School of Electronics, Electrical Engineering and Computer Science

Investigating Information Leakage from IoT devices

ELE8095 Individual Research Project

Student Name: *Angela McKeown*

Student Number: %%%%

Academic Supervisor: *Dr Kieran McLaughlin*

20th April 2018

Abstract

This paper looks at the problem of information leakage from Internet of Things devices, primarily through the lens of children's consumer toys. It examines the increased threat of personally identifiable information leaked from such a vulnerable group, looks at some examples of examinations of devices of this type, and then uses a derived methodology to examine three such devices.

The children's IoT devices this study compares and contrasts are; the ToyFi teddy from Dragon-i Toys (previously manufacturer of the CloudPets range), My Friend Freddy bear from Genesis Toys (previously manufacturer of the Cayla doll), and NuNu from ToyMail (a new company). An examination of each device is undertaken using the methodology previously derived, followed by an analysis and comparison of their security and privacy features and associated risks, assigns a risk score, and recommendations mitigation strategies for the consumer and manufacturer.

The methodology and risk assessment process are then analysed for robustness, with a view to their use in the examination and rating of future Toy devices.

Acknowledgements

It has been no secret to those close to this project that this year has been a particularly difficult one, personally. The several sudden illnesses, deaths and financial crises of close family have undoubtedly been very hard on everyone involved, and my Masters was quite rightly the last thing on everyone's mind. Yet these people still rallied around as much as possible to try and ensure I got the support I needed to continue, and I would not have had the ability to carry on or had a dissertation to submit without their help. Therefore, it is with absolute, heartfelt gratitude that I thank the following:

Dr Kieran McLaughlin, for constantly steering me back to the path and helping me retain some agency over the outcome, in the face of everything veering off track. Also, specifically, for making the effort to see past my non-academic background and sometimes too-jovial demeanour, and for tirelessly telling me off for my colloquialisms. I look forward to hearing his voice in my head correcting all my future writing.

My mother and father, daughter, and brother, who variously provided financial support, cooked meals, provided much needed Wi-Fi, and put up with me wittering at them for months about the vagaries of Bluetooth, Android versioning and packet sniffing. Their patience and kindness despite their own problems is a testament to them.

The MSc students in the Keybase group who have exchanged encouragement, tips they picked up, and clarifications about reports or concepts. Having companions to travel with on this journey has made it much better.

The fantastic NI Women in Tech and everyone in NI Tech & Design Slack, who are too many to name but who constantly checked if I was alright and just kept on believing in me anyway, when I'd quietly already given up. Having someone who isn't family tell you they think you can do something is a powerful thing, and I will always be grateful and amazed by it.

Declaration of Originality

"I certify that this dissertation is my own original work, except where stated and referenced.

Signed: ..... Date....."

Table of Contents

Introduction

As the number of IoT devices continues to increase exponentially, the leakage of personal information from them has become more of a concern for the consumer, particularly since so many children's toys are being marketed as 'smart' and connected and children are ill-equipped to understand what is safe to disclose.

The available security studies of IoT devices range from rigorous but confusing for consumers, to journalistic hyperbole, making it difficult to truly assess risk. There is a gap which can be filled by testing a good methodological approach and coupling it with a usable privacy risk assessment.

The selected project makes a comparison of security and privacy risks between three commonly available IoT toys for young children, examining the different methods they use to transmit and store information, and connect to other devices.

This will provide a tested methodology for use in other children's IoT projects and a privacy methodology designed with these in mind.

It will also provide rudimentary testing of three IoT Toy devices used as examples.

The rest of this paper is organised as follows: an overview of several previous IoT toy scandals precedes a brief review of some of the literature on possible attacks on IoT devices, and a review of some examinations of IoT devices from others in the field. The aims of the study are then more clearly set forth, and the methodology laid out in detail. The three device examinations follow, and then a discussion of the findings, including an analysis of the methodology, a discussion of the privacy risk assessment matrix, some suggested future work and research challenges, and finally some suggested mitigations for the consumer and manufacturer.

Investigating Information Leakage in IoT Devices

1. Key issues

The state of the art in information leakage in IoT devices appears to be divided into several domains, as IoT is a very wide remit covering everything from personal devices and wearables, to in-home gadgets and toys, right through to large-scale public or business sensor networks. While all of these can leak information, the consequences can be very different for each.

Children's IoT devices present a unique concept in security and privacy risk as they are used by one of the most vulnerable and easily compromised sections of our population, who are often not able to make good security decisions for themselves. The devices frequently deal with personally identifiable information (PII), and often have limited power and storage capabilities, yet still demand enough processing power for interactivity functionality. In addition, they are subject to the usual market forces of reducing manufacturing costs to improve sales and profit margins.

Recently this difficult balance of factors has led to manufacturing shortcuts resulting in compromised security and public outcry for several high-profile Toy devices.

Compounding these problems, due to the urgency of providing security information as quickly as possible once new products are released to market, research tends to take place in an ad-hoc, piecemeal manner, appearing online on blogs, news sites and company websites with vastly varying degrees of detail and no standard method of advising consumers of risk.

1.1. Background info

An early example of a connected toy scandal is the My Friend Cayla doll, which was released in Nov 2014. The doll was able to answer questions and recognise objects via a connection to Google, but unfortunately had open Bluetooth pairing allowing anyone nearby to access the camera or mic. Astonishingly, despite several security warnings, the UK Toy Retailers Association is reported by the *BBC* to have responded to the allegations by saying the toy posed 'no special risk' [1], and went on to subsequently grant the toy two annual awards. The lack of effort on adding pairing security despite

the work put into the Google connection, and the Toy Retailers Association stance, only emphasises the idea that business is often willing to put profit before any other consideration. The Toy was eventually banned in Germany for falling afoul of their surveillance laws.

December 2015 saw Hello Barbie cause problems. Wi-Fi Barbie was an innovative invention which allowed young children to tell Barbie their worries and other thoughts, which their parents could later listen to. *The Register* claims [2] the Wi-Fi Barbie app featured hard-coded security credentials and a predictable Access Point name, making spoofing a connection easy and traffic susceptible to surveillance, and that their servers used the insecure SSLv3 for encryption making them susceptible to flaws including protocol downgrading attacks like POODLE [3]. Unfortunately, this was not the only problem Barbie faced. As it transpired that conversations were being uploaded to the internet, which many consumers had not at first understood, people became paranoid that the device was recording conversations in their homes to analyse for Mattel's marketing purposes, and soon the internet was aflame with the scandal [4], despite the conversation's not even going to servers owned by Mattel. This evidences the idea that giving people an accurate but understandable way to assess risk is important, as they are often unable to find dependable sources of information or assess security complexities competently on their own.

In early 2016 *Rapid7* did some research into the Fisher-Price interactive Smart Toy [5], which utilises a mobile app and Wi-Fi. They found that their web platform was not verifying API calls and so anyone could extract private consumer profile details. This was an excellent example of responsible disclosure working well.

The Cloudpets data breach in February 2017 saw an open database of 821 thousand customer records exposed [6], including insecure links to 2.2 million voice recordings, which was stolen by hackers shortly afterwards. In separate research, the Toy device itself was also found to be insecure [7]. In contrast to the previous problem, this was an example of the company utterly failing to respond to responsible disclosure and an enormous data breach and scandal ensuing, and absolutely highlights the importance of having a procedure for responding to disclosures in place.

Unfortunately, there is no sign of any reduction in consumer interest for connected Toy devices, and many, just like My Friend Cayla and CloudPets, continue to be manufactured in China, where privacy regulation is much less strict.

The newly applicable EU General Data Protection Regulations (GDPR) [8] introduced in May 2018 by the European Union, place very strict rules and hefty fines on any company which does not handle the data of European Citizens with very strict and defined care, including security protection, privacy statements, and a very clear statement of who will be handling the data and for exactly what purpose. Further protections are prescribed for the storage and processing of data belonging to minors.

In the U.S. businesses must also comply with the Children's Online Privacy Protection Act (COPPA) [9] if they collect, use or disclose personal information from or about children under thirteen on the internet.

In Germany, hidden surveillance devices are banned under their Abuse of Telecommunications act [10], so many IoT toys which have hidden record and transmit capabilities are illegal to own there.

In the newly emerging area of IoT the field is moving fast, and the proliferation of devices makes it hard for print literature to keep up. As we have touched on, again and again, these technical investigations have revealed numerous problems with Wi-Fi and Bluetooth insecurity due to poor app design, poor protocol implementation, and general lazy development, as manufacturers concern themselves mainly with shortening time-to-market or providing excitement at the cost of best practice. Despite many device examinations and data breaches being revealed via internet blogs, many of the exploits used are still developed through traditional research, and a review of the literature revealed an ecosystem replete with vulnerabilities and weaknesses.

1.2. Theoretical foundations

A review of the literature of general device data leakage makes it very clear that any leakage at all is a problem [11]. It is relatively obvious that any leakage of Personal Data is egregious and possibly illegal, but what the Torre *et al* paper on "Preventing Disclosure of Personal Data in IoT Networks" makes clear is that Inference Attacks are

actually a hugely underestimated problem – in which seemingly innocuous information is revealed or leaked but can then later be combined with other known or public information to create data which is much more personal or damaging. The authors give a very basic example of a user withholding their birth date from an app, but the app inferring the information by simply scraping their 'wall' for posts from friend's that include the words 'Happy Birthday'. Many other types of information can be combined with public data – the combination of GPS data from Strava fitness wearables with public maps data earlier in the year unintentionally revealed the location of military bases, amongst other things [12],and we know that stalking victims may find it dangerous if their location data is vulnerable in ways that normal people do not care about. Thus, it is clear, we must regard any information leakage from an IoT device as important and potentially sinister.

The literature review also highlighted that there are many, varied ways to attack a device, from the hardware itself, to the transmissions sent out, to the information stored online, or through an accompanying mobile application:

The paper on CEMA side channel attacks on AES128 [13] identifies which hardware modules and frequencies the Arduino boards leak at during the encryption process, using an EM probe on the flash memory, databus and SRAM. This is a really interesting insight into the full extent of what is possible with an electromagnetic side-channel attack, including revealing the AES secret key.

In "Using Histograms to Remotely Detect Skype Traffic" [14], Atkinson et al show that Skype traffic, and by extension many other traffic types, can be detected over WiFi based on the frame size and frame arrival times via passive monitoring just by being within the receiving range of the transmissions. They found they could discern between a mixture of Skype, Web Browsing and BitTorrent traffic using the 'Random Forests' machine learning model, and conclude the ML model could easily be applied to other user network activities.

Wearables by their very nature carry a lot of personal data which can be joined up with other data to be very damaging, and yet they are often poorly protected because IoT devices are so small that there is little room for onboard security. “Smart Attacks against Intelligent Wearables in People-Centric Internet of Things” [15] comprehensively outlines a whole variety of possible attacks on wearables such as;

- Attacks on data integrity: Wi-Fi/Bluetooth sniffing, MITM attacks/injections. They use an Ubertooth One for a BLE attack. Fitbit doesn’t check the JavaScript sent.
- Attacks on data authenticity: mule attacks (fool sensors with false data). Sniff and modify Bluetooth firmware download traffic to inject with MITMPROXY, then pivot into main device.
- Attacks on data privacy: mole attack gathers private info via side-channel sensors – smart watches password protection is only enabled when unpaired. When passcode and USB debugging are enabled micro USB access often allows a command line shell.

This informative paper highlights that most wearables are very insecure and mitigation for these kind of attacks is extra work that many companies producing devices in bulk just don’t bother doing.

Flaws in the Android OS come up regularly and although they are often patched in new versions users don’t always update their phones so quickly, and IoT devices running Android operating systems may not even have update capabilities. Zhang et al discuss some possible side-channel attacks in “App-level Protection Against Runtime Information Gathering on Android” [16] in which a malicious app running concurrently with another legitimate application may exploit it via shared OS information channels, even if the legitimate app does not have explicit implementation flaws of its own. Of course, app developers can have little control over information exposed by the OS, they cannot disable recording by another app for example, and adding noise to their data may or may not help but it will certainly impact on performance.

The paper cites examples of these side-channel attacks on the Global Resources outside the usual app sandbox, such as audio, video, memory, network, CPU, GPS and Bluetooth;

- the Belkin Wi-Fi camera home-security system which utilises an Android operating system – yet a side-channel attack on the app allowed researchers to uncover whether the user was at home or not, and whether they were currently monitoring the surveillance cameras.
- A game app with Bluetooth permission for connecting to its playpad can also download patient data from a Bluetooth Glucose meter.
- Any app with 'Network' permission can ask to snap a picture of the screen – continuous screenshots will allow a recording of a user entering a password.

Inference attacks can also be used on this data, such as using the 'isMusicActive' API status sequences in correlation with Google Navigator route data to uncover driving routes.

This section discussed the key issues of vulnerability in IoT devices, especially those sold as children's toys, and found that not only have widely-reported scandals already occurred but that manufacturers appear to prioritise cost and complexity savings over safety and security. It also took a brief overview of research from the literature review which supports the idea that there are quite a large variety of ways to successfully attack IoT devices. We will proceed to consider some previous Toy examinations more specifically in order to derive an appropriate methodology.

1.3. Related studies and experiments

In order to derive an appropriate foundation for examination and analysis, research was undertaken into existing work in this field. This research surfaced several IoT investigations of recent years, as follows.

As previously mentioned, many of the experimental projects examining information leakage in consumer 'in-home' IoT devices lack scientific papers, and are instead carried out as hobby 'hacking' projects by individuals or interested businesses and posted on sites such as GitHub, hackster.io or their own blogs, which can lead to varying amounts and quality of data available. These more unorthodox sources are also important in IoT because it is growing so fast as a new field and with so many heterogeneous products being released it is difficult for security research to keep up. With the current popularity of the maker movement, the continued embracing of Open Source in the business community, and the increasing tendency for hardware dev kits

(from companies such as Nordic **[17]** or Ubertooth **[18]**) to be made available at consumer prices, small companies and private individuals can now make an active contribution to advancing knowledge.

Paul Stone, Principal Security Consultant from Context does a detailed examination [7] of the Cloud Pets toy vulnerabilities. This includes online research, examining BLE advertising data, reverse engineering the android app, and then a more detailed experiment using clues from the app data to demonstrate further insecurities in the BLE Characteristics and manipulate the device. He then discusses a responsible disclosure policy, gives consumer guidance for toy owners based on the vulnerabilities found, and identifies that next steps may be to tackle possible insecurities identified in the firmware update mechanism.

The Stone blog also links to Troy Hunt's contemporaneous online report of the CloudPets data breach [6], which includes a comparison of password hashes against user data found in an open Amazon S3 bucket found via Shodan, which he shows can often be subsequently used to expose user profile pictures and voice recordings.

Simone Margaritelli makes an examination of the Nike+ FuelBand BLE wristband [19], including reverse engineering the Android app and comparing this to the BLE data, which results in a successfully compromising the wristband and discovering commands not meant to be in the production release. While this blog post is technical and relatively short, Simone is a subject matter expert and so it still contains some useful ideas about finding a connection between BLE advertising data and smali code, and working out what the app is doing.

Margaritelli has also produced a detailed step-by-step guide to reversing Android Apps [20], which covers everything from system setup, through examining network traffic, to understanding the difference between Smali-based dex files and their Java counterparts.

Mark Stanislav from Rapid7 did an excellent job finding and disclosing the open API vulnerabilities associated with the Fisher Price Smart Toy, and lists 6 different APIs that were readable as a result of the exploit, as well as discussing the impact on the consumer, as well as providing a disclosure timeline and that they followed their company Responsible Disclosure procedure, and also sent details to CERT.

As the aim is to derive a viable project methodology which will function with a range of children's toys which involve connection to an Android app to function, it seems that a project methodology which features a device as closely aligned with these features as possible is advisable. It is hoped that the privacy and security concerns of these toys may also be given heightened consideration by choosing a methodology which has a similar focus. Therefore, it is proposed that the Stone CloudPets teardown be used as the main framework for the project methodology in this case.

However, it is notable that Margaritelli provides a much more detailed strategy for reverse engineering, and is widely considered an expert in the field, being the author of Bettercap, speaking at many industry functions and contributing widely to open source and on Twitter, and therefore his reverse engineering tutorial can be considered a definitive guide.

It is therefore proposed that his methodologies for reversing applications and combining exposed app data with BLE data also be merged with the Stone methodology to strengthen the overall chance of success, and that reference be made to both throughout the project.

While the Hunt OSINT and hashing article is useful it is presenting as an adjunct to journalistic reporting of a data breach, and lacks technical detail about how each stage was accomplished, rather than as a sound methodology in itself. In addition, these ideas may not be applicable to all devices, as many do not hold customer information online in this manner. Nevertheless, information leakage in this manner is very serious, and it would be remiss not to include it as part of an investigation.

Stanislav, while clearly having done important work, has presented the findings as more of an informative post about the accomplishments of Rapid7 than a breakdown of how it was accomplished, and so it has not been considered useful to incorporate it into the methodology. Nevertheless, it is another good example of a responsible disclosure procedure. It also highlights the vast difference between the company blog that Stone wrote, which students may learn from and reproduce, and this company blog, which serves only as an announcement. This perhaps highlights once again the varying difference in quality between online research reports, and emphasises the usefulness of having a standard methodology to work and report against.

Hunt and Margaritelli are subject-matter-experts and well-known names in their respective fields. Stone, although less well known, is covering a tear-down of a well-known insecure device in a relatively thorough fashion in his professional capacity, and thus can be considered a good source for these purposes. The methodology chosen, then, will comprise of Stone's framework, filled in with some additional logical steps from Margaritelli, and some steering from Hunt for any work with online servers.

2. Aims and Objectives

2.1. Practical Aims

As previously discussed, much of the current research into consumer IoT goods happens in the 'professional hobbyist' and 'corporate blog' space, where the quality of research, skill and methodology is highly variable. This is also very unlikely to change, as the response time is key between new products coming to market and security warnings being released to warn the public, particularly in the realm of children's toys.

Therefore, the key aims of this project are to take some prominent existing methodologies and test them against a small range of connected Toy devices, expand on the effectiveness of the methodology where possible, and suggest a method of privacy risk assessment to complement the methodology.

It is important to understand that in a device investigation aimed at understanding threat to consumer privacy, not all avenues are necessarily investigated. Indeed, it often depends on the investigator's interest, specialism, or what has been covered by others in the past. We can see this in the Hunt coverage of the CloudPets breach [6], as he does excellent investigation into the available open source intelligence (OSINT) sources and open S3 buckets, but does not touch at all on the device itself. Simultaneously yet separately, Stone [7] had done an investigation of the device, but had not investigated any online information. Neither investigated compromising the actual electronics.

The purposes of the following examinations, and indeed any future use of the methodology then is to use it not exhaustively, but following the key interests of consumer privacy, so that a solid basis for the risk assessment can be derived.

It is also hoped that any PCAPs and technical knowledge gained from the project can be made available for others to learn from, where permitted by license and trademark.

2.2. The methodology

2.2.1. Information Gathering

The methodology followed in each case begins with information gathering about the device in order to build up a picture of how it functions and its potential vulnerabilities. FCC filings, packaging, and device information from the manufacturer's website or online blogs, will be the first port of call in building up an initial picture of how the device is intended to function, it's start up and communication procedures, and any potential flaws or issues that users or the company itself are experiencing with it, which could then be used as a vulnerable point for ingress or just better understanding of the device and any online infrastructure it depends on.

2.2.1.1. Tools and Methods

Packaging and paperwork that accompanies the device, URLs or Manufacturer references from the packaging, internet search engines, and any identification numbers marked on the device are all used to accomplish this.

To be legally sold in the United States, wireless devices must be independently tested to ensure they conform to US regulations. The results of these tests, plus user manuals, documentation and photographs are then registered with the Federal Communications Commission (FCC), who then assign a unique ID which must be displayed on the device. This means that device emissions, operating frequencies, and a great deal of other supplied information can be obtained about devices simply by searching online against their FCC ID.

Possible vulnerabilities are enumerated so that these can be more closely examined during analysis of the connection and APK code.

2.2.2. Device Advertisement/ Connection Sniffing

Packet sniffing' will be carried out during both the setup and connection phases of the Bluetooth connection with the Android app (essentially a Man In The Middle attack) in order to determine if it is connecting securely or if information is being leaked in any way about the WiFi network, about the payload, or if the payload itself is accessible. Liu et al [15] show that Bluetooth data integrity, authenticity and privacy is highly susceptible to attack in this way. Advertising packets being broadcast from the device must also be captured. The Stone example [7] uses the Ramble app and the nRF app to accomplish this. The use of BLE protocols is noted, and where possible which authentication method

is used, whether extra Characteristics are present, and Google or the BLE documentation is referred to for additional information on whatever is found.

The accompanying app for the Toy will also be downloaded and packet sniffing carried out while the app is attempting connection, and while interacting with the user, in order to capture any data transfer or difference in behaviour then - this stage will include Wi-Fi sniffing in addition to Bluetooth Low Energy.

2.2.2.1. Tools and Methods

2.2.2.1.1. Advertisement/Connection Sniffing

At first it was anticipated that the device and the Android phone would send Bluetooth LE packets to each other, which will be intercepted by the nRF51 dongle plugged into the computer (Figure 1). These will then be recorded in Wireshark for further analysis, hopefully allowing us to see if any information about the advertising packets, connection or the payload of the packet has been leaked. It was hoped the use of a sniffer which interfaces with Wireshark directly would facilitate simultaneous Wi-Fi and BLE sniffing, resulting in richer and more useful logs.

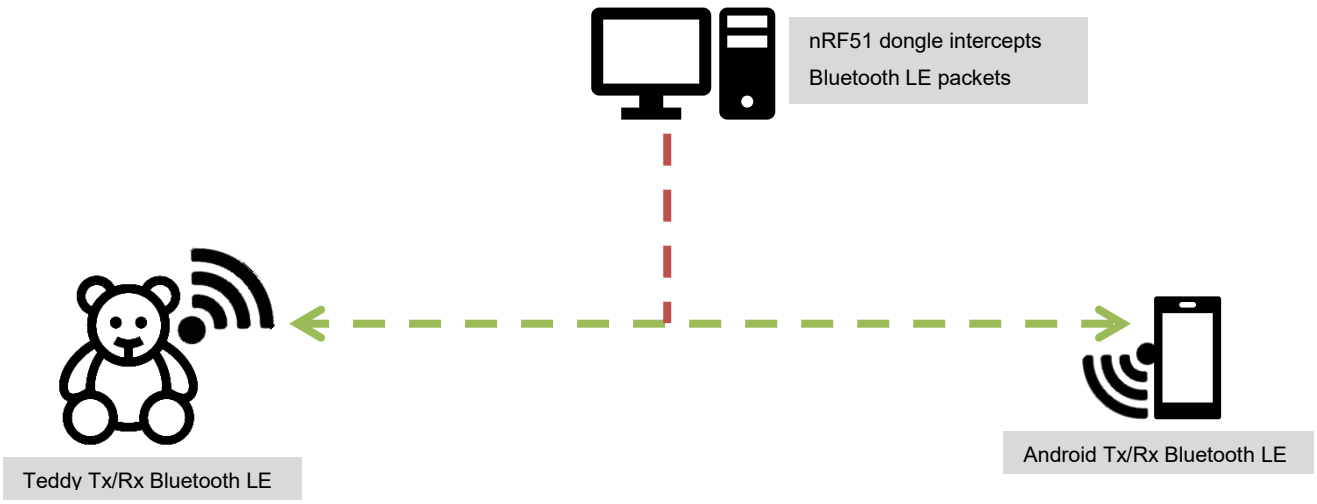


Figure 1- Network Diagram for Bluetooth packet capture

The dongle itself is a simple USB device supported by drivers provided by the manufacturer (Figure 2).

Figure 2 - nRF51 dongle (PCA10031 Nordic Semiconductor)

MITM Test Device

As it is imperative that packets from the setup phase are captured correctly, the dongle/Wireshark setup must first be tested and verified as working with another regular household Bluetooth object before the project experiment commences.

MITM Project Device

Then the stages of activating the toy device, registering with the app, creating and sending messages between the device and the phone, must all be separately and carefully captured so as not to miss any data.

However, unexpected problems arose during the setup of the test environment and subsequent test device packet captures, which resulted in changes to the original plans for packet capture, as detailed below.

Test Environment Setup

First, online research was undertaken on a selection of Bluetooth sniffing devices, with particular reference to Afaneh's hardware guide [21], as he is a well-known Bluetooth subject-matter-expert producing a comprehensive book on BLE and a highly subscribed industry newsletter on BLE mesh networking technologies. Research found that those in a consumer price range were restricted to the Ubertooth One [18] and the Nordic Semiconductor nRF51 [17]. Further research into a comparison of the two devices was undertaken on the manufacturer's forums and this piece [22] by Hughes which provides a walkthrough of using each device. The primary system available for use during the investigation was a Windows system and the initial project device used Bluetooth LE. Nordic Semiconductor are a major founder and contributor to the Bluetooth SIG in charge of the Bluetooth specifications and publish a series of highly respected apps and command line tools. nRF51 also has a Windows, Linux and Android client and interfaces

directly with Wireshark, whereas the Ubertooth is only natively supported on Linux. Both sniffers may suffer minor packet loss issues due to channel hopping, but this is to be expected when not using higher-end equipment. For this reason, the Nordic nRF51 was selected as the most suitable sniffer for the project.

It was then a matter of following the setup guidelines in the nRF sniffer User Guide, available from Nordic Semiconductor.

Firstly, the software environment was setup:

- Software downloaded to make the sniffer work included Segger jLink v6.16 and Python 2.7.
- The nRF51 was then connected via USB port and Windows automatically detected and installed drivers.
- Wireshark was then launched. The External Capture folder was located (as detailed in Help -> About -> Extcap path) and the previously downloaded **nrf_sniffer_<version>_<hash>** file was unzipped to that location.
- It was verified that Python was installed successfully, and callable from the command line.

```
C:>python --version
```

Then the nRF51 dongle was prepared for use:

Firmware was installed on the dongle using the jlink.exe program, downloaded as part of Segger.

- jlink.exe was opened from the command line.
- The **erase** command was used to ensure the dongle was empty and begin a series of automatic formatting prompts, with the following values specified from the User Guide.

Device type	**nRF51422_XXAC**
SWD interface	**s**
Speed	**1000**

- The extcap hexfile was then loaded onto the device with **loadfile,** then **<Path to Wireshark>\extcap\nrf_sniffer_<version>_<**
- **hash>\hex\sniffer_<board name>_<hash>.hex**
- Then **r** to reset the board, and finally **g** to run the board firmware.

It was then verified that the sniffer firmware was running correctly by activating a nearby BLE advertising device and checking that the onboard LED was flashing, indicating that packets were being received.

Some issues were then encountered, as the sniffer was not present as expected in the Wireshark interfaces menu, and no packets were being received in Wireshark from the sniffer, although Wi-Fi packets were appearing successfully. After some searching for solutions on the Nordic DevZone forums the version of Wireshark was downgraded to 2.4.2 for compatibility, as suggested by Nordic. A new hex file for the sniffer dongle was also downloaded and flashed to the dongle. Python 3 was also uninstalled in case unknown conflicts were arising, although this proved difficult as Visual Studio kept auto-reinstalling it, so it ultimately had to be uninstalled via regedit.

A decision was taken to manually execute the required Python file in the Wireshark directory to see what was happening there, and the error 'No module named serial' appeared. This was an unknown python module dependency, and was then installed separately via 'Pip install serial'. After this the file appeared to run correctly.

```
Collecting pyserial
  Downloading https://files.pythonhosted.org/packages/0d/e4/2a744dd9e3be04a0c0907414e2a01a7c88bb39
15cbe3c8cc06e209f59c30/pyserial-3.4-py2.py3-none-any.whl (193kB)
    100% |################################| 194kB 2.3MB/s
Installing collected packages: pyserial
Successfully installed pyserial-3.4
You are using pip version 9.0.1, however version 10.0.1 is available.
You should consider upgrading via the 'python -m pip install --upgrade pip' command.

C:\Program Files\Wireshark\extcap>nrf_sniffer.bat --extcap-interfaces
extcap {version=2.0.0}{display=nRF Sniffer}{help=http://www.nordicsemi.com/eng/Products/Bluetooth-
low-energy/nRF-Sniffer#Downloads}
interface {value=COM3}{display=nRF Sniffer COM3}
control {number=0}{type=selector}{display=Device}{tooltip=Device list}
control {number=1}{type=string}{display=Passkey / OOB key}{tooltip=6 digit temporary key or 16 byt
e Out-of-band (OOB) key in hexadecimal starting with '0x', big endian format. If the entered key i
s shorter than 16 bytes, it will be zero-padded in front'}{validation=\b^(([0-9]{6})|(0x[0-9a-fA-F
]{1,32}))$\b}
control {number=2}{type=string}{display=Adv Hop}{default=37,38,39}{tooltip=Advertising channel hop
 sequence. Change the order in which the siffer switches advertising channels. Valid channels are
37, 38 and 39 separated by comma.}{validation=^\s*((37|38|39)\s*,\s*){0,2}(37|38|39){1}\s*$}{requi
red=true}
control {number=3}{type=button}{role=help}{display=Help}{tooltip=Access user guide (launches brows
er)}
control {number=4}{type=button}{role=restore}{display=Defaults}{tooltip=Resets the user interface
and clears the log file}
control {number=5}{type=button}{role=logger}{display=Log}{tooltip=Log per interface}
value {control=0}{value= }{display=All advertising devices}{default=true}

C:\Program Files\Wireshark\extcap>
```

Figure 3 - Installing pyserial and configuring Wireshark extcap interface to display the sniffer

Wireshark was then launched to see if the nRF51 appeared successfully as an interface, which it finally did.

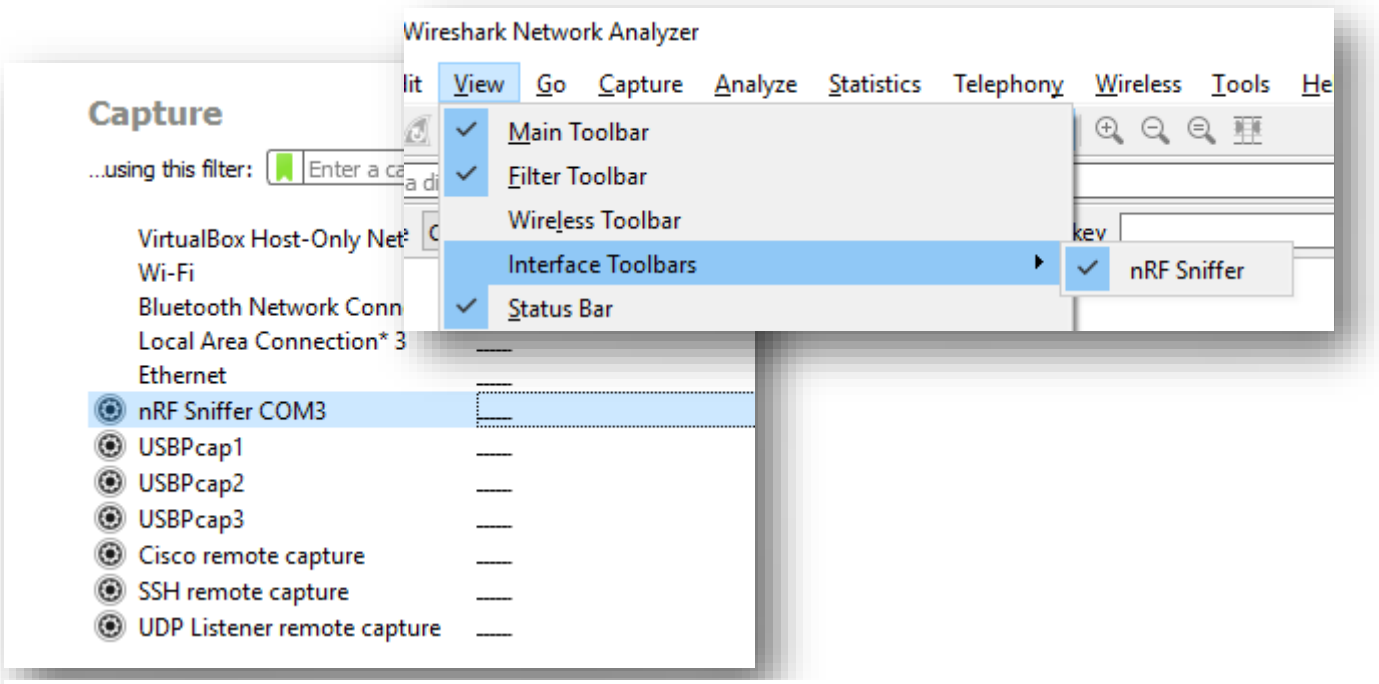


Figure 4 - Two images showing the sniffer successfully appearing in the Wireshark interface

However, this was sadly not the end of problems with the nRF51.

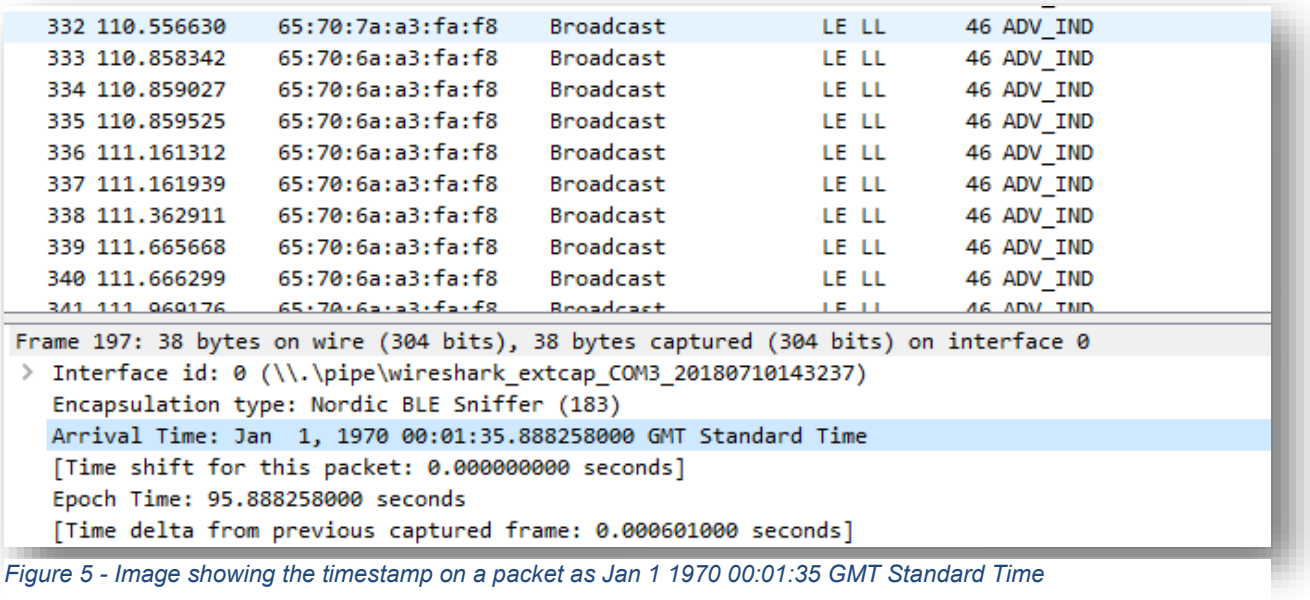


Figure 5 - Image showing the timestamp on a packet as Jan 1 1970 00:01:35 GMT Standard Time

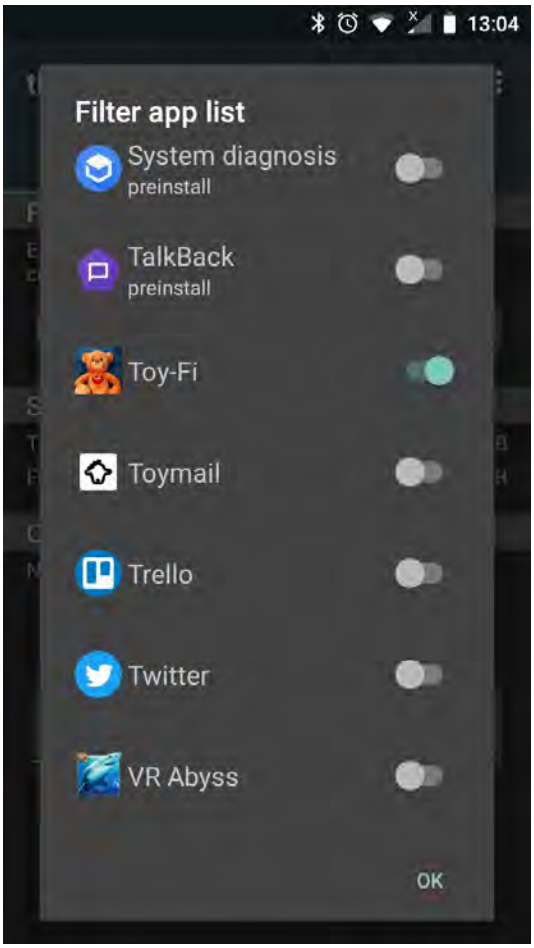
Unfortunately, it was soon discovered that the timestamp was wrong on all PCAPs. It is definitely not 1970, but Network time stamps are always counted in seconds since January 1, 1970, 00:00:00 UTC (also known as UNIX time or Epoch time). Timestamps in PCAPs are derived from the clock on the machine performing the packet capture [23], so this is an additional difficulty of working with the nRF51 which we may surmise has an

onboard clock which appears to start when it is plugged into the USB slot, rather than deriving its time from the system clock of the machine itself.

Further into the project, the nRF51 stopped capturing packets on several occasions. Erasing firmware and reinstalling everything from scratch did not help. Downgrading the firmware version helped temporarily. Speculation about the extremely hot weather affecting the device (which already ran hot) was the only theory. Although people with similar issues were posting on the manufacturer forums [24] [25], they do not seem to have found solutions.

An Android Nokia 5 phone had been obtained for use with any apps during the project. As a contingency plan, in order to prevent progress stalling on the project, BLE logging was considered on the phone instead.

For Bluetooth on the Nokia 5 this was achieved by opening Settings – System – Build Number and tapping on it Seven times to activate Developer Options. This then reveals extra menu options. By clicking on Developer Options “Enable Bluetooth HCI snoop log” can now be selected” as well as “Show Bluetooth devices without names” (so that we can now see BLE devices that only advertise a MAC), “Enable Wi-Fi verbose logging” and “USB debugging”. For convenience, “Bug Report Shortcut” was also selected, which displays a button in the power menu for taking a bug report instead of having to access this menu again. A bug report is an easy way to export Wi-Fi and Bluetooth logs from the phone without rooting it, although it could also be pulled over ADB.



Research showed that many apps for phone Wi-Fi packet capture required rooting the phone. However, several online forums and articles mention tpacketcapture [26] as an app which will let you capture Wi-Fi packets only from one specific app, without rooting your phone.

The pcap can be shared from within the app, and then opened in Wireshark. This seemed like a very useful solution as it would eliminate the large amount of background noise from normal phone and network Wi-Fi operations.

The following procedure was drawn up for each capture to avoid mistakes:

- The phone Bluetooth cache is cleared
- The tpacketcapture app is started and set to record captures only from the app being investigated
- The Toy device is switched on
- The corresponding Toy app is launched
- Interaction with the app takes place as planned
- The Toy app is closed
- The tpacketcapture app is stopped
- A Bug Report is taken from the phone in order to export the Wi-Fi and Bluetooth logs

Packet sniffing is also desirable for the Toy devices when no corresponding phone app is active, and on the phone app when no corresponding Toy device is active, to check if an inability to detect their corresponding Toy/app causes any problems or security flaws. It is important to remember that packet captures must also be taken from the app, as well as just the Toy device.

Stone and Margaritelli both perform packet sniffing with a number of different methods, depending on the test location and whether they are trying out new tools, and in fact Stone switches between two during his examination. Ultimately, it does not make an

enormous amount of difference what sniffing method is used as long as the examiner is aware of what data is available during the process. For example, I may use the LightBlue sniffer knowing it will not display device Types, but that is not a problem if I am not interested in a device type, but I know one is available should I want it. However, the Ramble app, though it will not display un-named devices or Characteristics, will store GPS and datetime data on where they were sighted, providing additional OSINT. It follows therefore, not to worry overly about the correct tool choice in this instance.

2.2.2.1.2. Analysis of PCAPs

The captured packets (PCAP files) must then be analysed in Wireshark for any data they contain about the WiFi, the phone, the audio payloads, or any other user information. Sample Bluetooth packets can be found via the Wireshark wiki [27] so we can make a comparison there for information which is unusual, but there are regrettably few of them.

A note was made of devices and their BLE MAC addresses in the vicinity in order to assist with identification and calibration of filters **[28]**. The Nokia5 is the main phone used for app communication with the toy devices ToyFi, Freddy and NuNu. Other devices are those expected to be used in the vicinity in general, except for temporary ones which may be neighbours or passers by simply close enough to be in signal range.

Nokia5	58:%%%%%%B9
iPad	2C:%%%%%%:5E
Toy_Fi	78:A5:04:15:F5:A0
Freddy	00:11:67:11:16:FF
NuNu	*0c:2a:69:0e:fa:2b
iPhone	B4:%%%%%%:CA
Apple watch	30:%%%%%%37
Hudl	60:0%%%%%%:F9
Echo Dot	00:%%%%%%B9
Windows PC	20:%%%%%%:94

*These are all Bluetooth MACs except the NuNu, which is the device MAC address.

2.2.3. APK Reversal

APK reversal is an important part of the investigation. Although analysing data in transmission can give great insight into functionality when the device and application are

running, examination of the APK codebase allows the examination of all possible functionality, even those events in the code which are not often triggered, and it is often these overlooked edge cases which can provide a vulnerable weak spot, such as the firmware.
Reverse engineering of the Android app will be done in several stages, commencing with some basic network analysis, followed by a static analysis of the app manifest and other key information files, and finally the main code. Dynamic analysis is unlikely to be necessary in an app of this small size and simplicity but will not be ruled out.

Apps are distributed to Android phones in the Android Package Kit file format (APK). The APK can be downloaded to a computer (usually from the Android phone, to ensure it is the manufacturer's intended version and not an illicitly modified one) and reversed in order to understand how it functions internally. Note is taken of authentication information in comparison to the previously gleaned BLE authentication data. The app will also be searched for any strings gleaned from any additional BLE characteristics in an effort to find any code related to them. Any insecure coding practices within the APK code can be noted.

2.2.3.1. Tools and Methods

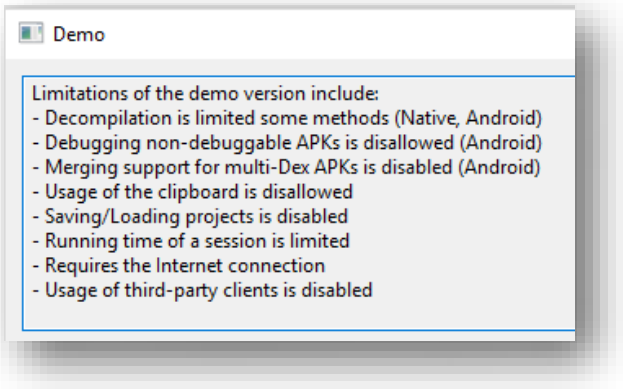
A Reverse Engineering environment on the Windows machine was set up, as per the Margaritelli methodology [20]. Java Runtime Environment (JRE) was first installed. Android Debug Bridge (ADB) was also installed and added to the Windows path. ADB enables communication with Android devices and provides a variety of actions, such as "installing and debugging apps, and it provides access to a Unix shell that you can use to run a variety of commands on a device." [29]

The APK can be downloaded from the Google Play Store to ensure it is the most widely used and up-to-date version for that toy, so ADB was used to pull them from the phone wherever possible. Where this was not possible the APK was downloaded from the manufacturer's official website.

Once obtained, the APK is examined for Malware and authenticity via upload to VirusTotal.com. This has the added advantage of listing any permissions which may be problematic, and any interesting strings, which can provide further pointers on things to look out for when viewing the app code.

As per Margaritelli, OpenSSL was also downloaded and installed in order to view the certificate part of the APK, to check whether it has been signed by the app developer or a third party and any other pertinent information.

A method of viewing the APK dex files in their original Java code was also required. Several methods were trialled, including jADX, dex2jar and Jeb as suggested in the Margaritelli methodology.



Of the three, Jeb was clearly the superior product, with colour coding to increase code readability and displaying a directory tree, but was unfortunately so restricted in the demonstration version that it was not suitable to continue with. The paid version, which also features arm decompilation, would be extremely useful for anyone working full-time in this area, but is over £1200 so is out of reach for the amateur.

jADX seems to have slightly more features than dex2Jar and a nicer interface so it was selected in the end, though there wasn't an enormous different. It is certainly the case, however, that reading Smali files is difficult and an interpreter is preferable, unless the intent is to modify code for re-signing and pushing back to the phone. Using the powerful jADX search feature is a significant part of the process at this point, as the codebase is searched for strings, IP addresses, Characteristic names and other relevant data from the information gathering and sniffing stages.

The functionality of the app is also looked at, especially any potentially problematic permissions and functions, and code around vulnerable areas like login, transmission of data, encryption and interfaces.

Notes are made of any discoveries. Any understanding of how data may be transmitted to or stored on the device or online may enable the next stages of the investigation.

2.2.4. Device Manipulation

If appropriate, an app or custom interface may then be used in an attempt to write control strings back to any open Characteristics on the device, if these can be gleaned from the APK. For example, Stone used Web Bluetooth to build a web interface to automate control of the CloudPets toy [30]. The nRF and LightBlue app also have features to push hex strings to connected devices.

2.2.5. Further Online Investigation

Any links found in the APK code to online locations can be investigated further with Search Engines, or other network/OSINT tools such as WHOIS, nmap, and Bettercap. Hunt also mentions in his write-up that the CloudPets database was on Shodan [6], which is another useful source of search information for vulnerable IoT devices. For more thorough and automated investigation Spiderfoot [31] may be of interest, which will automatically search "over 100 public data sources to gather intelligence on IP addresses, domain names, e-mail addresses, names and more", and is particularly good for correlating data for identifying possible inference attacks.

2.2.6. Conclusions and Assessment

Each investigation is then concluded with some discussion bringing together all the findings, and their security implications, recommendations to consumers and manufacturers, and details of any responsible disclosure made to the company, to a CERT, and finally to the public.

The device is then Risk Assessed on the following criteria:

1. Is personal info stored on the device? How difficult is it to access?
2. Can pairing take place? How secure is it?
3. Is personal information transmitted to/from the device? How secure is transmission?
4. Is personal data stored in an online location? How secure is storage and transmission?

The Risk Scores are suggested as None – 0, Low – 2, Med – 5, High – 10, and combine to give each device an overall score for comparison.

A further discussion of risk assessment appears later in this report.

In this section a methodology for the examination of IoT Toy devices was presented, with reference to what three experts have done previously in their own investigations. This is intended to offer a framework to draw from to ensure key areas are not missed, and somewhat homogenise the process, rather than an exhaustive examination process suitable for every device. Importantly, the inclusion of a simple risk assessment allows for basic consumer insight and comparison, which has been somewhat lacking in the past.

The following sections go on to test the methodology on some toy devices, and then analyse its effectiveness.

2.3. ToyFi Device Examination

ToyFi teddy is a children's toy designed to exchange audio messages between the child and a parent or other absent adults with whom the child has a bond. It is marketed as a charming way to keep in touch with grandparents and military parents on deployment. It claims to use Bluetooth LE and mobile app(s) to manage this exchange.

2.3.1. Information Gathering

The packaging directs to the manufacturer's website, but no mention can be found of ToyFi. However, the User Guide filed with the FCC reiterated the packaging claims of connecting and sending an audio message through Bluetooth LE, reply to messages by recording on the teddy, and connect to mobile devices within ten metres.

A physical examination of the toy led to the conclusion that despite some differences in embroidery and packaging, the ToyFi device is strikingly similar to the CloudPets teddy(Figure 6), and may therefore suffer from the same device security flaws.

Note the light-up red heart on the chest, and the record and playback buttons on each paw denoted by embroidered Wi-Fi signals.

Figure 6 - ToyFi teddy (L) and CloudPets (R) comparison photo with light up heart (top arrows) and playback buttons on paws (bottom arrows)

The FFC filing is by Dragon-I Toys Ltd in May 2014. Both the CloudPets(FCC ID: 2AD3BJAP85110) and ToyFi (FCC ID: 2ACBM80620) toys have FCC filing documentation which shows the internal physical mouldings are very similar visually(Figure 7,Figure 9), and the internal circuit board for the CloudPets toy is stamped ToyFi_v2.4.0 while the ToyFi one is stamped ToyFi Altium v2.2 (Figure 8). This further supports the idea that they have been manufactured by the same company.

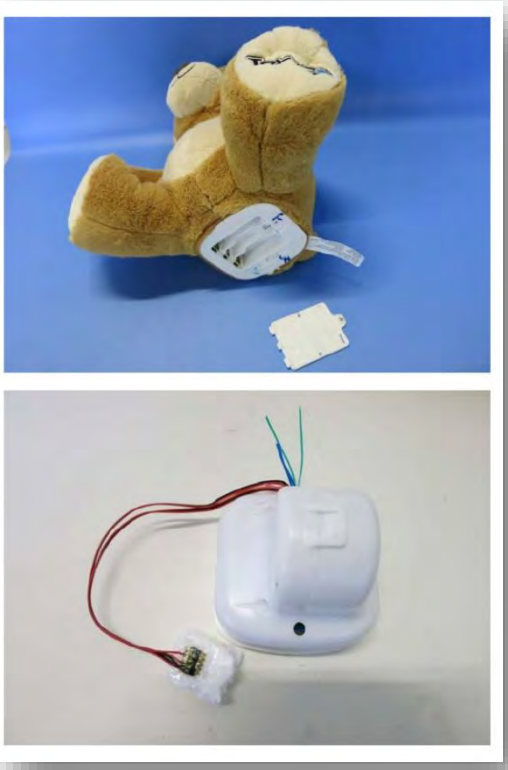


Figure 7 - ToyFi internals

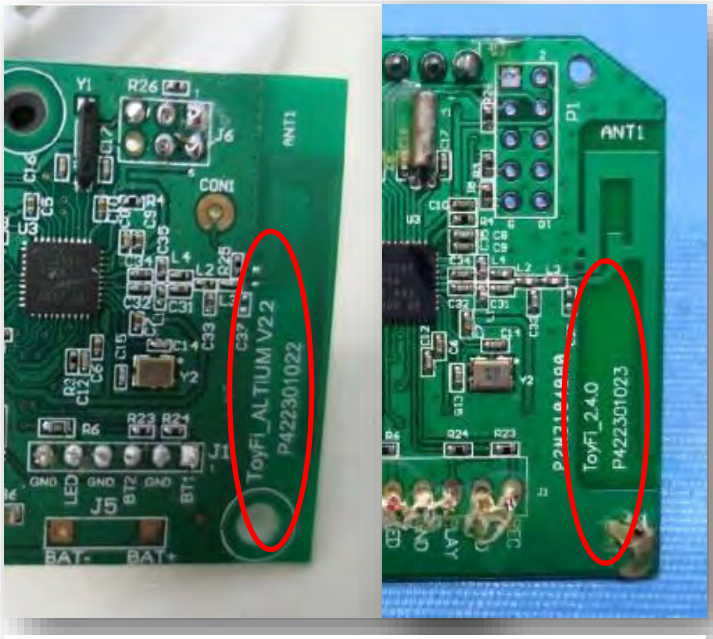


Figure 8 – ToyFi circuit board (L) and CloudPets circuit board (R)

Figure 9 - CloudPets internals

While following a methodology may not usually involve such close comparison to another physical original device, it provided valuable insight during the course of this investigation.

2.3.2. Device Advertisement Sniffing

The nRF sniffer app on the Android phone was used to see what BLE advertising data the ToyFi device was broadcasting (Figure 10).

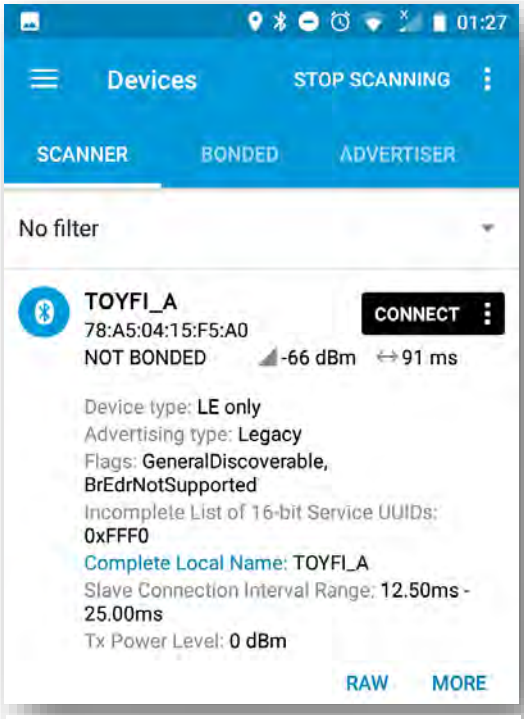


Figure 10 - The nRF app shows advertising data

It shows the Bluetooth Media Access Control (MAC) address the device is broadcasting on as 78:A5:04:15:F5:A0. A note is made of this, although from 4.0, BLE has the ability to change to a new, randomly generated Bluetooth MAC address periodically to avoid user tracking [32], so we need to be mindful that this address may change.

We can use the 'Raw' data (Figure 11) to verify the information shown to us by the app, to check that it's visual interpretation of the Bluetooth specification is accurate:

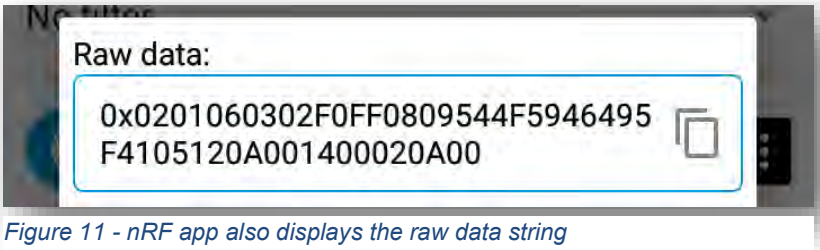


Figure 11 - nRF app also displays the raw data string

The raw data is reported as:
0x0201060302F0FF0809544F5946495F4105120A001400020A00

It can be split into Length, Type and Value, as follows:

02,01,06.03,02,F0FF.08,09,544F5946495F41.05,12,0A001400.02,0A,00.

Length (Bytes)	Type	Meaning	Value
02	0x01	Flags	0x06
03	0x02	Incomplete List of 16-bit Service Class UUIDs	0xF0FF
08	0x09	Complete Local Name	0x09544F5946495F41
05	0x12	Slave Connection Interval Range	0x0A001400
02	0x0A	Tx Power Level	0x00

'Meaning' is derived from the type column, from the Bluetooth Generic Access Profile Assigned Numbers reference [33]

The **Flags** declaration is set to 06, or 00000110. This means bit positions 1 and 2 are set, which translates to:

Bit 1 : "*LE General Discoverable Mode*"

These tell us that it's a Bluetooth LE device, rather than a Bluetooth Classic device, and that it's Generally Discoverable, meaning it will advertise its presence constantly. By contrast, 'Limited Discovery' devices save on battery by only advertising for 30 seconds each time they are triggered, and tend to be those with keyboards or other human interface devices (HID), and devices will generally be set to 'Non-Discoverable' after they are paired.

Bit 2: "*BR/EDR Not Supported*"

This is short for Bluetooth Basic Rate/ Enhanced Data Rate, and is a Classic Bluetooth point-to-point continuous data streaming protocol.

Incomplete List of 16-bit Service Class UUIDs tells us that the device has not declared the full list of 128-bit services it supports as part of this advertising string.

For **Complete Local Name**, converting the hex number '09544F5946495F41' to text gives 'TOYFI_A'.

Slave Connection Interval Range

BLE devices connect and send data in short bursts to preserve battery life, so the peripheral device (which is usually the lower powered of the two) defines a preferred connection interval range whose minimum depends on battery considerations and whose maximum depends on available buffer size. The Central (master) device should then use this information when establishing a connection. The first 2 octets (0A00) define the minimum and the second 2 octets (1400) define the maximum [34].

Tx Power Level refers to the power the advertising packet was transmitted at and is useful for calculating path loss. It is given as 0dBm, which is what the app is reporting. It is worth noting that this is within the usual power range for BLE devices of -30 to 0dBM [35].

In addition, the 'device type' and 'advertising type' are also sent as part of the mandatory 'GAP service' data.

Aside from basic connection information we can tell from the **Incomplete List of 16-bit Service Class UUIDs** that we can connect to our device and query for more Services information (Figure 12), and on doing so we can see the following additional details:

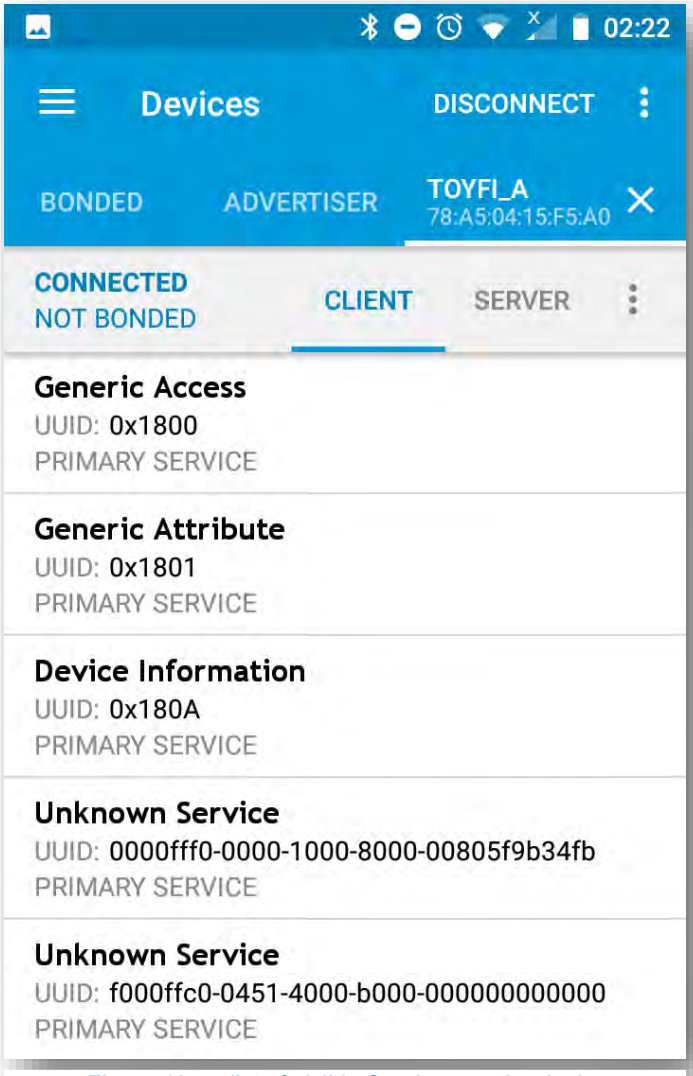


Figure 12 - a list of visible Services on the device

Generic Access (GAP) attributes contains mandatory details such as *Device Name, Appearance, Peripheral Privacy Flag (Privacy is disabled in this device), Reconnection Address* and *Peripheral Preferred Connection Parameters*. These are mostly concerned with managing the low-level details of the connection with the Central.

Generic Attribute contains the *Service Changed* characteristic which allows the addition of new services without re-bonding. This alerts the client when updates have taken place.

Device Information has Characteristics which contain values for *System ID, Model Number String, Serial Number String, Firmware Revision String, Hardware Revision String, Software Revision String, IEEE 11073-20601 Regulatory Certification Data List, Manufacturer Name String* and *PnP ID (which shows as Texas Instruments Inc, Product Version: 272)*

The two Unknown Services are of particular interest, however, as these are custom to the device and may therefore represent custom functionality. Figure 13 is a screen shot from the 'LightBlue' iOS BLE sniffer app, which shows all Characteristics of the Services and their Properties on one screen, which is a little easier to read.

Each Service has a UUID and a number of Characteristics. Each Characteristic also has a UUID, a Handle, and a number of Properties which determine how it can be interacted with.

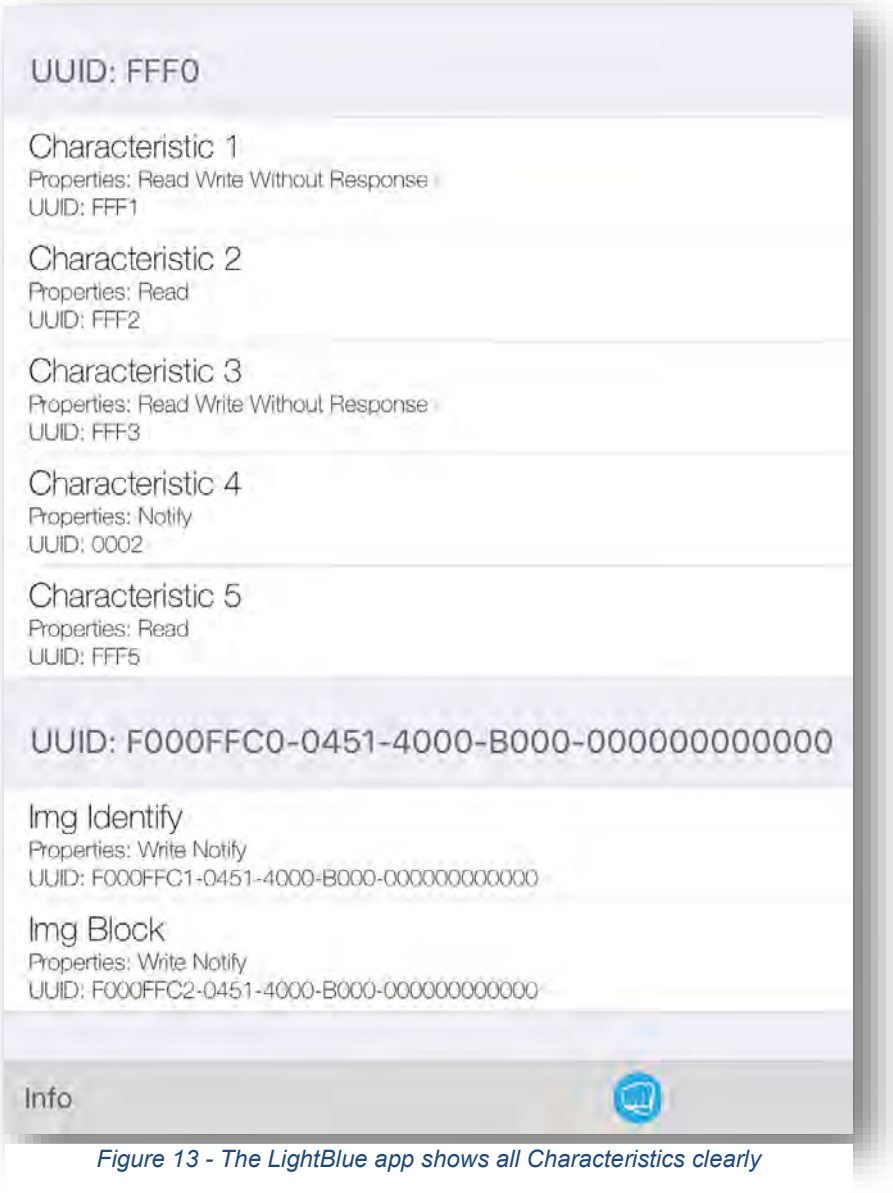


Figure 13 - The LightBlue app shows all Characteristics clearly

For example:

Service UUID: FFF0		
Handle	Characteristic 1	The name assigned to the characteristic
UUID	FFF1	The unique ID assigned
Properties	Read, Write Without Response	Can be read by anyone connected, can be written to by anyone connected, without waiting for a response between packets (this increases data throughput [36])

As has already been revealed, connection to the device is open and unauthenticated.

Although the BLE GATT layer provides for both encryption and authentication properties [37], neither has been used for any of the Characteristics on this device.

The UUIDs and Handles of these Characteristics are noted for comparison with the Android app code later. It is an early theory that they may be used to store the audio messages or other user data within the device.

2.3.3. Connection Sniffing

Dynamic testing of the communication between the device, the mobile app, and the internet was also carried out.

The APK file was downloadable directly from a link from the toy manufacturer (). The hash of the downloaded file was checked to ensure it had not been altered in any way from the original. It was also checked via VirusTotal, which checks the file for Viruses and other malicious payloads.

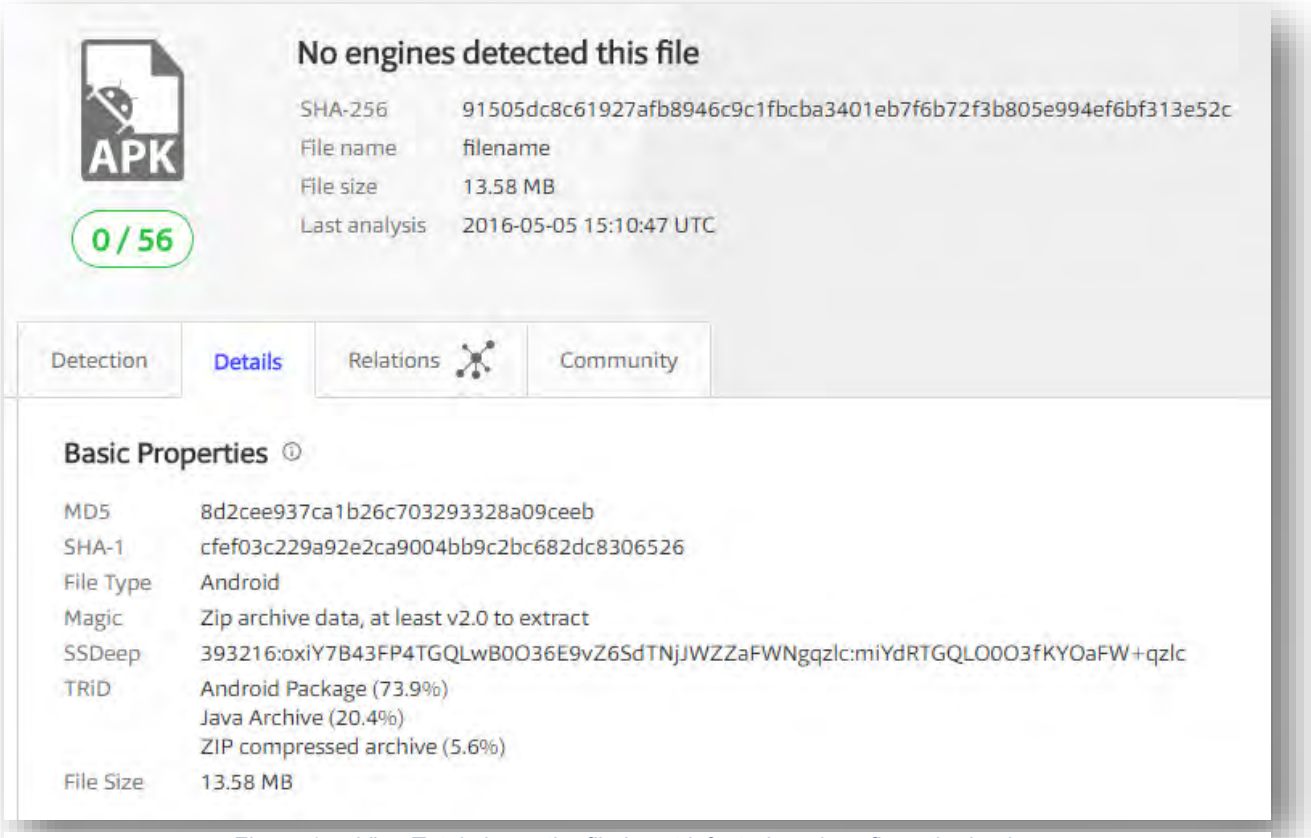


Figure 14 - VirusTotal shows the file is not infected, and confirms the hashes

It also points out any interesting or suspicious permissions (Figure 16) and strings(Figure 15), which can then be located within the code to uncover further context about their use.

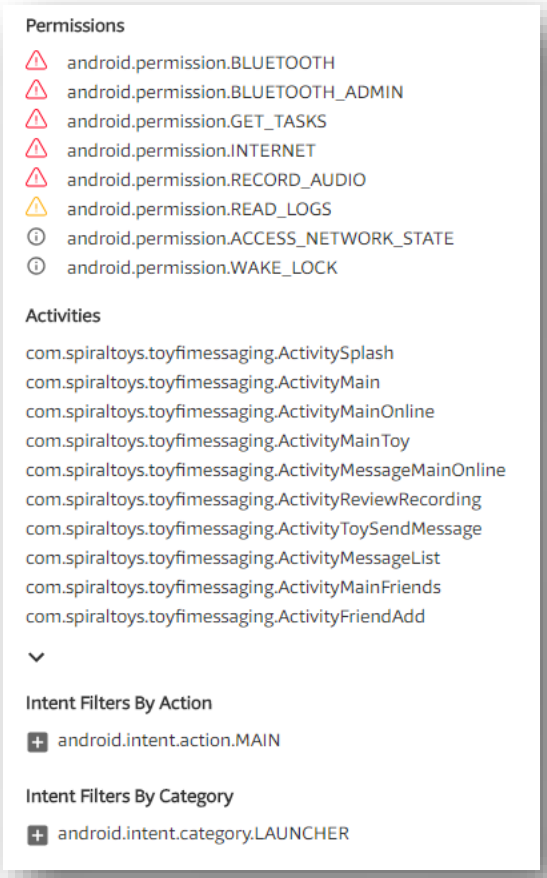


Figure 16 - VirusTotal highlights the main activities in the app, and potentially concerning permissions

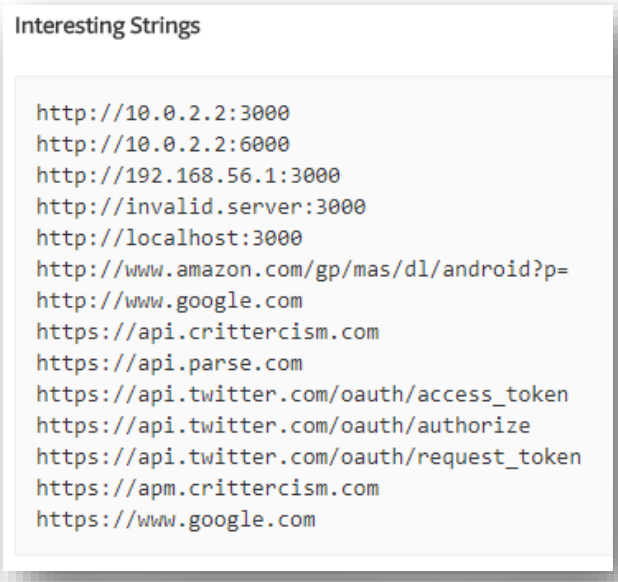


Figure 15 - VirusTotal picks out interesting strings from the APK

The APK was then pushed onto a Nokia 5 Android phone running Android 8.0 Oreo, via Android Debug Bridge (ADB).

The tPacketCapture app was then used in conjunction with the HCI snoop log to ensure that all Bluetooth and Wi-Fi communication to and from the app was captured.

However, the app did not function as expected at this juncture, repeatedly refusing to allow the creation of a user account. It also transpired that no communication with the device was possible without a user account, which was also unexpected, as there had been little previous mention of online accounts, and some rudimentary functionality was expected even if online functions were restricted or non-functional. The limited functionality therefore greatly the scope of the packet captures. However, these did provide some insight about the problem when they were examined in Wireshark.

```
ec2-54-215-17-250.us-west-1.compute.amazonaws.com  10.8.0.1    TLSv1.2    296 New Session Ticket, Change Cipher
```

The app does manage to connect to an amazon aws server on port 443, indicating https, and the presence of TLSv1.2 Record Layer: Application Data Protocol: http-over-tls, but after sending only a relatively small amount of encrypted data the server sends an Encrypted Alert and closes the connection. This happens every time a connection is attempted.

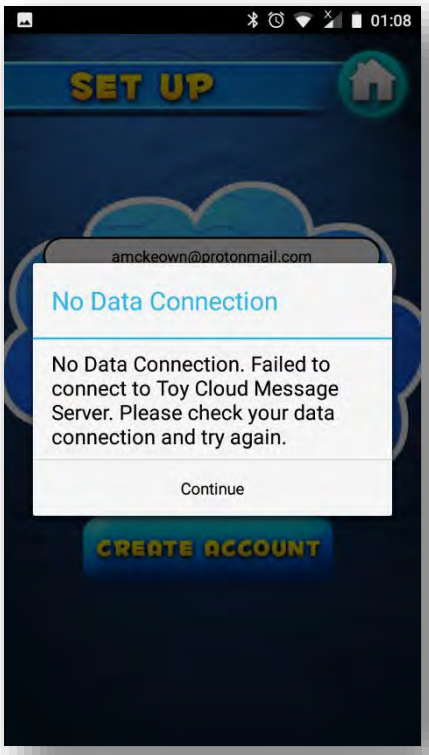


Figure 17 - The app unable to connect

The app itself continues to report an error message that it is unable to find a data connection (Figure 17). The data connection on the phone was activated in case this was the problem, and the operation repeated, but the same error occurred.

Online searching shows other users complaining of the same problem [38].

2.3.4. APK Reversal

Given that there have already been similarities with the CloudPets toy, it was considered of interest to also reverse the CloudPets app and make a cursory comparison between the two in case this could provide parallels with and insight from existing investigations. A more in-depth comparison with the CloudPets app was considered of limited usefulness and outside the scope of this project.

The reversal followed the Margaritelli methodology [20] looking first at the app manifest to try and gain some insight into the major parts of the application.

After connecting the phone via USB, ADB is used to list all the app packages (Figure 18) within the phone, in order to determine the specific names of the ones we want to pull to the Windows machine for reversal.

Packages with both ToyFi and Cloudpets in the names were found, and then ADB was used to find the full path and pull the contents into separate folders.

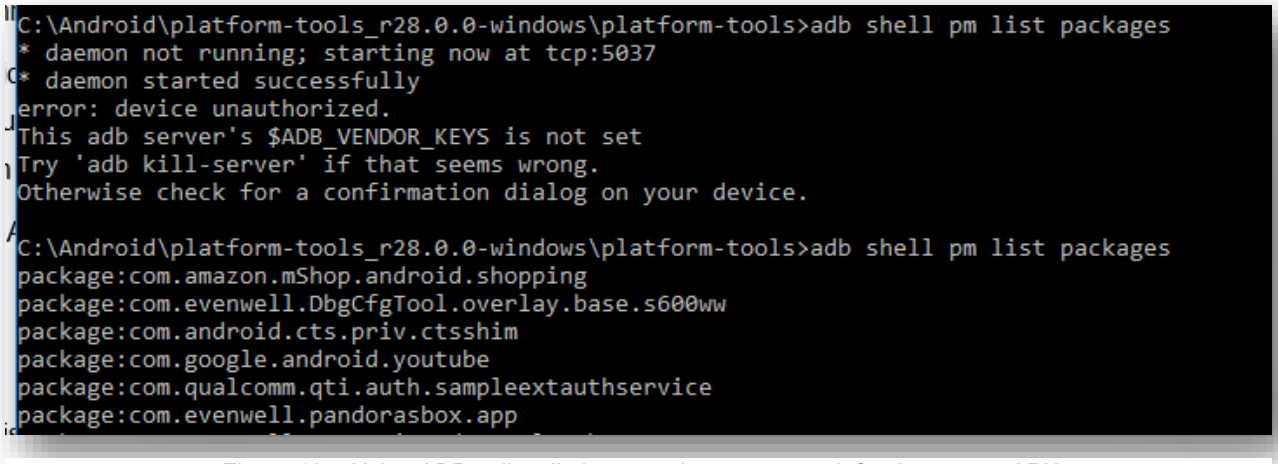


Figure 18 – Using ADB to list all phone packages to search for the correct APK

This further cements the idea that the devices are developed by the same company, as the main app package names are both *spiraltoys (*Figure 19 *and* Figure 20*)*.

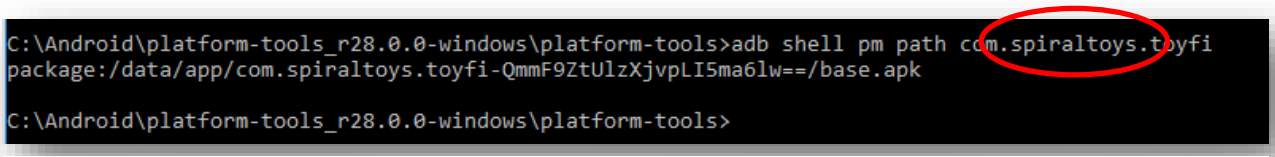


Figure 19 - Finding out the path of the ToyFi APK. The main package name is spiraltoys.

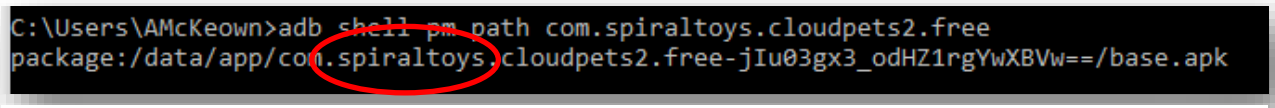


Figure 20 - Finding out the path of the Cloudpets APK. The main package name is also spiraltoys.

Indeed, when the certificates are examined using OpenSSL they also show the same issuer. It is also of note that the ToyFi app certificate has been issued on Jun 29 2014, 9 months before the CloudPets one on Apr 1 2015 (Figure 21), so one possible implication is that the ToyFi app could be an earlier and less secure implementation.

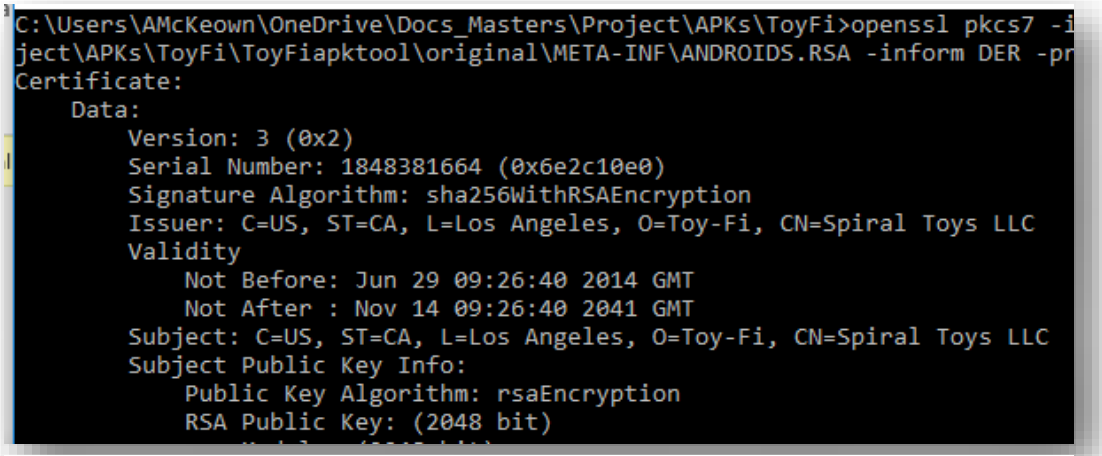


Figure 21 - ToyFi app certificate (via OpenSSL) dated Jun 29 2014, Spiral Toys LLC of Los Angeles

The manifest lists the main ‘activities’ used within the app. These are the screens that the app will display to the user. It also shows the system permissions that the app will request access to. It can be seen that this is a relatively simple app as it has fewer than twenty screens, and only requests a handful of permissions. As can be expected, given the feature list, the permissions requested include Internet, Record_Audio, Bluetooth and Bluetooth_Admin.

The activities list helps to give an quick list of where to focus efforts, as things like ToySendMessage, RecordMessage, Login, ChangePassword and FriendAdd are more likely to be security critical.

There is also a meta-data value called “crittercismKey” stored in the manifest with a value of “53b650f6466eda5116000004”, which hooks into an online service for tracking app crashes and other analytics.

The CloudPets manifest is a little more complicated, and makes use of the Firebase platform for some analytics, as we can see this mentioned several times throughout the manifest.

Moving on from the manifest, an examination is made of the app code itself via jADX.

Further connections with the CloudPets app can be found within the ToyFi Java code, where the CloudPets string is used as an ID within the app, though the only call to it is to getSharedPreferences so it is not clear why.

In package com.spiraltoys.toyfimessaging.ToyApp we find the following:

```
public class ToyApp extends Application {
    private static final String PERSIST_ID = "cloudpets";
```

A folder named ‘Wappworks’ would seem to suggest that (using online OSINT research) this Canadian company may have been used to create all or part of the app.

Using a string search it was found that package com.spiraltoys.toyfimessaging.toy.ToyDef deals with the Bluetooth GATT service and contains references to the UUIDs relating to the Device Name, ImgBlock, and other custom characteristics noted earlier from the advertising packet.

```
public class ToyDef {
    public static final String[] DEVICENAME_FIRMWARE_A_PREFIXES = new String[]{"Toy-Fi A", "TOYFI_A"};
    public static final String[] DEVICENAME_PREFIXES = new String[]{"Toy-Fi", "TOYFI"};
    public static final UUID GATTSERVICE_MAIN_UUID = UUID.fromString("0000fff0-0000-1000-8000-00805f9b34fb");
    public static final UUID GATTSERVICE_OTAUPDATE_UUID = UUID.fromString("f000ffc0-0451-4000-b000-000000000000");

    public enum GattMainCharacteristic {
        SEND("0000fff1-0000-1000-8000-00805f9b34fb"),
        RECEIVE("00000002-0000-1000-8000-00805f9b34fb");

        public final UUID uuid;

        private GattMainCharacteristic(String descUuid) {
            this.uuid = UUID.fromString(descUuid);
        }
    }

    public enum GattOtaUpdateCharacteristic {
        OTAUPDATE_IMAGENOTIFY("f000ffC1-0451-4000-b000-000000000000"),
        OTAUPDATE_IMAGEBLOCKREQ("f000ffC2-0451-4000-b000-000000000000");

        public final UUID uuid;

        private GattOtaUpdateCharacteristic(String descUuid) {
            this.uuid = UUID.fromString(descUuid);
        }
    }
}
```

We can see from this code that GATTSERVICE_MAIN has a UUID string which correlates with custom Characteristics 1-5 from the advertising packet examined earlier, and GATTSERVICE_OTAUPDATE has a UUID string which correlates with the custom Image characteristics.

Strangely, it is noted that only Characteristic 1 (FFF1) is used for storing and Characteristic 4 (0002) is used for receiving from the device, and the two Img Characteristics are used for Over The Air (OTA) updates which are device firmware updates in this case (ToyTaskUpdateFirmware calls this method), and custom Characteristics 2, 3 or 5 do not appear to be utilised at all.

Unfortunately, we can also see from the other code that the application never gets this far, as it fails as soon as it is unable to successfully connect to a server at the password screen, and so the BLE parts of the toy are never initialised during our tests.

Although ActivitySignUp (which is called when the user attempts to sign up for a new user account, as demonstrated earlier) only requires a very simple success check of a 1 to be returned if the server connection has been successful to allow progress to ActivityMainOnline, we can see from the code therein that multiple checks are made thereafter to the server for the presence of waiting messages and friends, and so server connection is intrinsic to general app functionality.

We can also inspect the signup code, which asks for the user to enter a password and confirm it. The only check made is this:

```
(password.length() <= 0 || !password.equals(passwordConfirm)
```

We can also examine how audio files are stored on the device.

From com.spiraltoys.toyfimessaging.server.ModelSecurity we can see that a 'security key' is generated during signup.

```
27    public static String generateSecurityKey(ParseUser user) {
28        SimpleDateFormat dateFormat = new SimpleDateFormat("MM-dd-yyyy_HH-mm-ss");
30        String securityKey = String.format("%s-%s", new Object[]{dateFormat.format(new Date()), user.getEmail()});
```

It is a simple concatenation of a datetime format and the user's email address. It is created during the signup process and then uploaded to the server when a user logs in (toyfimessagingserver.ServerTaskLogin), presumably to be compared to the one created during the signup process. It is also built into the BLE data when a message is sent to the device.

```
        runOp(new ToyOpGattConnect(gattManager));
        try {
            messageChunks = new BleMessageAudio(audioData, this.senderUuid, this.securityKey).build();
```

2.3.5. Device Manipulation

It is somewhat confusing that Pairing requires authentication, as we already know we can read openly from the important Characteristics of the device and that data is sent there in plaintext. However, it is still a good security measure that the device speakers and microphone cannot just be connected to by anyone within range as easily as a Bluetooth headset.

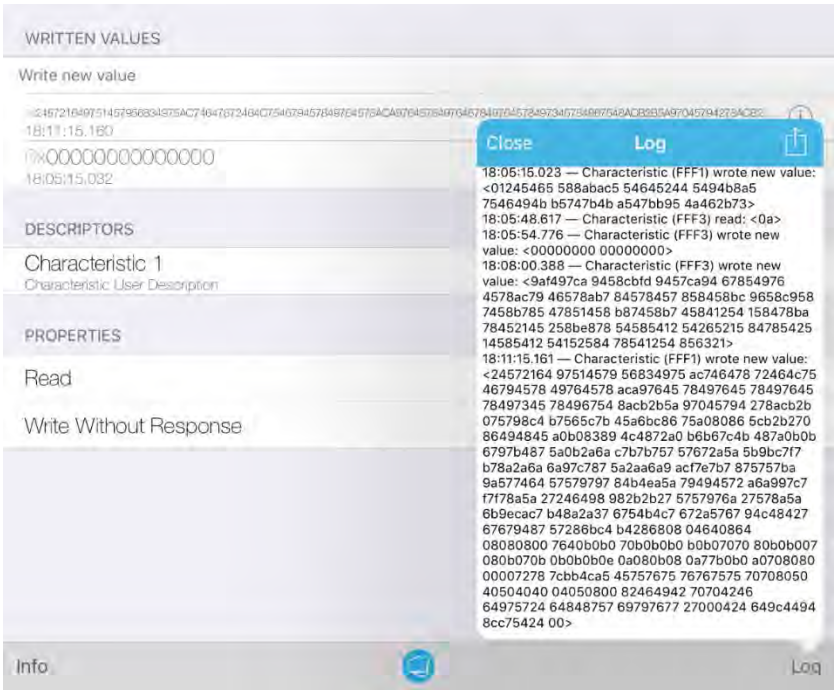


Figure 22 - Proof of concept writing new values to Characteristic 1

Each Characteristic with open Write ability can still be written to, however, and this only requires the app or technical ability to do so. Stone uses Web

Bluetooth to build a web interface for the CloudPets toy, for example. The LightBlue and nRF apps also give the ability to write strings back to these Characteristics (Figure 22), which could easily then be read straight into the app itself, causing buffer overflows or other attacks. Unfortunately, it is impossible to test as the app is non-functional without the server connection.

2.3.6. Further Online Investigation

For further correlation with CloudPets, the FAQ from the ToyFi mobile application actually opens the URL “http://www.cloudpets.com/Cloud-Pets-FAQs.dtm”. This is now a parked URL with godaddy, but it’s clear that the link with CloudPets, despite how different the apps are, is irrefutable.

As such, it seems likely that further OSINT and online work will be fruitless as far as revealing further vulnerabilities goes. Since the app is not working, the Toy device cannot function for the consumer and so can pose no harm.

2.3.7. ToyFi Conclusions and Assessment

As the examination of the ToyFi device proceeded the Stone methodology choice became increasingly vindicated, as it slowly became clear that the CloudPets manufacturer had simply rebranded and relaunched the problematic toy under a different product and company name. This was not known when the device was purchased.

The inability to connect to the online server and therefore interact with the app made it sadly impossible to do further research into the device functionality, thus severely limiting the intended scope of the original examination for this device. However, it has some clear and identifiable flaws.

1 - Is personal info stored on the device? How difficult is it to access? HIGH RISK

The Security Key is not encrypted or hashed in any way, and therefore seems mainly to be used as a timestamp and identifier rather than a robust security token.

Because we already know that data can be freely read from this Characteristic in plaintext, and the format of the data is a concatenation of Audio data, sender UUID and SecurityKey, we could read data stored in this Characteristic from any device within range (the BLE specifications and device documentation put this at 10m) and derive the users SecurityKey and UUID from this. We could also listen to any private Audio message left on the device.

Another possibility with the ability to write data freely to this Characteristic without constraint is Buffer Overflow or Code Injection attacks on the app, causing instability or remote code execution on the Android phone.

2 - Can pairing take place? How secure is it? - MEDIUM RISK

Pairing needs a code, which is a reasonable security measure, and prevents non-expert users from connecting easily to the speakers and microphone of the device.

3 - Is personal information transmitted to/from the device? How secure is transmission? HIGH RISK

It can be seen from the APK that a normal audio codec is used for transmission of the audio data, and it is simply concatenated into a packet with the Security Key and UUID, and no encryption is employed. Unfortunately, there was no way to verify and evidence this with pcaps due to the lack of app functionality.

4 - Is personal data stored in an online location? How secure is storage and transmission? HIGH RISK

It has been decided to designate this as *high risk *untested* on the balance of evidence, due to the clues within the app as to how personal data is transmitted to the server with a SecurityKey that can be spoofed, no enforcement of strong passwords, and the knowledge of the very close ties to the CloudPets app which operated an open Mongo database with customer credentials. The product is no longer functional and is withdrawn from the market, and so speculation will not be damaging to sales.

The UUID and SecurityKey could also be used in a spoofing attack to send modified packets to the online server as if we were the user, possibly gaining access to additional personal messages, and receiving or sending messages with the Victim's credentials.

Unfortunately, as the servers had been taken offline at the time of testing it was not possible to demonstrate this attack.

It is difficult to make a fair assessment of the privacy implications of this Toy device due to the inability to correctly assess the app component, which is where a lot of OSINT information may leak from.

It is clear that a Device Risk Score of 35 is very high, and it is recommended that consumers avoid the device altogether in its current form as both the app and device

appear to be replete with security flaws which should make consumers very wary about trusting any personal information to either.

The failure of the manufacturer to utilise basic good practice when it comes to enforcing strong passwords, adding encryption to code, and turning on basic Bluetooth security features, does not engender trust and each of these things must be corrected urgently.

As the device and app have been removed from sale, no responsible disclosure is planned.

Despite the fact that this Toy device was essentially too broken to fully test, the thoroughness of the early elements of the Stone and Margaritelli examinations meant that a good amount of data was nonetheless generated, and conclusions inferred, in a way that a standalone examination would not, perhaps, have been able to achieve. Stone has also demonstrated that more can be done with regard to device manipulation, although his path there using pcap information cannot be followed in this instance and another route must be found. Such an examination is outside the scope and timeframe of this project, however.

2.4. Freddy Device Examination

My Friend Freddy bear is marketed as a totally safe interactive toy which will talk to your child using personalised details and fun, developmental games and stories. Family details and short audio messages can also be stored in the app. It claims to use Bluetooth and an Android or iOS mobile app to manage this, but absolutely no Wi-Fi.

It is claimed to be completely safe because it never connects to the internet, the app does not use Wi-Fi, and because the information is “not of a highly personal nature such as last name, address or telephone”.

2.4.1. Information Gathering

The packaging is in German and the user manual that shipped with the device points to a UK website which comes up blank. However, using the Internet Archive’s *Wayback Machine* facility we can uncover [39] that Freddy is using Bluetooth (Figure 23 - Website image showing Bluetooth 3.0), rather than BLE. This means that external sniffing is not possible, as the apps and dongles available for this examination only sniff for BLE packets, but interception via the internal Android phone logs can still take place.

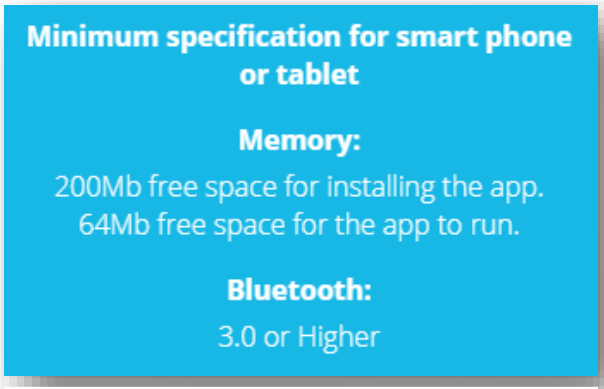


Figure 23 - Website image showing Bluetooth 3.0

The online documentation also assures users that there is a ‘bad words filter’ within the app, and this was noted for later examination.

The FCC filing (FCC ID: NS685585-BT) is in July 2015 by *Manly Toys Ltd*, and gives detailed specifications about the internal electronics of the device, including that it supports Bluetooth 3.0, corroborating our earlier finding.

The Google Play store shows that the app has over ten thousand installs, but reviews show that a very large number of users had problems getting the app to accept their information as no return key was displayed [40]. The only fix is alleged to be the installation of a third-party keyboard in order to regain control of the data submission, and this was noted for later testing.

At first glance, the amount of personally identifying information that the Freddy app asks for is quite alarming, from a privacy context. The fifty personalisation questions include

information such as your child's name, age, birthday, favourite colour, sibling's names, best friend's name, favourite book, film, pet's name, type of cake, lunch, dinner, snack, things you like to do at Grandma's house, in the garden. All information which could be devastating in the hands of someone wanting to groom a child, for example.

2.4.2. APK Reversal

As with the ToyFi device, reversal followed the Margaritelli methodology [20] looking first at the app manifest to try and gain some insight into the major parts of the application.

After connecting the phone via USB, ADB is used to list all the app packages within the phone, in order to determine the specific names of the ones we want to pull to the Windows machine for reversal. A package with Freddy in the name was found, and then ADB was used to find the full path and pull the contents into a separate folder(Figure 24).

```
C:\Android\platform-tools_r28.0.0-windows\platform-tools>adb shell pm path com.toyquest.Freddy.EN
package:/data/app/com.toyquest.Freddy.EN-ED9QfKhaPxTlrcmOtG4Eew==/base.apk

C:\Android\platform-tools_r28.0.0-windows\platform-tools>
```

Figure 24 – Finding out the path of the Freddy APK

A look at the app certificate with OpenSSL shows that the issuer is 'Egg Cartonstudios', which is another, different name to add to the list of those associated with the production of this device/app.

```
C:\Users\AMcKeown\OneDrive\Docs_Masters\Project\APKs\Freddy\freddy\META-INF>openssl pkcs7 -in
cert.rsa -inform DER -print_certs -text -noout
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 1433753844 (0x557558f4)
        Signature Algorithm: sha1WithRSAEncryption
        Issuer: C=CH, ST=HongKong, L=HongKong, O=Cartonstudios, OU=Egg, CN=Egg Cartonstudios
        Validity
            Not Before: Jun  8 08:57:24 2015 GMT
            Not After : May 15 08:57:24 2115 GMT
        Subject: C=CH, ST=HongKong, L=HongKong, O=Cartonstudios, OU=Egg, CN=Egg Cartonstudios
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
            RSA Public Key: (1024 bit)
                Modulus (1024 bit):
```

Figure 25 - Freddy certificate dated June 8 2015, Egg Cartonstudios, Hongkong

The app manifest is very short and shows only one main activity so it is assumed that, rather than this being a monolithic app with only one screen, instead it calls to some kind of framework. Again, the expected Android system permissions are requested based on the features stated and include Bluetooth and Bluetooth_Admin, but interestingly the permission to Record_Audio is also there although this functionality is not mentioned in

the documentation. This may be a planned future feature, and is noted for further investigation.

An intent-filter (essentially a listener which can trigger the activity launch) is also defined for something called *Nuance* [41], which some online investigation reveals to be a text-to-speech generator.

There is also a service set up to download *Cocos2dx* expansion files, and listen for updates to these. Google Play store requires APKs to be no larger than 100Mb or 50Mb before 28 Sept 2015 [42], so APKs which are larger utilise Expansion Files [43] of up to 2Gb to get around the limit. These are usually declared in the manifest and then downloaded seamlessly along with the APK when it is selected from the store. *Cocos2d-x* [44] is a mobile app/game development framework, so this may explain why there is only one single activity declared.

The APK is uploaded to VirusTotal, as previously, and is clean. Interestingly it shows the two arm libraries; Cocos2d and NuanceVocalizer.

2.4.3. Device Advertisement Sniffing/Connection Sniffing

As mentioned, sniffing for this device was only possible via the Android phone logs, and so these were captured after APK installation and imported into Wireshark for analysis.

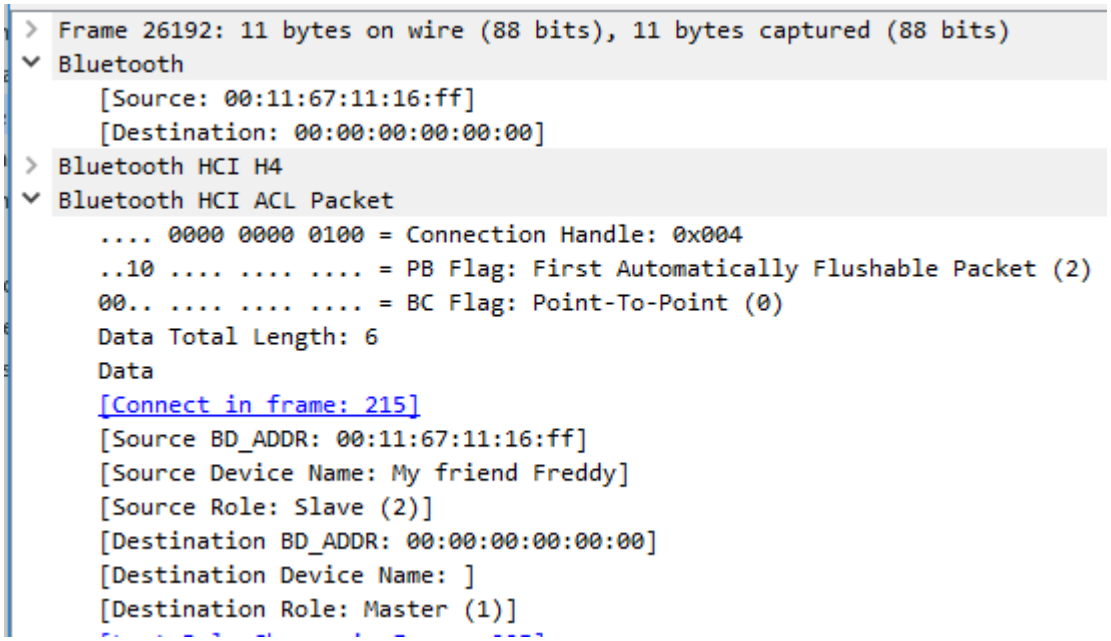
Once the Toy device is powered on it starts talking, but no personal data is heard. However, once it is paired and the app is started, personal data is used in the conversation. If the app crashes, loses focus on the phone, or the pair is lost, the Toy device reverts to general conversation. When the Toy device is turned on a pairing request is sent to the phone immediately.

This time the PCAP features a multitude of protocols (Figure 26), as Bluetooth audio connections are

Protocol	Percent Packets
Frame	100.0
Bluetooth	100.0
Bluetooth HCI H4	100.0
Bluetooth HCI Event	44.3
Bluetooth HCI Command	1.5
Bluetooth HCI ACL Packet	54.2
Bluetooth L2CAP Protocol	53.9
Malformed Packet	0.0
Bluetooth Security Manager Protocol	0.0
Bluetooth SDP Protocol	0.1
Bluetooth RFCOMM Protocol	0.4
Bluetooth HFP Profile	0.4
Bluetooth AVDTP Protocol	0.6
Bluetooth Attribute Protocol	0.0
Bluetooth A2DP Profile	41.7
Real-Time Transport Protocol	41.7
Bluetooth A2DP Content Protection Header SCMS-T	41.7
Bluetooth SBC Codec	41.7
Malformed Packet	41.2

Figure 26 - Protocol hierarchy statistics from Freddy pcap

clearly a lot more complicated than simple BLE. What seems particularly striking is the high percentage of malformed packets, and this tallies with the aural experience, as Freddy often misses words or drops the start or end of a sentence entirely.



We can see from several packets that the phone is the Master Role and Freddy is the slave, which puts the phone in charge of the frequency hopping calculations.

The PCAP is a steady flow of multiple L2CAP connection requests, HCI Events and Malformed packets as the app selects and sends new audio snippets to the Toy device.

As we already know from the APK reversal that only audio is being sent out to the device by the app, further focus on connection sniffing complex Bluetooth audio protocols was deemed to be unnecessary from a privacy standpoint.

2.4.4. Device Manipulation

Freddy is designated as a Bluetooth headphone device, and so it is easy to manipulate within a ten metre range. Whilst in the midst of casual conversation the device was paired with and music played directly through the device, but any audio file could have been selected, or even a microphone connected.

In addition, if a phone call is received while the teddy is paired, even if the app is active, the caller's audio will be heard through the teddy. This means an attacker who gains the mobile number of the paired mobile device can, if they can induce the adult to answer the phone while still paired, speak or play audio directly to the child who has the toy. The chip inside the Toy device has built-in capability to transfer audio back to the phone, but the app code has not utilised this.

2.4.5. Further Online Investigation

The FCC documentation shows the US manual which gives a link to the US website www.myfriendteddy.com, which shows the toy is still currently available at Walmart.

Further online research reveals that the famously insecure My Friend Cayla doll which was mentioned in the Background info section of this report is also produced by this manufacturer. The FCC ID (NS631837-BT) for the My Friend Cayla doll can be found by following the manufacturer link back and looking at other FFC toy applications.

https://www.genesis-toys.com/ also shows the Cayla doll on their page, and their awards page makes multiple mentions of the awards the Cayla doll has been given.

URLS associated with device: www.myfriendfreddybear.co.uk, www.myfriendteddy.com

Company names associated with device: Manley Toys Ltd, Genesis Toys, toyquest, Egg Cartonstudios.

2.4.6. Freddy Conclusions and Assessment

The Wayback Machine is a useful addition to the methodology, as it can unearth information which manufacturer's may have published openly in the past but then removed when it later proved to be problematic.

It was not mentioned anywhere else on the packaging or docs that this device was not using BLE, so it was not immediately obvious that sniffing would not be possible with our usual apps, and some time was wasted wondering if tool functionality was at fault before this step was added and it became clear that the device was using normal Bluetooth. Therefore, this is a good step to add to the new methodology for times when older devices may be examined, or information may otherwise be buried in the past.

The dangers of phone calls while paired is also new and not mentioned in other examinations, and could be worth adding more specifically as something to test for. As was noted previously, the facility to guard against this problem is built into the hardware, it just was not utilised.

1 - Is personal info stored on the device? How difficult is it to access? LOW RISK

The device makes sensible use of the Bluetooth Central/Peripheral paradigm, so that all personal data is collated, parsed by the Nuance software and only then sent to the

device if it is already paired. This means that no data need ever exist on the device, and if the pairing is lost, all data remains on the app. Due to Android sandboxing the data is not at significant risk on the phone either, except from expert users who may be able to extract it forensically with personal access to the device. This is partly because it is data which is not being transmitted ‘as is’, but rather is parsed into audio first and then transmitted. Many of the attacks mentioned in the Theoretical foundations section by Zhang et al [16] rely on data being handled by communal Android system processes, which is not happening here.

However, as was previously noted, the installation of a third-party keyboard was required by many users in order to complete the fifty questions to personalise the device, and these keyboards explicitly request permission to send all data entered back to their servers, ostensibly to improve predictive text. This is an obvious privacy flaw, and depending on the keyboard these transmissions may or may not be secure. Whether a consumer chooses to accept this risk will depend on their threat model.

2 - Can pairing take place? How secure is it? - HIGH RISK

The enormous security flaw in this device is that it can be openly paired with as a Bluetooth speaker. This means that anyone within roughly ten metres, (the specification stipulates a 33-foot minimum), can send inappropriate audio to the device.

It is not difficult to imagine a worst-case scenario with a child going to sleep with their favourite teddy after a bedtime story, daddy’s phone unpairing because he has wandered too far away, and a stranger sending scary or inappropriate audio to the device.

The added danger of receiving a nasty phone call while paired with Freddy is also concerning.

3 - Is personal information transmitted to/from the device? How secure is transmission? LOW RISK

Personal information is stored by the app, converted into audio, and transmitted to the device only when paired.

4 - Is personal data stored in an online location? How secure is storage and transmission? NONE

As this device does not make any use of Wi-Fi or online storage at all there is no threat at all to personal data online.

As has been discussed, the personal information is relatively safe on the app, but there is such a tremendous amount of it that it would be fair to believe the whole family could be at risk if it was released. The birth dates and names of extended family, pet names, and favourite activities and foods are certainly enough to groom a child, but possibly also break into bank accounts and fool grandma out of some inheritance money. Given than the app itself is not password protected, this could make a simple phone theft worthwhile.

At a Device Risk Score of 14, consumers can feel reasonably confident that this device is likely safe enough if they live and use it in a relatively remote location, always supervise use and remember to turn the Toy device off when not in use.

The manufacturer would be well advised update the app to password protect the settings section, and utilise a pairing pin and bonding with Bluetooth to prevent strangers being able to connect to it easily. In future device firmware updates they should also utilise the hardware facility which auto-redirects the speaker to the phone headset when a phone call is received.

2.5. NuNu Device Examination

NuNu and the other toys in this range bill themselves as a way to communicate with your kids remotely without giving them a phone. The toys record and send messages to an adult's phone, and also to each other when authorised (Figure 27), and can download games, stories and songs to play. They claim to achieve this securely with a mobile app and home Wi-Fi connection.

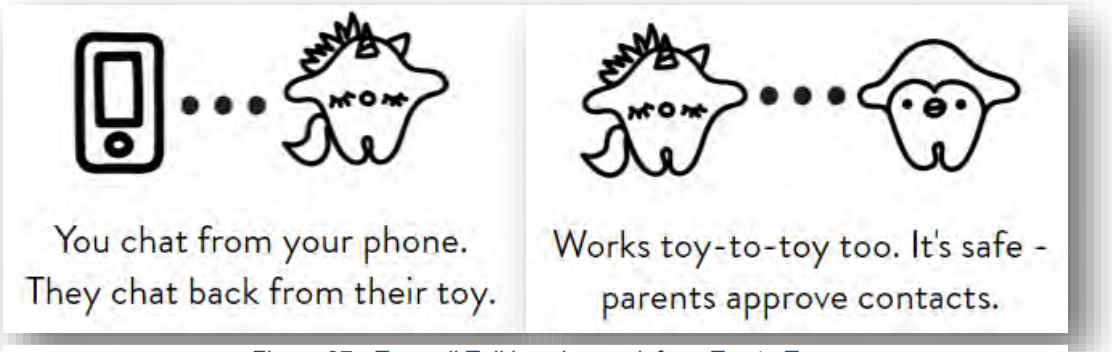


Figure 27 - Toymail Talkies also work from Toy to Toy

2.5.1. Information Gathering

A key consideration for information gathering is how the device makes an initial connection to the phone, in this case, as there is no indication from the packaging of any Bluetooth and there is no screen on which to enter Wi-Fi details.

The FCC filing on 08 Dec 2016 (ID: 2AJENTMAIL01) is registered to Toymail, Inc. New York, United States, but contains surprisingly personal registration by Gauri Nanda, CEO, and a personal email address gauri@toymailco.com.

This time the FCC internal photos are very blurry and dirty, as if the components are purposely obscured. The company have asked for confidentiality of the description, parts and schematics also, so these are unavailable for viewing.

An examination of the device at hand instead shows an optical sensor and LED which flashes red constantly as soon as the device is powered. Disassembly and internal examination of the device did not reveal anything additional about the manufacturer or design at this stage but photos were taken in the case that something might come up later in the investigation.

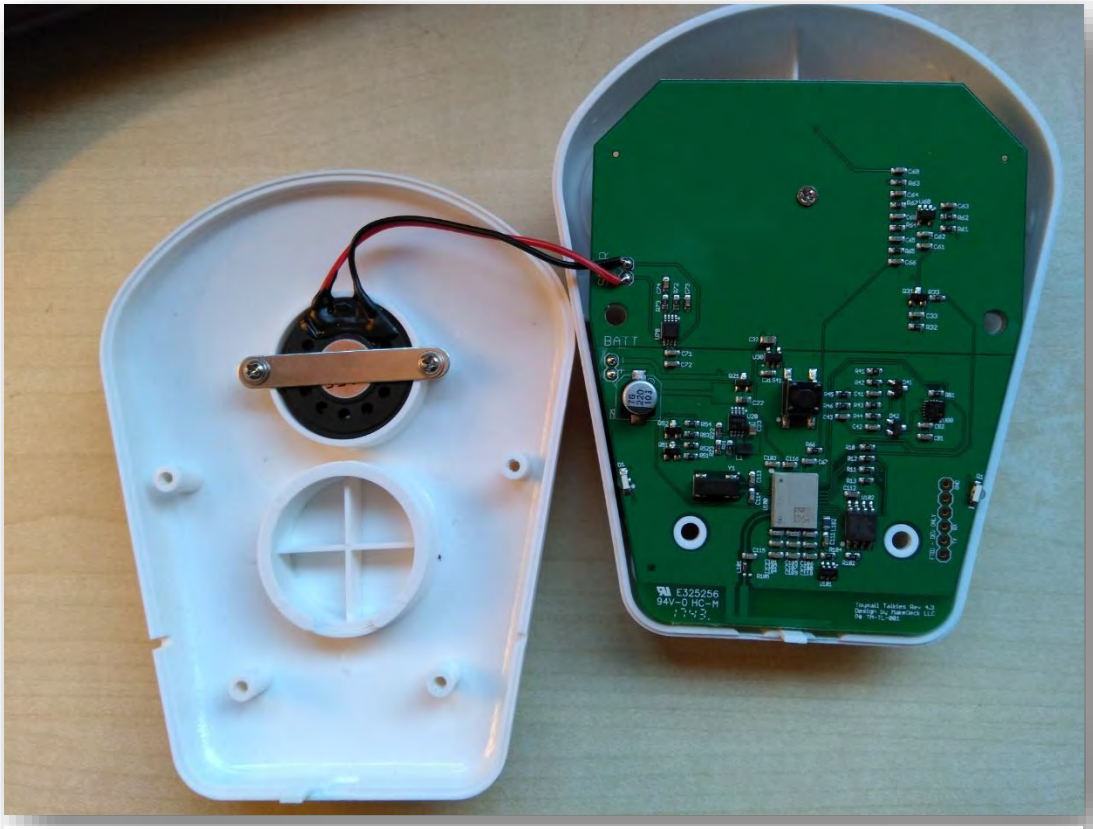


Figure 29 - NuNu internal photo #1

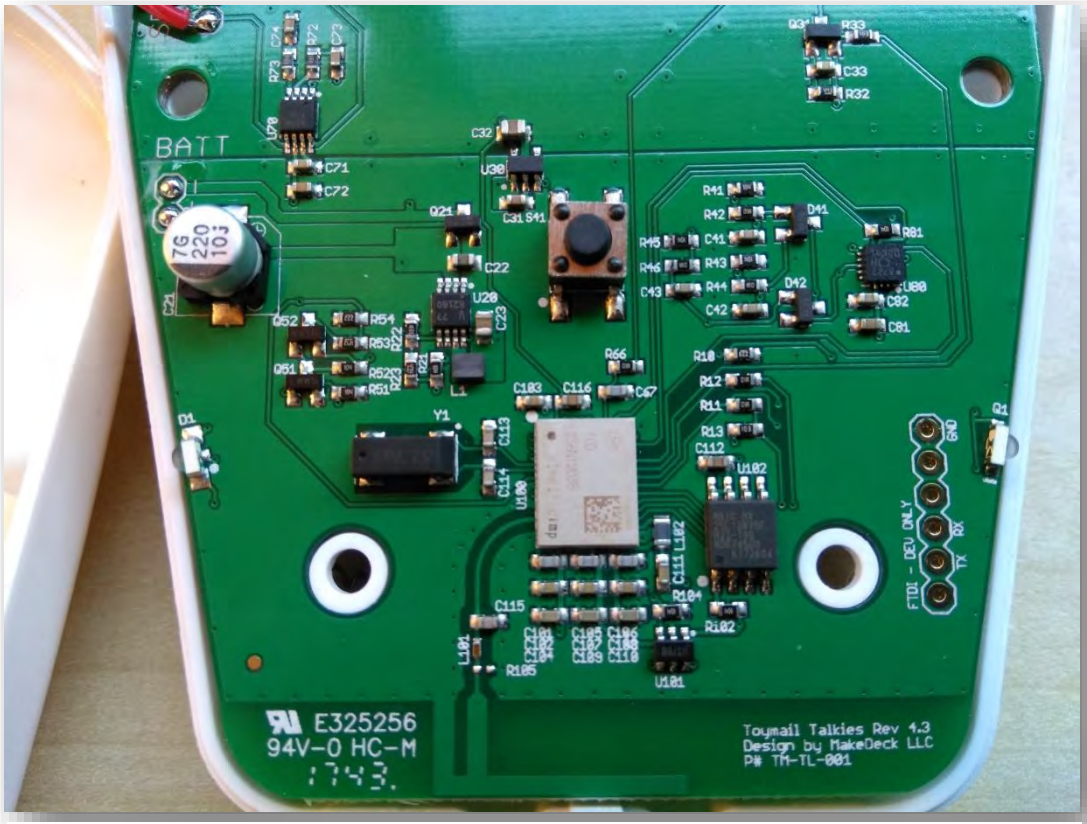


Figure 28 - NuNu internal photo #2

The QR code (via the Quikmark reader) simply contains the ID: 30000c2a690efa44

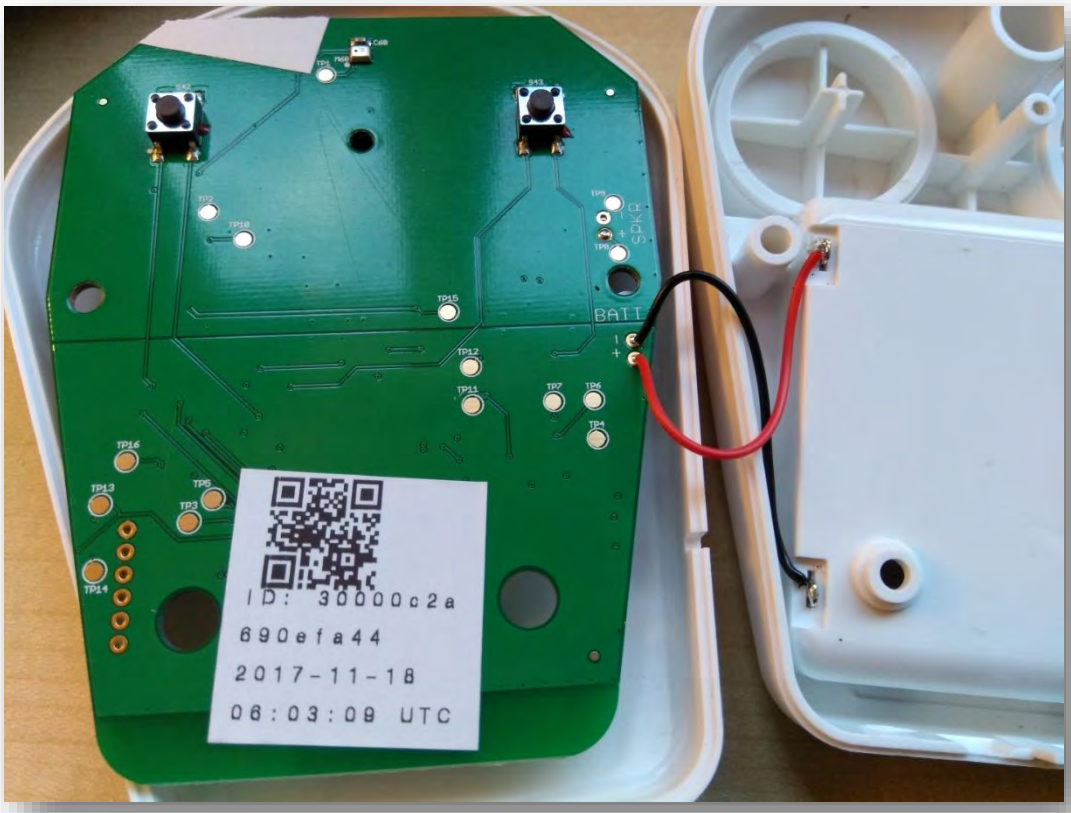


Figure 30 - NuNu internal photo #3

Stamped on the board it says the design is by MakeDeck LLC. The user manual uploaded to the FCC reveals that device initialisation to the Wi-Fi network is accomplished via placement of the optical sensor on top of the phone screen while the app is running.

An examination of the manufacturer's website yielded a substantive privacy policy [45] which, thanks to GDPR, helpfully listed all the third party service providers which may handle user data on behalf of the company. These included a company called ElectricImp, which is listed as used for "device provisioning, pairing and networking services, and shares device and network configuration information". Detailed investigation of their product range uncovers that they supply a Wi-Fi hardware module which supports on board ram, connection to their secure cloud platform, and most importantly supports *Blink-Up*, their hardware module which "is a patented method used for communicating Wi-Fi and device-registration credentials optically from an iOS or Android device to an imp-enabled product" [46].

2.5.2. Device Advertisement Sniffing/ Connection Sniffing

The NuNu device does not broadcast Bluetooth or BLE, but instead connects to the household Wi-Fi. It gains these details via an optical sensor using the proprietary BlinkUp

method, combining ElectricImp hardware components in the device, and a software Application Programming Interface (API) in the accompanying mobile app.

Despite ensuring activity before testing, Nmap reports that the device is either down or has no open ports.

The PCAP is taken from tpacketcapture after account signup and some message interaction have taken place. Over the course of the whole PCAP, several IPs appear, but each resolve to AWS servers, presumably as part of a load balancing operation. The initial connections are made with TLSv1.2, and then the bulk of the transmissions are done with QUIC.

QUIC (short for Quick UDP Internet Connections) is a protocol invented at Google and then open sourced, aimed at faster connections. New connections can be established and secured with just one single round trip and if subsequent connections are between the same client and server the client can often send application data immediately [47]. Wireshark has built-in decryption functionality for the QUIC protocol, which now only uses TLS 1.3 and is designed to drop connections which attempt to negotiate TLS versions below that.

Despite the robust security features in QUIC using TLS 1.3, we can see from the PCAP that previous connections utilise TLS 1.2, and this leaves the connection open to some attacks. X.509 certificates do not contain information about which cipher suite they are used for, and so if a legitimate certificate is obtained via a TLS 1.2 attack, a Bleichenbacher [48] attack on QUIC need only be performed once for the server to be impersonated for much longer than only one session [49], in what is known as a cross-ciphersuite attack. The attack on QUIC is particularly devasting specifically because of it's 'feature' of using the same signature over many sessions.

2.5.3. APK Reversal

As previously, the package is located and pulled from the phone via ADB (Figure 31).

```
C:\Android\platform-tools_r28.0.0-windows\platform-tools>adb shell pm path com.toymailco.chat
package:/data/app/com.toymailco.chat-Qfz5HqzuuQJiVErtOmgZHw==/base.apk

C:\Android\platform-tools_r28.0.0-windows\platform-tools>
```

Figure 31 - Finding out the path of the NuNu APK

Then the certificate is examined. We can see that the subject Organisation Unit (OU) is Toymail the manufacturer, and the Common Name (CN) is given as Gauri Nanda, who is also referenced as the CEO in the FCC filing mentioned earlier. The certificate signing

date is surprisingly early, April 2014, despite the FCC filing date being December 2016. Although this can not necessarily be trusted, as tools like OpenSSL can be used to set new start and end validity dates, it is hard to see why the developer would have reason to change them in this circumstance.

```
C:\Users\AMcKeown\OneDrive\Docs_Masters\Project\APKs\NuNu\nunu\META-INF>openssl pkcs7 -in
cert.rsa -inform DER -print_certs -text -noout
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 1396639393 (0x533f06a1)
        Signature Algorithm: sha1WithRSAEncryption
        Issuer: OU=Toymail, CN=Gauri Nanda
        Validity
            Not Before: Apr  4 19:23:13 2014 GMT
            Not After : Mar 29 19:23:13 2039 GMT
        Subject: OU=Toymail, CN=Gauri Nanda
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
            RSA Public Key: (1024 bit)
                Modulus (1024 bit):
```

Figure 32 - NuNu certificate dated Apr 4 2014, Toymail, Gauri Nanda

Next, we look at the app manifest to try and gain some insight into the major parts of the application. We can see immediately from the enormous manifest that this app is much, much more complicated than the previous two. As well as the usual read/write permissions, it asks for Camera permissions, Record_Audio, and access to Billing for in-app purchases. It also asks to access the user's Contacts. There are also well over eighty activities declared in the manifest, several background services, and several intent-filters which listen for events occurring, including one for the installation of new components.

Examination of the directory structure reveals the use of Crashlytics [50], a fabric plugin which handles Android crash reporting, Swipelistview, an opensource Android List View implementation with support for drawable cells and other swipe-related features, and okhttp3, which is an HTTP client for Android applications.

In addition, there are several arm libraries that are not examinable with the software available, and the whole application is replete with classes called a, b, c, d, e…which make it extremely difficult to follow the internal logic (Figure 33).

```
r.a(a);
r.c(new w.a().a().b());
g = r.m();
com.toymailco.toymail.api.a.a();
```

Figure 33 - NuNu codebase is complicated

2.5.4. Device Manipulation

No way was found to manipulate the device within the project scope. The device is not open for optical pairing after the first time, unless re-authorised from within the phone app for a different Wi-Fi network, at which time the sensor will reactivate. Multiple messages were sent and received from a variety of phones and tablets and free 'applets' were downloaded to access extra features, but in the end all the extra features are push audio features, some of which are timed, and therefore not introducing any extra security or privacy issues other than uploading bedtime to a remote server.

2.5.5. Further Online Investigation

The website claims that "Talkies are built on top of a platform certified to UL-2900-2-2 standard for Cybersecurity." but this is a standard for network components of healthcare devices, so this appears to be a slightly odd side-claim to make.

They also claim to transmit data securely using "HTTPS, SSL, and 128bit encryption". This claim is also very odd, as we know that SSL is outdated and 128bit is no longer considered to be strong (the NSA announced in 2015 that all classified material should be protected by AES256 or higher), but we also know that they are actually using TLSv1.2 or more and decent encryption, so perhaps the website is just confused marketing?

In what is perhaps an acknowledgement of concerns over previous Toy device breaches, they explain that although customer data is stored in secure Amazon S3 infrastructure this does not guarantee security, but that their engineers have also implemented AWS in a properly secure fashion.

In all, the language around security, while not perfect, is still much better and references more technical details than that of the other two toy manufacturer websites that have been examined.

2.5.6. NuNu Conclusions and Assessment

1 - Is personal info stored on the device? How difficult is it to access? LOW RISK

Personal audio messages are the only thing stored on the device. As there is no way to pair with the device without removing the hardware and running the app, and it does not advertise, personal information seems reasonably safe. However anyone with physical access to the device can press the button on the front to listen to the messages.

2 - Can pairing take place? How secure is it? - LOW RISK

BlinkUp is a slight security risk in that once the app is on the users phone it can be run and their Wi-Fi details leaked, however it would take someone to have physical access to the phone and be skilled to interpret the results, which are likely more accessible from the phone in another manner, so it is a low risk for regular consumers. If the Wi-Fi details are available from the Toy device it seems only likely to be by hardware extraction, but further testing would be necessary to ascertain that.

3 - Is personal information transmitted to/from the device? How secure is transmission? LOW RISK

Although an identified threat exists in the form of the Bleichenbacher attack against QUIC, it is not an easy one to carry out and would require a very competent adversary and significant time with the device.

4 - Is personal data stored in an online location? How secure is storage and transmission? LOW RISK

When signing up for an account the password requirements were much more stringent than ToyFi, for example. In this case a minimum of 8 characters were required, and at least one number and one special character.

It is striking that so much of the NuNu material is so personalised with Gauri Nanda's name, when companies usually spend a lot of time depersonalising and removing liability from products. The app certificate and FCC filing are both in her name, and she appears in the "Dragon's Den" video on the company website. In many ways this increases both consumer and developer trust in the product, as it takes a lot of confidence to so boldly put your name against something with such surety. Further investigation shows she is an alumni of MIT media lab and worked for Apple, and invented another smart product (a runaway alarm clock) before this one, further increasing the likelihood that the company is competent and has an understanding of the technology issues.

With a Device Risk Score of only 8, consumers can be confident that this Toy device is safe to use within the usual bounds of caution about what personal information you put online.

The methodology has yielded few results in this examination, and there is a feeling that it brings the lower Device Risk Score into question slightly. The lack of a reliable way to show positive security, rather than just the absence of security flaws, is something which will require further consideration. It has been remarked that a more robust password policy is a positive step, but everything else appears to be only able to be described in terms of ‘the lack of’ pairing/advertising/open S3 buckets.

3. Discussion

The three Toy devices examined were each very different and deceptively cuddly, and yet had vastly different risks associated with them that were not obvious from either looking, or from the online information available about them.

Even just comparing FCC filings, the treasure trove of information suggested by Stone, it was almost impossible to assess which toys would be problematic.

They were also each very different with advertising, with NuNu doing none, Freddy doing Bluetooth only, and ToyFi giving away far too much information.

Each of the Toy devices was more vulnerable to attacks based on close physical proximity, but the Bluetooth based devices especially so because of the possibility of remote connections such as connecting and pairing.

It was clear from the literature review, and mentioned at the outset, that many vulnerabilities with IoT devices occur due to the small size of the devices and manufacturer laziness at implementing features, and our examinations have proved that out. We have seen in both ToyFi BLE and Freddy Bluetooth that key security features already built into the basic protocols were simply not flagged as on.

Ignorance of the most basic security practice was also evident, such as merely requiring a single character password for ToyFi, and transmitting data in plaintext, and yet this was in stark contrast to NuNu's strong password security. There has been some suggestion that Chinese manufacturers are subject to fewer privacy controls in China and so are more lax than Western manufacturers, and this would anecdotally seem to bear out, although an argument could also be made for new products having learned from the scandals of the past.

It was initially thought that assessing Risk against a common framework might be difficult for such a disparate range of devices with different features, but that it was nevertheless worthwhile and necessary from a consumer standpoint. Having settled on points which fit roughly against stages within the chosen testing methodology, however, it was not as difficult as first imagined. The main foreseeable drawback is those devices which do not offer any kind of online personal data storage, which eliminates an entire assessment category, but it is felt that it is such an important category when it is present (as can be evidenced by the breaches discussed at the outset), that it is imperative it be included.

Concerns are outstanding about some areas that remained lacking during the overall investigatory process. The lack of ability to investigate ARM libraries, for example, due to the cost of the tools. And simply the fact that a wider reaching methodology requires a much larger skillset, such as a deep understanding of Bluetooth audio to make a proper and true assessment of Risk. It leads inevitably to the thought that perhaps that is the true reason why the hobby blogs and even professional sites have a piecemeal approach; that rather than lacking a full methodology to ensure every aspect is tested, they are lacking a full skillset and a full toolset.

4. Conclusions

In summary, the proposed methodology retains the best features of some of the most useful IoT investigations of recent years and builds them together into a useful whole, which can be further added to. It has been tested across a small number of devices and performed well in coming up with a risk score for each. While it is not necessary to follow it in its entirety for every device, it should act as a reminder of what is possible, and, coupled with the Device Risk Score act as a guide for comparing devices when purchasing decisions are being made.

In conclusion, however, it must also be admitted that there may be a flaw in the aim of the research itself; in that perhaps the case for a full methodology, while still useful, has other prohibiting factors to its use.

The main advice to others is; Manufacturers must focus on building in the already pre-existing security mechanisms. Consumers should remember not to give away personal details where possible.

4.1. Analysis of the methodology

The chosen methodology was an amalgamation of the Margaritelli and Stone analysis methods, with some additions to attempt to add rigour during the various stages.

Information Gathering

Stone's suggestion to use FCC filings as a resource has been a trove of useful OSINT and has added valuable extra information to each investigation, either by corroboration, the ability to look at device internals, or by looking at previous devices by the same manufacturer. When manufacturers have something to hide this is an excellent source of

information. The addition of use of the Wayback Machine is a great compliment to this process, uncovering older or sometimes purposefully hidden information.

Device Advertisement Sniffing/ Connection Sniffing

Problems with the nRF51 sniffer were not prohibitive in conducting an analysis, and in fact using Android sniffing resulted in much cleaner Wi-Fi pcaps, and a separate Bluetooth log from the Android device. Stone and Margaritelli's emphasis on establishing key items of interest during sniffing strengthens and gives direction to the later reversal of the app code.

The tpacketcapture app being able to capture wifi directly from the phone was an excellent addition, and as mentioned previously the various different properties of the BLE sniffer apps compliment each other so that each can be used for a different purpose.

Although the Toy devices tested did not give a good opportunity to bear this out fully as they were restricted in operation due to missing online infrastructure, they remain a sound addition to the process which ensures all aspects of the connection process are fully and methodically examined.

In retrospect a useful toolkit item may have been a Bluetooth sniffer, which was not initially anticipated as it was expected all the devices would be using BLE and nothing as old as Bluetooth 3.0. While this was not an issue for this time because our device was communicating directly with our app, if we had wanted to sniff for any other traffic it would have been useful.

APK Reversal

While an upload to VirusTotal was not part of the methodology of either Stone or Margaritelli it was deemed prudent as part of protecting the system, in light of how many infected apps are rumoured to be part of the app stores now. It had the added benefit of providing some further guidance on what to look for during review of the code, and so is considered to be a useful addition to the methodology.

Margaritelli's guide was very in-depth about APK reversal so there is little necessary to add here, except to reiterate the problem about toolsets and skillsets being very broad, as understanding ARM libraries and having the tools to decompile them is yet another element of this.

Device Manipulation

The device manipulation is heavily predicated on the results of all prior stages, and is also the biggest evidence of the presence of Risk. However it can also rely on having the skill to accomplish the task, and the tools available, and the time to spend. It is important at this stage (at least within the scope of assessing consumer risk) to consider whether it is worth the effort to prove that the device can be manipulated, or if enough evidence has already been gathered, or if the type of manipulation that can be carried out is not relevant to a consumer Threat model. Being able to break AES if you can get an electrical probe onto the circuit board is not relevant for a child's toy that doesn't store information, for example. For this case, it was useful to have the Risk Assessment questions to measure against at the end to keep things within scope.

In this, the chosen methodologies diverged from the aims, as both Stone and Margaritelli seem to take a more hobbyist approach and build interfaces to test if they work.

Further Online Investigation

The methodology struggles somewhat in this section, partly due to the lack of a good source of step-by-step technical instructions, and partly because at this late stage in the process so much will depend on what information has been uncovered earlier that it is difficult to give guidance on what steps may be necessary as the range of options has branched in many directions. Therefore, advice given has focussed around various OSINT and network tools available, but without the more thorough direction which made the earlier sections so useful. In the future this section could be greatly expanded on to go into more detail about the range of online examinations that could take place and perhaps link to examples showing step-by-step guidance. The addition of Spiderfoot [31] and Shodan as OSINT and vulnerability search tools was touched on but not really expanded on in detail.

General Conclusions

Summing up, and coming to an assessment of risk, is an important step and indeed the key point driving this report. Consumers need an accessible way to understand the findings, and the findings are more useful if they can be compared with other findings and other devices to build up a bigger picture.

Stone, and indeed Stanislav at Rapid7, also mention company policies for responsible disclosure, and Stanislav includes alerting a CERT and these should not be overlooked. Too light a touch has been made on those and it should be fleshed out to show a more

step-by-step suggested approach on how to go about these things in order to be comparable to the other sections of the methodology.

4.2. Assessing Privacy Risks

Using information that is gathered during examination, a privacy and security assessment can be made of the Toy device using a risk matrix to come up with an approximate overall privacy risk score. This can make for a clearer comparison of which Toy device excels in certain scenarios, and which may be best overall for privacy. The ratings given during each examination have been taken and tabulated, and a suggested score assigned to each one.

The devices were assessed on the following criteria:

1. Is personal info stored on the device? How difficult is it to access?
2. Can pairing take place? How secure is it?
3. Is personal information transmitted to/from the device? How secure is transmission?
4. Is personal data stored in an online location? How secure is storage and transmission?

	Risky device storage?	Risky pairing?	Risky transmission?	Risky online storage?	Risk Total
ToyFi	High	Medium	Easy	High (untested)	35
Freddy	Low	High	Low	None	14
NuNu	Low	Low	Low	Low	8

Low – 2 points, Medium – 5 points, High – 10 points

In addition, consideration needs to be made of an individual's threat model. For example, a military family will have different vulnerability and tolerance for risk than a rural farming family might.

To this end it is proposed that end users also add the following information to the calculation, where possible:

Information Sensitivity	Threat Agent Capability	User Vulnerability
High (government/military job)	**High** (known stalker, state actors)	**High** (technically inexperienced, not security conscious)
Medium (corporate, teacher, access to data)	**Medium** (corporate rival, angry ex)	**Medium** (security conscious)
Low (rural, community business)	**Low** (no known)	**Low** (technically capable)

High – 10 points, Medium – 5 points, Low – 2 points

Information Sensitivity roughly translates to Impact on the OWASP Risk Rating Methodology [51], in the sense that it accounts for the type of individual that is at risk if a breach of personal data occurs. It is not necessarily that military or school data would be leaked directly from the Toy device by the individual's conversation (although it may), but more that personally identifiable information about them might be leaked which could then be used in an inference attack, or the device could be used to pivot into another point on their network which may contain more valuable information. Although the impact is, therefore, diluted in some sense, it must still be accounted for.

Talking of **Threat Agents** may appear overly dramatic in reference to children's toys, however as previously mentioned, child sexual exploitation continues to rise year-on-year, including grooming activity [52], high-value adults may be targeted through their children, and abuse victims still overwhelmingly have their devices compromised by ex partners. This is an even greater issue for 'blended family' children who may see their biological parent for contact time at weekends or holidays, and may even be given gifts with embedded tracking as a way of tracking down the other parent's address. A report by NPR [53] which surveyed 72 US domestic violence shelters found that 85% of victims were tracked by GPS by their abusers. Even if the biological parent intends no harm, they may not be aware of the threat model of the new family and so could be unwittingly used as a point of compromise without realising.

User Vulnerability is also a consideration, as more security aware users are less likely to give away personal information in audio or text messages, and more likely to activate security or privacy features on their phones and Wi-Fi networks, thus helping to prevent some attacks.

For example:

A family giving a ToyFi teddy device to their child to communicate with a low-ranking military parent on deployment, but who is likely to be relatively security conscious about what information they relay, so their Privacy Risk may look like the following:

Device info broadcast freely via side channels	High	10
Other devices can pair without auth	Medium	5
Personal info hackable via side channels	Easy	10
Risky cloud data storage	High (untested)	10
Information Sensitivity	High (military job)	10
Threat Capability	Low (no known)	2
User Vulnerability	Medium (security conscious)	5
		52

When you consider that there is only a limited amount that users can do to change their Threat Model score (by becoming more security aware and IT literate), it then becomes more obvious that it is important to choose a Toy device which lowers their score (and therefore their overall risk).

While there is no way to give concrete advice and users are urged to make their own assessment of their individual circumstances, it is suggested that under 25 is a good score, between 25-40 caution is urged, and for anything over 40 the device should not be used.

However, any single Toy device category that scored high should be a cause for more investigation on the part of the parent, in order to assess the risks on an individual basis. For example, parents who live in large houses in a rural location may feel unconcerned about toys which openly pair, as they are unlikely to feel anyone would get close enough for that to be a considerable threat. Inner-city parents may be understandably more cautious.

4.3. Contributions

As there are extremely few Bluetooth and BLE PCAPs available for comparison purposes [54], it is hoped it will be useful to upload these to an online repository with accompanying notes, for other people to compare and study.

As there is a lot of fragmentation in the Android ecosystem and many manufacturers and Android versions treat the snoop log slightly differently, it is hoped a blog post about the BT snoop log, enabling it, finding it in the filesystem, and different ways to export and view it, would be a useful contribution.

The methodology and accompanying privacy rating table will also be made available online so it can be used for future examinations.

4.4. Future Work

Due to the restricted scope of the project, and simple time constraints, there is a number of areas where possible future work could be carried out.

As previously discussed, in his CloudPets work Stone took advantage of the Web Bluetooth protocol to make a webpage which would automate the connection to any close CloudPets toy and present a series of buttons which use its basic and open protocols to control it [30]. This could be used by anyone with a BLE compatible phone and a web browser to connect and use a CloudPets toy, and there is a possibility that it could be forked and adapted for use with the ToyFi toys as they are so similar. It may be thwarted by the inability to capture the necessary data from the app, however.

Since the devices are not generally available for sale any longer, this would not be of much consumer interest, however it could make an interesting demonstration about web Bluetooth functionality or a code project.

4.5. Research Challenges

Despite the mandatory registration process with bodies like the FCC, it is clear that highly insecure devices are still being put on the market. A framework for reliably testing that the required security controls have been activated, or regulatory requirements for such, would go a long way towards protecting consumers.

Inspired (somewhat ironically) by the way in which Freddy kept all of the personal data on the phone and never stored any on the device, and how it disappears as soon as the

connection did: if there was somehow a way for us to always keep our personal data on our own machine and only transmit it temporarily to a place that needs it, like the bank, for instance, or an online forum only while we are logged in, but then it disappears again, it seems that would be much safer than any of the current systems.

5. Mitigations

5.1. For the Consumer

Increase security awareness of all users, even children, can go a long way to preventing inference attacks or compromises by preventing the leakage of personal information and reducing opportunity. Turning off Toy devices when not in use, disabling Bluetooth on the phone when not in use, and never leaving phones or Toy devices unattended reduces risk drastically. Reducing the amount of personal information available by being mindful about what you say or enter into devices is an effective measure for combatting many attacks.

GPS tracking and pairing cannot occur if the Toy device is switched off. Personal information cannot be leaked if you didn't put it in to begin with.

Increase technical capability where possible, though this may be admittedly more difficult with very elderly or very young users, but even simple things like increasing password strength and using lock screens mitigates many attacks. Ensure phones, devices and computers are always updated. Don't connect any devices to Wi-Fi which is not your own or a trusted friend.

If a data breach occurs, a strong password may still prevent your data being stolen. If you leave your phone unattended a lock screen may prevent an attacker compromising it. Manufacturers release the latest security patches in updates, and vulnerabilities are still being found regularly. For instance, the 'Blueborne' exploit made Windows Bluetooth vulnerable to MITM spoofing attacks [55].

Research connected Toy devices before purchase. Security researchers often take only months to examine and blog about new Toy devices if they are problematic, so wait and do some thorough online searching before you rush out to buy something brand new. Check on the manufacturer's website whether online storage will be used, how it will be protected, and whether they will adhere to GDPR standards if there is a breach. In addition, it can be seen that some manufacturers produced insecure devices previously, such as Genesis with the Cayla doll, and thus online research may help to identify a potentially problematic manufacturer.

Although these recommendations are clearly borne out of the findings of this report, most of them are also strongly in line with the FBI Alert on Internet-Connected Toys from July 2017 [56].

5.2. For Manufacturers

Bluetooth and Bluetooth Low Energy has good security already built in. Using it effectively is much better than ending up in the middle of a scandal.

If at all possible, give users the option of an offline mode for those that need to reduce their risk. If using online storage for customer data, using established third-party providers for cloud services instead of rolling your own and introducing possible mistakes is a great plan, but be careful the implementation is subsequently tested for known exploits. Have several layers of storage security.

Always ask for the minimum possible personally identifiable information from customers, take password hashes rather than the plaintext, and encrypt all data in transport and in storage so that in the event of a breach exposure is minimised.

Don’t underestimate how many people are looking for apps and devices to compromise out of curiosity or to make a name for themselves, let alone to cause actual harm to consumers. Particularly in the area of children’s toys, where scandal is a PR disaster, having proper Pentesting done would be considered Best Practice.

Do some threat modelling with industry tools such as STRIDE [57] or Security Cards [58] a diverse team so that a good variety of abuse cases are caught. Assume that your app will be reversed, and your data will be breached, and have a mitigation plan in place.

References

[1] “German parents told to destroy Cayla dolls over hacking fears,” BBC News, 17 Feb 2017. [Online]. Available: https://www.bbc.co.uk/news/world-europe-39002142.

[2] I. Thomson, “Goodbye, Hello Barbie: Wireless toy dogged by POODLE SSL hole,” The Register, 04 Dec 2015. [Online]. Available: https://www.theregister.co.uk/2015/12/04/wireless_barbie_slipshod_security/.

[3] T. D. K. K. Bodo Möller, “This POODLE Bites: Exploiting The SSL 3.0 Fallback, Security Advisory,” Google, Sept 2014. [Online]. Available: https://www.openssl.org/~bodo/ssl-poodle.pdf.

[4] A. Willis, “Creepy Barbie doll records children’s voices and uploads them to the internet,” Metro, 13 Mar 2015. [Online]. Available: https://metro.co.uk/2015/03/13/creepy-barbie-doll-records-childrens-voices-and-uploads-them-to-the-internet-5102331/.

[5] M. Stanislav, “R7-2015-27 and R7-2015-24: Fisher-Price Smart Toy® hereO GPS Platform Vulnerabilities (FIXED),” Rapid7 Blog, 02 Feb 2016. [Online]. Available: https://blog.rapid7.com/2016/02/02/security-vulnerabilities-within-fisher-price-smart-toy-hereo-gps-platform/.

[6] T. Hunt, “Data from connected CloudPets teddy bears leaked and ransomed, exposing kids' voice messages,” TroyHunt.com, 28 Feb 2017. [Online]. Available: https://www.troyhunt.com/data-from-connected-cloudpets-teddy-bears-leaked-and-ransomed-exposing-kids-voice-messages/.

[7] Paul Stone, “Hacking Unicorns with Web Bluetooth,” Context, 28 Feb 2017. [Online]. Available: https://www.contextis.com/blog/hacking-unicorns-web-bluetooth.

[8] European Parliament and the Council of the European Union, “Regulation (EU) 2016/679 of the European parliment and of the council (General Data Protection Regulation),” *Official Journal of the European Union,* pp. L119/1-L119/88, 2016.

[9] “Children's Online Privacy Protection Rule ("COPPA") 1998 78 FR 4008,” Federal Trade Commission, 17 Jan 2013. [Online]. Available: https://www.ftc.gov/enforcement/rules/rulemaking-regulatory-reform-proceedings/childrens-online-privacy-protection-rule.

[10] F. M. o. J. &. C. Protection, “90 Abuse of broadcast or other telecommunications equipment,” Telecommunications Act (TKG), 1998. [Online]. Available: https://www.gesetze-im-internet.de/tkg_2004/__90.html.

[11] I. Torre, G. Adorni, F. Koceva and O. Sanchez, “Preventing Disclosure of Personal Data in IoT Networks,” in *12th International Conference on Signal-Image Technology & Internet-Based Systems*, Naples, Italy, 2016.

[12] “http://www.bbc.co.uk/news/technology-42853072,” BBC, 29 Jan 2018. [Online].

[13] A. A. Pammu, K.-S. Chong, W.-G. Ho and B.-H. Gwee, “Interceptive Side Channel Attack on AES-128 Wireless Communications for IoT Applications,” in *IEEE Asia Pacific Conference on Circuits and Systems (APCCAS)*, Jeju, South Korea, 2016.

[14] J. S. Atkinson, M. Rio, J. E. Mitchell and G. Matich, “Your WiFi Is Leaking: Ignoring Encryption, Using Histograms to Remotely Detect Skype Traffic,” in *IEEE Military Communications Conference*, Baltimore, MD, USA, 2014.

[15] J. Liu and W. Sun, “Smart Attacks against Intelligent Wearables in People-Centric Internet of Things,” *IEEE Communications Magazine,* vol. 54, no. 12, pp. 44-49, 2016.

[16] N. Zhang, K. Yuan, M. Naveed, X. Zhou and X. Wang, “Leave Me Alone: App-level Protection Against Runtime Information Gathering on Android,” in *IEEE Symposium on Security and Privacy*, San Jose, CA, USA, 2015.

[17] “nRF Sniffer,” Nordic Semiconductor, [Online]. Available: https://www.nordicsemi.com/eng/Products/Bluetooth-low-energy/nRF-Sniffer.

[18] “Project Ubertooth,” 2018. [Online]. Available: http://ubertooth.sourceforge.net/.

[19] S. Margaritelli, “NIKE+ FUELBAND SE BLE PROTOCOL REVERSED,” 29 Jan 2015. [Online]. Available: https://www.evilsocket.net/2015/01/29/nike-fuelband-se-ble-protocol-reversed/.

[20] S. Margaritelli, “Android Applications Reversing 101,” 27 Apr 2017. [Online]. Available: https://www.evilsocket.net/2017/04/27/Android-Applications-Reversing-101/ .

[21] M. Afaneh, “How to use a Bluetooth (BLE) sniffer without pulling your hair out!,” Novel Bits, 21 Jul 2016. [Online]. Available: http://www.novelbits.io/bluetooth-low-energy-sniffer-tutorial/.

[22] M. Hughes, “Troubleshooting Tools for Your Next Bluetooth LE Project: Ubertooth and the Nordic nRF Sniffer,” All About Circuits, 14 Jul 2017. [Online]. Available:

https://www.allaboutcircuits.com/projects/troubleshooting-tools-bluetooth-LE-project-ubertooth-Nordic-nRF-sniffer/.

[23] “Understanding time stamps in Packet Capture Data (.pcap) files,” Elvidence Computer Forensic Investigations, 25 Sept 2015. [Online]. Available: https://www.elvidence.com.au/understanding-time-stamps-in-packet-capture-data-pcap-files/.

[24] N. Brunner, “nrf51 dongle sniffer not working,” Nordic DevZone Q&A, 04 Aug 2017. [Online]. Available: https://devzone.nordicsemi.com/f/nordic-q-a/24083/nrf51-dongle-sniffer-not-working.

[25] micallef25, “nrf sniffer not picking up any packets,” Nordic DevZone Q&A, 11 Apr 2017. [Online]. Available: https://devzone.nordicsemi.com/f/nordic-q-a/21294/nrf-sniffer-not-picking-up-any-packets/83409#83409.

[26] V. Krishna, “6 Best Wireshark Alternatives for Android,” Techwiser, 27 Dec 2017. [Online]. Available: https://techwiser.com/wireshark-alternatives-for-android/.

[27] “Wireshark - Bluetooth,” Wireshark, 19 Feb 2017. [Online]. Available: https://wiki.wireshark.org/Bluetooth.

[28] C. Maynard, “Display Filters,” 23 Jan 2017. [Online]. Available: https://wiki.wireshark.org/DisplayFilters.

[29] “Android Studio: Android Debug Bridge (adb),” Android Open Source project, 04 Sept 2018. [Online]. Available: https://developer.android.com/studio/command-line/adb.

[30] pdjstone, “CloudPets Web Bluetoooth Demo,” 27 Feb 2017. [Online]. Available: https://github.com/pdjstone/cloudpets-web-bluetooth.

[31] S. Micallef, “Spiderfoot,” 2018. [Online]. Available: https://www.spiderfoot.net/.

[32] M. Woolley, “Bluetooth Technology Protecting Your Privacy,” Bluetooth Blog, 02 Apr 2015. [Online]. Available: https://blog.bluetooth.com/bluetooth-technology-protecting-your-privacy.

[33] “Generic Access Profile,” Bluetooth Specifications, 2018. [Online]. Available: https://www.bluetooth.com/specifications/assigned-numbers/generic-access-profile.

[34] B. SIG, “Advertising and Scan Response Data Format,” in *Bluetooth 4.0 Specification (legacy)*, Bluetooth SIG, 2010, pp. 11.1.8, p.377.

[35] C. C. A. R. D. Kevin Townsend, “Chapter1: Introduction/Operating Range,” in *Getting started with Bluetooth Low Energy*, California, USA, O'Reilly, 2014, p. 8.

[36] A. m. I. D. Platform, “ Introduction to mbed BLE: High data rate, low latency transfers,” ARM Ltd, 2016. [Online]. Available: https://docs.mbed.com/docs/ble-intros/en/latest/Advanced/HighData/#transfer-without-waiting-for-a-response.

[37] A. C. C. K. T. Robert Davidson, “Chapter 4. GATT (Services and Characteristics),” in *Getting Started with Bluetooth Low Energy*, California, USA, O'Reilly, 2014, pp. 54-55.

[38] “ToyFi App,” AppRecs, [Online]. Available: https://apprecs.com/android/com.spiraltoys.toyfi/toy-fi. [Accessed 2018].

[39] W. Machine, “MyFriendFreddyBear.co.uk,” Internet Archive, 06 Mar 2016. [Online]. Available: https://web.archive.org/web/20160306231207/http://myfriendfreddybear.co.uk/.

[40] “My Friend Freddy (UK English) App: Reviews,” Google Play Store, 19 Aug 2015. [Online]. Available: https://play.google.com/store/apps/details?id=com.toyquest.Freddy.EN&showAllReviews=true.

[41] “Vocalizer Text-To-Speech,” Nuance, Aug 2016. [Online]. Available: https://www.nuance.com/content/dam/nuance/en_us/collateral/mobile/data-sheet/170303%20Nuance_Vocalizer_Datasheet_BLU_A4_rgb.pdf.

[42] K. Glick, “Support for 100MB APKs on Google Play,” Android Developers Blog, 28 Sept 2015. [Online]. Available: https://android-developers.googleblog.com/2015/09/support-for-100mb-apks-on-google-play.html?linkId=17389451.

[43] “APK Expansion Files,” Android Developer Documentation, 13 Sept 2018. [Online]. Available: https://developer.android.com/google/play/expansion-files?utm_campaign=100MB-APKs-Play-928&utm_source=dac&utm_medium=blog.

[44] “Cocos2d-x,” CoCos2d, Sept 2018. [Online]. Available: http://www.cocos2d-x.org/products#cocos2d-x.

[45] “Privacy Policy,” Toymail, 23 May 2018. [Online]. Available: https://toymail.co/pages/privacy.

[46] “How To Design And Tune Your BlinkUp Circuit,” electric imp Dev Center, 2018. [Online]. Available: https://developer.electricimp.com/hardware/blinkuptuning.

[47] S. T. R. H. M. Thomson, “Using Transport Layer Security (TLS) to Secure QUIC,” IETF Trust, 03 Oct 2018. [Online]. Available: https://quicwg.org/base-drafts/draft-ietf-quic-tls.html.

[48] J. S. Y. Hanno Böck, “Return Of Bleichenbacher’s Oracle Threat (ROBOT),” in *27th USENIX Security Symposium*, Baltimore, MD, USA, 2018.

[49] J. S. J. S. Tibor Jager, “On the Security of TLS 1.3 and QUIC Against Weaknesses in PKCS#1 v1.5 Encryption,” in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security* , Denver, Colorado, USA, 2015.

[50] “Crashlytics,” fabric, [Online]. Available: https://fabric.io/kits/android/crashlytics/summary.

[51] “OWASP Risk Rating Methodology,” OWASP, 07 Aug 2018. [Online]. Available: https://www.owasp.org/index.php/OWASP_Risk_Rating_Methodology.

[52] C. Copyright, “National Strategic Assessment of Serious and Organised Crime 2018,” National Crime Agency, 2018.

[53] A. Shahani, “Smartphones Are Used To Stalk, Control Domestic Abuse Victims,” NPR, 15 Sept 2014. [Online]. Available: http://www.npr.org/blogs/alltechconsidered/2014/09/15/346149979/smartphones-are-used-to-stalk-control-domestic-abuse-victims.

[54] “Sample Captures,” Wireshark, [Online]. Available: https://wiki.wireshark.org/SampleCaptures. [Accessed Oct 2018].

[55] G. V. Ben Seri, “BlueBorne,” Armis, 2017.

[56] P. S. Announcement, “Consumer Notice: Internet-Connected Toys Could Present Privacy and Contact Concerns for Children I-071717(Revised)-PSA,” Federal Bureau of Investigation, 17 Jul 2017. [Online]. Available: https://www.ic3.gov/media/2017/170717.aspx.

[57] S. Davis, “Threat Modeling with STRIDE,” WebTrends, 16 Apr 2015. [Online]. Available: https://www.webtrends.com/blog/2015/04/threat-modeling-with-stride/.

[58] B. F. a. T. K. Tamara Denning, “The Security Cards,” University of Washington, 2013. [Online]. Available: http://securitycards.cs.washington.edu/.

[59] “80620 Toy-Fi / Toy-Fi Teddy User Manual D80620003,” FCC ID io, 28 Feb 2014. [Online]. Available: https://fccid.io/2ACBM80620/User-Manual/User-Manual-2276741.

[60] M. Afaneh, “How to use a Bluetooth (BLE) sniffer without pulling your hair out!,” Novel Bits, Mar 2017. [Online]. Available: http://www.novelbits.io/bluetooth-low-energy-sniffer-tutorial/.

[61] M. Hughes, “Troubleshooting Tools for Your Next Bluetooth LE Project: Ubertooth and the Nordic nRF Sniffer,” 14 Jul 2017. [Online]. Available: https://www.allaboutcircuits.com/projects/troubleshooting-tools-bluetooth-LE-project-ubertooth-Nordic-nRF-sniffer/.

[62] “Nordic Devzone,” Nordic Semiconductor, 2018. [Online]. Available: https://devzone.nordicsemi.com/.

[63] M. Helwig, “Hacking Android Apps with Frida I,” 13 Mar 2017. [Online]. Available: https://www.codemetrix.net/hacking-android-apps-with-frida-1/.

[64] B. Harrison, “Location Based Sleep Scheduling for Target Tracking Wireless Sensor Network Applications,” Faculty of Eng. & Phy. Sci. Queens Uni, Belfast, 2010.

Table of Figures