EZVIZ camera
Model: CS-C6-21WFR-8
Firmware: V5.2.7 build 170628

The “davinci” application in the firmware does not correctly verify the server certificate. Therefore, the SSL/TLS connection between the camera and the cloud server can be decrypted with a man in the middle attack using a self-signed certificate.

This is caused by the application only calls SSL_CTX_set_verify(SSL_CTX *ctx, int mode, SSL_verify_cb verify_callback) to verify the server certificate. However, the application passes “SSL_VERIFY_NONE” to the mode flag which configure the application does not check the server certificate.

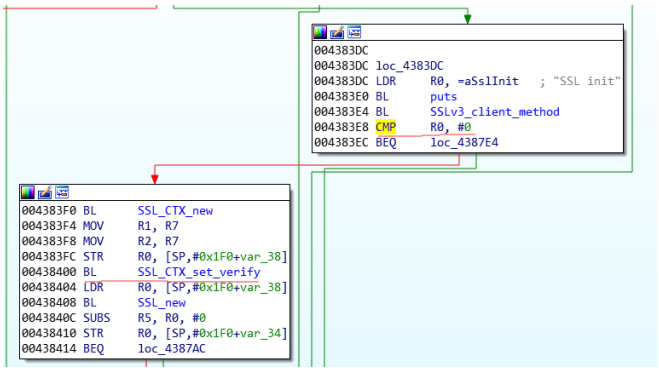

Figure 1

The following figure shows we can decrypt the traffic with Man-in-the-middle attack.

```
濫  SOH NUL NUL NUL NUL NUL NUL ETX <BS BEL NUL NUL (3 NUL NUL NUL NUL NUL NUL NUL xE8 NUL NUL NUL NUL
<?xml version="1.0" encoding="utf-8"?>
<Request>
    <DevSerial>807605214</DevSerial>
    <Authorization>f0b061c867eb4d058654264c9b30bbf9</Authorization>
    <Verificationcode>c527fd069fc5cedc4c2727a8bc373fa8</Verificationcode>
</Request>
8bcaf74f44cd4247f539cd68b6caa0d2


濫  SOH NUL NUL NUL NUL NUL NUL ETX NUL NUL NUL NUL NUL NUL (4 NUL NUL NUL NUL NUL NUL NUL x81 NUL NUL NUL NUL
<?xml version="1.0" encoding="utf-8" ?>
<Response>
    <Result>0</Result>
    <Crypto Algorithm="AES128" Key="3c3e4c92fede4d09"/>
</Response>
016c25cb662ab4b55886d9cb0be91989
```

Figure 2

We further analyze the detailed workflow of the device authentication process (see the black lines). There are mainly four steps. (1) The camera communicates with server 1 (fixed IP address) to request the IP addresses of other two servers: cloud server 2 and cloud server 3. (2) It attempts to request an authorization code from server 2 using the device identity information. The authorization code is encrypted using AES-128-CBC. (3) The device decrypts the authorization code and sends the decrypted authorization code to server 3 to get an AES-128 key which is used for the encryption and decryption of the following communication. (4) Finally, the device finishes the authentication to the cloud servers by sending some other device information to server 3.

By intercepting the device identity information which is transmitted in the second step of the device authentication process, we could create a fake device for the victim one and authenticate it to the

cloud servers. The workflow of our phantom device attack is illustrated in the red lines of Figure 3. In the first step, our phantom device requests to the server 1 to get the addresses of server 2 and server 3. In the second step, our phantom device sends the intercepted identity information of the victim device to server 2 and gets an encrypted authorization code. By reversing the binary, we find that the key and the initialization vector used for decrypting the authorization code are a constant string of 8 characters and a constant string of 32 characters, respectively. Therefore, we successfully get the decrypted authorization code. In the third step, the phantom device requests an AES key from server 3 with the decrypted authorization code and proceeds the forth step with server 2. After these four steps, the cloud servers would recognize our phantom device as the victim one. Note that even the victim device has been authenticated with the cloud, our attack shows that the phantom device could still replace the victim device in the cloud.

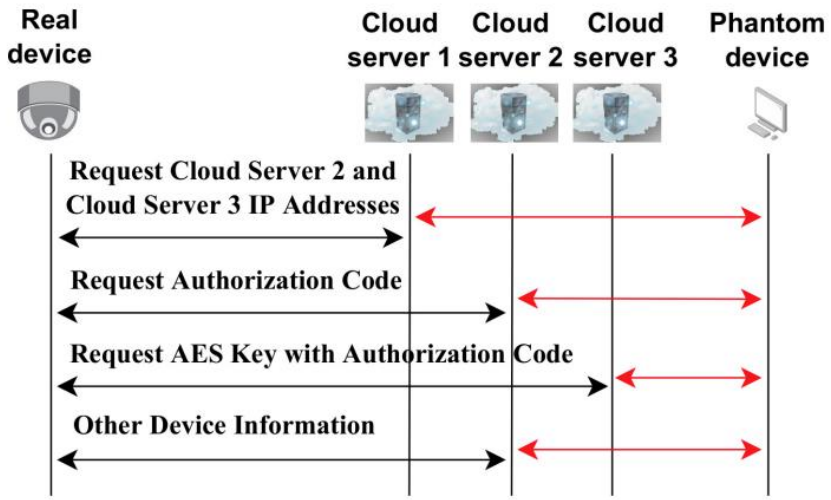

Figure 3