

Date: 2019-09-03 Author: Ian Rash Index: #3

---

I figured since a majority of people would be using the Android client, I would start my audit there. The Android app, Eye4, is available in two separate downloads, one from the [Eye4 site](#), and one from the [Play store](#). They also have a [Windows download](#), which I will audit later.

The first thing I wanted to do is compare the Play store APK with the Eye4 site APK to see if there is any foul play going on, but unfortunately their personal download on the site didn't work and lead to a blank page.

We want to start our network sniffer, and begin our testing process.

First thing we want to do is boot the camera up after resetting it for the first time. This makes sure we have a clean slate when working with the camera. We will be resetting the camera multiple times to make sure we have lots of samples to compare.

Without connecting to any client, we will boot the camera up, connect to our sniffer, capture all the boot network data, then reset the camera, stop capture, and restart the whole thing. We do this about 2-3 times to get an idea of what the booting process looks like.

After that, we will connect the camera to the client, by searching for LAN in the client and connecting that way. Delete the camera, reset, then try it again. Do this multiple times to get multiple captures.

After that, we will fully connect the camera to the app, changing the password, as well as using various features, such as camera control, taking videos/pictures, changing settings, etc.

Once we are satisfied with our captures, we will begin dissecting them packet by packet.

## Boot Sequence

First impressions of boot sequence captures, what do we notice?

1. Take a look at some of the packets in the capture. 1. Do we see packets that have similar contents? 2. Can we see similar parts of packets that have small contents changed or replaced. 3. Do we see plaintext words/phrases? 4. Do we see commonly known numbers, ids, serial numbers, etc?

If you answered yes to any of these then the device most likely doesn't use encryption. This also means the device leaks data.

So let's go through the packets.

| No. | Time         | Source            | Destination               | Protocol | Length | Info                |
|-----|--------------|-------------------|---------------------------|----------|--------|---------------------|
| 19  | 31.999986862 | MagicCon_00:f9:97 | Spanning-tree-(for-bri... | STP      | 52     | Conf. Root = 32768/ |
| 20  | 34.015976380 | MagicCon_00:f9:97 | Spanning-tree-(for-bri... | STP      | 52     | Conf. Root = 32768/ |
| 21  | 35.363577412 | 192.168.1.126     | 255.255.255.255           | UDP      | 64     | 6808 → 6809 Len=18  |
| 22  | 35.373036978 | vstarcam          | Broadcast                 | ARP      | 64     | Who has 192.168.1.1 |
| 23  | 35.383437563 | 192.168.1.126     | 255.255.255.255           | UDP      | 64     | 6808 → 6809 Len=18  |
| 24  | 35.403186200 | 192.168.1.126     | 255.255.255.255           | UDP      | 64     | 6808 → 6809 Len=18  |
| 25  | 35.423186585 | 192.168.1.126     | 255.255.255.255           | UDP      | 64     | 6808 → 6809 Len=18  |
| 26  | 35.443189066 | 192.168.1.126     | 255.255.255.255           | UDP      | 64     | 6808 → 6809 Len=18  |
| 27  | 35.463183353 | 192.168.1.126     | 255.255.255.255           | UDP      | 64     | 6808 → 6809 Len=18  |
| 28  | 35.483813510 | 192.168.1.126     | 255.255.255.255           | UDP      | 64     | 6808 → 6809 Len=18  |
| 29  | 35.503315005 | 192.168.1.126     | 255.255.255.255           | UDP      | 64     | 6808 → 6809 Len=18  |
| 30  | 35.523187059 | 192.168.1.126     | 255.255.255.255           | UDP      | 64     | 6808 → 6809 Len=18  |
| 31  | 35.543181067 | 192.168.1.126     | 255.255.255.255           | UDP      | 64     | 6808 → 6809 Len=18  |
| 32  | 35.734312599 | 0.0.0.0           | 255.255.255.255           | DHCP     | 594    | DHCP Discover - Tra |

|   |                                                                                                         |
|---|---------------------------------------------------------------------------------------------------------|
| ▶ | Frame 21: 64 bytes on wire (512 bits), 64 bytes captured (512 bits) on interface 0                      |
| ▶ | Ethernet II, Src: vstarcam (48:02:2a:0b:db:b4), Dst: Broadcast (ff:ff:ff:ff:ff:ff)                      |
| ▶ | Internet Protocol Version 4, Src: 192.168.1.126 (192.168.1.126), Dst: 255.255.255.255 (255.255.255.255) |
| ▶ | User Datagram Protocol, Src Port: 6808 (6808), Dst Port: 6809 (6809)                                    |
| ▶ | Data (18 bytes)                                                                                         |

|      |                         |                         |                 |
|------|-------------------------|-------------------------|-----------------|
| 0000 | ff ff ff ff ff ff 48 02 | 2a 0b db b4 08 00 45 00 | .....H. *....E. |
| 0010 | 00 2e 00 00 40 00 40 11 | 78 99 c0 a8 01 7e ff ff | ...@. x.....    |
| 0020 | ff ff 1a 98 1a 99 00 1a | cf b2 48 65 6c 6c 6f 2c | .....Hello,     |
| 0030 | 49 27 6d 20 6f 6e 20 6c | 69 6e 65 21 d5 f8 03 5f | I'm on line!... |

The first weird thing I noticed is that the camera picked up a weird IP address for my network. It started out with the ip address 192.168.1.126, but my internal network range is 192.168.11.(100-254) meaning this camera had already come with this configuration. It also broadcasts an ARP looking for 192.168.1.1, and sends out a bunch of "Hello I'm online!" messages. Already this doesn't bode well, as it looks like they aren't using any encryption. Maybe it's a fluke, let's dive deeper.

I also saw that the device also contacts a couple external hosts

s2.vstarcam.com via UDP s3.vstarcam.com via UDP s2.ey4.cn via UDP An Amazon EC2 instance via UDP Google's public DNS, looking for baidu, wshifen.com, a.shifen.com An Alibaba cloud instance which it talks to via HTTPS. A Tencent cloud instance which it talks to via HTTP time.windows.com via NTP

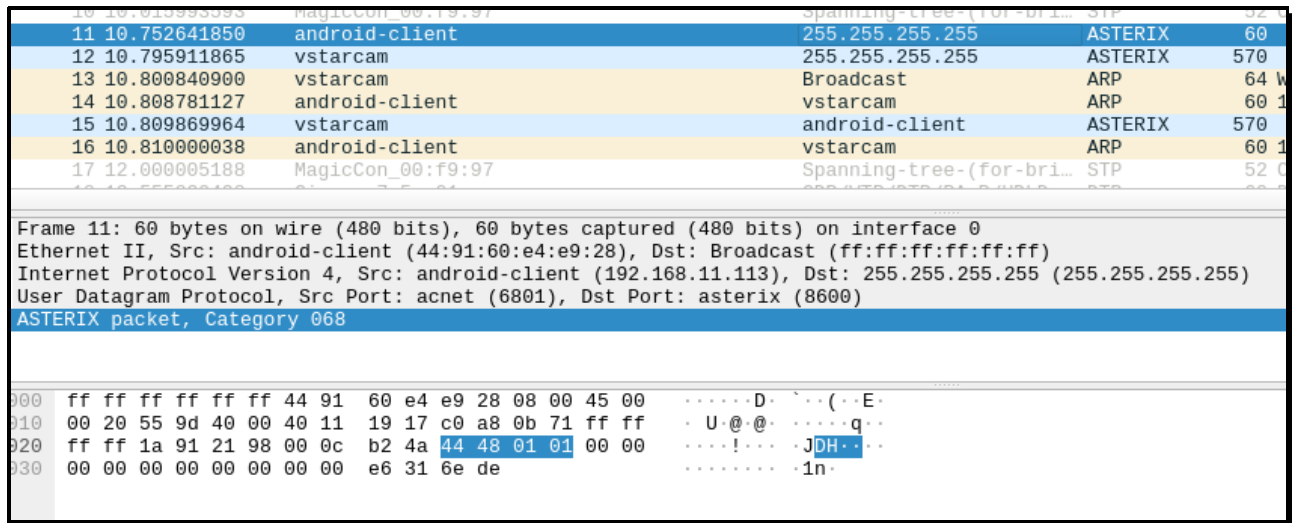
What did we learn?

The device may not use encryption The device may leak data as a result.

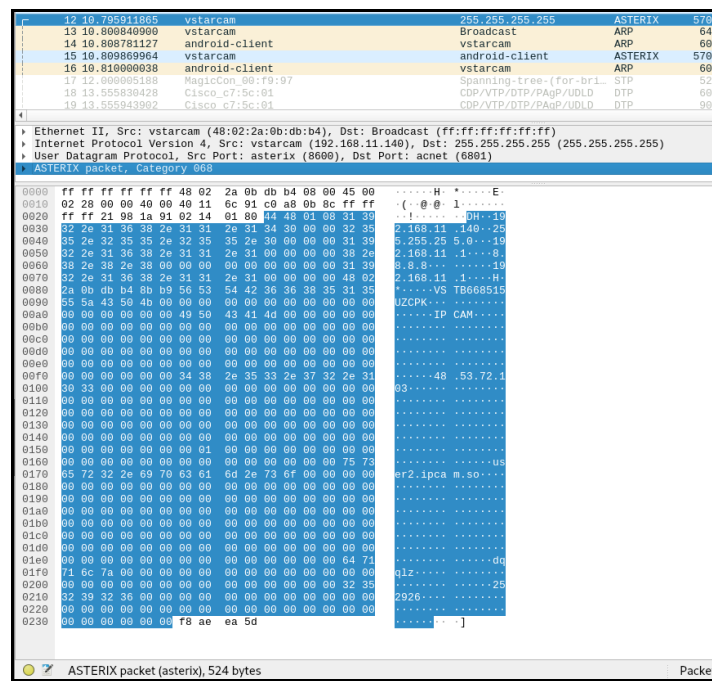
Let's take a look at the client's first registering of the camera, and it's first communications.

## Registering the camera

I loaded the app onto an android emulator and went to town. I started by registering the camera to the client by using the "search for LAN" feature.



First thing we notice is that when the Android client initiates the connection, it doesn't talk directly to the camera, it instead sends a UDP packet with the contents 0x44480101. We will call this the **Discovery Broadcast Packet** or **DBP** for short. The camera then replies back with a 0x44480108, and then dumps a bunch of settings information to the broadcast address. We will call this the **Discovery Broadcast Reply** or **DBR**.



For some reason Wireshark labels this packet as an **ASTERIX** packet, which when I looked it up seems to have nothing to do with this protocol. Maybe it shares a port or something and Wireshark incorrectly labels it.

When you look at the data incoming, we can start to see how this packet is laid out. The first 4 bytes are a signal to the client that it's looking for it, the followed by a space specifically carved out for an IP address, plus one 0x00 byte to delimit. The way we can tell this is because if we were to count the chars in a full IP address string (XXX.XXX.XXX.XXX which is 15 bytes, plus a 0x00 byte to delimit, making it 16 bytes.) We can see the next IP address (Which is the netmask) would be adjacent to that.

The camera then sends an exact copy of the packet it just broadcast, directly to the client.

After this little dance the camera and client talk directly to each other. We see it chooses the port 47499, which when

converted into hex is 0xB98B. We can see in the 0x80 row of the large packet above, there is a 0xB98B in little endian. This must be how the camera negotiates what port the client will talk to after the discovery phase.

When we compare this port number to our other captures, (remember, I captured the registration process, then reset the device, then did it again to see if I could notice any patterns.) we notice that the port changes with every reboot.

| No. | Time         | Source             | Destination               | Protocol | Length | Info                                                       |
|-----|--------------|--------------------|---------------------------|----------|--------|------------------------------------------------------------|
| 46  | 37.482882794 | vstarcam           | android-client            | TCP      | 78     | 47499 → 55146 [SYN, ACK] Seq=0 Ack=1 Win=14480 Len=0       |
| 47  | 37.486634849 | android-client     | vstarcam                  | TCP      | 66     | 55146 → 47499 [ACK] Seq=1 Ack=1 Win=87808 Len=0 TS=0       |
| 48  | 37.486849502 | android-client     | vstarcam                  | HTTP     | 300    | GET /get_camera_params.cgi/?user=admin&pwd=888888 HTTP/1.1 |
| 49  | 37.487715460 | vstarcam           | android-client            | TCP      | 70     | 47499 → 55146 [ACK] Seq=1 Ack=235 Win=14480 Len=0          |
| 50  | 37.977975244 | vstarcam           | android-client            | TCP      | 119    | 47499 → 55146 [PSH, ACK] Seq=1 Ack=235 Win=14480 Len=0     |
| 51  | 38.015998781 | MagicCon_00:f9:... | Spanning-tree-(for-bri... | STP      | 52     | Conf. Root = 32768/0/00:05:1b:00:f9:97 Cost = 0            |
| 52  | 38.043756242 | android-client     | vstarcam                  | TCP      | 66     | 55146 → 47499 [ACK] Seq=235 Ack=50 Win=87808 Len=0         |
| 53  | 38.044357169 | vstarcam           | android-client            | TCP      | 133    | 47499 → 55146 [PSH, ACK] Seq=50 Ack=235 Win=14480 Len=0    |
| 54  | 38.047692712 | android-client     | vstarcam                  | TCP      | 66     | 55146 → 47499 [ACK] Seq=235 Ack=113 Win=87808 Len=0        |
| 55  | 38.048748100 | vstarcam           | android-client            | TCP      | 95     | 47499 → 55146 [PSH, ACK] Seq=113 Ack=235 Win=14480 Len=0   |
| 56  | 38.050459511 | vstarcam           | android-client            | HTTP     | 924    | HTTP/1.1 200 OK (text/html)                                |

Transmission Control Protocol, Src Port: 55148 (55148), Dst Port: 47499 (47499), Seq: 1, Ack: 1, Len: 203

Hypertext Transfer Protocol

GET /get\_status.cgi?user=admin&pwd=888888 HTTP/1.1\r\nUser-Agent: Dalvik/2.1.0 (Linux; U; Android 8.1.0; SM-N960U Build/M1AJQ)\r\nHost: 192.168.11.140:47499\r\nConnection: Keep-Alive\r\n\r\n

0000 48 02 2a 0b db b4 44 91 60 e4 e9 28 08 00 45 00 H\*...D:..(..E  
0010 00 ff a8 13 40 00 40 06 f9 97 c0 a8 0b 71 c0 a8 ...@.@:....q..  
0020 0b 8c d7 6c b9 8b 29 af 2b f3 27 c7 db 13 80 18 ...l...):+.....  
0030 01 57 ae 47 00 00 01 01 08 0a 00 27 83 e4 ff ff ..W.G.....  
0040 d2 b9 47 45 54 20 2f 67 65 74 5f 73 74 61 74 75 ..GET /get\_statu  
0050 73 2e 63 67 69 3f 75 73 65 72 3d 61 64 6d 69 6e s.cgi?us er=admin  
0060 26 70 77 64 3d 38 38 38 38 38 20 48 54 54 50 &pwd=888 888 HTTP  
0070 2f 31 2e 31 0d 0a 55 73 65 72 2d 41 67 65 6e 74 /1.1..Us er-Agent  
0080 3a 20 44 61 6c 76 69 6b 2f 32 2e 31 2e 30 20 28 : Dalvik /2.1.0 (  
0090 4c 69 6e 75 78 3b 20 55 3b 20 41 6e 64 72 6f 69 Linux; U ; Androi  
00a0 64 20 38 2e 31 2e 30 3b 20 53 4d 2d 4e 39 36 30 d 8.1.0; SM-N960  
00b0 55 20 42 75 69 6c 64 2f 4d 31 41 4a 51 29 0d 0a U Build/ M1AJQ)..  
00c0 48 6f 73 74 3a 20 31 39 32 2e 31 36 38 2e 31 31 Host: 19 2.168.11  
00d0 2e 31 34 30 3a 34 37 34 39 39 0d 0a 43 6f 6e 6e .140:474 99..Conn  
00e0 65 63 74 69 6f 6e 3a 20 4b 65 65 70 2d 41 6c 69 ection: Keep-Ali  
00f0 76 65 0d 0a 41 63 63 65 70 74 2d 45 6e 63 6f 64 ve..Acce pt-Encod  
0100 69 6e 67 3a 20 67 7a 69 70 0d 0a 0d 0a ing: gzi p....

Looking at the contents of the HTTP GET request shows a sad truth, the devices doesn't use any encryption for it's client communications. This super concerning because not only does this compromise security of the device, especially on wireless, in an earlier capture, we have SEEN the device actually use SSL. So they are just choosing not to?

Looking at the reply brings much of the same sadness.

|                                                 |                     |
|-------------------------------------------------|---------------------|
| 3a 20 63 6c 6f 73 65 0d 0a 0d 0a 76 61 72 20 63 | : close...var C     |
| 61 6d 65 72 61 74 79 70 65 3d 33 3b 0d 0a 76 61 | amematyp e=3;..va   |
| 72 20 72 65 73 6f 6c 75 74 69 6f 6e 3d 32 3b 0d | r resolu tion=2;..  |
| 0a 76 61 72 20 72 65 73 6f 6c 75 74 69 6f 6e 73 | ..var res olutions  |
| 75 62 3d 30 3b 0d 0a 76 61 72 20 72 65 73 6f 6c | sub=0;..v ar resol  |
| 75 74 69 6f 6e 73 75 62 73 75 62 3d 31 3b 0d 0a | utionsub sub=1;..   |
| 76 61 72 20 76 62 72 69 67 68 74 3d 31 32 38 3b | var vbri ght=128;.. |
| 0d 0a 76 61 72 20 76 63 6f 6e 74 72 61 73 74 3d | ..var vc ontrast=   |
| 31 32 36 3b 0d 0a 76 61 72 20 76 68 75 65 3d 31 | 126;..va r vhue=1   |
| 32 36 3b 0d 0a 76 61 72 20 76 73 61 74 75 72 61 | 26;..var vsatura    |
| 74 69 6f 6e 3d 31 32 36 3b 0d 0a 76 61 72 20 4f | tion=126 ;..var 0   |
| 53 44 45 6e 61 62 6c 65 3d 30 3b 0d 0a 76 61 72 | SDEnable =0;..var   |
| 20 6d 6f 64 65 3d 30 3b 0d 0a 76 61 72 20 66 6c | mode=0;..var fl     |
| 69 70 3d 30 3b 0d 0a 76 61 72 20 65 6e 63 5f 73 | ip=0;..v ar enc_s   |
| 69 7a 65 3d 32 3b 0d 0a 76 61 72 20 65 6e 63 5f | ize=2;.. var enc_   |
| 66 72 61 6d 65 72 61 74 65 3d 31 35 3b 0d 0a 76 | framerat e=15;..V   |
| 61 72 20 65 6e 63 5f 6b 65 79 66 72 61 6d 65 3d | ar enc_k eyframe=   |
| 31 35 3b 0d 0a 76 61 72 20 65 6e 63 5f 71 75 61 | 15;..var enc qua    |
| 6e 74 3d 30 3b 0d 0a 76 61 72 20 65 6e 63 5f 72 | nt=0;..v ar enc_r   |
| 61 74 65 6d 6f 64 65 3d 30 3b 0d 0a 76 61 72 20 | atemode= 0;..var    |
| 65 6e 63 5f 62 69 74 72 61 74 65 3d 31 30 32 34 | enc_bitr ate=1024   |
| 3b 0d 0a 76 61 72 20 65 6e 63 5f 6d 61 69 6e 5f | ..var e nc.main.    |
| 6d 6f 64 65 3d 30 3b 0d 0a 76 61 72 20 73 75 62 | mode=0;..var sub    |
| 5f 65 6e 63 5f 73 69 7a 65 3d 30 3b 0d 0a 76 61 | enc_siz e=0;..va    |
| 72 20 73 75 62 5f 65 6e 63 5f 66 72 61 6d 65 72 | r sub_en c.framer   |
| 61 74 65 3d 31 35 3b 0d 0a 76 61 72 20 73 75 62 | ate=15;..var sub    |
| 5f 65 6e 63 5f 6b 65 79 66 72 61 6d 65 3d 31 35 | enc_key frame=15    |
| 3b 0d 0a 76 61 72 20 73 75 62 5f 65 6e 63 5f 71 | ..var s ub_enc_q    |
| 75 61 6e 74 3d 30 3b 0d 0a 76 61 72 20 73 75 62 | uant=0;..var sub    |
| 5f 65 6e 63 5f 72 61 74 65 6d 6f 64 65 3d 30 3b | enc_rat emode=0;    |
| 0d 0a 76 61 72 20 73 75 62 5f 65 6e 63 5f 62 69 | ..var su b_enc_b1   |
| 74 72 61 74 65 3d 35 31 32 3b 0d 0a 76 61 72 20 | trate=51 2;..var    |
| 73 75 62 5f 73 75 62 5f 65 6e 63 5f 73 69 7a 65 | sub_sub_enc_size    |
| 3d 31 3b 0d 0a 76 61 72 20 73 75 62 5f 73 75 62 | =1;..var sub_sub    |
| 5f 65 6e 63 5f 66 72 61 6d 65 72 61 74 65 3d 31 | enc_fra merate=1    |
| 30 3b 0d 0a 76 61 72 20 73 75 62 5f 73 75 62 5f | 0;..var sub_sub     |
| 65 6e 63 5f 6b 65 79 66 72 61 6d 65 3d 31 30 3b | enc_keyf rame=10;   |
| 0d 0a 76 61 72 20 73 75 62 5f 73 75 62 5f 65 6e | ..var su b_sub_en   |
| 63 5f 71 75 61 6e 74 3d 30 3b 0d 0a 76 61 72 20 | c_quant= 0;..var    |
| 73 75 62 5f 73 75 62 5f 65 6e 63 5f 72 61 74 65 | sub_sub_enc_rate    |
| 6d 6f 64 65 3d 30 3b 0d 0a 76 61 72 20 73 75 62 | mode=0;..var sub    |
| 5f 73 75 62 5f 65 6e 63 5f 62 69 74 72 61 74 65 | _sub_enc_bitrate    |

Looking at the HTTP headers, we can also see that the web server the device is GoAhead webs, an embedded web server, and it also hasn't been updated since 2004. JUICY!

So this is where it gets interesting.

So, they already have an HTTP server to talk to the camera and serve CGI scripts, but they decided to add an extra protocol on top of that. This is where some strange UDP protocol starts to come into play.

Even though the camera and client were literally just talking via HTTP, it seems to forget all about that and try again to search for a new camera. The client sends out a 0xf1300000 packet to my subnet's broadcast address (192.168.11.255). We will call this packet the **Broadcast Connection Packet** or **BCP** for short.

|     |              |                |                |     |     |       |   |       |         |
|-----|--------------|----------------|----------------|-----|-----|-------|---|-------|---------|
| 153 | 81.569365118 | android-client | 192.168.11.255 | UDP | 60  | 11618 | → | 32108 | Len=4   |
| 154 | 81.571525648 | vstarcam       | android-client | UDP | 70  | 12669 | → | 11618 | Len=24  |
| 155 | 81.575177020 | android-client | vstarcam       | UDP | 66  | 11618 | → | 12669 | Len=24  |
| 156 | 81.576270465 | vstarcam       | android-client | UDP | 70  | 12669 | → | 11618 | Len=24  |
| 157 | 81.590151332 | vstarcam       | android-client | UDP | 70  | 12669 | → | 11618 | Len=24  |
| 158 | 81.609923188 | vstarcam       | android-client | UDP | 70  | 12669 | → | 11618 | Len=24  |
| 159 | 81.630547064 | vstarcam       | android-client | UDP | 70  | 12669 | → | 11618 | Len=24  |
| 160 | 81.669683540 | android-client | 192.168.11.255 | UDP | 60  | 11618 | → | 32108 | Len=4   |
| 161 | 81.731030083 | vstarcam       | android-client | UDP | 64  | 12669 | → | 11618 | Len=4   |
| 162 | 81.735686057 | android-client | vstarcam       | UDP | 146 | 11618 | → | 12669 | Len=104 |
| 163 | 81.736025129 | android-client | vstarcam       | UDP | 60  | 11618 | → | 12669 | Len=4   |
| 164 | 81.754179180 | vstarcam       | android-client | UDP | 64  | 12669 | → | 11618 | Len=10  |
| 165 | 81.765522053 | android-client | vstarcam       | UDP | 133 | 11618 | → | 12669 | Len=91  |
| 166 | 81.773769086 | vstarcam       | android-client | UDP | 64  | 12669 | → | 11618 | Len=10  |
| 167 | 81.810146541 | vstarcam       | android-client | UDP | 64  | 12669 | → | 11618 | Len=4   |
| 168 | 81.813478510 | android-client | vstarcam       | UDP | 60  | 11618 | → | 12669 | Len=4   |
| 169 | 81.854172615 | vstarcam       | android-client | UDP | 122 | 12669 | → | 11618 | Len=76  |
| 170 | 81.867520643 | android-client | vstarcam       | UDP | 60  | 11618 | → | 12669 | Len=10  |
| 171 | 81.957426287 | android-client | vstarcam       | UDP | 130 | 11618 | → | 12669 | Len=88  |
| 172 | 81.969185249 | android-client | vstarcam       | UDP | 387 | 11618 | → | 12669 | Len=345 |
| 173 | 81.977845379 | android-client | vstarcam       | UDP | 130 | 11618 | → | 12669 | Len=88  |

▶ Frame 153: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface 0  
▶ Ethernet II, Src: android-client (44:91:60:e4:e9:28), Dst: Broadcast (ff:ff:ff:ff:ff:ff)  
▶ Internet Protocol Version 4, Src: android-client (192.168.11.113), Dst: 192.168.11.255 (192.168.11.255)  
▶ User Datagram Protocol, Src Port: 11618 (11618), Dst Port: 32108 (32108)  
▶ Data (4 bytes)

|      |                         |                         |                  |
|------|-------------------------|-------------------------|------------------|
| 0000 | ff ff ff ff ff ff 44 91 | 60 e4 e9 28 08 00 45 00 | .....D..(..E..   |
| 0010 | 00 20 38 9a 40 00 40 11 | 69 72 c0 a8 0b 71 c0 a8 | ..8..@..ir...q.. |
| 0020 | 0b ff 2d 62 7d 6c 00 0c | cb 15 f1 30 00 00 00 00 | ..b}l...0....    |
| 0030 | 00 00 00 00 00 00 00 00 | e6 ad 4a 0f             | .....J..         |

The camera then replies back with a small part of its UID and the letter A. The client repeats this message back to the camera. We will call this the **Broadcast Packet SYN** or **BPS** for short.

|     |              |                |                |     |     |       |   |       |         |
|-----|--------------|----------------|----------------|-----|-----|-------|---|-------|---------|
| 154 | 81.571525648 | vstarcam       | android-client | UDP | 70  | 12669 | → | 11618 | Len=24  |
| 155 | 81.575177020 | android-client | vstarcam       | UDP | 66  | 11618 | → | 12669 | Len=24  |
| 156 | 81.576270465 | vstarcam       | android-client | UDP | 70  | 12669 | → | 11618 | Len=24  |
| 157 | 81.590151332 | vstarcam       | android-client | UDP | 70  | 12669 | → | 11618 | Len=24  |
| 158 | 81.609923188 | vstarcam       | android-client | UDP | 70  | 12669 | → | 11618 | Len=24  |
| 159 | 81.630547064 | vstarcam       | android-client | UDP | 70  | 12669 | → | 11618 | Len=24  |
| 160 | 81.669683540 | android-client | 192.168.11.255 | UDP | 60  | 11618 | → | 32108 | Len=4   |
| 161 | 81.731030083 | vstarcam       | android-client | UDP | 64  | 12669 | → | 11618 | Len=4   |
| 162 | 81.735686057 | android-client | vstarcam       | UDP | 146 | 11618 | → | 12669 | Len=104 |
| 163 | 81.736025129 | android-client | vstarcam       | UDP | 60  | 11618 | → | 12669 | Len=4   |
| 164 | 81.754179180 | vstarcam       | android-client | UDP | 64  | 12669 | → | 11618 | Len=10  |
| 165 | 81.765522053 | android-client | vstarcam       | UDP | 133 | 11618 | → | 12669 | Len=91  |
| 166 | 81.773769086 | vstarcam       | android-client | UDP | 64  | 12669 | → | 11618 | Len=10  |
| 167 | 81.810146541 | vstarcam       | android-client | UDP | 64  | 12669 | → | 11618 | Len=4   |
| 168 | 81.813478510 | android-client | vstarcam       | UDP | 60  | 11618 | → | 12669 | Len=4   |
| 169 | 81.854172615 | vstarcam       | android-client | UDP | 122 | 12669 | → | 11618 | Len=76  |
| 170 | 81.867520643 | android-client | vstarcam       | UDP | 60  | 11618 | → | 12669 | Len=10  |
| 171 | 81.957426287 | android-client | vstarcam       | UDP | 130 | 11618 | → | 12669 | Len=88  |
| 172 | 81.969185249 | android-client | vstarcam       | UDP | 387 | 11618 | → | 12669 | Len=345 |
| 173 | 81.977845379 | android-client | vstarcam       | UDP | 130 | 11618 | → | 12669 | Len=88  |

▶ Frame 154: 70 bytes on wire (560 bits), 70 bytes captured (560 bits) on interface 0  
▶ Ethernet II, Src: vstarcam (48:02:2a:0b:db:b4), Dst: android-client (44:91:60:e4:e9:28)  
▶ Internet Protocol Version 4, Src: vstarcam (192.168.11.140), Dst: android-client (192.168.11.113)  
▶ User Datagram Protocol, Src Port: 12669 (12669), Dst Port: 11618 (11618)  
▶ Data (24 bytes)

|      |                         |                         |                   |
|------|-------------------------|-------------------------|-------------------|
| 0000 | 44 91 60 e4 e9 28 48 02 | 2a 0b db b4 08 00 45 00 | D..(H..*.....E..  |
| 0010 | 00 34 00 00 40 00 40 11 | a2 6b c0 a8 0b 8c c0 a8 | ..4..@..@..k..... |
| 0020 | 0b 71 31 7d 2d 62 00 20 | 55 7d f1 41 00 14 56 53 | ..q1}-b..U}-A..VS |
| 0030 | 54 42 00 00 00 00 00 0a | 33 63 55 5a 43 50 4b 00 | TB.....3cUZCPK..  |
| 0040 | 00 00 79 e2 a8 d9       |                         | ...y...           |

The camera then sends the same packet back, this time with the A changed to a B. We will call this the **Broadcast Packet ACK** or **BPA** for short

|     |              |                |                |     |    |       |   |       |        |
|-----|--------------|----------------|----------------|-----|----|-------|---|-------|--------|
| 154 | 81.571525648 | vstarcam       | android-client | UDP | 70 | 12669 | → | 11618 | Len=24 |
| 155 | 81.575177020 | android-client | vstarcam       | UDP | 66 | 11618 | → | 12669 | Len=24 |
| 156 | 81.576270465 | vstarcam       | android-client | UDP | 70 | 12669 | → | 11618 | Len=24 |
| 157 | 81.590151332 | vstarcam       | android-client | UDP | 70 | 12669 | → | 11618 | Len=24 |
| 158 | 81.609923188 | vstarcam       | android-client | UDP | 70 | 12669 | → | 11618 | Len=24 |
| 159 | 81.630547064 | vstarcam       | android-client | UDP | 70 | 12669 | → | 11618 | Len=24 |
| 160 | 81.669683540 | android-client | 192.168.11.255 | UDP | 60 | 11618 | → | 32108 | Len=4  |
| 161 | 81.731030083 | vstarcam       | android-client | UDP | 64 | 12669 | → | 11618 | Len=4  |

▶ Frame 156: 70 bytes on wire (560 bits), 70 bytes captured (560 bits) on interface 0

▶ Ethernet II, Src: vstarcam (48:02:2a:0b:db:b4), Dst: android-client (44:91:60:e4:e9:28)

▶ Internet Protocol Version 4, Src: vstarcam (192.168.11.140), Dst: android-client (192.168.11.113)

▶ User Datagram Protocol, Src Port: 12669 (12669), Dst Port: 11618 (11618)

▶ Data (24 bytes)

|      |                         |                         |                               |
|------|-------------------------|-------------------------|-------------------------------|
| 0000 | 44 91 60 e4 e9 28 48 02 | 2a 0b db b4 08 00 45 00 | D · · · (H · * · · · · E ·    |
| 0010 | 00 34 00 00 40 00 40 11 | a2 6b c0 a8 0b 8c c0 a8 | · 4 · · @ · @ · · k · · · · · |
| 0020 | 0b 71 31 7d 2d 62 00 20 | 55 7c f1 42 00 14 56 53 | · q1 } · b · U   · B · · VS   |
| 0030 | 54 42 00 00 00 00 00 0a | 33 63 55 5a 43 50 4b 00 | TB · · · · · 3cUZCPK ·        |
| 0040 | 00 00 4a 4b c6 0a       |                         | · · JK · ·                    |

Finally we arrive at the main phase of the connection, here we can observe three things happening.

1. The client and the camera are constantly engaging in a "ping pong" pattern message loop, where if either the client or the camera receives a 0xF1E0000 or 0xF1E10000, it replies back with the other packet. So for example, if a 0xF1E0000 is received it will reply back with 0xF1E10000. For reference 0xF1E0000 is ping, and 0xF1E10000 is pong. *You can illustrate this best by using the Wireshark filter below* frame contains F1:E1:00:00 || frame contains F1:E0:00:00 \* 
2. The client sends GET requests with a special header, via UDP to the camera. The client receives replies back denoting success or failure, and the results of the query, sometimes broken up into multiple packets. 3. The client receives video and picture data from the camera. *livestream.cgi snapshot.cgi \* audiostream.cgi* Let's focus a little more on number two, since one is already solved, and three requires more extensive knowledge of video formats and decoding.

|     |              |                |                |     |  |  |  |  |  |
|-----|--------------|----------------|----------------|-----|--|--|--|--|--|
| 162 | 81.735686057 | android-client | vstarcam       | UDP |  |  |  |  |  |
| 163 | 81.736025129 | android-client | vstarcam       | UDP |  |  |  |  |  |
| 164 | 81.754179180 | vstarcam       | android-client | UDP |  |  |  |  |  |
| 165 | 81.765522053 | android-client | vstarcam       | UDP |  |  |  |  |  |
| 166 | 81.773769086 | vstarcam       | android-client | UDP |  |  |  |  |  |
| 167 | 81.810146541 | vstarcam       | android-client | UDP |  |  |  |  |  |
| 168 | 81.813478510 | android-client | vstarcam       | UDP |  |  |  |  |  |

▶ Frame 162: 146 bytes on wire (1168 bits), 146 bytes captured (1168 bits) on interface 0

▶ Ethernet II, Src: android-client (44:91:60:e4:e9:28), Dst: vstarcam (48:02:2a:0b:db:b4)

▶ Internet Protocol Version 4, Src: android-client (192.168.11.113), Dst: vstarcam (192.168.11.140)

▶ User Datagram Protocol, Src Port: 11618 (11618), Dst Port: 12669 (12669)

▶ Data (104 bytes)

|      |                         |                         |                                    |
|------|-------------------------|-------------------------|------------------------------------|
| 0000 | 48 02 2a 0b db b4 44 91 | 60 e4 e9 28 08 00 45 00 | H · * · · · D · · · · ( · · E ·    |
| 0010 | 00 84 37 bc 40 00 40 11 | 6a 5f c0 a8 0b 71 c0 a8 | · · 7 · @ · @ · j _ · · · · q · ·  |
| 0020 | 0b 8c 2d 62 31 7d 00 70 | 54 5b f1 d0 00 64 d1 00 | · · · · b1 } · p T [ · · · · d · · |
| 0030 | 00 00 01 0a 00 00 58 00 | 00 00 47 45 54 20 2f 63 | · · · · · · X · · · GET /c         |
| 0040 | 68 65 63 6b 5f 75 73 65 | 72 2e 63 67 69 3f 6e 61 | heck_use r.cgi?na                  |
| 0050 | 6d 65 3d 33 30 30 31 32 | 37 36 33 30 26 6c 6f 67 | me=30012 7630&log                  |
| 0060 | 69 6e 75 73 65 3d 61 64 | 6d 69 6e 26 6c 6f 67 69 | inuse=ad min&logi                  |
| 0070 | 6e 70 61 73 3d 38 38 38 | 38 38 38 26 75 73 65 72 | npas=888 888&user                  |
| 0080 | 3d 61 64 6d 69 6e 26 70 | 77 64 3d 38 38 38 38 38 | =admin&p wd=88888                  |
| 0090 | 38 26                   |                         | 8&                                 |

The first 16 bytes of the packet are some sort of specialized header. We are going to want to get more samples of these to be able to learn how to forge our own. We also want to get samples to see if it might be vulnerable to replay

tactics as, the header might block potential replay attacks. After that comes the GET request. Right off the bat we are hit with another oddity of the client, credential leakage, and unnecessary duplication. Not only are the communications not encrypted, so it also leaks the password on every GET interaction but, it also repeats the credentials twice just in case the first time wasn't good enough.

|     |              |                |                |     |
|-----|--------------|----------------|----------------|-----|
| 164 | 81.754179180 | vstarcam       | android-client | UDP |
| 165 | 81.765522053 | android-client | vstarcam       | UDP |
| 166 | 81.773769086 | vstarcam       | android-client | UDP |
| 167 | 81.810146541 | vstarcam       | android-client | UDP |
| 168 | 81.813478510 | android-client | vstarcam       | UDP |

ame 164: 64 bytes on wire (512 bits), 64 bytes captured (512 bits) on interface 0  
hernet II, Src: vstarcam (48:02:2a:0b:db:b4), Dst: android-client (44:91:60:e4:e9:28)  
ternet Protocol Version 4, Src: vstarcam (192.168.11.140), Dst: android-client (192.168.11.113)  
er Datagram Protocol, Src Port: 12669 (12669), Dst Port: 11618 (11618)

ta (10 bytes)

|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 44 | 91 | 60 | e4 | e9 | 28 | 48 | 02 | 2a | 0b | db | b4 | 08 | 00 | 45 | 00 | D | . | . | . | . | ( | H | . | * | . | . | . | . | E | . |
| 00 | 26 | 00 | 00 | 40 | 00 | 40 | 11 | a2 | 79 | c0 | a8 | 0b | 8c | c0 | a8 | . | & | . | . | . | @ | . | @ | . | . | . | . | y | . | . |
| 0b | 71 | 31 | 7d | 2d | 62 | 00 | 12 | 45 | c3 | f1 | d1 | 00 | 06 | d1 | 00 | . | q | 1 | . | . | . | . | . | . | . | . | . | E | . | . |
| 00 | 01 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | c9 | da | e1 | 60 | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . |

A short time afterwards the camera sends a packet that looks like this. This seems to acknowledge that the command went through.

|     |              |                |                |     |     |       |   |       |        |
|-----|--------------|----------------|----------------|-----|-----|-------|---|-------|--------|
| 165 | 81.765522053 | android-client | vstarcam       | UDP | 133 | 11618 | → | 12669 | Len=91 |
| 166 | 81.773769086 | vstarcam       | android-client | UDP | 64  | 12669 | → | 11618 | Len=10 |
| 167 | 81.810146541 | vstarcam       | android-client | UDP | 64  | 12669 | → | 11618 | Len=4  |
| 168 | 81.813478510 | android-client | vstarcam       | UDP | 60  | 11618 | → | 12669 | Len=4  |

me 165: 133 bytes on wire (1064 bits), 133 bytes captured (1064 bits) on interface 0  
hernet II, Src: android-client (44:91:60:e4:e9:28), Dst: vstarcam (48:02:2a:0b:db:b4)  
hernet Protocol Version 4, Src: android-client (192.168.11.113), Dst: vstarcam (192.168.11.140)  
r Datagram Protocol, Src Port: 11618 (11618), Dst Port: 12669 (12669)

a (91 bytes)

|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 48 | 02 | 2a | 0b | db | b4 | 44 | 91 | 60 | e4 | e9 | 28 | 08 | 00 | 45 | 00 | H | . | * | . | . | D | . | . | . | . | ( | . | . | E | . |   |   |
| 00 | 77 | 37 | c0 | 40 | 00 | 40 | 11 | 6a | 68 | c0 | a8 | 0b | 71 | c0 | a8 | . | w | 7 | . | . | . | @ | . | . | . | . | j | h | . | . | q | . |
| 0b | 8c | 2d | 62 | 31 | 7d | 00 | 63 | 96 | 07 | f1 | d0 | 00 | 57 | d1 | 00 | . | . | . | . | b | 1 | . | . | . | . | . | . | . | . | . | . |   |
| 00 | 01 | 01 | 0a | 00 | 00 | 4b | 00 | 00 | 00 | 47 | 45 | 54 | 20 | 2f | 74 | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . |   |
| 72 | 61 | 6e | 73 | 5f | 63 | 6d | 64 | 5f | 73 | 74 | 72 | 69 | 6e | 67 | 2e | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . |   |
| 63 | 67 | 69 | 3f | 63 | 6d | 64 | 3d | 32 | 30 | 30 | 31 | 26 | 63 | 6f | 6d | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . |   |
| 6d | 61 | 6e | 64 | 3d | 30 | 26 | 6c | 6f | 67 | 69 | 6e | 75 | 73 | 65 | 3d | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . |   |
| 61 | 64 | 6d | 69 | 6e | 26 | 6c | 6f | 67 | 69 | 6e | 70 | 61 | 73 | 3d | 38 | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . |   |
| 38 | 38 | 38 | 38 | 38 | 38 | 38 | 38 | 38 | 38 | 38 | 38 | 38 | 38 | 38 | 38 | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . |   |

88888

Here we see another GET Request but this one is called "trans\_cmd\_string". No idea what this did at the time, but I took note of the two arguments cmd, and command.

|     |              |                    |                           |     |
|-----|--------------|--------------------|---------------------------|-----|
| 169 | 81.854172615 | vstarcam           | android-client            | UDP |
| 170 | 81.867520643 | android-client     | vstarcam                  | UDP |
| 171 | 81.957426287 | android-client     | vstarcam                  | UDP |
| 172 | 81.969185249 | android-client     | vstarcam                  | UDP |
| 173 | 81.977845379 | android-client     | vstarcam                  | UDP |
| 174 | 81.978150064 | android-client     | vstarcam                  | UDP |
| 175 | 82.015991365 | MagicCon_00:f9:... | Spanning-tree-(for-bri... | STP |
| 176 | 82.018520257 | android-client     | vstarcam                  | UDP |
| 177 | 82.018771297 | android-client     | vstarcam                  | UDP |

me 169: 122 bytes on wire (976 bits), 122 bytes captured (976 bits) on interface II, Src: vstarcam (48:02:2a:0b:db:b4), Dst: android-client (44:91:60:01:75:37), Ethernet Protocol Version 4, Src: vstarcam (192.168.11.140), Dst: android-client (192.168.11.113), Internet Protocol Version 4, Src Port: 12669 (12669), Dst Port: 11618 (11618)

a (76 bytes)

|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 44 | 91 | 60 | e4 | e9 | 28 | 48 | 02 | 2a | 0b | db | b4 | 08 | 00 | 45 | 00 | D | . | . | . | ( | H | . | * | . | . | . | E | . |   |
| 00 | 68 | 00 | 00 | 40 | 00 | 40 | 11 | a2 | 37 | c0 | a8 | 0b | 8c | c0 | a8 | . | h | . | . | @ | . | @ | . | . | . | 7 | . | . | . |
| 0b | 71 | 31 | 7d | 2d | 62 | 00 | 54 | 50 | db | f1 | d0 | 00 | 48 | d1 | 00 | . | q | 1 | } | - | b | . | T | P | . | . | . | H | . |
| 00 | 00 | 01 | 0a | a0 | 60 | 3c | 00 | 00 | 01 | 72 | 65 | 73 | 75 | 6c | 74 | . | . | . | . | . | . | . | . | . | . | . | . | . | . |
| 3d | 20 | 30 | 3b | 0d | 0a | 76 | 61 | 72 | 20 | 63 | 75 | 72 | 72 | 65 | 6e | . | . | . | . | . | . | . | . | . | . | . | . | . | . |
| 74 | 5f | 75 | 73 | 65 | 72 | 73 | 3d | 31 | 3b | 0d | 0a | 76 | 61 | 72 | 20 | . | . | . | . | . | . | . | . | . | . | . | . | . | . |
| 6d | 61 | 78 | 5f | 73 | 75 | 70 | 70 | 6f | 72 | 74 | 5f | 75 | 73 | 65 | 72 | . | . | . | . | . | . | . | . | . | . | . | . | . | . |
| 73 | 3d | 34 | 3b | 0d | 0a | f8 | f3 | 99 | 3c | .  | .  | .  | .  | .  | .  | . | . | . | . | . | . | . | . | . | . | . | . | . | . |

.....(H \*.....E.  
 .h...@...@...7.....  
 .q1}-b.T P...H..  
 .....<...result  
 = 0;...va r curren  
 t\_users= 1;...var  
 max\_supp ort\_user  
 s=4;... <

We see a standard reply, again, comes with a special 16 byte header, and what seems to be some sort of Javascript or some sort of configuration. This was the reply to check\_users.cgi.

|     |              |                |          |     |
|-----|--------------|----------------|----------|-----|
| 172 | 81.969185249 | android-client | vstarcam | UDP |
| 173 | 81.977845379 | android-client | vstarcam | UDP |
| 174 | 81.978150064 | android-client | vstarcam | UDP |

ernet II, Src: android-client (44:91:60:e4:e9:28), Dst: vstarcam (48:02:2a:0b:db:b4), Ethernet Protocol Version 4, Src: android-client (192.168.11.113), Dst: vstarcam (192.168.11.140), Internet Protocol Version 4, Src Port: 11618 (11618), Dst Port: 12669 (12669)

(345 bytes)

|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 48 | 02 | 2a | 0b | db | b4 | 44 | 91 | 60 | e4 | e9 | 28 | 08 | 00 | 45 | 00 | H | . | * | . | . | D | . | . | . | . | . | . | . | . |
| 01 | 75 | 37 | cc | 40 | 00 | 40 | 11 | 69 | 5e | c0 | a8 | 0b | 71 | c0 | a8 | . | u | 7 | . | @ | . | @ | . | . | . | . | . | . | . |
| 0b | 8c | 2d | 62 | 31 | 7d | 01 | 61 | d2 | 9a | f1 | d0 | 01 | 55 | d1 | 00 | . | . | . | . | . | . | . | . | . | . | . | . | . |   |
| 00 | 03 | 01 | 0a | 00 | 00 | 4f | 00 | 00 | 00 | 47 | 45 | 54 | 20 | 2f | 67 | . | . | . | . | . | . | . | . | . | . | . | . | . |   |
| 65 | 74 | 5f | 66 | 61 | 63 | 74 | 6f | 72 | 79 | 5f | 70 | 61 | 72 | 61 | 6d | . | . | . | . | . | . | . | . | . | . | . | . | . |   |
| 2e | 63 | 67 | 69 | 3f | 6c | 6f | 67 | 69 | 6e | 75 | 73 | 65 | 3d | 61 | 64 | . | . | . | . | . | . | . | . | . | . | . | . | . |   |
| 6d | 69 | 6e | 26 | 6c | 6f | 67 | 69 | 6e | 70 | 61 | 73 | 3d | 38 | 38 | 38 | . | . | . | . | . | . | . | . | . | . | . | . | . |   |
| 38 | 38 | 38 | 26 | 75 | 73 | 65 | 72 | 3d | 61 | 64 | 6d | 69 | 6e | 26 | 70 | . | . | . | . | . | . | . | . | . | . | . | . | . |   |
| 77 | 64 | 3d | 38 | 38 | 38 | 38 | 38 | 38 | 01 | 0a | 00 | 00 | 48 | 00 | 00 | . | . | . | . | . | . | . | . | . | . | . | . | . |   |
| 00 | 47 | 45 | 54 | 20 | 2f | 67 | 65 | 74 | 5f | 70 | 61 | 72 | 61 | 6d | 73 | . | . | . | . | . | . | . | . | . | . | . | . | . |   |
| 2e | 63 | 67 | 69 | 3f | 6c | 6f | 67 | 69 | 6e | 75 | 73 | 65 | 3d | 61 | 64 | . | . | . | . | . | . | . | . | . | . | . | . | . |   |
| 6d | 69 | 6e | 26 | 6c | 6f | 67 | 69 | 6e | 70 | 61 | 73 | 3d | 38 | 38 | 38 | . | . | . | . | . | . | . | . | . | . | . | . | . |   |
| 38 | 38 | 38 | 26 | 75 | 73 | 65 | 72 | 3d | 61 | 64 | 6d | 69 | 6e | 26 | 70 | . | . | . | . | . | . | . | . | . | . | . | . | . |   |
| 77 | 64 | 3d | 38 | 38 | 38 | 38 | 38 | 38 | 01 | 0a | 00 | 00 | 4d | 00 | 00 | . | . | . | . | . | . | . | . | . | . | . | . | . |   |
| 00 | 47 | 45 | 54 | 20 | 2f | 73 | 6e | 61 | 70 | 73 | 68 | 6f | 74 | 2e | 63 | . | . | . | . | . | . | . | . | . | . | . | . | . |   |
| 67 | 69 | 3f | 26 | 72 | 65 | 73 | 3d | 31 | 26 | 6c | 6f | 67 | 69 | 6e | 75 | . | . | . | . | . | . | . | . | . | . | . | . | . |   |
| 73 | 65 | 3d | 61 | 64 | 6d | 69 | 6e | 26 | 6c | 6f | 67 | 69 | 6e | 70 | 61 | . | . | . | . | . | . | . | . | . | . | . | . | . |   |
| 73 | 3d | 38 | 38 | 38 | 38 | 38 | 38 | 26 | 75 | 73 | 65 | 72 | 3d | 61 | 64 | . | . | . | . | . | . | . | . | . | . | . | . | . |   |
| 6d | 69 | 6e | 26 | 70 | 77 | 64 | 3d | 38 | 38 | 38 | 38 | 38 | 38 | 01 | 0a | . | . | . | . | . | . | . | . | . | . | . | . | . |   |
| 00 | 00 | 4d | 00 | 00 | 00 | 47 | 45 | 54 | 20 | 2f | 73 | 6e | 61 | 70 | 73 | . | . | . | . | . | . | . | . | . | . | . | . | . |   |
| 68 | 6f | 74 | 2e | 63 | 67 | 69 | 3f | 72 | 65 | 73 | 3d | 31 | 26 | 6c | 6f | . | . | . | . | . | . | . | . | . | . | . | . | . |   |
| 67 | 69 | 6e | 75 | 73 | 65 | 3d | 61 | 64 | 6d | 69 | 6e | 26 | 6c | 6f | 67 | . | . | . | . | . | . | . | . | . | . | . | . | . |   |
| 69 | 6e | 70 | 61 | 73 | 3d | 38 | 38 | 38 | 38 | 38 | 38 | 26 | 75 | 73 | 65 | . | . | . | . | . | . | . | . | . | . | . | . | . |   |
| 72 | 3d | 61 | 64 | 6d | 69 | 6e | 26 | 70 | 77 | 64 | 3d | 38 | 38 | 38 | 38 | . | . | . | . | . | . | . | . | . | . | . | . | . |   |
| 38 | 38 | 26 | .  | .  | .  | .  | .  | .  | .  | .  | .  | .  | .  | .  | .  | . | . | . | . | . | . | . | . | . | . | . | . | . | . |

This is another oddity, some times the GET request would have multiple stacked inside, starting with a 16 byte header, and padding each request with 8 bytes in the front.

What did we learn?

The device has some sort of HTTP service open, but the port changes for some reason, most likely with reboot. The device has some strange UDP protocol that encapsulates GET requests sent to the netcam. The device leaks username and password, as well as has confusing and strange argument duplication The device uses no encryption what-so-ever to encrypt video feeds and pictures sent by the device to the client. UDP GET requests have a special packet header The protocol is written in faulty ways, exemplified by the duplication of elements, strange request packing, and odd behavior.

TAGGED SKILLS:

 Hardware Security

 Reverse Engineering

 Security Research