All Rights Reserved. © 2024 Bitdefender.
All trademarks, trade names, and products referenced herein are the property of their respective owners.

Bitdefender

PUBLIC REPORT

Vulnerabilities Identified in Owlet Cam

Trusted. Always.

Contents

As the creator of the world's first smart home cybersecurity hub, Bitdefender regularly audits popular IoT hardware for vulnerabilities. This research paper is part of a broader program that aims to shed light on the security of the world's best-sellers in the IoT space. This report covers vulnerabilities discovered while researching Owlet Cam next-generation smart HD video baby monitor.

NOTE: the vulnerabilities presented in this paper have been responsibly disclosed to the affected vendor. An automatic device update to firmware versions 4.2.11 (Cam V1) and 4.2.10 (Cam V2) fixes the issues.

Vulnerabilities at a glance

- Bitdefender researchers have identified three vulnerabilities in a communication framework called ThroughTek Kalay (TUTK). This solution is used in a variety of IoT devices, including Owlet Cam 2.
- CVE-2023-6323 allows a local attacker to leak the **AuthKey** secret by impersonating the P2P cloud server used by the device.
- CVE-2023-6324 leverages a vulnerability where a local attacker can infer the pre-shared key for a DTLS session by forcing an empty buffer.
- CVE-2023-6321 allows an authenticated user to run system commands as the root user leading to the full compromise of the device.
- The three vulnerabilities (CVE-2023-6323, CVE-2023-6324, and CVE-2023-6321) can be chained together to allow an attacker to obtain root access from the local network and then to execute commands on the device.

Disclosure timeline

- Oct 23, 2023: Sent email asking for security team's contact information.
- Oct 30, 2023: Bitdefender submits a similar contact request via LinkedIn
- Oct 31, 2023: Bitdefender gets in touch with the vendor and submits the findings of this report
- Nov 8, 2023: The vendor confirms the validity of the issues
- Nov 17, 2023: The vendor confirms that 2 out of 3 issues have been resolved
- Jan 3, 2024: The vendor requests a 60-day extension. Bitdefender grants the request.
- May 1st, 2024: The vendor confirms that fixes have been delivered to the affected user base
- May 15, 2024: This report becomes public as part of the coordinated vulnerability disclosure process

Technical walkthrough

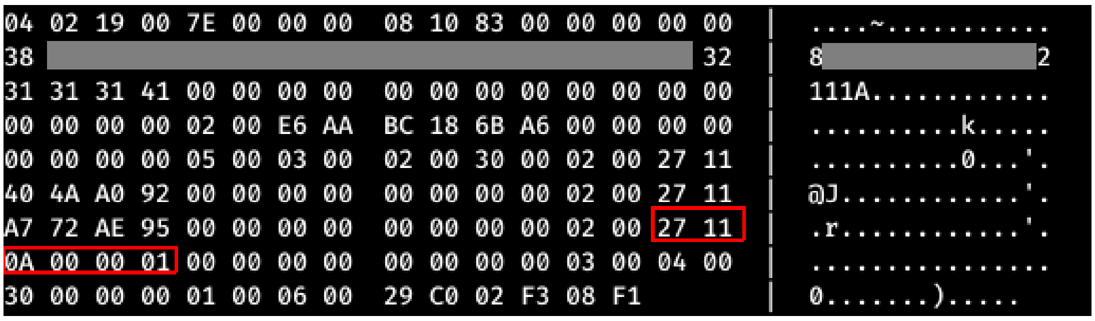
Owlet Cam 2 uses the ThroughTek Kalay solution to communicate with clients over the Internet. This functionality is implemented through the TUTK SDK. Connections from the smartphone app to the device are all handled through this service.

To connect to the IoT device, the smartphone app needs to know a secret string called **AuthKey**. The device will refuse any connection that does not have the correct key. However, an attacker can leak the AuthKey using the approach described in CVE-2023-6323:

First, we need to obtain the P2P ID of the device. This is straightforward, as the TUTK SDK service will listen for broadcast messages querying for devices (on port 63616), and responds with the device's P2P ID, local IP, and UDP port on which the SDK is listening. This port is randomized at each startup and is used to communicate with the cloud servers and local clients.

Under normal circumstances, the device communicates through UDP with several cloud servers. These servers have authority over the device and can issue commands to it. Each UDP packet exchanged represents a message and contains an ID describing its type. Among these types is message 0x1008 that tells the device to connect to an alternative P2P cloud server. The SDK does not verify the authenticity of the received messages.

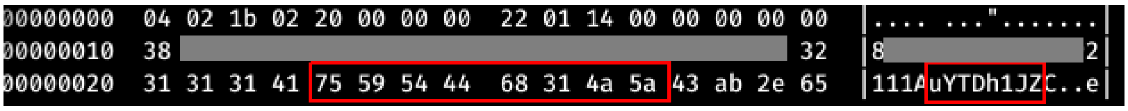
As we are in the same local network as the device, we can spoof a message with ID 0x1008 and make it appear as if the authoritative cloud server sent it. This message will instruct the SDK to connect to a server controlled by us and trust it as a P2P server, giving us control over the device. Before encoding, the spoofed packet looks like this:



```
04 02 19 00 7E 00 00 00  08 10 83 00 00 00 00 00  | ....~...........
38 [redacted] 32  | 8[redacted]2
31 31 31 41 00 00 00 00  00 00 00 00 00 00 00 00  | 111A............
00 00 00 00 02 00 E6 AA  BC 18 6B A6 00 00 00 00  | ..........k.....
00 00 00 00 05 00 03 00  02 00 30 00 02 00 27 11  | ..........0...'.
40 4A A0 92 00 00 00 00  00 00 00 00 02 00 27 11  | @J............'.
A7 72 AE 95 00 00 00 00  00 00 00 00 02 00 27 11  | .r............'.
0A 00 00 01 00 00 00 00  00 00 00 00 03 00 04 00  | ................
30 00 00 00 01 00 06 00  29 C0 02 F3 08 F1        | 0.......)....
```

The packet contains the P2P ID, obtained in the first step, [redacted in the image above for obvious reasons]. The first highlighted part, hex 0x2711, specifies the target port, and 0x0A000001 is the hex-encoded target IP (in this case our IP, 10.0.0.1).

Upon receiving this packet, the device initiates communication with the specified server. It begins by sending its **AuthKey** to the new server, that is now controlled by the third party, using the command 0x122. In some cases, it might first ask for its public IP, as part of NAT traversal functionality, through the 0x8003 command. If we respond with 0x8004 it will then send the 0x122 packet containing the key.



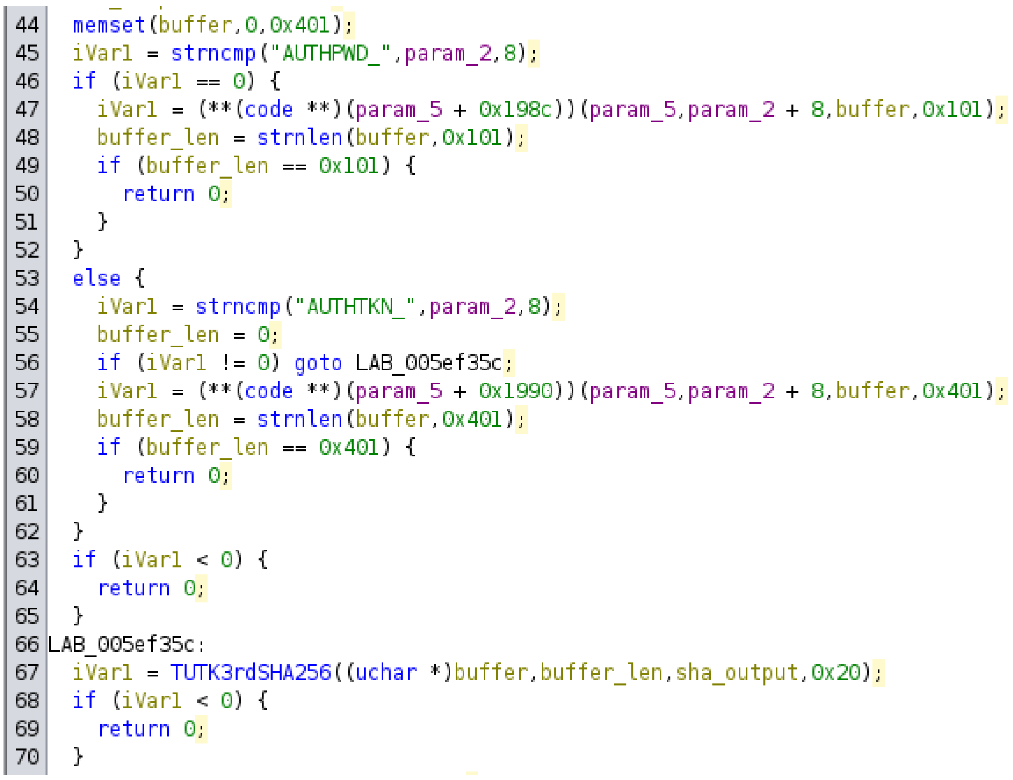
```
00000000  04 02 1b 02 20 00 00 00  22 01 14 00 00 00 00 00  |.... ...".......|
00000010  38 [redacted] 32  |8[redacted]2|
00000020  31 31 31 41 75 59 54 44  68 31 4a 5a 43 ab 2e 65  |111AuYTDh1JZC..e|
```

In this case the **AuthKey** is **uYTDh1JZ**. This key is random for each device.

Having obtained the **AuthKey**, we are now one step closer to authenticating with the device.

With the P2P ID and AuthKey now revealed, we can connect to the camera, but we cannot establish a DTLS session that is required to control the device. During normal operation, a client requires a secret value which is a random key used as a pre-shared key (PSK) when establishing the DTLS session.

There are two PSK identities that tell the SDK which PSK to use: **AUTHPWD_** and **AUTHTKN_**. For each identity the handling function will retrieve their respective secret and will compute its SHA256 hash value, which will then be used as the PSK for the DTLS session. The identity is specified by the connecting client.



```
  memset(buffer,0,0x401);
  iVar1 = strncmp("AUTHPWD_",param_2,8);
  if (iVar1 == 0) {
    iVar1 = (**(code **)(param_5 + 0x198c))(param_5,param_2 + 8,buffer,0x101);
    buffer_len = strnlen(buffer,0x101);
    if (buffer_len == 0x101) {
      return 0;
    }
  }
  else {
    iVar1 = strncmp("AUTHTKN_",param_2,8);
    buffer_len = 0;
    if (iVar1 != 0) goto LAB_005ef35c;
    iVar1 = (**(code **)(param_5 + 0x1990))(param_5,param_2 + 8,buffer,0x401);
    buffer_len = strnlen(buffer,0x401);
    if (buffer_len == 0x401) {
      return 0;
    }
  }
  if (iVar1 < 0) {
    return 0;
  }
LAB_005ef35c:
  iVar1 = TUTK3rdSHA256((uchar *)buffer,buffer_len,sha_output,0x20);
  if (iVar1 < 0) {
    return 0;
  }
```

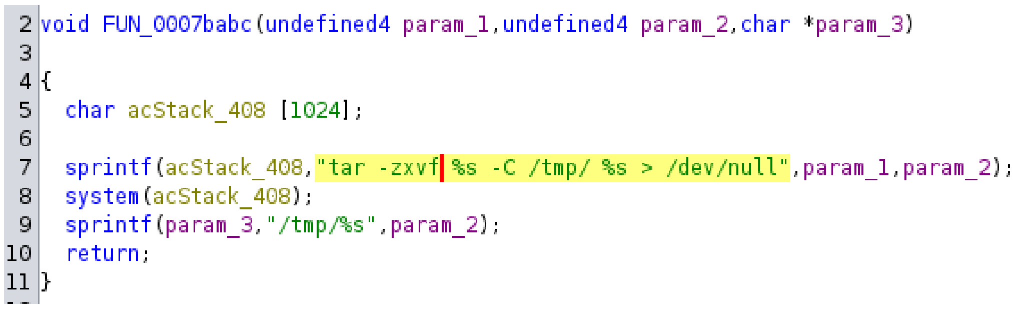
The code checking the requested PSK identity will leave the buffer empty if it encounters an identity that does not match any of the expected values. It will then perform the SHA256 hash on the empty buffer, resulting in a known hash value that will then be used as the PSK for the DTLS session.

This vulnerability, tracked as **CVE-2023-6324**, allows us to establish a DTLS session with the device without knowing the secret value. We simply use the same known hash value for the empty buffer as the PSK while specifying an invalid PSK identity.

Now that we have established a DTLS session, we have the same privileges as a legitimate client, and we can issue the same commands.

We have found an authenticated command injection vulnerability, tracked as **CVE-2023-6321**, that allows us to run system commands as the root user.

The TUTK SDK allows vendors to integrate custom handlers for actions specific to their products. These actions are triggered through special messages called IOCTLs. Only authenticated users can send those messages. A command injection vulnerability exists in the handler of IOCTL message 0x6008E, which is used to unpack archives containing OTA updates.



```
void FUN_0007babc(undefined4 param_1,undefined4 param_2,char *param_3)

{
  char acStack_408 [1024];

  sprintf(acStack_408,"tar -zxvf %s -C /tmp/ %s > /dev/null",param_1,param_2);
  system(acStack_408);
  sprintf(param_3,"/tmp/%s",param_2);
  return;
}
```

The message contains two parameters: the first one specifies the name of the archive containing the update files and the second one is hardcoded to “OTA_files_sha256.txt”. The first parameter is used as argument to a shell command without any sanitization.

By inserting shell metacharacters in the name of the archive we trigger the command injection vulnerability and execute system commands as the root user.

Bitdefender is a cybersecurity leader delivering best-in-class threat prevention, detection, and response solutions worldwide. Guardian over millions of consumer, business, and government environments, Bitdefender is one of the industry's most trusted experts for eliminating threats, protecting privacy and data, and enabling cyber resilience. With deep investments in research and development, Bitdefender Labs discovers over 400 new threats each minute and validates around 40 billion daily threat queries. The company has pioneered breakthrough innovations in antimalware, IoT security, behavioral analytics, and artificial intelligence, and its technology is licensed by more than 150 of the world's most recognized technology brands. Launched in 2001, Bitdefender has customers in 170+ countries with offices around the world.

Romania HQ
Orhideea Towers
15A Orhideelor Road,
6th District,
Bucharest 060071
T: +40 21 4412452
F: +40 21 4412453

US HQ
3945 Freedom Circle,
Suite 500, Santa Clara,
CA, 95054

bitdefender.com

Bitdefender-PReport-owlet-7745
CD-05/15/24 • MD-May 15, 2024 8:49 am