An investigation into the security of the Smarter FridgeCam

Tia C

CMP320: Ethical Hacking 3

Ethical Hacking - Year 3

2020/21

CONTENTS

1. INTRODUCTION

1.1 BACKGROUND

With thirty-one billion new internet connected devices set to be installed by the end of 2021 and one-hundred and twenty-five billion by 2030, it is clear to see that IoT devices are becoming a large part of people's everyday lives. *(How Many IoT Devices Are There in 2021? More than Ever!, 2021)* To bring the first figure into perspective, that is almost four times the world's current population (7.6 billion). The question is – are they secure?

The 'Internet of Things' (IoT) is made up of many different devices, such as smart watches, smart assistant, and even planes are all part of the IoT ecosystem. Take an ordinary item and connect it to the internet to function and you have an IoT device. There are many examples of IoT security not being fit for purpose, from 'innocent' children's toys like the Cayla Doll which has been banned in Germany due to security concerns to full industrial control systems used for transportation, manufacturing, energy, and food production. Some of these vulnerabilities include no-lockout policies and IoT systems being connected to subsystems within an organisation.

This report will focus on devices that are built to be smart home devices, specifically the Smarter FridgeCam. Most people utilise smart assistants within their home which can also be used to connect to other smart utilities such as smart bulbs and TV's. This is a lot of devices that could all be potentially accessed, if just one of the devices is successfully exploited. At home, people expect to be safe and secure but IoT devices may have a part to play in invading this.

There are countless reports of devices such as IP cameras and baby monitors being hacked. These devices allowed hackers remote access into people's bedrooms and baby rooms, even letting them speak to victims causing alarm and distress. This invades people's personal space in a very concerning and new way – which if not monitored could go un-noticed. These companies are improving their security, but users should also protect themselves by changing the default passwords and updating their devices regularly.

The Smarter FridgeCam was chosen as the subject of this report as there was not much documentation on the security of the device and its application that is used to work with it. Due to the pandemic, the tester did not have access to the hacklab where the tools and support would be readily available. Instead the tester had to carry out the testing using their own personal machine, this meant that some aspects of the testing did not go as planned.

1.2 AIM

The aim of this investigation is to examine the Smarter FridgeCam for vulnerabilities that potential attackers may attempt to take advantage of. The tester will not carry out any offensive behaviour towards the application as they do not have permission to do a penetration test. This test is simply a black box/research project to analyse the security of the device.

The tester will carry out threat-modelling of the Smarter FridgeCam, analyse it's firmware, android application and hardware aspects. The tester will then look at the results of the investigation and

research and discuss potential countermeasures that could be applied to further protect both the device and the user.

2. PROCEDURE AND RESULTS

2.1 OVERVIEW OF PROCEDURE

The tester decided to use the methodology from the *IoT Penetration Testing Cookbook (Guzman, Gupta, 2017).* This methodology was chosen as it was highly recommended from experts in the IoT field and covered many aspects of penetration testing for IoT devices. This methodology was also used as it was considered suitable for testers that have not done IoT and hardware hacking before.

The methodology consisted of:

1. *IoT Threat Modelling*
2. *Analysing and Exploiting Firmware*
3. *Exploitation of Embedded Web Applications*
4. *Exploiting IoT Mobile Applications*
5. *IoT Device Hacking*
6. *Radio Hacking*

There were sections of the methodology which the tester has left out as they are not relevant to the application that was being tested. The sections that were missed out were Exploitation of Embedded Web Applications and Radio Hacking as they were not required for the FridgeCam.

The tools that were used throughout this investigation were:

- Kali Linux
- Flashrom
- CH341A SPI Programmer USB board with SOIC clip
- Android Studio
- Rex Vulnerability Scanner
- MITRE ATT&CK
- Exploit Database
- iFixit Essential Electronics Toolkit

2.2 OVERVIEW OF SMARTER FRIDGECAM

The Smarter FridgeCam is an internet connected device that is installed into a fridge to monitor the contents within. The barcodes of items in the fridge can be added as well as use by dates, that the FridgeCam will notify users of to help reduce food waste. The Fridgecam can also add items to online shopping lists such as Tesco and Amazon Fresh. It will also connect to other IoT devices such as Alexa and IFTTT (If this then that) services, such as Siri and Google Assistant.

The FridgeCam requires users to make an account and register their fridgecam in the application. When setting the device up, it uses a technique called ‘BlinkUp’ which is to allow the device to connect to the internet. The BlinkUp method causes the screen to flash black and white, whilst the camera is facing directly into the screen. These flashes are optically transmitting data to the internal components. This

data could be a device identification token and network configuration information. *(How To Add BlinkUp To Your Mobile App | Dev Center, 2021)*

When the device has been connected to the internet, the user is taken through steps to calibrate their device in the fridge. When the device has been calibrated and attached to the inside of the fridge door, it will take its first photo and upload this to the Smarter servers. When the user opens the Smarter App, they will see the photo with a time stamp on their app. When the fridge is closed, the device enters a sleep state and disconnects from the internet.

Each time that the fridge is opened, the FridgeCam will wake up and will connect to the internet. When the device detects that the fridge is being closed, it will take a photo with the flash and upload this to the internet.

2.3 IoT Threat Modelling

Typically threat modelling is done during the development phase, however it is important for the tester to understand the attack surface and the possible threats that exist on this surface. These threats are purely theoretical and may not exist within the device.

The methodology carried out for threat modelling the FridgeCam was:

1. *Identifying the Assets*
2. *Creating an architecture overview*
3. *Decomposing the IoT device*
4. *Identifying Threats*
5. *Documenting threats*
6. *Rating the Threats*

For the threat model, the attack surface was drawn out as a map, including the functionalities and features of each of the dependencies. When the map was completed, the possible threats that have been identified were then assessed individually. The threats were then rated using the DREAD system, which is an acronym for – ***Damage Potential, Reproducibility, Exploitability, Affected Users, Discoverability.***

MITRE ATT&Ck was used to identify the possible techniques that an attacker may utilise in a threat case. MITRE ATT&CK is the new industry standard for threat categorization and is replacing Lockheed Martin's kill chain. The threat model diagrams were created with the use of draw.io.

2.2.2. Threat Modelling IoT Device

The threat modelling exercise is to gain a better understanding of how an attacker may plan to compromise the device. This is done by looking at assets, entry points and what an attacker may gain from an attack. To begin the threat modelling exercise, the investigator had to consider the device and its architecture wholly before looking at each individual area.

2.2.2.1. IDENTIFYING THE ASSETS

Looking at the FridgeCam and what it interacts with, the tester was able to determine that there were multiple assets required for the device to operate as standard. The assets were added to table 1, with a description of each asset's functionality.

Table 1 - Detailing assets of the FridgeCam

ID	Asset	Description
1	Smarter FridgeCam	Smarter FridgeCam provides a real-time camera within the fridge, takes images of the fridge when it is opened and closed, alerts users when items are going out of date, automatically adds items to amazon/Tesco shopping carts, connected to router to upload images to a cloud server.
2	Router	Router handles network communication to and from devices within the house. Firewalls can block and allow traffic to pass to and from internal and external devices.
3	Mobile Applications	Mobile applications are used to control the FridgeCam. Receives notifications from FridgeCam for various alerts and activities.

2.2.2.2. CREATING AN ARCHITECTURE OVERVIEW

Once the assets were identified, an overview of the IoT architecture was carried out to gain an understanding of how the FridgeCam worked and what technologies were being used. To begin this process, a use case for users interacting with the FridgeCam was developed, this then allowed for a diagram of the FridgeCam ecosystem to be created.

Use Case of the FridgeCam: User is viewing FridgeCam through Mobile Device.

1. User downloads app to mobile device.
2. User creates account on app and registers the device.
3. Device is connected to the router.
4. Device is installed within fridge
5. Fridge door is opened and closed, prompting camera to take a photo of inside fridge
6. Image is uploaded to the Smarter cloud server
7. Image is shown in app, with timestamp of when image was taken

The diagram of the architecture below in figure 1, demonstrates how the components interact with each other to execute the task demonstrated in the use case.

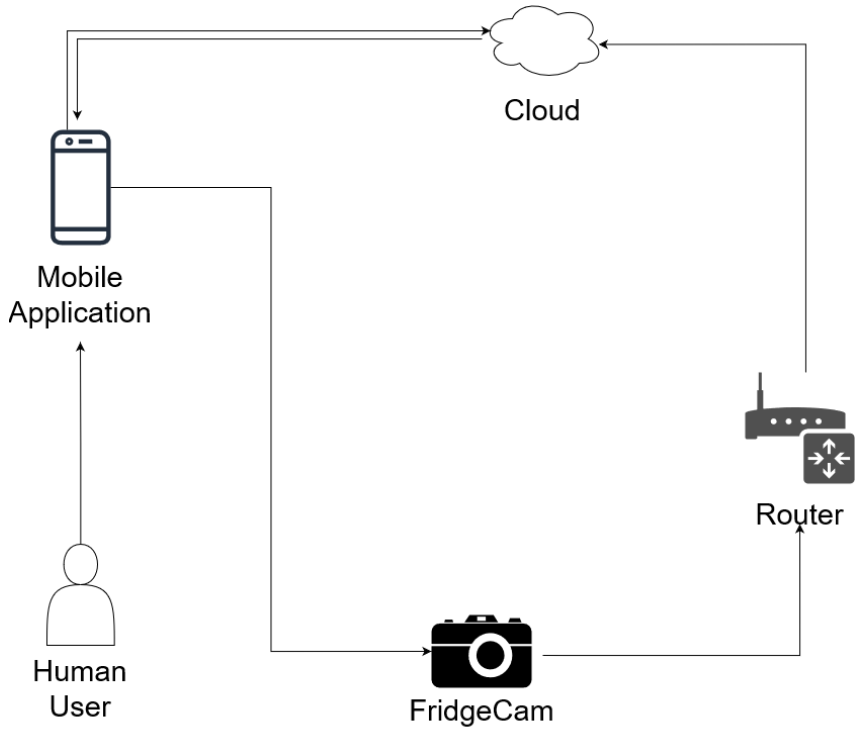

Figure 1 - Overview of FridgeCam architecture

Once the architecture had been drawn out, the tester examined the technologies that were being used. This was done to identify any potential vulnerabilities that existed within these technologies to understand possible threat vectors. The cloud has been left out of the table below, as it is out of scope of the test and falls under the category of the FridgeCam. Most of the information on the FridgeCam's technologies were publicly available, whilst information on the protocols being used to transmit data were significantly more difficult to find information about. Table 2 contains this information.

Table 2 - Table detailing FridgeCam technologies and protocols

Component/Technology	Details
FridgeCam	Communicates over secure HTTP and TCP/IP, connects to cloud server, 802.11 b/g/n standard, WPA/WPA2, only functions on 2.4GHz frequency.
Router	Sky Router, dual band (2.4GHz/5GHz), range 46m, makes use of two range boosters
Mobile Application	iOS and android apps connect to Smarter service to retrieve images of fridge. Mobile device requires internet connection to see images.
Protocol: HTTPS	Uses HTTPS to upload fridge images to Smarter Cloud server.

2.2.2.3. DECOMPOSING THE IOT DEVICE

The tester then analysed the technologies and architecture further to locate any possible entry points that may be vulnerable to a malicious actor. This allowed the investigator to comprehend the attack surface and how an attacker may gain access and/or data from the FridgeCam and its surrounding ecosystem.

The diagram of the architecture was referred to again, to decompose how the data was transmitted from the user to the FridgeCam and vice versa. These entry points and their relevant information was then compiled into the table 3.

Table 3 - Detailing possible entry points in the FridgeCam ecosystem

Entry Point	Description
Mobile App	The Mobile App is the only way for a user to access the FridgeCam. The app allows users to create and edit accounts, add more products, change network configurations, and is used to connect to the *Smarter* cloud.
FridgeCam	The FridgeCam is in an offline state until the fridge door is opened. When the door is being shut, the FridgeCam takes a photo and connects to the router, to send the image to the *Smarter* cloud.
Firmware	The firmware is encrypted, has signed boot, and executes only from on-die RAM. *(Hugo Fiennes, 2021)*
Wireless Communication	The mobile applications and device itself communicate over wireless technologies. This is over 802.11 for both the app and the device and possibly over mobile data for the app.

2.2.2.4. IDENTIFYING THREATS

The tester moved onto attempting to identify where possible risks and threats may exist within the device and application. To do this, they used the STRIDE model. The STRIDE model stands for:

- ***S**poofing Identity*
- ***T**ampering with data*
- ***R**epudiation*
- ***I**nformation disclosure*
- ***D**enial of Service*
- ***E**levation of privileges*

Threats that have been considered have been included in table 4. As the tester has access to the full schematics of the FridgeCam, they will also investigate any possible issues of the components that are installed within the device. These threats are completely theoretical and may not be present when the tester fully investigates the device.

Table 4 - Identifying threats for the FridgeCam

Threat Types	Analysis
Spoofing Identity	Attacker may be able to spoof identity through brute-forcing authentication
Tampering with Data	Protocols used by device to communicate with router and cloud server may not be secure and could lead to data tampering.
Repudiation	Attackers may be able to access areas such as entry points without being logged.
Information Disclosure	Device, application, and technologies used may be vulnerable to data leakage.
Denial of Service	Possibility of malicious actor locking out users through forgotten password abilities in application.
Elevation of Privileges	Possibility of malicious actor being able to escalate privileges in application.
Supply Chain Issues	Have full access to FridgeCam schematics, checking each component for potential vulnerabilities and issues.

2.2.2.5. DOCUMENTING THREATS

To assess the level of risk each of the threats may possess, the tester produced threat use cases. The threat cases include a description, threat target, attack technique and countermeasures to get an overall indicator of the risk level. The device only connects to the internet when the fridge is opened, leaving a small window to attack this entry point.

Table 5 - Threat Case #1

Threat description	**Attacker takes over user's account**
Threat Target	Smarter FridgeCam users, Smarter Application
Attack Techniques	Brute-forcing user credentials, socially engineering users for credentials, phishing attacks
Countermeasures	Application locks account if too many attempts, 2FA if signing in from a new device, enforcing strong password policies
MITRE ATT&CK Techniques	**Reconnaissance –** Phishing for Information(T1598), **Credential Access** – Brute Force(T1110), **Initial Access** – Phishing(T1566)

Table 6 - Threat Case #2

Threat description	**Attacker adds account to 'family' account**
Threat Target	Smarter FridgeCam users, Smarter Application
Attack Techniques	When user account takeover successful, adding own account
Countermeasures	Same as above, to prevent this threat happening
MITRE ATT&CK Techniques	**Persistence –** Create Account(T1136), **Initial Access** – Valid Accounts(T1078)

Table 7 - Threat Case #3

Threat description	**Attacker locks user out of account**
Threat Target	Smarter FridgeCam users, Smarter Application
Attack Techniques	When user account takeover successful, changing user details and email to email hacker has access to
Countermeasures	Email/SMS verification to old email/phone number for details to be changed
MITRE ATT&CK Techniques	**Persistance –** Account Manipulation(T1098),

Table 8 - Threat Case #4

Threat description	**Attacker intercepts traffic being transmitted to and from server**
Threat Target	Smarter FridgeCam users, Smarter Application, Communication Protocols
Attack Techniques	Using tools such as OWASP ZAP and burp, traffic could be intercepted, and data accessed by attacker – Credentials and/or images may be captured
Countermeasures	Secure network communications
MITRE ATT&CK Techniques	**Credential Access/Discovery –** Network Sniffing(T1040)

2.2.2.6. RATING THE THREATS

These threats will now be rated according to their DREAD ranking. DREAD standing for, ***Damage Potential, Reproducability, Exploitability, Affected Users, Discoverability.*** The scoring for each of the items are 3 being the highest risk, 2 being a medium risk and 1 being the lowest risk. The final risk rating is as follows:

- **High – 12-15**
- **Medium – 8-11**
- **Low 5-7**

The first and second threat case's ratings are shown in tables 9 to 12 below.

Table 9 - First threat case's DREAD rating

Threat Risk Rating: Attacker takes over user's account	
Item	**Score**
Damage Potential	**1**
Reproducibility	**1**
Exploitability	**2**
Affected Users	**1**
Discoverability	**1**
Risk rating score: **Low**	**6**

Table 10 - Second threat case's DREAD rating

Threat Risk Rating: Attacker adds account to 'family' account	
Item	**Score**
Damage Potential	**1**
Reproducibility	**1**
Exploitability	**1**
Affected Users	**1**
Discoverability	**1**
Risk rating score: **Low**	**5**

Table 11 - Third threat case's DREAD rating

Threat Risk Rating: Attacker locks user out of account	
Item	**Score**
Damage Potential	**1**
Reproducibility	**1**
Exploitability	**3**
Affected Users	**1**
Discoverability	**2**
Risk rating score: **Medium**	**8**

Table 12 - Fourth threat case's DREAD rating

Threat Risk Rating: Attacker intercepts traffic being transmitted to and from server	
Item	**Score**
Damage Potential	**3**
Reproducibility	**2**
Exploitability	**2**
Affected Users	**1**
Discoverability	**2**
Risk rating score: Medium	**10**

The first two threats have a low DREAD rating, which means that they are very unlikely to have a large impact however it could still cause reputational issues with the Smarter company. It is also likely to cause a large amount of distress to the victim due to having their privacy breached.

The third threat is a medium threat as it will lock the user out of the account meaning that they do not have access to their device at all. The fourth threat is also a medium risk threat, due to the damage potential it could cause, as well as the reproducibility and exploitability factor. It is not a high-risk threat as the attacker would need to be within the vicinity of either the FridgeCam or a user interacting with the application.

2.4 ANALYSING AND EXPLOITING FIRMWARE

When the threat models had been completed, the tester moved on to beginning the practical aspect of the investigation. To start, they attempted to analyse and exploit the firmware. As the firmware was not publicly available, the tester was going to try and dump the firmware directly from the device. To do this, the tester had to physically tear the FridgeCam apart to access the chip. The full teardown of the Smarter FridgeCam can be found in Appendix A.

The schematics that are available online were used to identify the components, there are two flash chips on the board. There is the MXIC MX 25L1605AM2C flash chip, which is the chip that holds the BIOS and is the target chip. There is also the Winbond W25Q64JV flash chip – these chips are identified in figure 2 with the MXIC chip in red box and the Winbond chip in the blue box.

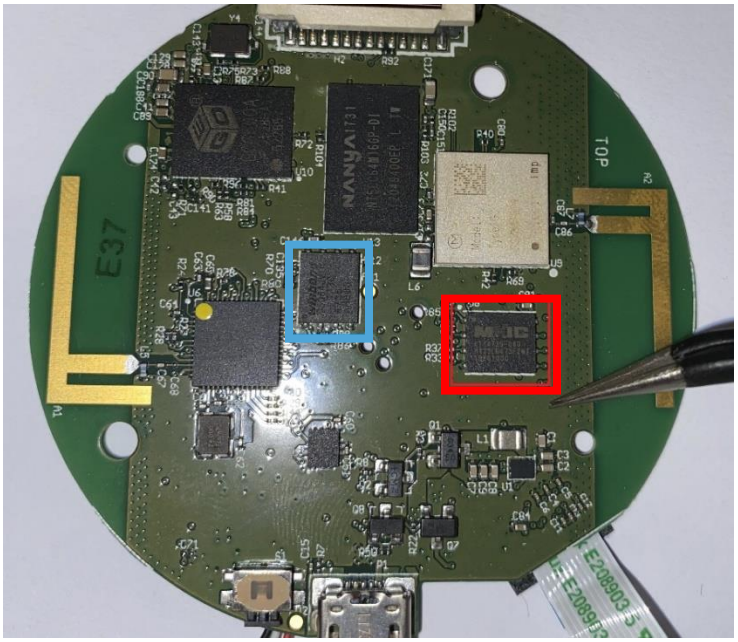

Figure 2 - SPI Chips on FridgeCam board.

When the chips were identified and labelled, the tester began the process of attempting to lift the firmware from the chip directly. To do this a CH341A SPI programmer was used, this can be seen in figure 3 with the SOIC clip attached to the MXIC chip.

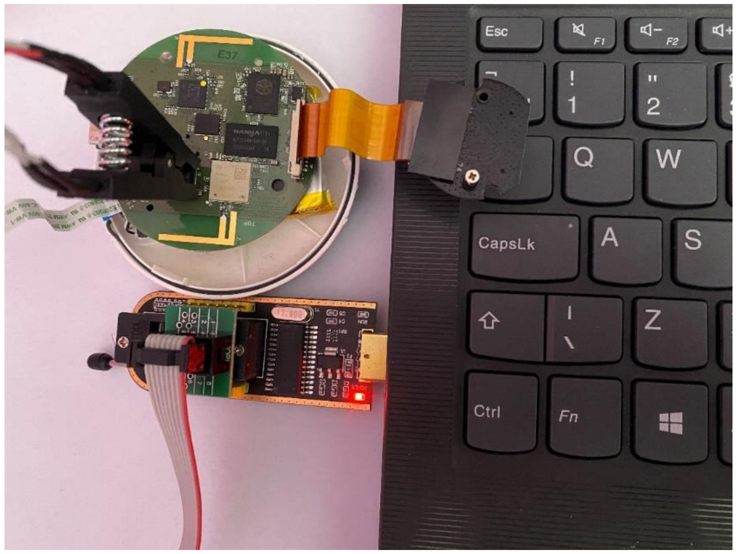

Figure 3 - SPI programmer attached to FridgeCam and plugged into computer

The Kali Linux virtual machine was used for the continuity of this section. This is due to the flashrom application on Windows having a limited number of programmers being supported. The tester then used the following command to attempt to dump the contents of the firmware from the BIOS, "sudo flashrom –programmer ch314a_spi -r spidump.bin". The programmer was recognized; however the command was throwing back an error message that the "Programmer initialization failed." This can be seen in figure 4 below.

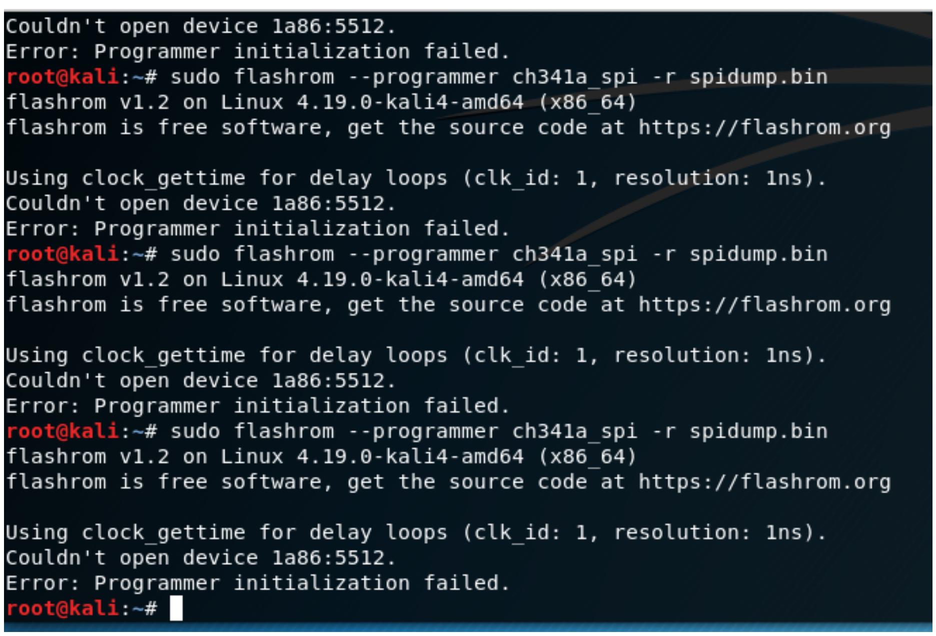

Figure 4 - Flashrom command with programmer error message

This means that the chip was not receiving power and that either the SOIC clip was not fitted properly, or the chip had not been wired into the adapter correctly. The tester made several attempts to fix this issue but unfortunately the chip was still not able to be read.

This meant that the firmware was unable to be analysed or exploited.

2.5 Analysing and Exploiting IoT Mobile Applications

Moving on from the firmware stage, the tester began to look at the Mobile Application for the FridgeCam. Due to the tester carrying out black box testing for research purposes only the source-code was analysed. As the iOS App was not obtainable for debugging and analysis, the Android APK was used instead. The APK was downloaded from APKPure which is a reputable website to get android application APK's. The version that was downloaded from APKPure was the same version as the one on the Google Play store. This is shown in figure 5.

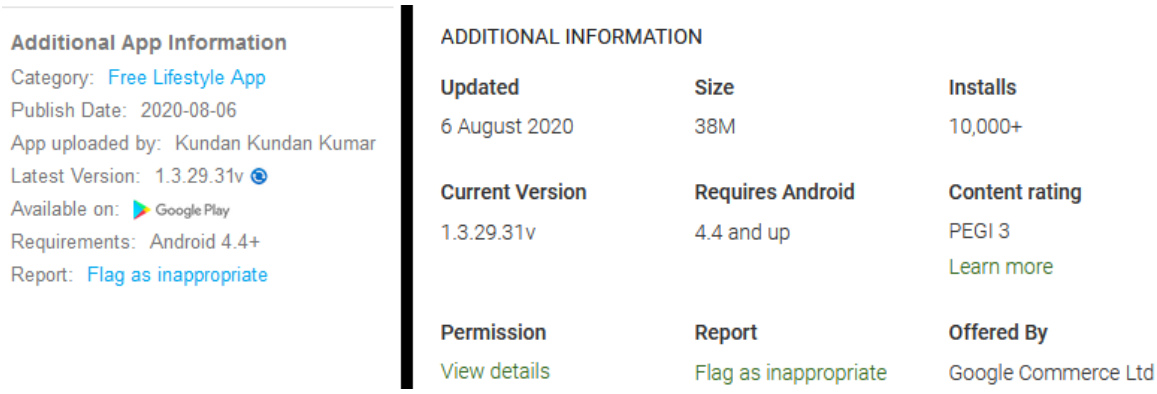

Figure 5 - Current Versions of Smarter3.0 on APKPure(L) and Google Play Store(R)

2.5.1. Static Analysis

As the version of the APK downloaded from the third-party website is the same version the tester continued with the analysis stage. Using the Rex Vulnerability Scanner tool, the tester uploaded the APK to be examined for vulnerabilities. Rex produced a report of sixty-six issues within the APK, with fifty medium and sixteen low severity vulnerabilities being identified. The full report is contained within appendix B.

Using the report as guide, the tester started collecting details about the vulnerabilities within the code using android studio. The tester was able to find table names from raw queries, that the application makes use of firebase databases and firebase messenger to send notifications. The tester also found that AES is used for the encryption of the secret key which is used when connecting the device to the smarter servers through the 'blink-up' activity. The snippets of code where these were identified can be found in Appendix C.

2.5.2. DYNAMIC ANALYSIS

The dynamic analysis stage of this process could not be carried out. This stage would have involved using OWASP ZAP to filter the traffic being sent between the app and the Smarter servers. As it is analysing the mobile layer as well as the backend services and API's used, the tester decided to err on the side of caution and ruled this section as out of scope.

2.6 IOT DEVICE HACKING

IoT Device Hacking is the hardware hacking aspect of this section. The tester has not carried out hardware hacking before and was learning whilst working through this section. This means that the results may not be representative of the actual security of the device.

2.6.1. EXTERNAL AND INTERNAL ANALYSIS OF THE DEVICE

Before any hardware hacking could take place, the tester had to research the hardware both inside and outside of the FridgeCam. This is to identify any potential vulnerabilities that an attacker may attempt to exploit.

For the external analysis, the whole of the FridgeCam was examined. The only peripheral that is available on the camera, is the micro-USB peripheral that is used to power the battery stored within – this is at the back of the device. There is also a reset button located to the left of the USB peripheral. Both can be seen in figure 6.

Figure 6 - Image of the Micro-USB peripheral and the reset button

On the top of the camera is where the camera and LED ring are located. This is protected by the plastic covering on the top of the device. There is a sticker on top to inform the user which way to install the device into the fridge. This can be seen in figure 7.

Figure 7 - Top of the FridgeCam where the camera and LED ring are situated

At the back of the device there is the brand and device information, as well as the FCC ID which if googled presents the FCC page of the FridgeCam which includes teardowns, user manuals and test results for the device. The label can be examined in figure 8.

Figure 8 - FridgeCam label with FCC ID included

For the internal analysis of the device the individual components of the FridgeCam would be looked at. As the teardown was available online and that the tester had also already taken the device apart, all that the tester had to do was identify each component. Table 13 contains the list of components and

external peripherals within the FridgeCam. The table has been colour and number coded, this relates to each component on figure 9.

Table 13 - Table listing components on the FridgeCam circuit board

Number	Component	Purpose	Colour on Figure
1	Nanya NT6DM16M32AC-T3	LPDDR1 DRAM Memory	Yellow
2	MXIC MX 25L1605AM2C	Flash EEPROM	Red
3	WinBond W25Q64JV	Serial NOR Flash	Blue
4	Electric Imp imp005	Internet-of-Things node	Orange
5	Cypress CY8C4248LQI-BL573	Semiconductor (Disabled)	Green
6	GEO GC6500A	Semiconductor	Purple
7	Micro-USB Peripheral	Power peripheral	Grey

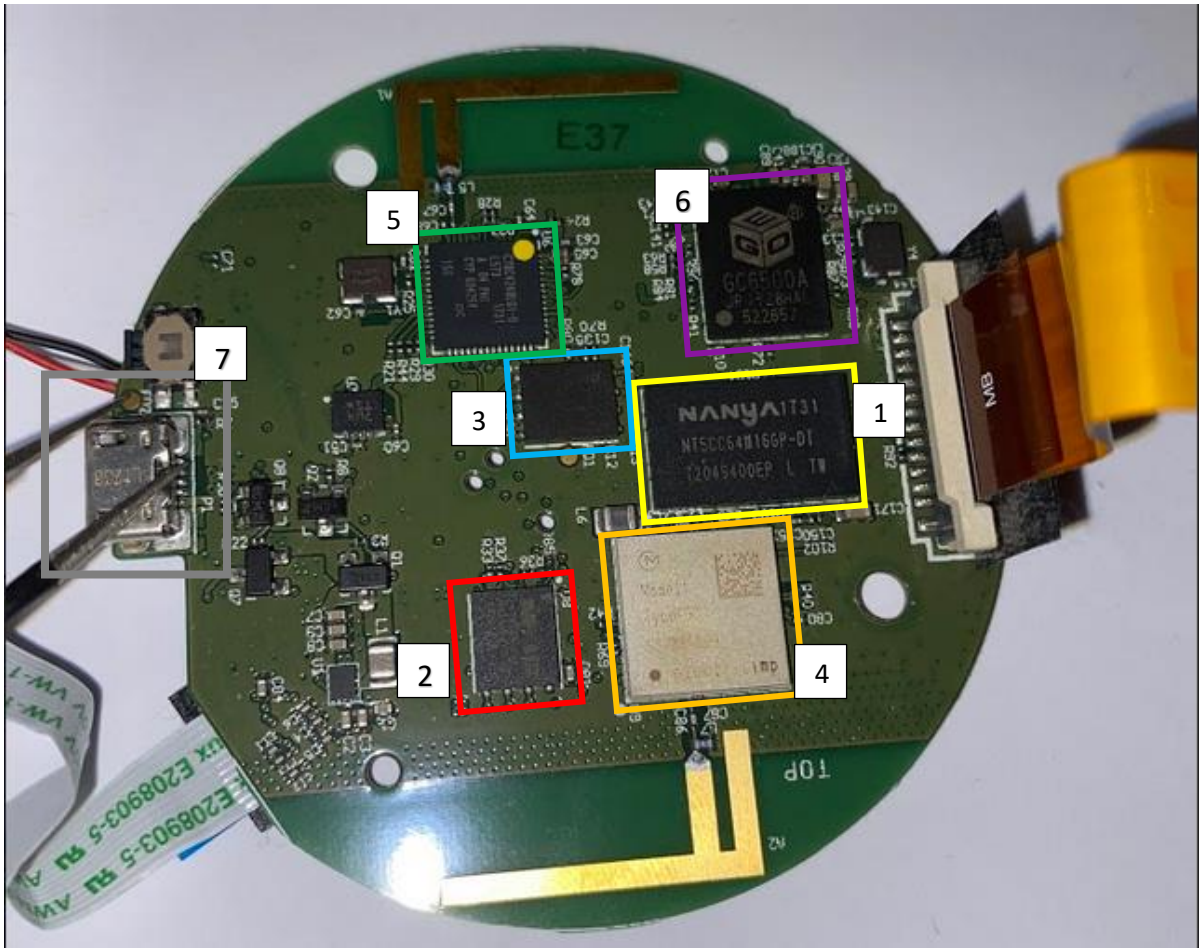

Figure 9 - Image of the circuit board with colour labels

2.6.2. IDENTIFYING COMMUNICATION INTERFACES

The Imp005 is the IoT node that allows the FridgeCam to connect to the internet. It is created by Electric Imp which specifically create secure hardware and partnered software products for IoT devices. Electric Imp has its own cloud server that data must pass through when being transmitted from the device to the customer cloud and vice versa. The data is also passed through a secure tunnel, meaning any data being uploaded and downloaded from the Smarter servers is going to be secure. *(Devices - Electric Imp, 2021)*

As well as the communication interface being secure, the imp module has also protected other hardware aspects of the device. For example, the firmware is encrypted as well as being signed with a key which is used within a secure boot process. The firmware also utilises Execute-Only memory (XOM), meaning that the firmware cannot be read or written to – only executed. *(Hugo Fiennes, Electric Imp CEO – Twitter, 2021)*

Due to this, the tester realised that the device will not be vulnerable to any exploits that they can carry out, at this time. This concluded the practical aspect of the investigation.

3. DISCUSSION

3.1 CONCLUSION

After the investigation into the Smarter FridgeCam, it can be concluded that the device's security is very good. The vulnerabilities found revolved around the Smarter 3.0 android application and most were mainly medium vulnerabilities surrounding deserialization, ciphers, cryptography, and raw queries that could be used for SQL injections. These vulnerabilities can be easily removed by changing the code where the vulnerabilities have been identified.

The hardware aspect of the device is very strong in terms of security due to the Electric Imp node that is built in. As mentioned in the IoT hacking section, the firmware is encrypted and has signed boot, as well as utilizing execute-only memory. This makes it incredibly difficult for any exploitation of the firmware. The communication process that is used to upload and view photos from the Smarter servers is also protected by Electric Imp through a secure tunnel and remote security monitoring. The device is regularly updated through Over-The-Air updates.

The first, second and third threat cases that were produced (User account takeover's and Account persistence) are still very real threats but are of a low likelihood. The last threat case is unlikely to occur due to the level of security implemented. When compared to the 'OWASP Top 10 IoT Vulnerabilities - 2018' which include:

- Weak, Guessable or Hardcoded Passwords
- Insecure Network Services
- Insecure Ecosystem Interfaces
- Lack of Secure Update Mechanism
- Use of Insecure or Outdated Components
- Insufficient Privacy Protection
- Insecure Data Transfer and Storage
- Lack of Device Management
- Insecure Default Settings
- Lack of Physical Hardening

The FridgeCam contained none of these vulnerabilities *(OWASP Internet of Things Project - OWASP, 2021).* Overall, the Smarter FridgeCam is a secure product and protects itself and users from a multitude of vulnerabilities that are commonly seen in other IoT devices.

3.2 COUNTERMEASURES

Even though the FridgeCam's security was very good, there are some countermeasures that every IoT device should implement to ensure a basic level of security.

Vulnerability Scanners

Vulnerability Scanners are incredibly useful for sniffing out potential vulnerabilities that humans may not be able to identify. There are scanners for both the firmware and mobile application of an IoT device.

For firmware scanning, there is FlawFinder which is a static code analysis tool that looks for security vulnerabilities in the firmware code.

For mobile applications there are many security scanners to choose from, for example there is Codified which is an iOS security scanner and REX (which was used in this investigation) which is an android security scanner. There are free and paid versions of each of these scanners, the paid versions typically allow for more scanning time, dynamic tests, etc.

Changing default password

Most IoT devices are shipped out to users with a default password. These users subsequently do not change the default password, meaning that attackers may be able to remotely access the devices. An example of this is the Mirai malware that searches the internet for vulnerable IoT devices, infecting them and adding them to a network of botnets. There is currently a new variant called Mukashi that is brute forcing devices with default passwords and adding them to their range of botnets. *(Trend Micro, 2021)*

To prevent devices being attacked in this way, the devices should require users to change the password of the device before it is deployed within their home. This way the credentials are not public and guessable, meaning there is less chance of the devices being affected by threats such as the Mirai malware.

Secure network communication

As the device is connected to the internet and is going to be transferring data back and forth between the user and the application endpoint, it is important the route is secure. If the communication is not secure, it is possible for attackers to sniff the network for personal data and credentials – this is a very dangerous vulnerability.

The network should be fully protected, ports should be closed when not in use and only opened when necessary, firewalls and intrusion detection systems should be in place as well as blocking unauthorized IP addresses from accessing the network. The network should also make sure that HTTPS is being used rather than HTTP, as well as TLS and SSL certificates to ensure that the data is not being sniffed or tampered with.

Security Monitoring

Malicious actors will attempt many ways of trying to access and interact with IoT devices. If there is not monitoring of the networks, these attempts may go unnoticed and allow the attacker to access the network.

Security monitoring should be implemented to deter and prevent these attackers. There are many different tools available such as Splunk that can be installed. These tools allow proactive approaches such as rules being made like a firewall to detect threats, there is also reactive analysis when an alert goes off to identify the reason for the alert such as on-going attacks or abnormal behaviour within a network.

Updating software regularly

Vulnerabilities can arise at any point, as well as new threats to devices. If the device's software is not updated regularly, this means that it may become increasingly insecure as time goes on. This will allow

attackers and other threats to be able to exploit these vulnerabilities causing damage and disrepute to the company as well as invading customer's privacy.

To ensure that the device is being updated regularly, over the air updates are available – this means that it does not rely on the user to be updated. The over-the-air update is deployed by the vendor and is automatically installed onto the device. This ensures that security patches are also installed, preventing possible vulnerabilities from surfacing.

3.3 FUTURE WORK

If the tester was going to repeat such an investigation, they would approach the company directly and ask for permission to carry out a white or grey-box test instead. This would have allowed for more testing and a more thorough investigation into the FridgeCam's security.

The tester would have also undertaken more learning surrounding hardware hacking and IoT technologies. This would possibly have allowed for a more in-depth investigation, however the device has been secured incredibly well, so it is likely the results would have stayed the same.

REFERENCES

APKPure. 2021. *Download APK free online downloader*. [online] Available at: <https://m.apkpure.com> [Accessed 13 May 2021].

App.rexscan.com. 2021. *REX*. [online] Available at: <https://app.rexscan.com> [Accessed 13 May 2021].

Bios-mods.com. 2021. *[GUIDE] Flash BIOS with CH341A programmer*. [online] Available at: <https://www.bios-mods.com/forum/Thread-GUIDE-Flash-BIOS-with-CH341A-programmer> [Accessed 14 May 2021].

Digipart.com. 2021. *GC6500A Price & Stock | DigiPart*. [online] Available at: <https://www.digipart.com/part/GC6500A> [Accessed 15 May 2021].

Electric Imp. 2021. *imp005 - Electric Imp*. [online] Available at: <https://www.electricimp.com/resourceset/imp005/> [Accessed 15 May 2021].

FCCid. 2021. *Smarter FridgeCam Application*. [online] Available at: <https://fccid.io/2AKC5-SFC01> [Accessed 19 April 2021].

Geekflare. 2021. *11 Mobile App Scanner to Find Security Vulnerabilities*. [online] Available at: <https://geekflare.com/mobile-app-security-scanner/> [Accessed 16 May 2021].

Mouser Electronics. 2021. *Cypress Semiconductor CY8C4248LQI-BL573*. [online] Available at: <https://www.mouser.co.uk/ProductDetail/Cypress-Semiconductor/CY8C4248LQI-BL573?qs=QTajy0ParP5P3mgeM1iWeQ%3D%3D> [Accessed 15 May 2021].

NT6DM16M32AC-T3, N., 2021. *Nanya NT6DM16M32AC-T3*. [online] Memory-distributor.com. Available at: <https://www.memory-distributor.com/nt6dm16m32ac-t3.html> [Accessed 15 May 2021].

Pentestpartners.com. 2021. *Finding wireless kettles with social networks | Pen Test Partners*. [online] Available at: <https://www.pentestpartners.com/security-blog/finding-wireless-kettles-with-social-networks/> [Accessed 18 April 2021].

Pentestpartners.com. 2021. *Hacking kettles & extracting plain text WPA PSKs. Yes really! | Pen Test Partners*. [online] Available at: <https://www.pentestpartners.com/security-blog/hacking-kettles-extracting-plain-text-wpa-psks-yes-really/> [Accessed 18 April 2021].

Smarter. 2021. *Smarter FridgeCam*. [online] Available at: <https://smarter.am/products/smarter-fridgecam> [Accessed 18 April 2021].

TechJury. 2021. *How Many IoT Devices Are There in 2021? More than Ever!*. [online] Available at: <https://techjury.net/blog/how-many-iot-devices-are-there> [Accessed 14 May 2021].

Useche, A., 2021. *Intro to Hardware Hacking - Dumping your First Firmware*. [online] Blog.nvisium.com. Available at: <https://blog.nvisium.com/intro-to-hardware-hacking-dumping-your-first-firmware> [Accessed 15 May 2021].

veswin.com. 2021. *MX25L1605AM2C-15G MXIC | MX25L1605AM2C-15G Electronic Chips Distributor | MX25L1605AM2C-15G MX29F016T4C-90 MX29LV320ETTI-70G MX25L8005MI-12G MX66C1024MC-70*

stock available from veswin.com. [online] Available at: <https://www.veswin.com/product-MX25L1605AM2C-15G.html> [Accessed 15 May 2021].

Wiki.owasp.org. 2021. *OWASP Internet of Things Project - OWASP*. [online] Available at: <https://wiki.owasp.org/index.php/OWASP_Internet_of_Things_Project#tab=IoT_Top_10> [Accessed 16 May 2021].

Win-raid.com. 2021. *[GUIDE] The Beginners Guide to Using a CH341A SPI Programmer/Flasher (With Pictures!)*. [online] Available at: <https://www.win-raid.com/t4287f16-GUIDE-The-Beginners-Guide-to-Using-a-CH-A-SPI-Programmer-Flasher-With-Pictures.html> [Accessed 14 May 2021].

Winbond.com. 2021. *W25Q64JV - Serial NOR Flash - Code Storage Flash Memory - Winbond -*. [online] Available at: <https://www.winbond.com/hq/product/code-storage-flash-memory/serial-nor-flash/?__locale=en&partNo=W25Q64JV> [Accessed 15 May 2021].

APPENDICES

APPENDIX A - FULL TEAR DOWN OF SMARTER FRIDGE CAM

Figure 10 - Bottom of FridgeCam

Figure 11 - Top of FridgeCam

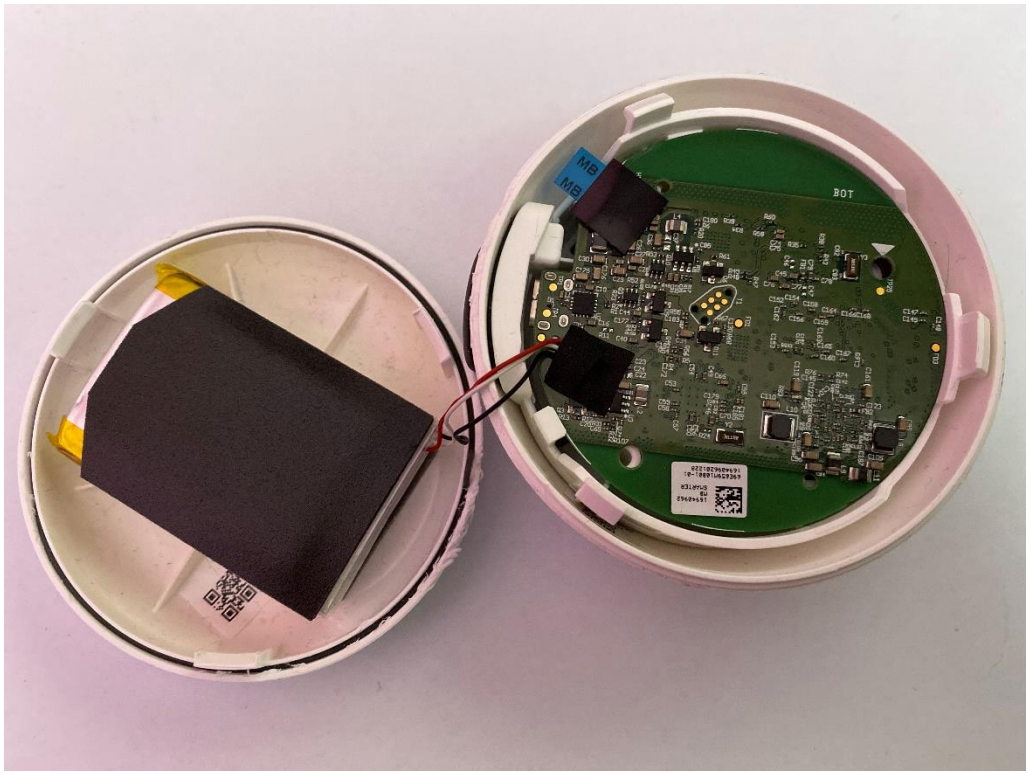

Figure 12 - Inside of FridgeCam displayed chip board and battery

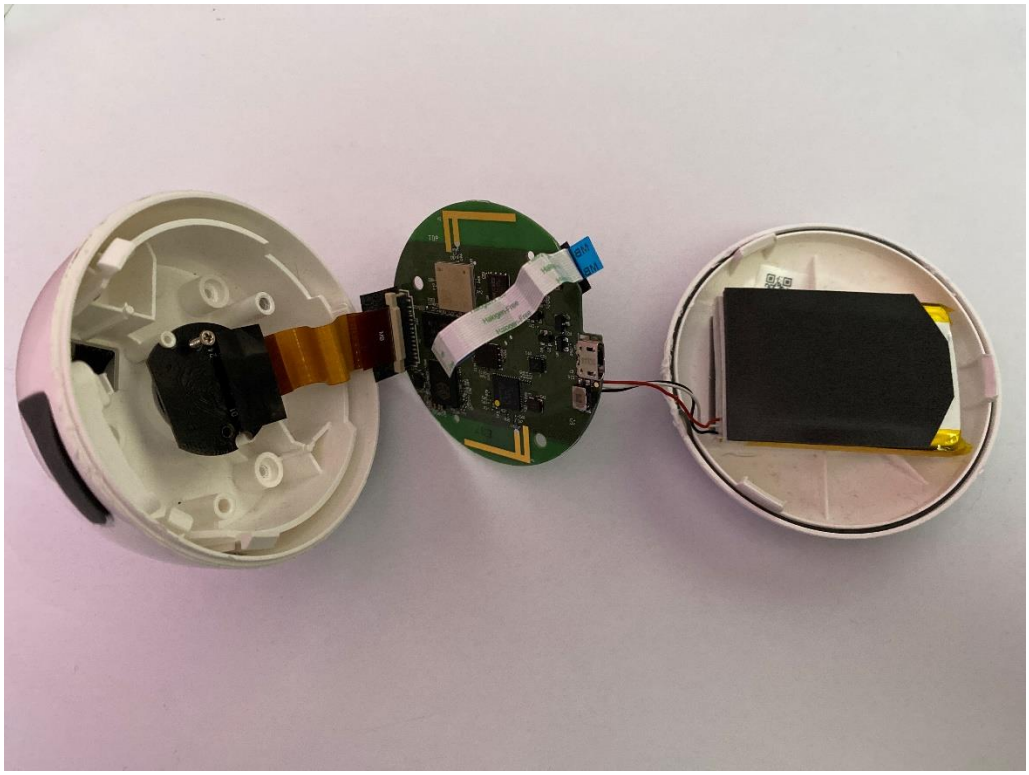

Figure 13 - Camera, chip board and battery within FridgeCam

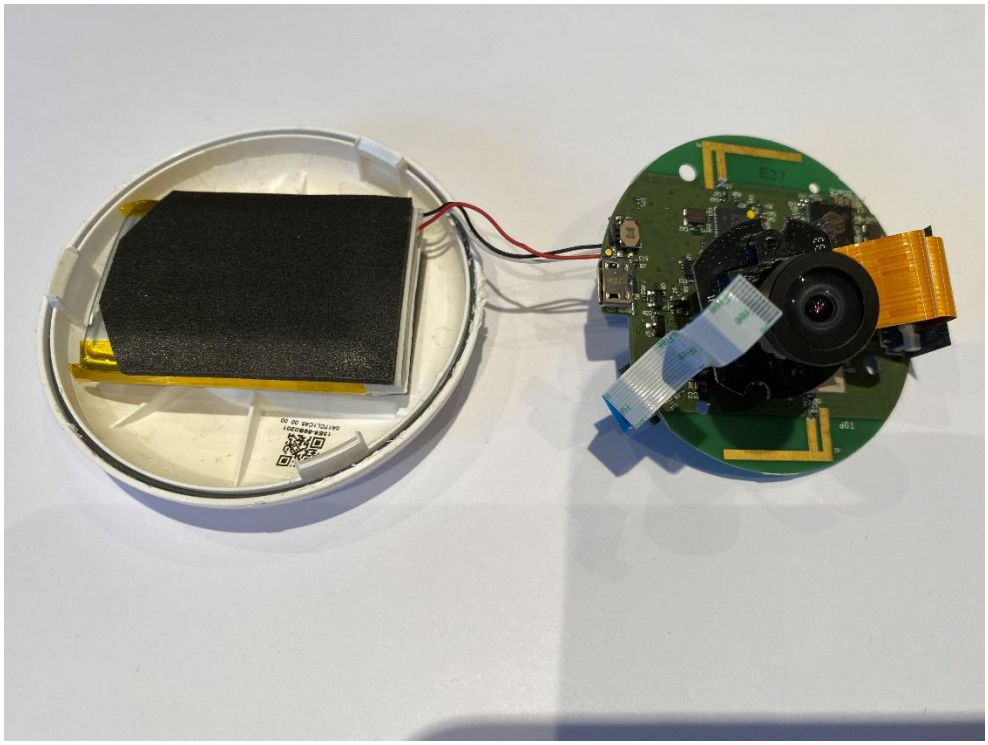

Figure 14 - Outer housing removed from FridgeCam

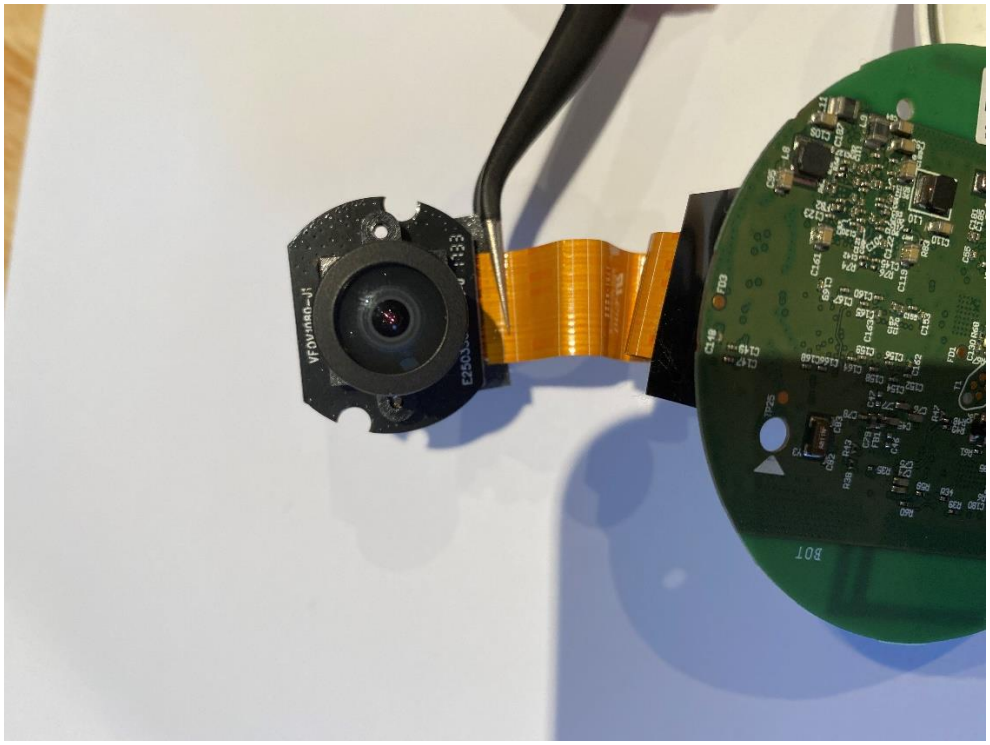

Figure 15 - Closer look at the camera used within the FridgeCam

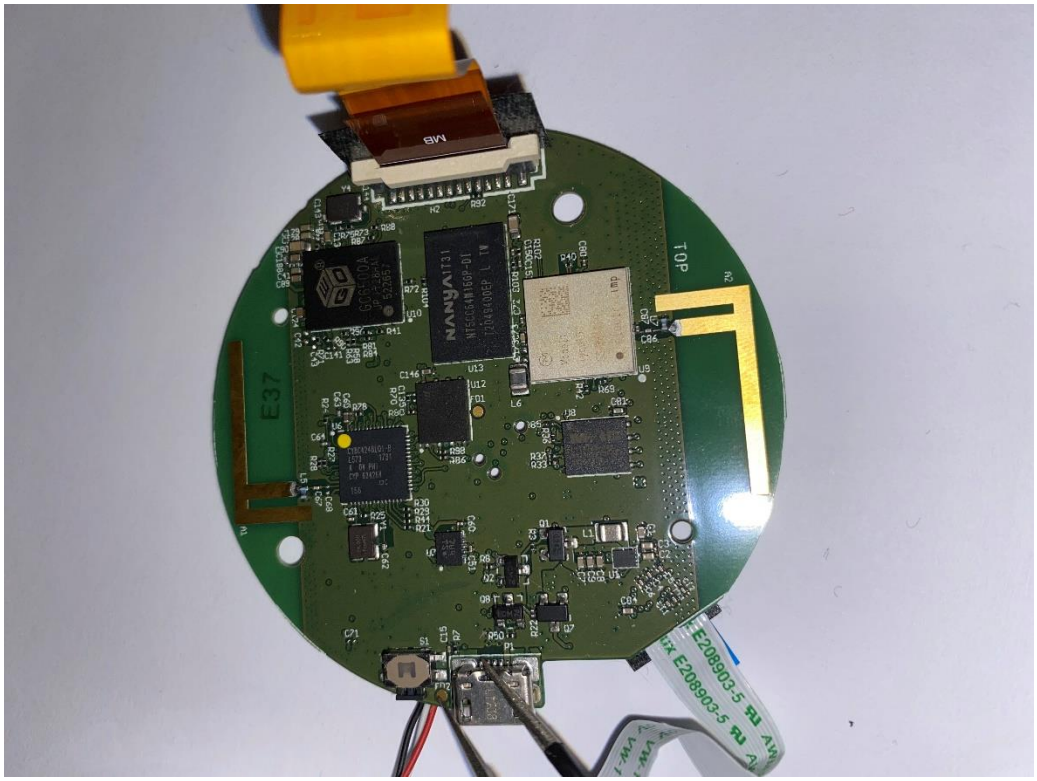

Figure 16 - Closer look at the chip board

Figure 17 - LED's used for flash

APPENDIX B - ANDROID APK VULNERABILITY SCANNER

Test Details

Status	**Complete**
Date Created	May 13th 2021, 22:23
Issues	66
Time Taken	765 seconds

Object deserialization is used in androidx.versionedparcelable.VersionedParcel.readSerializable()	10: Medium	1: High	Object deserialization is used in androidx.versionedparcelable.VersionedParcel.readSerializable()	View Details
Object deserialization is used in com.birbit.android.jobqueue.persistentQueue.sqlite.SqliteJobQueue$JavaSerializer.deserialize(byte[])	10: Medium	1: High	Object deserialization is used in com.birbit.android.jobqueue.persistentQueue.sqlite.SqliteJobQueue$JavaSerializer.deserialize(byte[])	View Details
Cipher with no integrity	10: Medium	1: High	The cipher does not provide data integrity	View Details
MD2, MD4 and MD5 are weak hash functions	10: Medium	1: High	MD5 is not a recommended cryptographic hash function	View Details
Object deserialization is used in com.facebook.appevents.AppEventStore.readAndClearStore()	10: Medium	1: High	Object deserialization is used in com.facebook.appevents.AppEventStore.readAndClearStore()	View Details
Object deserialization is used in com.facebook.appevents.AppEventsLogger$PersistedAppSessionInfo.restoreAppSessionInformation(Context)	10: Medium	1: High	Object deserialization is used in com.facebook.appevents.AppEventsLogger$PersistedAppSessionInfo.restoreAppSessionInformation(Context)	View Details
Object deserialization is used in com.google.android.gms.measurement.internal.zzil.zza(Object)	10: Medium	1: High	Object deserialization is used in com.google.android.gms.measurement.internal.zzil.zza(Object)	View Details
Object deserialization is used in com.loopj.android.http.PersistentCookieStore.decodeCookie(String)	10: Medium	1: High	Object deserialization is used in com.loopj.android.http.PersistentCookieStore.decodeCookie(String)	View Details
ECB mode is insecure	10: Medium	1: High	The cipher uses ECB mode, which provides poor confidentiality for encrypted data	View Details
Object deserialization is used in cz.msebera.android.httpclient.impl.client.cache.DefaultHttpCacheEntrySerializer.readFrom(InputStream)	10: Medium	1: High	Object deserialization is used in cz.msebera.android.httpclient.impl.client.cache.DefaultHttpCacheEntrySerializer.readFrom(InputStream)	View Details

Hard Coded Password	12: Medium	2: Normal	Hard coded password found	View Details
Predictable pseudorandom number generator	12: Medium	2: Normal	The use of java.util.Random is predictable	View Details
External file access (Android)	12: Medium	2: Normal	Files could be saved to external storage	View Details
Potential Path Traversal (file read)	12: Medium	2: Normal	java/io/File.<init>(Ljava/lang/String;)V reads a file whose location might be specified by user input	View Details
Potential Android SQL Injection	12: Medium	2: Normal	This use of android/database/sqlite/SQLiteDatabase.rawQuery(Ljava/lang/String;[Ljava/lang/String;)Landroid/database/Cursor; can be vulnerable to SQL injection	View Details
SHA-1 is a weak hash function	12: Medium	2: Normal	SHA is not a recommended cryptographic hash function	View Details
Unencrypted Socket	12: Medium	2: Normal	Unencrypted socket to am.smarter.smarter3.util.old_devices.DeviceSockets (instead of SSLSocket)	View Details
Potential Path Traversal (file read)	12: Medium	2: Normal	java/io/File.<init>(Ljava/io/File;Ljava/lang/String;)V reads a file whose location might be specified by user input	View Details
SHA-1 is a weak hash function	12: Medium	2: Normal	SHA-1 is not a recommended cryptographic hash function	View Details
Potential Path Traversal (file read)	12: Medium	2: Normal	java/io/File.<init>(Ljava/lang/String;Ljava/lang/String;)V reads a file whose location might be specified by user input	View Details
Broadcast (Android)	12: Medium	2: Normal	Broadcast intents could be received by a malicious application	View Details
Unencrypted Socket	12: Medium	2: Normal	Unencrypted socket to androidx.core.net.DatagramSocketWrapper (instead of SSLSocket)	View Details
Hard Coded Key	12: Medium	2: Normal	Hard coded cryptographic key found	View Details
Potential Path Traversal (file read)	12: Medium	2: Normal	java/io/FileInputStream.<init>(Ljava/lang/String;)V reads a file whose location might be specified by user input	View Details
Potential Path Traversal (file write)	12: Medium	2: Normal	java/io/FileOutputStream.<init>(Ljava/lang/String;)V writes to a file whose location might be specified by user input	View Details
Predictable pseudorandom number generator	12: Medium	2: Normal	The use of java.lang.Math.random() is predictable	View Details
WebView with JavaScript interface (Android)	12: Medium	2: Normal	WebView with JavaScript interface	View Details
Potential Android SQL Injection	12: Medium	2: Normal	This use of android/database/sqlite/SQLiteDatabase.execSQL(Ljava/lang/String;)V can be vulnerable to SQL injection	View Details
Potential Android SQL Injection	12: Medium	2: Normal	This use of android/database/sqlite/SQLiteDatabase.compileStatement(Ljava/lang/String;)Landroid/database/sqlite/SQLiteStatement; can be vulnerable to SQL injection	View Details
Potential Android SQL Injection	12: Medium	2: Normal	This use of android/database/sqlite/SQLiteDatabase.execSQL(Ljava/lang/String;[Ljava/lang/Object;)V can be vulnerable to SQL injection	View Details
Static IV	12: Medium	2: Normal	The initialization vector (IV) is not properly generated	View Details

SHA-1 is a weak hash function	12: Medium	2: Normal	SHA1 is not a recommended cryptographic hash function
TrustManager that accept any certificates	12: Medium	2: Normal	TrustManager that accept any certificates makes communication vulnerable to a MITM attack
Bad hexadecimal concatenation	12: Medium	2: Normal	Leading zeros are omitted in the concatenation increasing collision potential
Regex DOS (ReDOS)	12: Medium	2: Normal	The regular expression "^((?:[0-9A-Fa-f][1,4](?::[0-9A-Fa-f][1,4])*)?)::((?:[0-9A-Fa-f][1,4](?::[0-9A-Fa-f][1,4])*)?)$" is vulnerable to a denial of service attack (ReDOS)
Potential Android SQL Injection	12: Medium	2: Normal	This use of android/database/sqlite/SQLiteDatabase.delete(Ljava/lang/String;Ljava/lang/String;[Ljava/lang/String;)I can be vulnerable to SQL injection
Unencrypted Socket	12: Medium	2: Normal	Unencrypted socket to com.google.firebase.database.tubesock.WebSocket (instead of SSLSocket)
XML parsing vulnerable to XXE (XMLReader)	12: Medium	2: Normal	The use of XMLReader.parse(...) is vulnerable to XML External Entity attacks
Unencrypted Socket	12: Medium	2: Normal	Unencrypted socket to cz.msebera.android.httpclient.conn.MultihomePlainSocketFactory (instead of SSLSocket)
Unencrypted Socket	12: Medium	2: Normal	Unencrypted socket to cz.msebera.android.httpclient.conn.scheme.PlainSocketFactory (instead of SSLSocket)
Unencrypted Socket	12: Medium	2: Normal	Unencrypted socket to cz.msebera.android.httpclient.conn.socket.PlainConnectionSocketFactory (instead of SSLSocket)
HostnameVerifier that accept any signed certificates	12: Medium	2: Normal	HostnameVerifier that accept any signed certificates makes communication vulnerable to a MITM attack
DES/DESede is insecure	12: Medium	2: Normal	DES/DESede should be replaced with AES
Unencrypted Socket	12: Medium	2: Normal	Unencrypted socket to cz.msebera.android.httpclient.impl.client.ProxyClient (instead of SSLSocket)
Regex DOS (ReDOS)	12: Medium	2: Normal	The regular expression "(((\\p{Alnum}([\\p{Alnum}-]*\\p{Alnum})?\\.)*\\p{Alpha}([\\p{Alnum}-]*\\p{Alnum})?\\.?)|(\\d+\\.\\d+\\.\\d+\\.\\d+)|(\\:\\d*)?" is vulnerable to a denial of service attack (ReDOS)
Unencrypted Socket	12: Medium	2: Normal	Unencrypted socket to cz.msebera.android.httpclient.impl.pool.BasicConnFactory (instead of SSLSocket)
Predictable pseudorandom number generator	12: Medium	2: Normal	The use of java.util.concurrent.ThreadLocalRandom is predictable
Regex DOS (ReDOS)	12: Medium	2: Normal	The regular expression ":\\s*(?:([a-zA-Zo-9-!#$%&'*+.^_`\|\|~]+)=(?:([a-zA-Zo-9-!#$%&'*+.^_`||~]+)|"([^"]*)"))?" is vulnerable to a denial of service attack (ReDOS)
Unencrypted Socket	12: Medium	2: Normal	Unencrypted socket to okhttp3.internal.connection.RealConnection (instead of SSLSocket)
WebView contains a JavaScript interface	14: Medium	2: Normal	A JavaScript interface is registered to a WebView in bolts.WebViewAppLinkResolver$2.then(bolts.Task)
Format String Manipulation	15: Low	3: Low	Format string argument allowing user controlled parameters

Potential CRLF Injection for logs	15: Low	3: Low	This use of java/util/logging/Logger.log(Ljava/util/logging/Level;Ljava/lang/String;Ljava/lang/Throwable;)V might be used to include CRLF characters into log messages	View Details
Potential Path Traversal (file read)	15: Low	3: Low	java/io/FileReader.<init>(Ljava/lang/String;)V reads a file whose location might be specified by user input	View Details
Potential CRLF Injection for logs	15: Low	3: Low	This use of java/util/logging/Logger.logp(Ljava/util/logging/Level;Ljava/lang/String;Ljava/lang/String;Ljava/lang/String;)V might be used to include CRLF characters into log messages	View Details
Potential Android SQL Injection	15: Low	3: Low	This use of android/database/sqlite/SQLiteDatabase.query(Ljava/lang/String;[Ljava/lang/String;Ljava/lang/String;[Ljava/lang/String;Ljava/lang/String;Ljava/lang/String;Ljava/lang/String;)Landroid/database/Cursor; can be vulnerable to SQL injection	View Details
This class could be used as deserialization gadget	15: Low	3: Low	This class could make application using serialization vulnerable	View Details
Potential CRLF Injection for logs	15: Low	3: Low	This use of java/util/logging/Logger.fine(Ljava/lang/String;)V might be used to include CRLF characters into log messages	View Details
WebView has JavaScript enabled	17: Low	2: Normal	A WebView has JavaScript enabled in am.smarter.smarter3.ui.settings.ifttt.IfttttFragment.onCreateView(android.view.LayoutInflater, android.view.ViewGroup, android.os.Bundle)	View Details
WebView has JavaScript enabled	17: Low	2: Normal	A WebView has JavaScript enabled in am.smarter.smarter3.ui.settings.ifttt.IfttttFragment.onCreateView(android.view.LayoutInflater, android.view.ViewGroup, android.os.Bundle)	View Details
WebView has JavaScript enabled	17: Low	2: Normal	A WebView has JavaScript enabled in am.smarter.smarter3.ui.settings.ifttt.IfttttLoginActivity.onCreate(android.os.Bundle)	View Details
WebView has JavaScript enabled	17: Low	2: Normal	A WebView has JavaScript enabled in am.smarter.smarter3.ui.settings.ifttt.IfttttLoginActivity.onCreate(android.os.Bundle)	View Details
WebView has JavaScript enabled	17: Low	2: Normal	A WebView has JavaScript enabled in am.smarter.smarter3.ui.settings.ifttt.IfttttFragment.onCreateView(android.view.LayoutInflater, android.view.ViewGroup, android.os.Bundle)	View Details
WebView has JavaScript enabled	17: Low	2: Normal	A WebView has JavaScript enabled in am.smarter.smarter3.ui.settings.ifttt.IfttttFragment.onCreateView(android.view.LayoutInflater, android.view.ViewGroup, android.os.Bundle)	View Details
WebView has JavaScript enabled	17: Low	2: Normal	A WebView has JavaScript enabled in am.smarter.smarter3.ui.settings.ifttt.IfttttLoginActivity.onCreate(android.os.Bundle)	View Details
WebView has JavaScript enabled	17: Low	2: Normal	A WebView has JavaScript enabled in am.smarter.smarter3.ui.settings.ifttt.IfttttLoginActivity.onCreate(android.os.Bundle)	View Details
WebView has JavaScript enabled	17: Low	2: Normal	A WebView has JavaScript enabled in am.smarter.smarter3.ui.settings.ifttt.AlexaLoginActivity.onCreate(android.os.Bundle)	View Details
WebView has JavaScript enabled	17: Low	2: Normal	A WebView has JavaScript enabled in am.smarter.smarter3.ui.settings.ifttt.AlexaLoginActivity.onCreate(android.os.Bundle)	View Details
WebView has JavaScript enabled	17: Low	2: Normal	A WebView has JavaScript enabled in am.smarter.smarter3.ui.settings.support.SupportFragment.onCreateView(android.view.LayoutInflater, android.view.ViewGroup, android.os.Bundle)	View Details
WebView has JavaScript enabled	17: Low	2: Normal	A WebView has JavaScript enabled in bolts.WebViewAppLinkResolver$2.then(bolts.Task)	View Details
WebView has JavaScript enabled	17: Low	2: Normal	A WebView has JavaScript enabled in com.facebook.internal.WebDialog.setUpWebView(int)	View Details

APPENDIX C - SMARTER 3.0 VULNERABLE ANDROID CODE

```
.line 78
new-instance p2, Ljava/lang/StringBuilder;

invoke-direct {p2}, Ljava/lang/StringBuilder;-><init>()V

const-string v5, "SELECT tag,image,en,"

invoke-virtual {p2, v5}, Ljava/lang/StringBuilder;->append(Ljava/lang/String;)Ljava/lang/StringBuilder;

invoke-virtual {p2, v1}, Ljava/lang/StringBuilder;->append(Ljava/lang/String;)Ljava/lang/StringBuilder;

const-string v1, " FROM dialogs"

invoke-virtual {p2, v1}, Ljava/lang/StringBuilder;->append(Ljava/lang/String;)Ljava/lang/StringBuilder;

invoke-virtual {p2}, Ljava/lang/StringBuilder;->toString()Ljava/lang/String;

move-result-object p2

goto :goto_15c

:cond_15b
move-object p2, v0

:goto_15c
const/4 v1, 0x0

goto :goto_161

:cond_15e
:goto_15e
const-string p2, "SELECT tag,image,en FROM dialogs"

const/4 v1, 0x1
```

Figure 18 - Raw SQL Query detailing 'dialogs' table and columns

```
.field static final INSERTION_ORDER_COLUMN:Lcom/birbit/android/jobqueue/persistentQueue/sqlite/SqlHelper$Property;

.field static final JOB_HOLDER_TABLE_NAME:Ljava/lang/String; = "job_holder"

.field static final JOB_TAGS_TABLE_NAME:Ljava/lang/String; = "job_holder_tags"
```

Figure 19 - Variables storing table names

```
.class Lcom/google/firebase/database/FirebaseDatabase$1;
.super Ljava/lang/Object;
.source "FirebaseDatabase.java"
```

Figure 20 - Firebase Database code found in the database package

```
.class public final Lam/smarter/smarter3/notifications/MyFirebaseMessagingService;
.super Lcom/google/firebase/messaging/FirebaseMessagingService;
.source "MyFirebaseMessagingService.kt"


# annotations
.annotation system Ldalvik/annotation/SourceDebugExtension;
    value = "SMAP\nMyFirebaseMessagingService.kt\nKotlin\n*S Kotlin\n*F\n+ 1 MyFireb
.end annotation

.annotation runtime Lkotlin/Metadata;
    bv = {
        0x1,
        0x0,
        0x3
    }
    d1 = {
        "\u0000(\n\u0002\u0018\u0002\n\u0002\u0018\u0002\n\u0002\u0008\u0002\n\u0002
    }
    d2 = {
        "Lam/smarter/smarter3/notifications/MyFirebaseMessagingService;",
        "Lcom/google/firebase/messaging/FirebaseMessagingService;",
        "()V",
        "getNotificationIcon",
        "",
        "onMessageReceived",
        "",
        "remoteMessage",
        "Lcom/google/firebase/messaging/RemoteMessage;",
        "onNewToken",
        "refreshedToken",
        "",
        "saveRegistrationToken",
        "token",
        "sendNotification",
        "messageBody",
        "app_prodRelease"
    }
```

Figure 21 - Firebase Messaging code in notifications package

```
.line 31
new-instance p1, Ljavax/crypto/spec/SecretKeySpec;

iget-object p2, p0, Lcom/electricimp/blinkup/MCrypt;->SecretKey:Ljava/lang/String;

invoke-virtual {p2}, Ljava/lang/String;->getBytes()[B

move-result-object p2

const-string v1, "AES"

invoke-direct {p1, p2, v1}, Ljavax/crypto/spec/SecretKeySpec;-><init>([BLjava/lang/String;)V

iput-object p1, p0, Lcom/electricimp/blinkup/MCrypt;->keyspec:Ljavax/crypto/spec/SecretKeySpec;

:try_start_31
const-string p1, "AES/CBC/NoPadding"
```

Figure 22 - Secret Key encryption method