



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

> 数字孪生时代下的新安全 New Security in the Digital Twin Era





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。

↑ → qemu-4.0.0-rc2 → hw	0 1	∨ ♂ 搜索"hw"		
				_
9pfs	acpi	adc	alpha	arm
audio audio	land block	bt bt	char	core
cpu	cris	display	dma	gpio
hppa .	hyperv	i2c	i386	ide
input input	intc	ipack	ipmi	isa
lm32	m68k	mem	microblaze	mips
misc	moxie	net	nios2	nvram
openrisc .	pci	pci-bridge	pci-host	pcmcia
ppc	rdma	riscv	s390x	scsi
sd	sh4	smbios	sparc	sparc64
ssi	timer	tpm	tricore	unicore32
usb	vfio	virtio	watchdog	xen
xenpv	xtensa	☐ Kconfig	Makefile.objs	





漏洞介绍

```
//gemi-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 != 0) (
                                                                     此处用户传进来的数据被拷贝到
       p->status = USB RET STALL;
                                                                     s->setup buf数组用户的数据
       return;
                                                                     在检查之前赋值给s->setup len
   usb packet copy(p, s->setup buf, n->icv.size);
   s->setup index = 0;
   p->actual length
                                                                                   此处开始检查s->setup len显否大于
   s->setup len (s->setup buf[7] << 8) | s->setup buf[8];
                                                                                   s->data buf数组的size,因为s->
   if (s->setup len > sizeof(s->data buf)) 4
       fprintf(stderr,
                                                                                   setup len已经被赋值了,此处检查没有意
               "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] << 0) | s->setup_buf[1];
           = (s->setup_buf[3] << 8) | s->setup_buf[2];
           = (s->setup_buf[5] << 8) | s->setup_buf[4];
   index
```





```
static void do_token_in(USBDevice *s, USBPacket *p)
   int request, value, index;
   assert (p->ep->nr == 0) ;
   request = (s->setup buf[0] << 0) | s->setup buf[1];
   value = (s->setup buf[3] << |) | s->setup buf[2];
   index = (s->setup_buf[6] << 0) | 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) (
                                                                        这个bug号数s->setup len可以
               return;
                                                                        大于s->data buf放组的size
           s->setup_state = SETUP_STATE_IDLE;
           p->actual length = 0;
       break;
                                                                        此处s->data buf發细可以
   case SETUP STATE DATA:
                                                                        越界後,目直接该回處拟机
       if (s->setup buf[0] & USB DIB () (
           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;
```





```
//gemu-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 */
                                                                         此漏洞导致s->setup_len可以
        } else {
            /* ignore additional output */
                                                                         大于s->data buf数组的size
       break;
    case SETUP STATE DATA:
        if (!(s->setup buf[0] & USB DIB TN)) {
                                                                         s->data buf数组可以越界
           int len = s->setup len - s->setup index;
                                                                        写,而且写的内容完全可控
           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;
```





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;
    mint32 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;
                                                                    这个数组是前面提到的
    bool attached;
                                                                    data buf,可以越界读写
    int32 t state;
    mint8 t setup buf[8];
    uint8 t data buf[4096];
    int32 t remote wakeup;
                                                           每次读写data bufftl用setup index作为偏移,所
    int32 t setup state;
                                                           以利用越界写把setup index设置成想要的值,可
    int32 t setup len;
                                                          以实现相对偏移写
    int32 t setup index;
    USBEndpoint ep ctl;
    USBEndpoint ep in [USB MAX ENDPOINTS];
    USBEndpoint ep out[USB MAX ENDPOINTS];
```





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'qxlvga',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

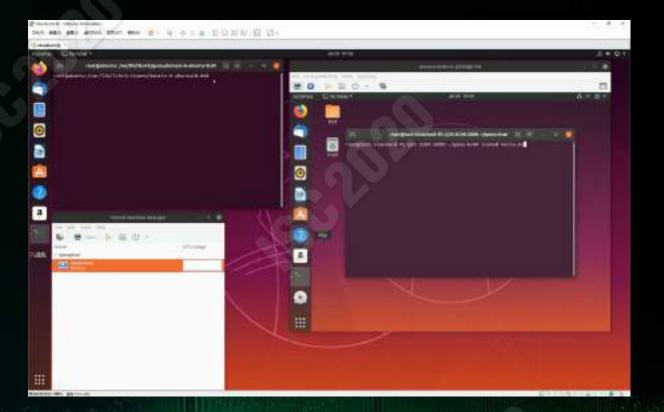
```
struct PCIDevice {
    PCIRegIDCache requester id cache;
   char name [64];//qxl-vga
                                                     保存普设备的名字,gxl-vga
    PCIIORegion io regions[PCI NUM REGIONS];
   AddressSpace bus master as;
   MemoryRegion bus master container region;
   MemoryRegion bus master enable region;
                                                        个函数指针,在虚拟
      do not access the following fields */
                                                      机里读取pci即置寄存器
   PCIConfigReadFunc *config read; -
                                                      可以触发config read的
    PCIConfigWriteFunc *config write;
    /* Legacy PCI VGA regions */
   MemoryRegion *vga regions[QEMU PCI VGA NUM REGIONS];
   bool has vga;
```





第八届互联网安全大会 360万 医阿安全中心

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(SLFBuffer buffer, SLFUrlEntry * urlentry)
(// openslp-2.0.0\common\slp v2message.c
  /* Enforce SLPVZ URL entry size limits. */
  if (buffer->end - buffer->curpos < 8)
     return SLP ERROR PARSE ERROR;
  /* Save pointer to opaque UEL entry black. */
  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;
  /* Perse authentication block. */
                                                                     2个空量量可以控制的,可
  urlentry->authcount = *buffer->curpos++/
                                                                     以不等于の
  if (urlentry->authcount)
     urlentry->authorray = xmalloc(urlentry->authount
           * sizeof(SLFAuthBlock));
     if (urlentry->authorray == 0)
                                                                 知思urlentry->authcount不够于0. 服
        return SLP ERROR INTERNAL ERROR;
                                                                 *urlentry->opaquelen-urlentry->urllen+6+一个值。
     memset (urlentry->authorray, 0, urlentry->authcount
           * sizeof(SLPAuthBlock));
                                                                 世就是说opaquelen可以大于urlan+6
     for (i = 0; 1 < urlentry->authcount; i++).
        int result = v2ParseAuthBlock(buffer, Surlentry->autharray[i]);
        if (result !- 0)
           return result;
  urlentry->opaqueler buffer->curpos - urlentry->opaque;
```





漏洞成因

```
static int ProcessFrvRqst(SLPMessage * message, SLPBuffer * sendbuf,
      int errorcode)
// opensip-2.0.0\mipd\mipd\mipd process.
  size = message->header.langtaglen + 107/+ 14 bytes for header
   if (db && errorcode == 1)
      for (i = 0; i < db->urlcount; i++)
         /* urlentry is the url from the db result */
         urlentry - db->urlarray[i];
                                                                        对于相一个urlentry,size只靠加了urllen+6
         size += urlentry->urlien + 17 /* 1 byte for reserved */
REPORT DWOLD SCHOOL SELECT
                                                                      这个特性默认是没有开启的、导致size少加了一个
         /* make room to include the authblock that was asked for */
                                                                       authblock->length
         if (G SlpdProperty.securityEnabled
               && message->body.srvrqst.spistrlen)
            for () = 0; j < urlentry->authocunt; j++)
               if (SLPCompareString (urlentry->authorray[j].spistrlen,
                     urlentry->autharray[j].spistr,
                     message->body.srvrqst.spistrlen,
                     message->body.srvrqst.spistr) - 0)
                                                                        opequelen區可以大于urlien+6的。特勢此处
                  authblock = &(urlentry->autharray[]]);
                                                                        的内存特到越界写,boom#1
                  size += authblock->length;
                  break;
   for (i = / i < db->urlcount; i++)
      mencpy (result->curpos, urlentry->opaque, urlentry->opaquelen);
```





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

```
static void IncomingStreamRead(SLPList * socklist, SLPDSocket * sock)
(// openslp-2.0.0\mlpd\slpd incoming.c
  typedef struct SLPBuffer 利用展別結構写成写sock->
                              recybuff/latartfilcurpos*FID
                              /*! < Sbrief Allows SLPBuffers to be lin
     SLPListItem listitem
                              /+! - Sbrief Allocated size of buffer.
     size t allocated:
     mint8 t * start
                              /*! < $brief Points to start of space.
                              /*!< @brief &p start < &c @p curpos < {
     minto t * curpos.
     mint8 t * end;
                              /* | < Shrint Foints to buffer limit, 1/
   * SLPBuffer:
   if (sock->state == STREAM READ FIRST)
     /* take a peek at the packet to get size information */
     bytesread = recvfrom(sock->fd, (char *)peek, 16, MSG PEEK,
            (struct sockaddr *) &sock->peeraddr, &peeraddrlen);
     if (bytesread > 0 && bytesread >= (*peek == 2? 5: 4))
                                   ● 每发起一个tcp连接就会分配一个任意大小的内存。
        recvien = PEEK LENGTH (peek) 识要连接不关闭内存就不会释放,利用这一点来推
         /* allocate the re buf big enough for the whole message */
         sock->recybuf = SLPBufferRealloc(sock->recybuf, recylen);
         if (sock->recybuf)
            sock->state = STREAM READ;
         else
           SLPDLog ("INTERNAL ERROR - out of memory! \n");
            sock->state = SOCKET CLOSE;
```





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。

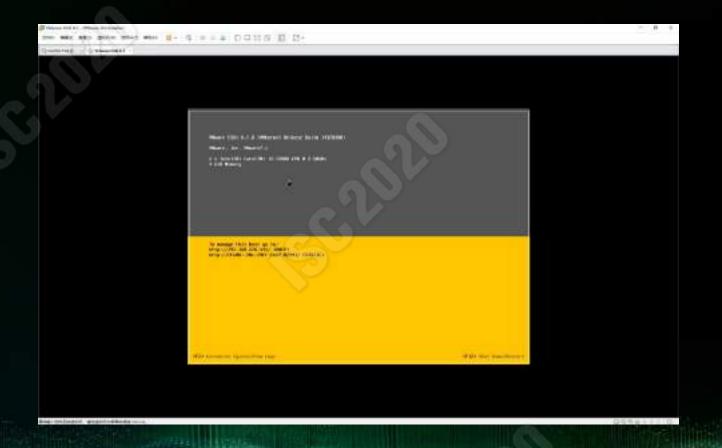
得到栈地址之后,利用任意地址写修改栈的内容实现rip control。

```
static int ProcessAttrRqst(SLPMessage * message, SLPBuffer * sendbuf,
     int errorcode)
   opensip-2.0.0\sipd\sipd process.c
   if (errorcode == 0)
                                   把查询的结果拷贝到一块内存中
      /* attr-list len */
                                db->attrlistlen);
     PutUINT16 (&result->curpos
     if (db->attrlistlen)
        memcpy (result->curpos, db->attrlist, db->attrlistlen);
     result->curpos += db->attrlistlen;
      /* authentication block */
     if (opsqueauth)
                                 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
         /* authcount */
         *result->curpos++ =
        memcpy (result->cytpos, opaqueauth, opaqueauthlen);
        result->curpos = opaqueauthlen;
     else
        *result->curpos++ = 0; /* authcount */
   *sendbuf = result;
```





Demo







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