以实现任意地址写入任意数据



泄露栈地址和控制rip

slpd模块没有开启ASLR,所以它的基址是固定的。利用任意地址写,在slpd的data节伪造一个database,最后在ProcessAttrRqst查询的时候实现任意地址读,从libc读取栈地址并泄露出去。

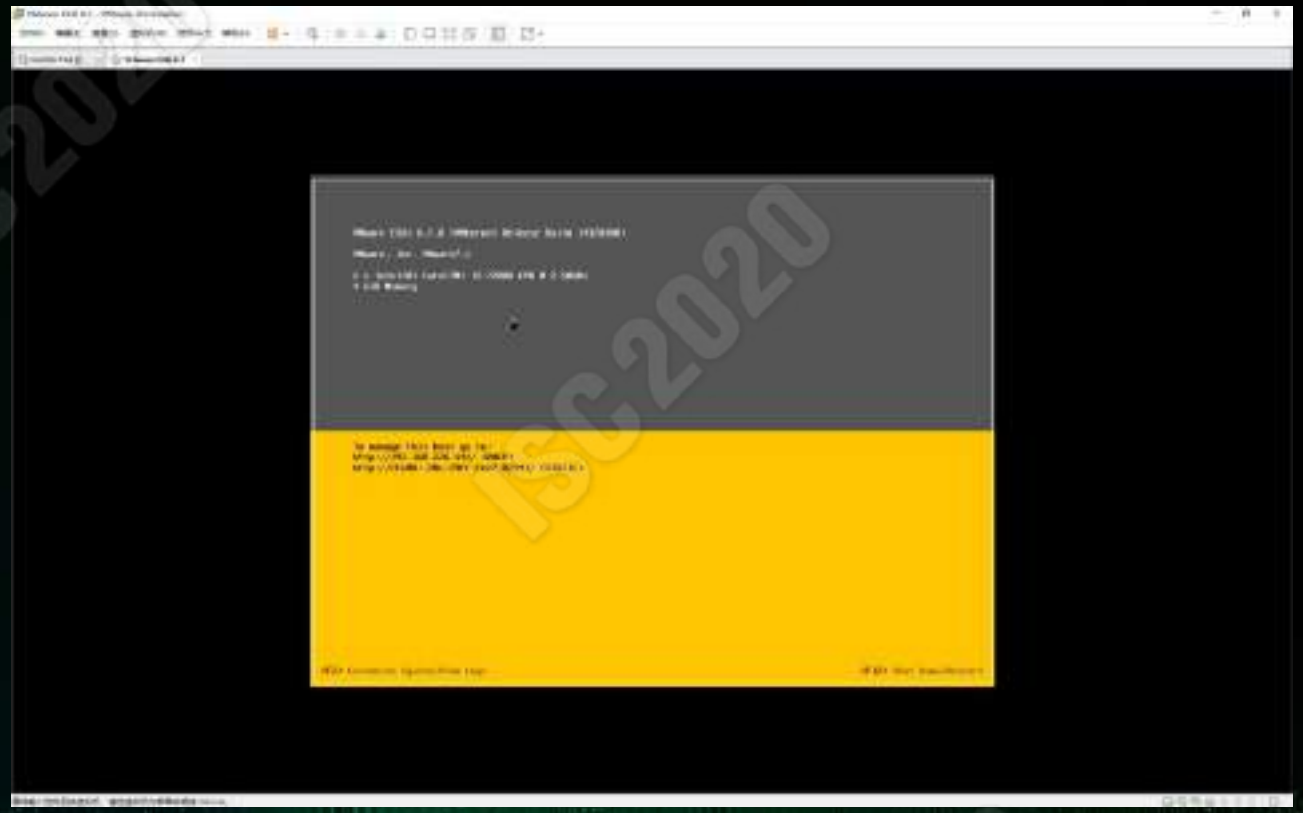
得到栈地址之后,利用任意地址写修改栈的内容实现rip control。

```
static int ProcessAttrRqst(SLPMessage * message, SLPBuffer * sendbuf,
      int errorcode)
{// openslp-2.0.0\slpd\slpd_process.c
   ...
   if (errorcode == 0)
   {
      /* attr-list len */
      PutUINT16(&result->curpos, db->attrlistlen);
      if (db->attrlistlen)
         memcpy(result->curpos, db->attrlist, db->attrlistlen);
      result->curpos += db->attrlistlen;

      /* authentication block */
#ifdef ENABLE_SLPv2_SECURITY
      if (opaqueauth)
      {
         /* authcount */
         *result->curpos++ = 1;
         memcpy(result->curpos, opaqueauth, opaqueauthlen);
         result->curpos += opaqueauthlen;
      }
      else
#endif
         *result->curpos++ = 0; /* authcount */
   }
   ...
   *sendbuf = result;
```

把查询的结果拷贝到一块内存中

最后会把查询的结果通过网络发送出去,利用这个特性来泄露栈地址

Demo



Q&A

邮箱

xiaowei-c@360.cn

THANKS

ISC 2020
第八届互联网安全大会
INTERNET SECURITY CONFERENCE 2020
数字孪生时代下的新安全
New Security in the Digital Twin Era