Security Analysis and Evaluation of Smart Toys

UNIVERSITY OF TURKU
Department of Future Technologies
Master of Science in Technology
Thesis
Networked Systems Security
May 2021
Konstantinos Theodoridis

Supervisors:
Dr. Antti Hakkala
Dr. Ali Farooq

UNIVERSITY OF TURKU
Department of Future Technologies
KONSTANTINOS THEODORIDIS: Security Analysis and Evaluation of Smart Toys
Master of Science in Technology Thesis, 72 p.
Information Security and Cryptography
Networked Systems Security
May 2021

During the last years, interconnectivity and merging the physical and digital technological dimensions have become a topic attracting the interest of the modern world. Internet of Things (IoT) is rapidly evolving as it manages to transform physical devices into communicating agents which can consecutively create complete interconnected systems. A sub-category of the IoT technology is smart toys, which are devices with networking capabilities, created for and used in play. Smart toys' targeting group is usually children and they attempt to provide a higher level of entertainment and education by offering an enhanced and more interactive experience.

Due to the nature and technical limitations of IoT devices, security experts have expressed concerns over the effectiveness and security level of smart devices. The importance of securing IoT devices has an increased weight when it pertains to smart toys, since sensitive information of children and teenagers can potentially be compromised. Furthermore, various security analyses on smart toys have discovered a worryingly high number of important security flaws.

The master thesis focuses on the topic of smart toys' security by first presenting and analyzing the necessary literature background. Furthermore, it presents a case study where a smart toy is selected and analyzed statically and dynamically utilizing a Raspberry Pi. The aim of this thesis is to examine and apply methods of analysis used in the relevant literature, in order to identify security flaws in the examined smart toy. The smart toy is a fitness band whose target consumers involve children and teenagers. The fitness band is communicating through Bluetooth with a mobile device and is accompanied by a mobile application. The mobile application has been installed and tested on an Android device.

Finally, the analyses as well as their emerged results are presented and described in detail. Several security risks have been identified indicating that developers must increase their efforts in ensuring the optimal level of security in smart toys. Furthermore, several solutions that could minimize security risks and are related to our findings are suggested, along with potentially interesting topics for future work and further research.

Keywords: Internet of Things, smart toys, networking capabilities, static and dynamic analysis

Table of Contents

1 Introduction

One of the main factors that contribute to the development of a child is the act of playing. Play is critical to the social, physical and emotional stability and well-being of children as it helps them unfold their creativity and cooperate with one another. Therefore, play is a mechanism for children to have fun and develop several skills necessary for them later in life [1].

Toys have been one of the main tools which initiate play among children and have been in existence from ancient civilization to the modern era. Starting from being merely sticks and rocks, toys have evolved into dolls and finally into modern toys that utilize current technological trends. One of those being the Internet of things, IoT technologies have seen rapid advancement and usage during the last years. One of the sub-fields of IoT are smart toys which I am defining as essentially toys with networking capabilities, that can provide a far more enhanced and immersive experience to children.

As with every rapidly advancing technology, security experts have expressed certain concerns related to the level of security in IoT devices. Enhancing security in IoT is a challenging task because of certain characteristics of IoT such as their physical limitations in available resources. However, it is highly important to ensure an adequate level of security in smart devices, a need of even greater importance when considering smart toys which can potentially mishandle children's sensitive data. Smart toys have specific capabilities such as network features in order to communicate with for example the mobile application, therefore increasing its attack surface [2].

However, despite the importance of enforcing smart toys' security, developers have to keep taking steps in the right direction and prioritize this even more. There are several cases where large smart toy manufacturers have made serious mistakes which led to data breaches. Some examples are

VTech's database breach where multiple accounts have been compromised, as well as the Hello Kitty breach [3]. Researchers have analyzed relevant smart toys and their findings are a raising concern amongst security experts; several critical flaws and vulnerabilities potentially indicate that developers do not pay the necessary attention to the device's security level [4].

This master's thesis aims to investigate and identify smart toys' security flaws, using as a basis already conducted research papers and specifically verify the methods used by Gordon Chu *et al* [4]. This is achieved by performing a static and dynamic analysis on a randomly selected smart toy and analyzing security issues that have been discovered during both parts of the analysis. In addition, all the methods used are described in detail from the starting point of setting up the necessary components and software used, to the methods used for identifying flaws of the mobile application accompanying the smart toy. The smart toy is a fitness band whose target consumers involve children and teenagers, and is accompanied by a relevant mobile application.

The outline of the thesis structure is described below:

In the first chapter the reader is introduced to the idea behind the writing of the thesis as well as the general outline of its structure. In the second chapter the main focus is defining and describing the fundamental concepts upon which the thesis is based upon. The reader is introduced to the topic of IoT, the general categories which consist the aforementioned technology as well as shortly defining the main areas it is applied on. The second main point is cybersecurity in general and the connection between security and IoT technology.

The third chapter introduces the reader to an even more specific topic of IoT, which is defining and analyzing security in relation with smart toys. In this chapter the reader gets accustomed with several methods of attacks performed on smart toys as well as the major discovered security flaws and risks.

The fourth chapter describes the case study of an analysis, both static and dynamic, performed in a randomly selected smart device. Firstly, the idea behind the chosen methods is described as well as the necessary steps in order to set up the environment necessary for performing the analysis. Moreover, the chapter consists of two main sections. The first section focuses on the statistic analysis, where all the findings are described in detail. The second section of this chapter focuses on the dynamic analysis as well as the steps taken in order to discover relevant security flaws.

The fifth chapter discusses the findings of the analysis and attempts to evaluate as well as to suggest alternative approaches and solutions to the discovered security flaws. Lastly, the sixth and final chapter concludes the master thesis as well as suggests additional areas for improvement and further research.

2 IoT overview

The Internet of Things or Internet of Everything is an idea that began taking shape, way before the actual use of the term [5]. Initially the idea of connecting several devices together with the goal of providing a way of interconnected communication was implemented by students at the Carnegie Melon University where a vending machine was Internet-connected with the help of photosensors [5]. As such anyone who had access to the Internet could determine the number of remaining drinks as well as their state (e.g., if they are cold or not). Almost a decade later the same principle of connecting devices through the Internet was applied to an internet-connected toaster; using TCP/IP and with the help of controller in order to turn it off or on [5].

2.1 Defining the IoT

In the modern era Internet of Things is identified as one of the most important and prominent areas in technology where an increasing number of industries has taken an interest in. Enterprises can take a significant advantage when utilizing the IoT technology as it can heavily impact customer support, inventory management as well as business analytics [6]. Supply demands can change rapidly and IoT can greatly help by increasing the speed of the response to those demands. Consecutively, the production as well as the function of the supply networks is optimized, as components of those areas, among others, are connected and communicating with one another.

As competition among technological firms intensifies, the need to adopt new technologies and utilize the advantages of it become more intense. This pressure reflects to IoT as this technology has seen a rapid increase in its popularity among firms and is expected to continue doing so. The estimations and the numbers given agree with the abovementioned claim; one example is the prognostic by Gartner made in 2014 where it is expected that the number of IoT devices will increase to 26 billion devices by 2020 whereas in 2009 the number was merely 0.9 billion.

Five of the more widely used IoT technologies are Radio frequency identification (RFID), Wireless sensor networks (WSN), Middleware, Cloud Computing and IoT application software.

2.1.1 Radio frequency identification (RFID)

Radio Frequency Identification is one the main technological achievements related to embedded computing that utilizes microchips in order to exchange data wirelessly [7]. Three basic components are used in order to achieve data transmission; radio waves, a tag that can store a bigger amount of data compared to barcodes as well as a reader [6]. When the RFID receiver triggers an electromagnetic request, the tag transmits data in order to identify itself. There are three main tag categories; Passive RFID tags where the tag does not have its own power supply but instead draws power from the tag reader. This category of tags is prominent in supply chain management as well as access control applications. Currently many road-toll tags as well as bank cards utilize passive tags. The second category is Active RFID tags where the RFID tag has its own battery supply. This feature allows for greater distance between the tag and the reader in order to communicate data. The final category is the Semi-passive tags where the tag utilizes batteries for powering, similar to active ones, while drawing power from the reader when communicating. Generally active and semi-passive tags have a greater cost than passive ones [6].

2.1.2 Wireless sensor networks (WSN)

Wireless sensor networks refer to spatially distributed sensors that can be attached to devices in order to measure conditions like temperature, sound levels, wind levels etc. The network consists of several nodes that form a network transmitting data throughout them and finally through the gateway sensor node which is the endpoint between network and the end user. The user then can measure and monitor the available data as well as configure the network. Typical examples of WSN usage are cold chain logistics, transportation as well as maintenance and tracking systems.

2.1.3 Middleware

Middleware is a software component that acts as the glue between software application layers making communication and interchanging data between them easier. It acts as a mask hiding the way inputs are handled (those technologies and methods of handling data that would be irrelevant to the IoT developers). In IoT, where a plethora of different devices with unique characteristics are involved, middleware helps simplify the process of integrating and developing new services and applications [6].

2.1.4 Cloud Computing

Cloud Computing enables easy and suitable on-demand access to computing resources such as networks, servers and applications while having a fast reaction to provisioning and releasing requests with minimal management effort. There are five essential elements relevant to cloud computing [8]:

- On-demand self-service: a party can benefit and utilize resources without the need of an external human provider related to those resources.
- Resource pooling: resources are pooled in order to serve multiple different consumers. The purpose of pooling is to mask the resources to the consuming parties (e.g., a consumer is not aware as to where their data is stored in the cloud).
- Broad network access: The resources are distributed over a network and are utilized by a plethora of different devices.
- Rapid elasticity: Provision of resources needs to be adaptable to the needs of the consumers as they can increase, as well as be able to instantly be release when they are no longer needed. Resources need to also appear infinite to the consumers are their needs can increase rapidly and is mandatory that those are met.

- Measured Service: As resources are pooled and shared between several consuming parties, the cloud model has to be able to monitor and track the number of resources utilized by each of those parties [8].

2.1.5 IoT application software

IoT applications provide the means of communication and interchanging data between devices as well as human and devices. Communications between different parties (either referring to device-device interaction or human – device interaction) have to established in a robust, steady and reliable manner. Thus, message have to be transmitted in a timely punctual manner in order for relevant actions to be immediately taken. An example can be the transportation industry where monitoring factors such as temperature, humidity etc. plays an important role on what actions have to be taken in case of a change in their values. Communications between devices usually do not require visualization of data and information. In systems where the human factor is present visualization of data is extremely useful in order to provide to the users a way of presenting data in an understandable way. Finally, IoT applications have to be built in a way that monitoring, interchange of data and resolution of issues that may arise does not need any sort of human intervention.

The Internet of Things technology has a big impact and plays a vital role in several different areas (domains) from healthcare to transportation or even in personal domains such as smart homes and devices. The table below presents the plethora of different domains that utilize the IoT technology as well as the way it is applied.

Table 1. Domains utilizing IoT and relevant applications ([5])

Paper	Domains	Applications
Atzori, Iera and Morabito (2010). "The Internet of Things: A	Transportations and logistics Healthcare	Logistics; assisted driving; mobile ticketing; environmental monitoring; augmented maps

Survey." *Computer Networks* 54 (15): 2787–2805.	Smart environment (home, office, plant) Personal and social Futuristic	Tracking;data collection; identification and authentication; sensing Comfortable homes and offices; industrial plants; Smart museums and gyms Social networks; losses and theft Robot Taxi; City information model; Enhanced game room
Perera et al. (2014)[b] "Context Aware Computing for the Internet of Things: A Survey." IEEE Communications Surveys & Tutorials 16 (1): 414–454.	Industry Society Environment	Suppy chain management; transportation and logistics; aerospace; aviation; automotive Telecommunication; medical technology; healthcare; smart building; home and office; media; entertainment; ticketing Agriculture and animal breeding; recycling; disaster alerting; environmental monitoring
Whitmore, Agarwal, and Xu (2015). "The Internet of Things - A Survey of Topics and Trends." Information Systems Frontiers 17 (2): 261–274.	Smart infrastructure Healthcare Supply chain and logistics Social application	Smart grids; smart homes and building; smart air quality; Intelligent traffic systems; smart parking; waste management Monitoring health; assisted living; medical practices Tracking products (RFID sensors); reducing counterfeiting; product traceability Integration with Facebook, Twitter; location-based interest; contact synchronization
Da Xu, He, and Li (2014)[a] "Internet of Things in Industries: A	-	Healthcare service; food supply chain; safer mining; transportation and logistics; firefighting

Survey." IEEE Transactions on Industrial Informatics 10 (4): 2233–2243.		
Farooq et al. (2015). "A Review on Internet of Things (IoT)." International Journal of Computer Applications 113 (1): 1–7.	-	Smart traffic system; smart environment; smart home; smart hospitals; smart agriculture; smart retailing and supply chain management.
Bandyopadhyay and Sen (2011). "Internet of Things: Applications and Challenges in Technology and Standardization." Wireless Personal Communications 58 (1): 49–69.	Industrial; social IoT; healthcare; infrastructure; security and surveillance	Aerospace and aviation; automotive; Telecommunications; medical and healthcare; independent living; pharmaceutical; retail, logistics and supply chain management; manufacturing; Process; environment monitoring; Transportaion; agriculture and animal breeding; media an entertainment industry; insurance; recycling
Vermesan et al. (2011)^b "Internet of Things Strategic Research Roadmap." Internet of Things-Global Technological and Societal Trends 1: 9–52.	Industrial; social IoT; healthcare; infrastructure; security and surveillance	Aerospace and aviation (systems status monitoring, green operations); automotive (systems status monitoring, V2V and V2I communication); telecommunications; intelligent buildings (automatic energy metering/home automation/ wireless monitoring); medical technology, healthcare, (personal area networks, monitoring of parameters, positioning, real-time location systems); independent living (wellness, mobility,

		monitoring of an ageing population); pharmaceutical; retail, logistics, supply chain management; manufacturing, product lifecycle management; processing industries – oil and gas; safety, security and privacy; environment monitoring; people and goods transportation; food traceability; agriculture and breeding; media, entertainment and ticketing; insurance; recycling
Al-Fuqaha et al. (2015). "Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications." IEEE Communications Surveys & Tutorials 17 (4): 2347–2376.	Industrial; social IoT; healthcare; infrastructure; security and surveillance	Smart home; smart building; intelligent transportation system; industrial automation; smart healthcare; smart grids
Zanella et al. (2014). "Internet of Things for Smart Cities." IEEE Internet of Things Journal 1 (1): 22–32.	Smart city	Structural health of buildings; waste management; improved air quality; noise monitoring; traffic congestion; city energy consumption; smart lighting; automation and salubrity of public buildings
Libelium (2015). "50 Sensor Applications for a Smarter World."	Smart cities	Smart parking; structural health; noise urban maps; smartphones detection; electromagnetic field levels; traffic congestion; smart lighting;

Accessed March 3, 2020. http://www. libelium.com/resources/top_50_iot_sensor_applications_ranking/.		waste management; smart roads
	Smart environment	Forest fire detection; air pollution; snow level monitoring; landslide and avalanche prevention; earthquake early detection;
	Smart water	Portable water monitoring; chemical leakage detection in rivers; swimming pool remote measurement; pollution levels in the sea; water leakages; river floods
	Smart metering	Smart grid; tank level; photovoltaic installations; water flow; silos stock calculations
	Sceurity and emergencies	Perimeter access control; liquid presence; radiations levels; explosives and hazardous gases
	Retail	Supply chain control; NFC payment; intelligent shopping applications; smart product Management
	Logistics	Quality of shipment conditions; item locations; storage incompatibility detections; fleet tracking
	Industrial control	M2M Applications; indoor air quality; temperature monitoring; ozone presence; indoor locations; vehicle auto-diagnosis
	Smart agriculture	Wine quality enhancing; greenhouses; golf

		courses; meteorological station network; compost; hydroponics
	Smart animal farming	Offspring care; animal tracking; toxic gas levels
	Domestic and home automation	Energy and water use; remote control appliances; intrusion detections systems; art and goods Preservation
	eHealth	Fail detections; medical fridges; sportsmen care; patients; surveillance; ultraviolet radiations
Miorandi et al. (2012). "Internet of Things: Vision, Applications and Research Challenges." Ad Hoc Networks 10 (7): 1497–1516.	Smart homes/smart buildings; smart cities; environment monitoring; healthcare; smart business/inventory and product management; security and surveillance	-

[a]Paper focuses on IoT in Industries.

[b]Perera et al. and Bandyopadhyay and Sen draw heavily on the CERP-IoT, hence similarities between the three.

2.2 Cybersecurity

Dan Craigen, Nadia Diakun-Thibault, and Randy Purse define cybersecurity as follows:

"Cybersecurity is the organization and collection of resources, processes, and structures used to protect cyberspace and cyberspace-enabled systems from occurrences that misalign de jure from de facto property rights" [9].

Generally, cybersecurity refers to the protection and prevention of computer systems and networks as well as the individual parts or assets that compose them from damage and disruption. Cyber-attacks can cause disruption or even misdirection of services, forcing them to behave in an unintended way in order to achieve individual goals of the attacking party. Assets refer to a plethora of elements such as human resources (personnel of a company), electronic devices, software and services, and every data and information interchanged inside the relevant environment [10].

Cybersecurity can be broken down into three individual parts whose presence is necessary for its definition. The first part are threats, which refer to any action that is taken in order to take advantage and maliciously exploit a weak point of the attacked system. Threats generally stem from either natural or human sources. In the context of cybersecurity, we are referring to those caused by human intervention [11]. The second part is vulnerabilities of the attacked system which refer to any weakness of the system that can be exploited by a malicious actor in order to disrupt or damage the system. Finally, the third part that constitutes the definition of cybersecurity is the assets that need to be protected.

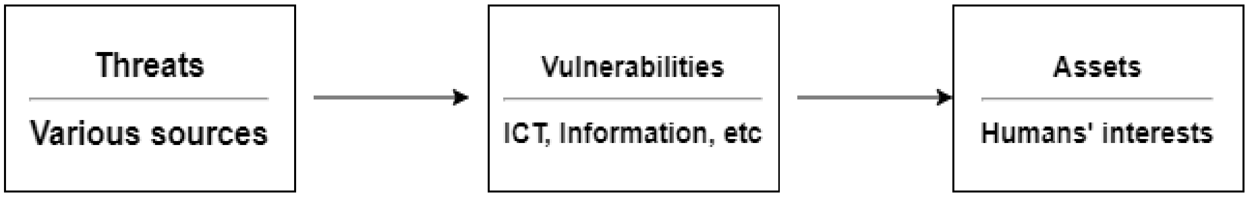


Figure 1: The concept of cybersecurity [10]

2.3 Security in the Internet of Things

As IoT technology is becoming increasingly prominent in the modern world, the need to protect and secure the relevant systems increases as well. IoT comes with certain limitations which can make this challenging to achieve. Some of the security challenges described by Carsten Maple [5] are:

1. **Existing physical limitations of IoT devices**

One of the most critical challenges when discussing IoT technology is the physical limitations of the relevant devices comprising the IoT system. The majority of the IoT devices are designed based on a main axis of relatively low cost and size dimensions. This results in a low amount of resources and computing capability. Furthermore, many devices deployed in the military, industrial and agriculture need to function for long periods of time without recharging them [12]. Three of the most important obstacles in securing the system is the limited battery supply as well as the physical specifications and resources or computing capability of the devices, particularly when it comes to memory capacity.

The problem of the limited battery power can be approached by three possible solutions that each of them presents relevant challenges. Firstly, the most obvious one would be to increase the battery capacity; this approach presents the problem of IoT devices' inherent size constrictions as the devices themselves need to be small in size in relatively lightweight. A second possible solution would be to decrease the security requirements, something that would present several risks especially when dealing with sensitive data and information. Finally, battery could be recharged using natural resources, a solution that could significantly increase the cost [13].

Due to the mentioned limitations IoT devices lack specific features such as memory management unit [MMU] thus are not able to take advantage of memory isolation, address space layout randomization and other features related to memory handling [12]. As the specification of IoT devices are limited e.g., limited memory and CPU, the use of advanced cryptographic algorithms is hard to achieve [14]. This can result to malicious parties being able to take advantage of memory vulnerabilities. Additionally, because of the physical limitations, IoT devices do not use encryption or in some occasions do not check the server's certificate when communicating, thus being susceptible to e.g., man-in-the-middle attacks.

Several solutions have been proposed by researches in order to address the physical limitations of IoT devices. One proposed solution, ARMor [15] is a system that utilizes software fault isolation that can protect code running on small embedded processors, but in some applications, which checked frequently addresses it caused high performance overhead. Another proposed solution for this challenge by Shafagh *et al.* [16] is an encrypted query processing algorithm that utilizes cloud in order to securely store encrypted IoT information, something that improves processing of queries made to the database in order to manipulate encrypted data. Another proposed approach by Kotamsetty and Govindarasu [17] aims on reducing the latency when processing queries over encrypted data, by reducing the amount of resulted data in smaller sets of data. A common approach or cryptography researchers is to design lightweight cryptography algorithms that use less resources than preexisting ones. However, it is very difficult that lightweight algorithms can provide the same level of security with classical ones.

2. Diverseness of devices and ad hoc nature

As the IoT concept is utilized in an increasing rate, the technologies and services involved are becoming more diverse with the common goal of collaborating with one another in order to create a robust system. IoT devices are deployed in a plethora of domains and as such need to perform a variety of tasks based on the needs of the domain's environment. Therefore, the capabilities and physical requirements of the devices vary vastly in each case. For example, the small temperature sensor in a smart home greatly differs from a military or industrial automatic machine. Additionally, the communication protocols are different depending on the scenario based on which IoT devices are deployed. Different authentication and communication protocols can exist even in the same IoT system (Amazon's AWS IoT and Alibaba's Alink) [12]. New protocols can have a variety of plausible security vulnerabilities. As manufacturers usually utilize a variety of third-party SDKs the workflow of the system can be complicated and if not carefully assessed can lead to privacy violations. For example, a research found that attackers could exploit Joylink protocol (shown in Fig.2) due to the lack of authentication [18].

Researchers have made several suggestions in order to tackle security issues related to the diverseness and heterogeneity of IoT devices deployed in a system by analyzing statically or dynamically the device's firmware. However, the proposed solutions have plausible drawbacks. Zaddach et al. [19] presented a framework called Avatar that enables dynamic analysis of embedded devices; the framework however needs to forward actions from emulator to device by physical connection and thus is not suitable for larger firmware analysis [12]. Another evaluation using the presented FIRMADYNE, an automated dynamic analysis system, by Chen *et al*. [20] aimed at larger scale firmware analysis. This led to the discovery of multiple and in some occasions previously undiscovered security vulnerabilities. However, in the case of FIRMADYNE the drawback is that the analysis is feasible only in Linux based systems.

The number of security threats related to IoT will increase as the combination of limited physical capabilities mentioned above and the diversity of the interconnected devices increases [21]. Several domains such as healthcare utilize mobile IoT devices; thus, it is clearly important that strategies of implementing security solutions adapt to the diverse standards, settings and protocols of each interconnected device [5].

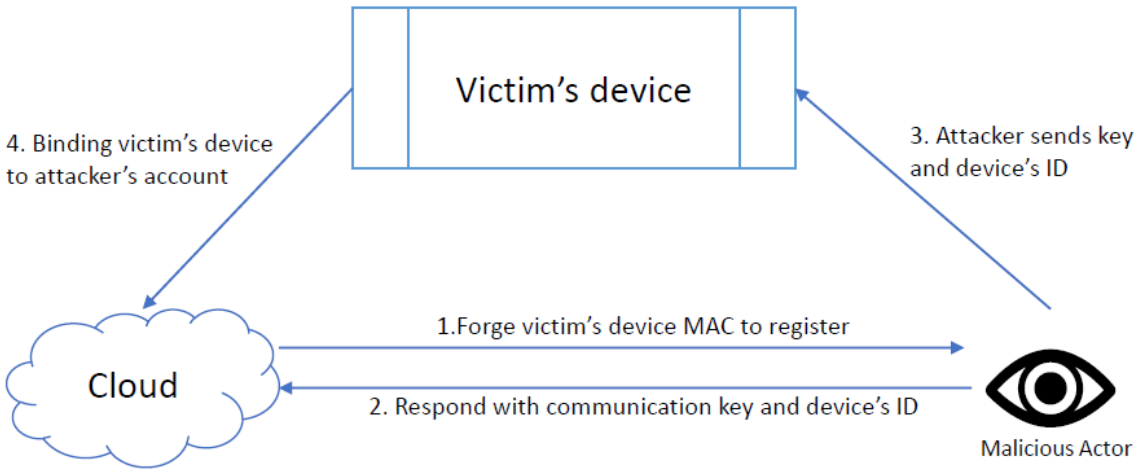

Victim's device
4. Binding victim's device to attacker's account
3. Attacker sends key and device's ID
1.Forge victim's device MAC to register
Cloud
2. Respond with communication key and device's ID
Malicious Actor

Fig. 2. Example of Joylink protocols hiijacking attack [12]

3. Authentication and Identity Management

One of the major system requirements related to IoT systems is authentication and Identity Management. In case of a malicious party pretending to be an authorized user the confidentiality and availability of the system could be at risk, as the attacker can view, delete, modify and generally compromise sensitive data. One of the most common practices of authentication is the use of a password and username. The use of a single password for all devices in the IoT system is less resource intensive but it creates a risk of a more vulnerable system since if the password is compromised the attacker has access to all interconnected devices [22]. Furthermore, additional authentication methods have seen an increase in usage such as biometric authentication, digital certificates and keys.

Identity management includes services and tools used to authenticate an entity that utilizes resources of the IoT system. The authentication of an entity relates more to the assurance of its genuineness rather than its identity [23]. Identity Management systems are relatively complex to implement and they require relatively homogenous systems in order to be applied. Thus, IoT systems and their increased diversity in devices, protocols etc., as well as the number of applicable domains (plethora of enterprises with different naming policies, protocols and security frameworks) present a major challenge in the appliance of IdM [5] [23].

4. Authorization and access control

The plethora and diverseness of the interconnected devices in an IoT system present the risk of a malicious party taking control or "inserting" their own devices in the system in order to utilize/consume resources. Thus, access control is a major consideration when it comes to IoT security as it prevents unauthorized access to the system's resources. Classic access control models include RBAC (role-based access control), ABAC (attribute-based access control) and CapBAC (capability-based access control) [24].

In RBAC the model revolves around roles such as "administrator" and "guest". Based on the rights that roles have, they can perform or have limitations on certain actions such as write, read and execute. ABAC focuses on policies that combine specific characteristics related to the several objects, subjects and environment of the system. The policies then based on those characteristics create a set of rules according to which different rights are given to users of the system [24]. In the CapBAC the entity required to present its authorization capability is the subject (or user) often with the use of a token (key and a ticket); this is the main difference between CapBAC and traditional ACL systems where in the latter ones the main entity or service provider needs to check the authorization levels of the user in order for a requested operation to be performed on the resources of the system [25].

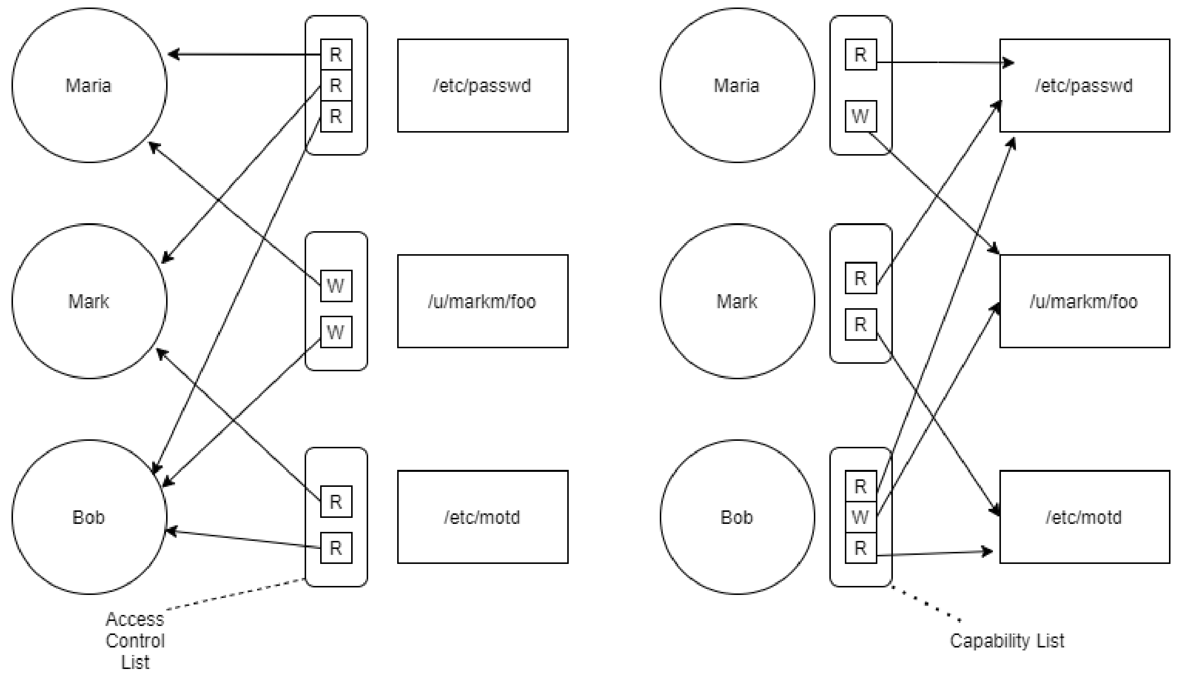


Figure 3. Difference between ACL and CapBAC [25]

5. Interdependence

With the constant increase of the IoT devices' complexity, the need for human interaction and intervention in order for communication to be established has decreased. IoT devices differ from traditional communication between laptops or smartphones. Instead, they can interact with each other based on the behavior of the devices and conditions that have been set by users through the Internet. For example, a thermometer can detect that temperature has fallen below the threshold defined by the set conditions, and thus enable the heating system of the smart house.

The interdependence of IoT devices creates however a wider range of security vulnerabilities by increasing the amount of access point for an attacker. While the device itself can be hard to be manipulated, the malicious party can attack and manipulate the behavior of other environmental factors in order to create a security breach. Therefore, by using the feature of interdependence the attacker can perform an indirect attack to the system. An example can be the scenario of a thermometer which after temperature exceeds a specific value, it checks if the air conditioner is on and if not the windows of the apartment automatically open. As shown in Figure 4 a malicious party can attack the power supply of the air conditioner in order to set it in “off” state, thus raising the temperature and opening the windows of the apartment in order to create a physical security breach [12].

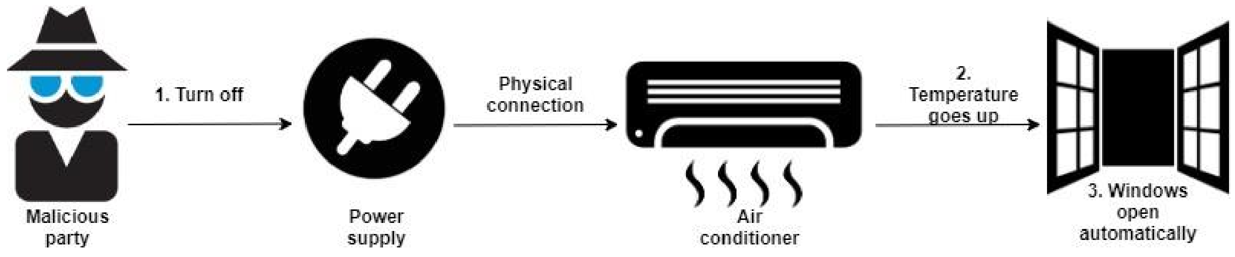


Figure 4. An attacker utilizes interdependence behaviors [12]

The security issues caused by interdependence are overlooked as most of the research done aims to protect the device itself. Furthermore, it is difficult to design access and privilege management for the interdependent devices. Therefore, one of the most common problems related to interdependence is that of “over privilege” where devices gain more permissions that are actually needed [12]. There have been several suggested solutions such as the proposition of Tianlong Yu *et al*. [26] for new policies and rules in order detect abnormal behavior and sketch a roadmap detecting challenges and plausible solutions. However, creating more rules leads to increased complexity especially considering the increasing amount of IoT devices.

Another proposed solution by Jie *et al*. [27] is ContexIoT, which is a context-based permission system for IoT systems aiming to help users perform efficient access control. Furthermore, it supports fine-grained context identification and context information prompting at runtime. Thus, allows the user to decide whether to allow or deny a specific behavior based on recorded information. Two main challenges accompany the abovementioned solution. The first one refers to the availability of context as in the majority of cases, after the initial setup procedure of the app the user is not involved or interacting with the app at runtime (majority of sensitive permission requests occur when user is not interacting with the app). Secondly the number of prompts is another important aspect that should be taken into consideration; the amount of permission requests is high thus leading to inconvenience or annoyance if users are prompted for every request.

6. Security issues in smart homes

IoT technology has impacted private homes and is increasingly being applied in order to create “smart” homes. Smart homes aim to provide enhanced security, power efficiency and comfort through interconnected devices and ease of use. Thus, smart home applications related to IoT technology has attracted interest from different domains such as home security providers and the energy industry [28]. Smart homes allow residents to remotely access smart devices and appliances and use them in order to automate a plethora of actions [29].

The figure below depicts the parts consisting a smart home system:

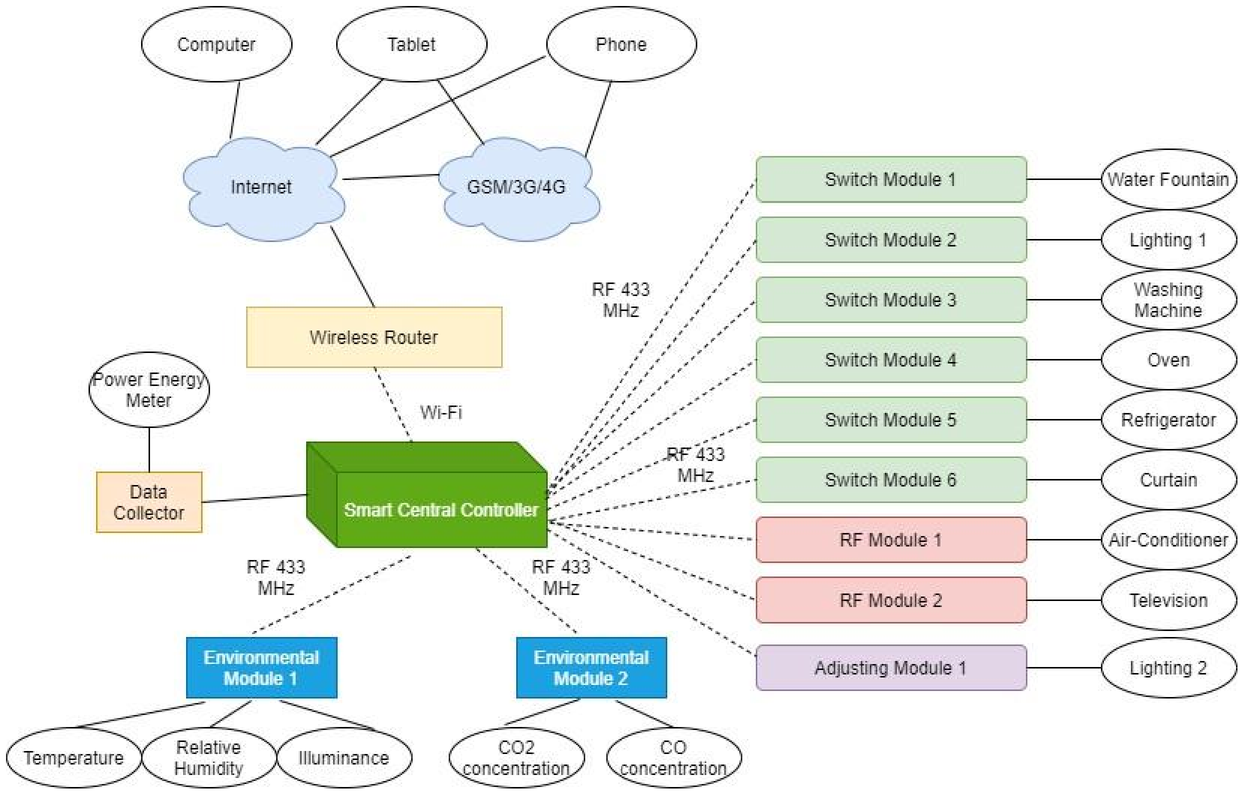


Figure 5: A smart home control system utilizing IoT technology [30]

The system depicted in the Figure 5 can be divided into two layers where the first includes all the user end points such as router, computers, tablets and phones. Those devices are used as controllers in order to remotely perform certain action such as turning off the light, washing machine etc. Before those actions can be completed, they need to pass through the lower layer of the system. The lower layer includes the smart central controller, the switch modules, environmental modules, RF modules, adjusting modules etc. The main part of the lower layer is the smart central controller which connects the two layers in order for the abovementioned user's actions to be performed.

The abovementioned system offers specific advantages. The modules (switch, RF etc.) are manufactured so that their size is convenient and easily mountable in e.g. walls. Furthermore, the utilization of a Wireless sensor network eliminates the need for extra wiring thus making it extremely adjustable as to the positioning of the required devices. Consequently, the smart home system is easy to install and maintain [30].

The increasing number of interconnected devices related to a smart home system subsequently increases the potential weak links or entry points of a performed attack, thus increasing security risks. Some major security risks are compromised confidentiality and privacy of the smart homeowner. A plethora of devices is susceptible to an attack and can potentially become entry points for the attacker and those include cameras, printers, home routers etc. The main danger is the combinations of all compromised devices in order to create a botnet that can have a wider impact [5].

A good example of an attack targeting an internet-connected control system was the attack on Target's payment and security systems. The malicious party deployed malware in order to steal sensitive data including credit card information used at the stores. The attackers gained access to the systems when they took advantage of a weak link between Target and Fazio Mechanical Services which was a contractor running Target's climate systems. Target provided credentials to Fazio but did not ensure that the latter one was not able to access the payment system. The connection was exploited for malware to be uploaded onto Target's systems [31].

3 Security in smart toys

Toys defined as an item used in order to play, have existed throughout ancient civilizations until the modern era. They have been a part of societies in order to educate, socialize and entertain. Toys become especially important in the proper development of a child's character, as it enables them to be creative and develop skills critical for them later in life. Thus, one of the main target groups for toy manufacturers throughout the years have been children.

Toys started from simple wooden sticks and stones and developed to teddy bears and dolls that can be interactive, process language and communicate. In a modern era where the goal of the majority of electronic devices is to offer a broader approach in connectivity (with IoT being a good example) while offering the means to do that conveniently, it comes with no surprise that toys start following the same path. Thus, toy manufactures started producing smart toys, a subset of IoT devices, which usually offer extreme interactivity. More specifically they often come equipped with some sort of sensors and are equipped with electronic parts that allow them to connect in a network. Usually, a mobile device is also required in combination with a mobile application in order to enable interaction with the smart toy. The connection happens in the majority of cases through a Bluetooth connection between the mobile phone and the smart toy. The collected data is then exchanged usually through a Wi-Fi connection and stored in the cloud. Moreover, smart toys can be equipped with a microphone or even a camera in order to further scan and provide additional data processing.

3.1 Identifying security risks related to smart toys

As technology changes the way children interact with smart toys during their playtime it creates certain advantages such as an enhanced method of improving their problem-solving skills through a two-way communication between them and the toy. Moreover, smart toys offer a far more personalized experience compared to conventional toys, while occasionally utilizing technological fields that are nowadays becoming dominant in people's everyday lives such as AI and machine learning.

Combining technology with toys consequentially can raise security concerns as well. Utilizing new ways of communication and interactivity increases the spectrum of vulnerabilities and makes the need for protecting data and ensuring that the privacy of the child remains intact an even more crucial task. One the first steps required in order to correctly identify the security issues that can arise from the use of a smart toy, is to identify its capabilities and available equipped means with which it can achieve those (e.g. equipped microphone, camera etc.). A smart toy that is for example capable of recording sound through a microphone and playing it back, can be susceptible to audio injection attacks, where an attacker can intercept and decrypt the traffic or even inject and make the toy play audio of his choice [32]. Furthermore, we need to also identify the way the different components surrounding the physical device (smart toy), such as the mobile app and the servers in which data are stored, communicate with each other.

In most cases, it is common that smart toys access a Wi-Fi network with the help of a mobile device and the correspondent app. The mobile device connects to the Wi-Fi access point and through the application the necessary network information is transmitted to the IoT device. A possible security issue can arise when in certain cases encryption during transmission is not available. This can lead to a "neighbor" eavesdropping on the traffic and getting access to sensitive data such as the network password. Other devices will attempt to use a different approach and transmit the network password over an encrypted connection with the help of cryptographic protocols such as the TLS. In certain cases, the smart-toy will receive a SSID and a password over a specific port and at this point a security issue can arise where a possible attacker can create an access point with the necessary name in order to trick the device and connect to the unsecure network giving him access to information transmitted from the correspondent application [32].

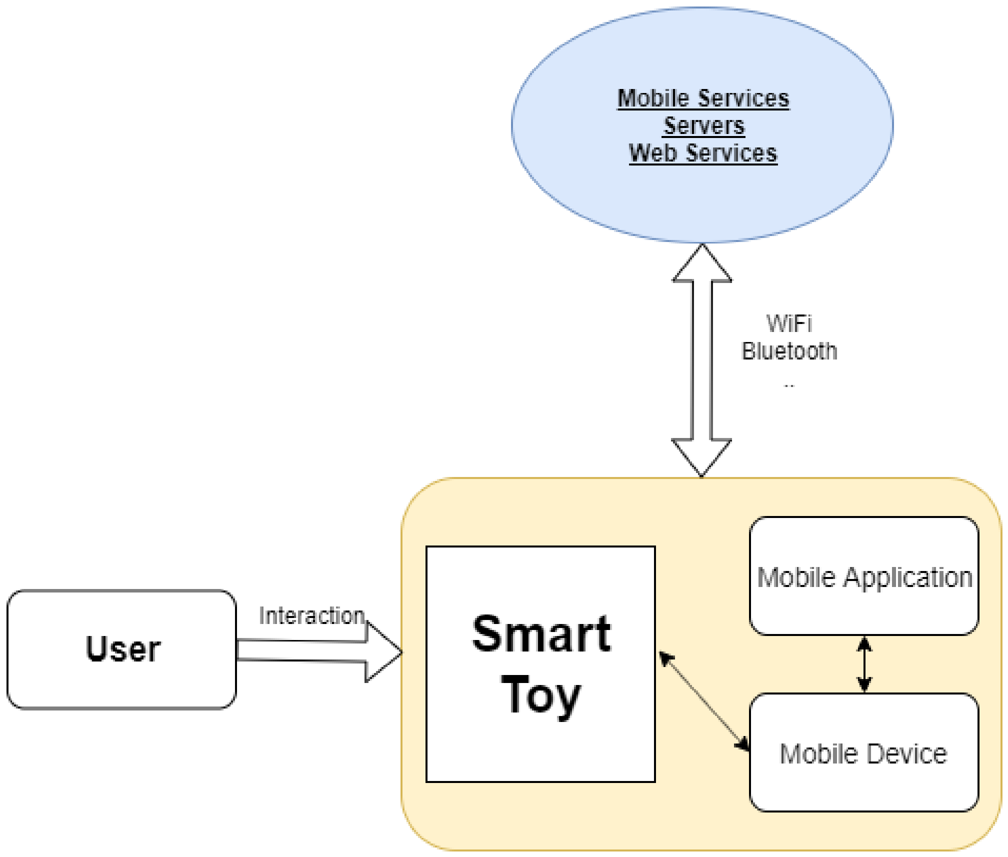


Figure 6. User interacts with the smart toy which communicates with the help of a mobile device. The data is then exchanged through the network with the use of the correspondent mobile application

Data leakage is considered one of the main concerns when examining security vulnerabilities in smart toys. The issue becomes larger when one considers the amount of sensitive data and personal information (such as age, sex and even location) collected and transmitted by the devices [33]. Especially considering the target group of those devices which is mainly children; a highly sensitive target group.

Smart toys can be divided into two main categories. The first one includes devices that children can directly interact with (an example of this category can be the Hello Barbie doll which can respond to the child, thus offering and increased amount of interaction). The second category includes cases when the device is used as a mean of communication (for example a smart walkie-talkie, children use to communicate with their parents). Furthermore, those devices utilize two architectural categories in order to function which include IoT to Cloud and IoT to App through Cloud.

In the first category the data processing is being done in the cloud backend servers, while another requirement is that the smart device is connected to a network with Internet access. In this case the child does not need to have access to a mobile device but rather physically interact with the smart toy (by pressing for example a button and then proceed on speaking in the microphone). The second category involves a mobile app which is connected with the toy through a network with Internet access. In this case children can for example use the smart toy to send a voice message to their parents' mobile device, enabling communication between two different parties. In the first category an attacker can intercept traffic between the smart toy and the cloud (Figure 7.a) while in the second category an attacker can intercept traffic either between the smart toy and the cloud or the cloud and the mobile device/app (Figure 7.b) [32].

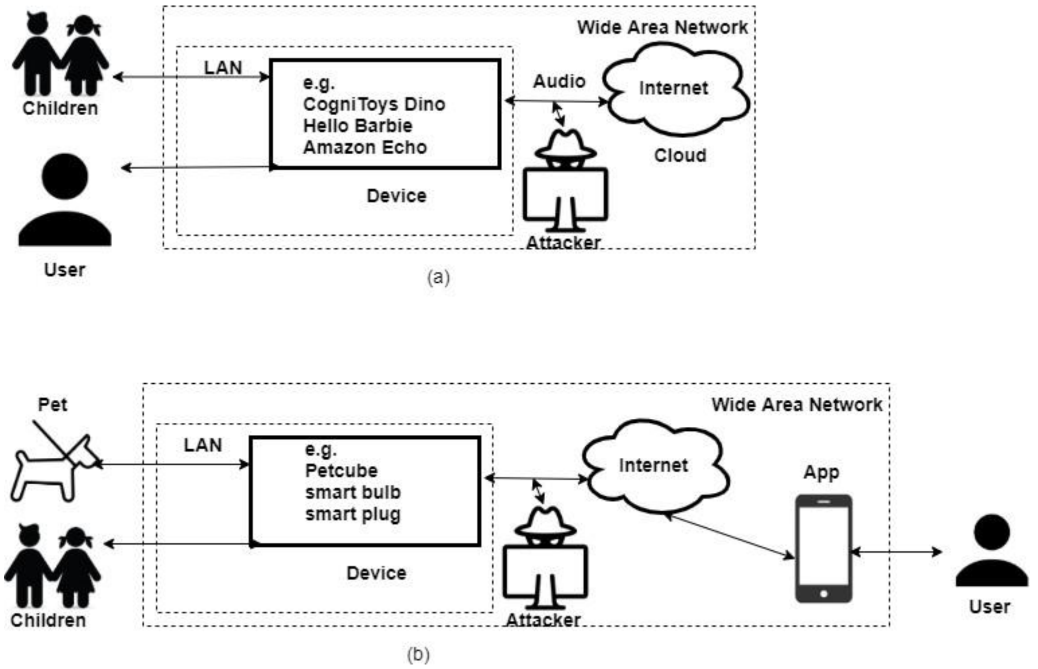


Figure 7. In (a) the malicious party attacks during the data exchange between the device and the cloud, while in (b) the attacker can act between the device and the cloud or the cloud and the mobile device [32]

Generally, we can identify and categorize security risks related to three main categories of the type of access an attacker can have to the smart toy. Physical access, where an attacker needs to have the smart toy or the mobile device with the companion app in his possession, nearby access where the attacker needs to be into close proximity to the smart toy and finally remote access where distance between the attacker and the smart device is irrelevant [34].

In the first category where the attacker has a direct physical access to the smart toy, we can identify a security vulnerability where a malicious party can configure the smart device in order to forward personal data into his account. In the second category where an attacker is in a relatively close distance to the smart toy when can identify several vulnerabilities that can lead to data leak. These include remote unauthorized control of the smart device as well as connecting to the hotspot of the toy in cases where there is no encryption in place. In the latter case a malicious person can even utilize a Man in the middle attack and sniff credentials or personal information that are being communicated in plaintext. Another issue can arise with smart toys that utilize Bluetooth in order to transmit data and particularly with those that make use of static MAC addresses since they are susceptible to persistent tracking as well as Man in the middle attacks.

In the last category of remote access there is a plethora of vulnerabilities that can lead to the attacker utilizing remote attacks in order to obtain private information. Firstly, there is password brute force attacks whereas an app that requires login and has unlimited number of tries can be susceptible to those. Another issue that must be considered is exposure of data to third parties, something that can increase the chance of personal data leakage if their security measures are inefficient or inadequate. Other vulnerabilities include insecure cookies (for example that do not expire) or Insecure TLS practices [34].

Finally, according to the STRIDE model, threats can be classified to specific categories depending on their type. The STRIDE model stands for **Spoofing** identity, **Tampering** with data, **Repudiation**, **Information** disclosure, **Denial** of service, and **Elevation** of privilege and it can provide an easy and comprehensible way to identify and categorize threats [35]. Below the following threats regarding smart toys are modeled using the STRIDE model [2]

Spoofing	• Attacker assumes identity of the child when using the device • Attacker uses another device to control the smart toy • Mobile service provider is fake
Tampering	• Altering the configuration file in the mobile device • Altering data in the database • Altering the information transmitted through the device and the smart toy
Repudiation	• User is not notified when an attacker uses purchase services
Information disclosure	• Leakage of data stored in the database • Leakage of data required to use mobile services e.g. Location • Leakage of data stored in the mobile device e.g. photos
DoS (Denial of Service)	• Attacker sends data to the database in order for it to reach full capacity • Smart-toy receives commands from more than one device simultaneously thus not being able to identify the real one and provide a correct answer • Access point which is utilized by the mobile services is shut down

Elevation of privilege	• Attacker eavesdrop to the data exchanged between the smart toy and the mobile device and alters it in order to gain access • Attacker eavesdrop to the data exchanged between the mobile device and the mobile app and alters it in order to gain access

Table 2. STRIDE model classification regarding threats performed on smart toys [2]

3.2 Assessment of vulnerabilities' impact and plausibility

A research made by Gordon Chu, Noah Apthorpe, and Nick Feamster [4] on three smart toys revealed several vulnerabilities, some of them not being patched for a considerable amount of time. It can then be apparent that vulnerabilities and security issues are indeed a realistic threat which leads to the need of raising (especially) parents' security awareness.

Lack of encryption and authentication is a common cause for security vulnerabilities on smart toys. An example derived from the abovementioned research where a hydration tracker (a smart device which tracks the daily water consumption) is investigated and HTTP requests (POST and GET) are unencrypted. A malicious party could eavesdrop on the exchange of information between the mobile app and the cloud servers and obtain private information (GET request to obtain the profile pic of the mobile app user) or even spoof a response that would contain random data in order to trigger the remote execution of a piece of code [4].

A second example is the smart pet which has more than ten thousand installs on android. It is observed again that there are Java files with source code containing constant variables in cleartext. Furthermore, there is use of unencrypted GET requests in order to obtain needed XML files, something that could lead to interception of those files and maliciously altered with a chance of remote code execution [4]. A conclusion that can be draw here is a repeated pattern of unencrypted requests or storage of information (in the above example the requests should have been made in HTTPS instead of HTTP).

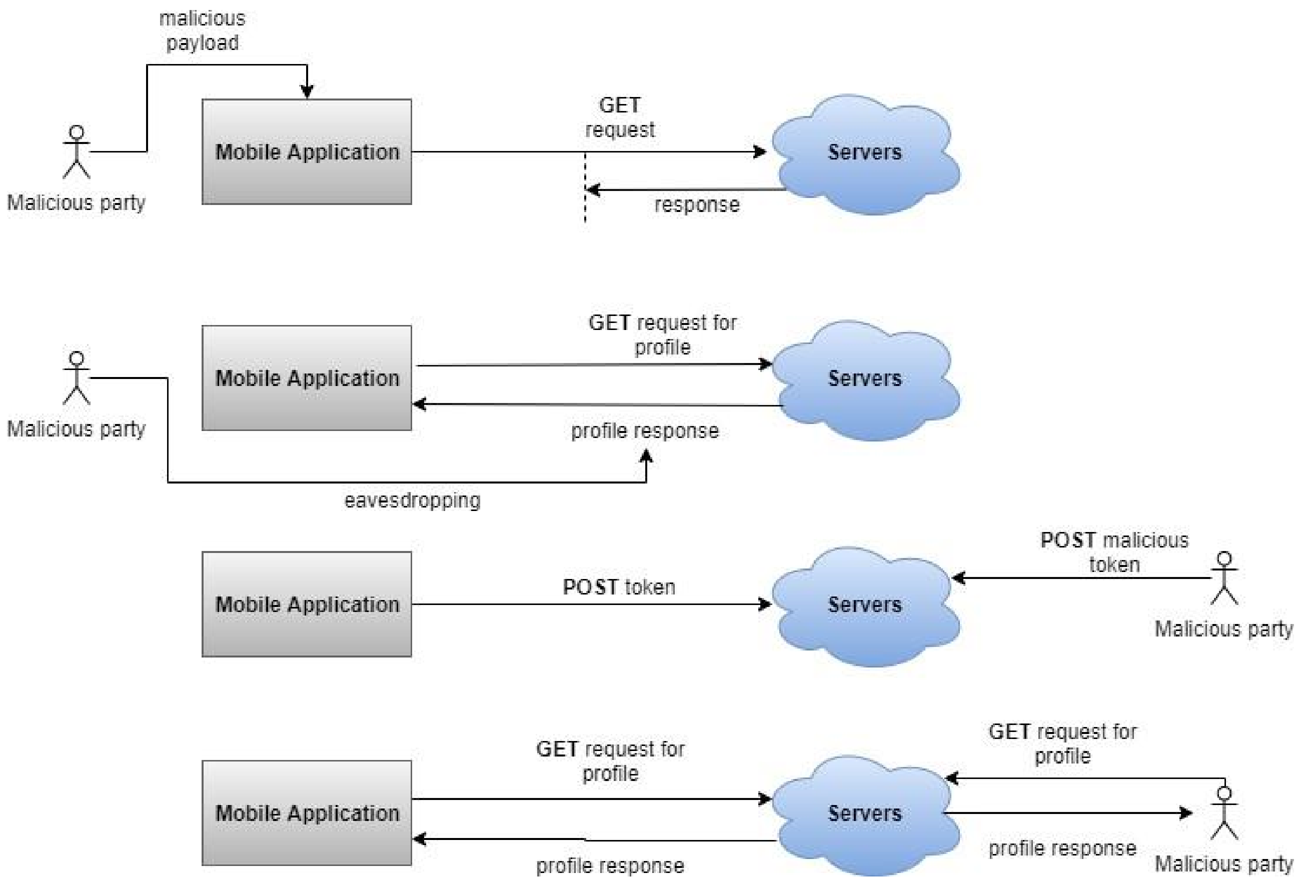


Figure 8. Malicious party utilizes unencrypted GET and POST http requests in order to communicate with the mobile application or the servers. [4]

We can observe that attacks on smart toys is an increasingly discussed topic and one of the main reasons is the companies' failure to implement the required measures in order to mitigate dangers related to leak of information, especially considering the sensitive consumer group most of those devices are targeting to. Because of this the impact of security breaches related to smart toys can cause serious concern. One of the reasons is that children generally don't have an increased awareness of what kind of information they share with the smart toy and how the information is recorded, something that is enhanced by the fact that a child can form a strong connection of trust with the toy [36]. Furthermore, potential leak of the child's location can put the child to physical danger as it can be possible that a child predator can obtain the information and track the location potentially trying to harm the child.

Similar security issues have been found in another research by Johann-Sebastian Pleban, Ricardo Band, Reiner Creutzburg [37] which examined the AR.Drone 2.0 quadcopter. After the toy boots up an insecure, unencrypted WIFI hotspot is established in which a malicious attacker can connect and perform a port scan revealing a connection to the FTP service as well as Telnet. In both cases there is lack of authorization measures in place which can lead to the attacker tampering with the configuration files or even injecting malicious files to the drone. Similarly, to the cases mentioned above, the smartphone app accompanying the toy offers more "opportunities" for malicious actions to the attacker as it can be vulnerable to eavesdropping and thus the attacker gaining access to e.g., video stream [37]. As before we can observe that using an encrypted WIFI hotspot instead of an unencrypted one could lead to a far more robust and secure infrastructure.

Security vulnerabilities related to smart toys are something common; as seen in the research paper by Sharon Shasha *et al*., the majority of the IoT devices currently on the market which are intended for younger ages have shown to be susceptible to attacks conducted either physically or remotely [34]. Furthermore, most of the tested devices are also gathering personal information which proceeded to communicate with several ads as well as analytics servers. Surprisingly, the gathering of personal identifying information was more intense when targeting younger audiences and children as one would expect that security measures would be more well thought in such vulnerable target groups. Additionally, the authors analyzed eleven smart devices of which nine were vulnerable to some form of attack depending on the access type: physical access, nearby point or remote access. The below table presents the percentage of plausible attacks accompanied by their access category:

Vulnerability	**Percentage of smart toys being susceptible**	**Type of access**
Insecure-Bluetooth-practice	54 %	Nearby
Unauthorized-config-physical	36 %	Physical
Unencrypted-comm-channels	36 %	Remote
URL-redirect	36 %	Remote

No-local personal identifiable information protection	27 %	Physical
Unauthorized-use	27 %	Nearby
Online-password-brute force	27 %	Remote
Insecure-session-cookies	27 %	Remote
Unauthorized-config-nearby	18 %	Nearby
Always-on	18 %	Nearby
Exposure-to-third-parties	18 %	Remote
Insecure-TLS-practices	18 %	Remote
Unencrypted-hotspot	9 %	Nearby
Weak-parental personal identifiable information control	9 %	Remote
Not having remote personal identifiable information protection	0 %	Remote

Table 3. Percentage of susceptible smart toys to vulnerabilities depending on their type of access required (Adapted from [34])

It can be observed that one of the most frequent weaknesses when it comes to security and smart toys is insecure Bluetooth implementation in development. Specifically, all the toys examined [34] were utilizing a static MAC address making the toy and the user (which plausibly is a child) trackable. Furthermore, certain toys accept unauthorized incoming Bluetooth connections which could lead to attacker connecting to the toy and altering its behavior. Implementing Bluetooth Low Energy could potentially help manufacturers solve the abovementioned issues through MAC address randomization and whitelisting.

A significant amount of high-profile attacks on smart toys shows that security attacks are indeed feasible and measures against them have been, in many cases, lacking and not given the necessary attention by the manufacturing companies. One such case took place in 2015 when the Hong Kong based toy manufacturer VTech reported that over 10 million accounts were compromised as the database of their online store related to their toys was breached. This resulted in a vast amount of personal information leaking such as names, email addresses, secret questions as well as the answers to them, user download history etc. Another report claimed that also photographs of younger users were leaked as well as chat logs. The attack was performed through a SQL injection. While VTech reported that images and audio files were encrypted with the AES algorithm the decryption keys on the other hand were lacking the necessary security measures [3].

Another case that indicates the necessity for manufacturing companies in putting more effort in securing not only the physical devices themselves but also their own services related to them is the Hello Kitty breach. The information of over three million accounts was leaked, including names, gender, age, emails and country of origin. The cause of the breach was a misconfiguration during the installation of the MongoDB database [3]. Breaches like the one in the Hello Kitty case should be quite alarming as identity theft is a serious issue when related to adults; even more so when the hacked accounts possibly include children as well. While Sanrio, the creator of Hello Kitty toys, initially declined that data was actually stolen, in the last statement admitted that the leaked data looked real and there has been indeed a major incident of data theft cause by the MongoDB misconfiguration.

There are several reasons that smart toy manufacturers neglect maintaining and enhancing the security measures of the device. One of the main reasons seem to be the limited budget of the manufacturers. Unfortunately, while trying to maximize profit companies that create IoT devices for children do not invest enough in the enhancement of the security measures. In the abovementioned case of VTech, the manufacturer was using an algorithm in order to hash passwords (MD5) which was already deemed obsolete due to increased computing power of available devices in the market since the hashing algorithm was first introduced [38]. Another factor that could lead to smart toys lacking the necessary security infrastructure is the manufacturer's lack of capable personnel which can lead to software bugs, security holes. Finally, lack of motivation and focusing on implementing smart toy features as fast as possible in order to be “pushed” to production (in an attempt to maximize profit), while neglecting the security aspect of the IoT device is another factor.

Even after an incident of a security breach is revealed, smart toy manufacturers fail to make serious attempt in order to strengthen security measures against malicious attacks. In the case of VTech for example, after the breach in their database, the manufacturer did not make any serious attempts in fixing the issues. Instead they revised the terms and conditions of the user agreement in order to abdicate responsibility and indicate that fault lies with the parents in case of a future data leak [38].

Another example of smart-toy manufacturers acting with irresponsibility towards consumers is presented in the research paper of Gordon Chu, Noah Apthorpe, and Nick Feamster where several devices violate the Children's Online Privacy Protection Act (COPPA). In the first case the presented hydro-tracker violates COPPA due to the possibility of the profile picture mining attack. As the consumer target group is likely children, the pictures are likely to be mostly of children which indicates that there are no adequate measures to protect the privacy and personal information of children. Also, personal information should be released to providers and third parties which have taken necessary steps to maintain the confidentiality and integrity of the data. Furthermore, when a new profile picture is set, the URL directing to the old one is still valid and working, something which violates the data retention policy of COPPA [4].

As shown above while manufacturers state that their privacy policy and implementation offer strong security and data protection, often their promises fall short. The hydration tracker is one such example but there is a plethora of other smart toys in the market with the same pattern; strong privacy policies that fail during the implementation phase. This is enforced especially with cases such as a smart toy using an encryption algorithm in order to protect voice data of children while at the same time using a fixed set of keys which makes decryption of data in transit between the device and the server plausible. Consequentially, there have been legal actions taken against toy manufacturers such as VTech. Thus, it is important that researchers continue to analyze and test the security level of smart toys as well as provide information when security vulnerabilities are discovered to the responsible actors [4].

There have been several researchers trying to identify the requirements that can be set to smart toy manufacturers, in order to comply with data and privacy protection regulations such as GDPR which has been set by the European Union Council. The requirement set can be identified based on e.g. Microsoft SDL (security software development) during the first two main phases of requirements and design [2]. It can be observed that the requirements can be grouped into specific categories based on their context. The category of “integrity” can include requirements that relate to the identification and statement of the reasons for specific actions from the manufacturers' side which target the customer side (e.g. reasons for collecting personal data). More specifically, this category can include requirements such as the manufacturers' obligation to state which information they collect as well as its use and methods of collection. Furthermore, included in the contextual category of "integrity" are the requirements of acquiring parental consent before attempting to collect and process sensitive data, as well as notifying the consumers in case of a data breach or leak. Finally, another example of a contextual category can be that of "clarity" which relates to the ways the abovementioned actions are handled and processed. An example falling under this category is the obligation of manufacturers to inquire for parental consent in a clear, transparent and intelligent form. Below the full table of contextual categories and associated requirements is presented.

Contextual Category	**Requirement**
Integrity	• State and identify the collected information • Acquire parental consent before attempting to collect and process sensitive data • Notify customers in the case of a data breach • Specify the type of the collected information • Acquire individual consent for collected data • Cease processing of information to third parties when data is erased
Clarity	• Asking for consent in an intelligent and transparent form • Document clearly and consistently the reason for collecting the necessary data
Consistency	• Same level of protection for data processed by third parties • Only promote the necessary personal information disclosure

	• Only store and process required information as long as necessary • Always keep personal information up to date and accurate as necessary
Efficiency	• Protect data's availability, confidentiality and integrity • Design procedure and actions to protect the collected data • Implement measures against theft or loss, unauthorized access and modification • Measures have to be designed from the beginning of the system's lifecycle

Table 4. Contextual categories and associated requirements

4 Evaluation of a commercially available smart toy

4.1 Method and used tools

For the purpose of the case study, I evaluated the potential security risks of a commercially available fitness band (Willful), whose target consumers include children and teenagers. The device is also accompanied by the relevant mobile application. The evaluation approach consists of a static and dynamical analysis in order to identify security vulnerabilities.

The evaluation process utilized tools used in penetration testing, malware analysis, security assessment etc., in order to perform the static analysis. Furthermore, for the dynamic analysis, a Raspberry Pi 3 Model A+ was used in order to create a Wi-Fi access point. After the Wi-Fi access point is created, a mobile phone device with the smart toy's relevant application installed, is connected to the access point. In order to observe the flow of web traffic an open-source proxy-software was used called "mitmproxy" [39]. The traffic was observed with a laptop connected to the same network as the Raspberry Pi is through an Ethernet cable, using the web interface of mitmproxy.

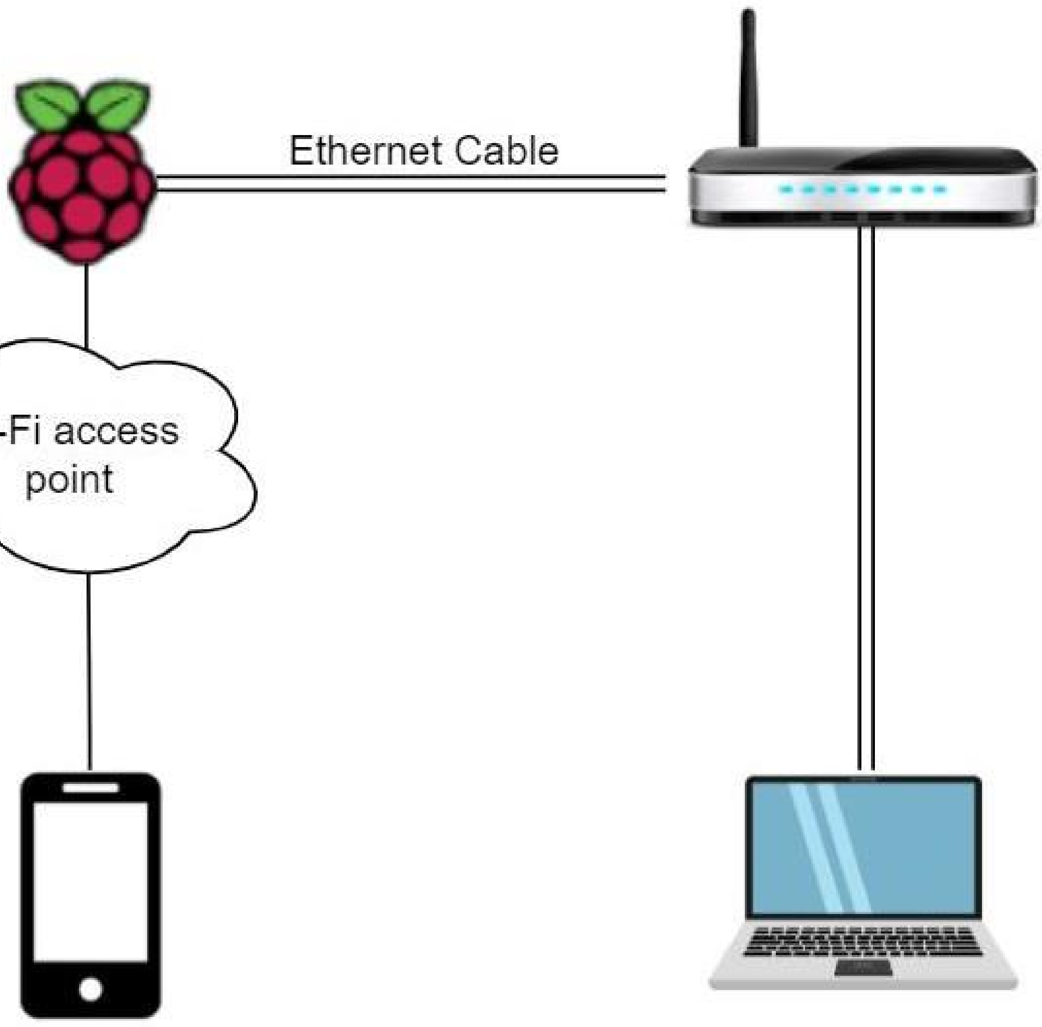


Figure 9. Setup used for dynamic analysis

In order to perform a static analysis Kali Linux running on VMware was used. Furthermore the apk of the smart device's relevant mobile application was obtained from https://apkpure.com/ and its analysis was achieved through MobSF [40].

4.2 Static Analysis

In order to create a static analysis on the APK of the mobile application, Kali Linux was used as the selected operating system. The OS was run in a created virtual machine on Windows 10 with the usage of VMWare. Furthermore, after the creation of the virtual machine and the necessary operating system's updates, MobSF was used in order to analyze the apk and obtain critical information about the mobile application.

4.2.1 Mobile Security Framework

Mobile Security Framework is a tool used in order to perform penetration testing in mobile applications, malware analysis and it is capable of performing both static and dynamic analysis. In this case study MobSF was used mainly in order to perform static analysis on the relevant mobile application as dynamic analysis required running MobSF outside of a virtual machine. The pre-requirements for MobSF in the Kali Linux operating systems is:

1. Installation of git
2. Installed version of Python v3.7
3. Installed JDK 8+
4. Installation of relevant dependencies as stated in the documentation [41]

Moreover, for the installation of MobSF on a Linux based system the following commands need to be executed on the terminal (clone from github, navigate to the directory of MobSF, and run setup.sh):

```
git clone https://github.com/MobSF/Mobile-Security-Framework-MobSF.git
cd Mobile-Security-Framework-MobSF
./setup.sh
```

Lastly in order to run MobSF we execute *./run.sh 127.0.0.1:8000* and in a browser of our choice navigate to localhost:8000 where the apk of the mobile application that will be investigated can be uploaded.

4.2.2 Static report

Firstly, the application is signed with v1 signature scheme, which can be a potential security risk especially on Android versions older than version 7.0 due to the Janus vulnerability. The attackers can modify the code of applications by adding extra bytes without altering the signature which leads to the Android runtime accepting the malicious code as legitimate update of the affected application [42].

Furthermore, the application requires multiple permissions, many of them are considered dangerous and can lead to potential security issues. Some of the permissions required are the following:

PERMISSION	INFO	DESCRIPTION
ACCESS_COARSE_LOCATION	Coarse(network-based) location	Access coarse location sources, such as the mobile network database, to determine an approximate phone location where available. Malicious applications can use this to determine approximately where the use is located.
ACCESS_FINE_LOCATION	GPS location	Access fine location sources such as GPS on the mobile device, where available.
ANSWER_PHONE_CALLS	Phone calls	Allows the application to answer an incoming phone call.
BLUETOOTH	Create Bluetooth connections	Allows an application to view the configuration of Bluetooth on the device as well as creating and accepting

		connections with paired devices.
BLUETOOTH_ADMIN	Bluetooth administration	Allows the application to configure the local Bluetooth and pair it with remote devices.
CALL_PHONE	Directly call phone numbers	Allows the application to call phone numbers without user intervention. Malicious application may cause unexpected call charges. The permission does not allow the application to call emergency numbers.
CAMERA	Take pictures and videos	Allows the application to take pictures and videos with the camera. This allows the application to collect images that the camera is seeing at any time
CHANGE_WIFI_STATE	Change Wi-Fi status	Allows an application to connect and disconnect from Wi-Fi access points and to make changes to configures Wi-Fi networks.
GET_TASKS	Retrieve running applications	Allows the application to retrieve information about currently running (or recently run) tasks. Malicious application may discover

		private information about other applications.
INTERNET	Full Internet access	Allows the application to create network sockets
READ_CONTACTS	Read contact data	Allows an application to read all of the contact data stored in the mobile device.
READ_EXTERNAL_STORAGE	Read SD card contents	Allows the application to read from SD card
READ_PHONE_STATE	Read phone state and identity	Allows the application to access the phone features of the device. An application with this permission can determine the phone number and serial number of the phone, whether a call is active, the number that call I connected to etc.
READ_SMS	Read SMS	Allows the application to read SMS messages stored in the mobile device or SIM card. A malicious application may read confidential information
RECEIVE_SMS	Receive SMS	Allows the app to receive and process SMS messages. Malicious applications may monitor your messages or delete them without letting the user know.
REQUEST_INSTALL_PACKAG ES	Request installing packages	Allows an application to request installing packages.

		Malicious applications can use this to try and trick users into installing additional malicious packages.
WRITE_CONTACTS	Write contact data	Allows the application to modify addresses stored on the mobile device. Malicious applications can use this to erase or modify contact data.
com.google.android.providers.gsf.permission.READ_GSERVICES	Google maps permission	This permission is related to the google maps API and allows the application to modify the google service map. However, the permission is not required anymore in order to use the Google Maps API.

Table 5. Sample of application's required permissions

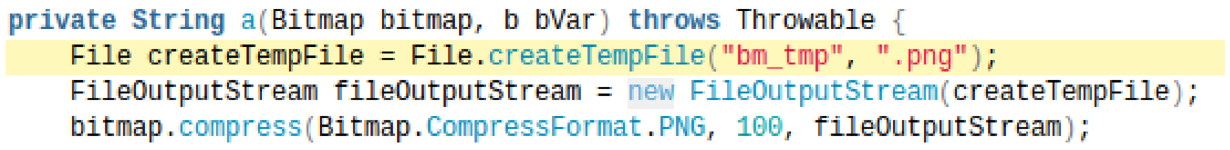
Apart from the aforementioned permissions, the Android manifest analysis also reveals certain security risks some of them categorized as highly severe:

- Launch Mode of several activities (*.module.home.MainActivity, .module.me.FeedbackActivity* etc) is not standard. An Activity should not have the launch mode attributes set to "singleTask/singleInstance" as it becomes root Activity and it is possible for other applications to read the contents of the calling intent. Using the "standard" launch mode attribute is recommended.
- Several activities (*com.mob.tools.MobUIShell*, *.module.home.WXEntryActivity* etc) are not protected. The activity is found to be shared with other application on the device therefore leaving it accessible to any other app on the mobile device.

- Service (*.common.location.LocationService*) is not protected as it is found to be shared with other apps on the device, thus leaving it accessible to any other application on the mobile device.
- Certain Services such as *.second.ui.services.IntelligentNotificationService* is protected by a permission, but the protection level of permission should be checked. The permission for the service is **android.permisson.BIND_NOTIFICATION_LISTENER_SERVICE.** The service is protected by the permission which is not defined in the analyzed application. Consecutively the protection level of the permission should be checked since in the case it is set in e.g. normal, a malicious application can request and obtain permission to interact with the component. If the permission is set to signature, only application signed with the same certificate can obtain the permission.

In addition to the Android manifest, analyzing the code of the application shows security risks some of the being severe in terms of importance. Firstly, the application uses SQLite Database and executes raw SQL queries. A malicious party can utilize this by inserting data to a raw SQL query which can cause an SQL injection. Furthermore, all sensitive information should be encrypted before written to the database.

Moreover, certain information is exposed (e.g. IP address) as it is included as plain text in the code of the application. The application can also read and write to an external storage which can present a security risk of another application reading the data which is written to the external storage. Another interesting mention is that the application creates a temp file which is a .png.


```
private String a(Bitmap bitmap, b bVar) throws Throwable {
    File createTempFile = File.createTempFile("bm_tmp", ".png");
    FileOutputStream fileOutputStream = new FileOutputStream(createTempFile);
    bitmap.compress(Bitmap.CompressFormat.PNG, 100, fileOutputStream);
```
Figure 10. Created temp file

The naming does not reveal further information but generally sensitive data (it is probable that the .png file is the photograph of the user) should not be written in a temp file. It is also worth mentioning that the application uses the MD5 hash function in order to encrypt specific data such as the username. MD5 has been known to be a weak hash function that has severe design flaws and hash collisions where given two different messages the final MD5 hash can be identical [43]. Additionally, in some instances the application uses the statistical pseudo-random number generator (java.util.Random) while it is recommended to use e.g. Java's SecureRandom class in order to generate a cryptographical secure number.

The application implements WebView insecurely since the execution of user-controlled code in WebView is a critical security hole. Usually developers use WebView to display secure websites, but in the case a user navigates to an untrusted malicious website, sensitive data can potentially be exposed. In addition, if the application loads an HTTP page and it is connected in an insecure Wi-Fi access point, it will be susceptible to a man-in-the-middle attack where malicious data can be inserted into the page [44].

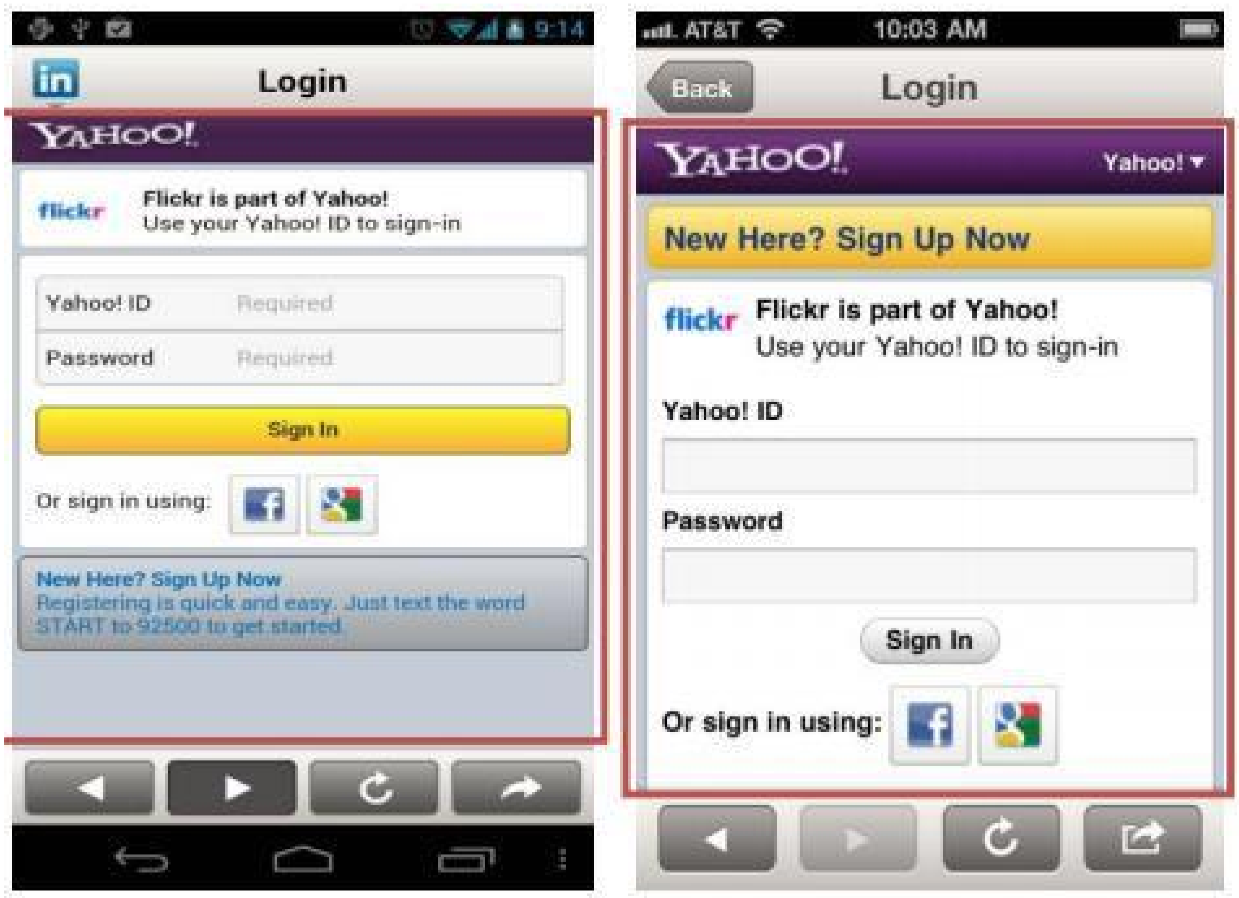


Figure 11: Accessing Yahoo! Login page from WebView-based LinkedIn apps in Android (left) and iOS(right) [45]**.**

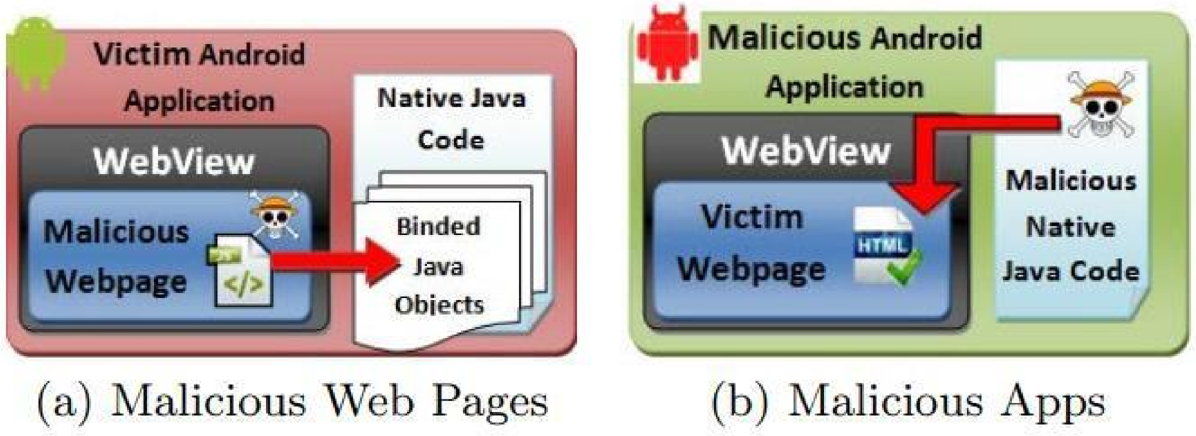


Figure 12. WebView threat models [46]

Another issue related to the how WebView is implemented in the application's code is that WebView ignores SSL Certificate errors and accepts any SSL Certificate something that can make it susceptible to a man-in-the-middle attack [45].

Furthermore, the application uses ECB mode where every block is encrypted independently and thus making it possible to process in a parallel way. ECB mode has significant drawbacks and presents a security risk. The fact that blocks are encrypted individually without relying on each other creates a security issue where a malicious party can hijack a block and replace a legitimate block with it without being detected. The aforementioned issue is called the block independency problem. Another issue with ECB mode is the fact that it results in identical ciphertext for identical blocks of plaintext [47].

In addition, it is worth mentioning that the application uses a Clipboard Manager in order to copy data to the clipboard. It is not clear from the source code if the string that is copied contains sensitive data but generally is it inadvisable that private or sensitive information is copied to the clipboard as other applications can have access to it.

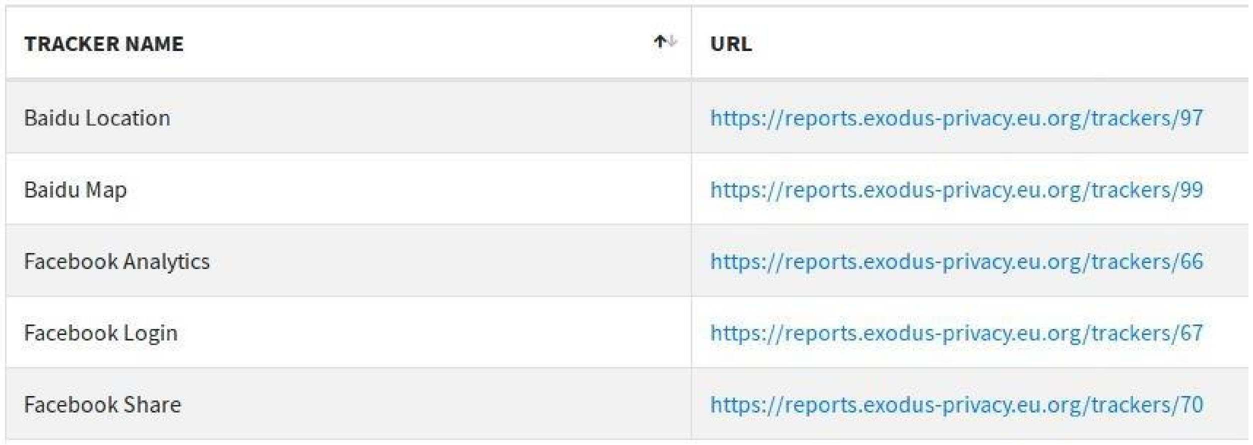
The performed APKiD analysis checks the available classes.dex files in order to identify code that could potentially indicate the presence of techniques similarly used by malware software (e.g. Anti-VM Code or Anti Debug Code). Malware authors can potentially use the aforementioned techniques in order to make the process of discovering malicious code difficult. The Anti-VM techniques are utilized in order to identify whether the code is being run in a Virtual Environment (as most security analysts prefer to run the analysis inside a virtual machine) and consequentially complicate the analysis. On the other hand, the Anti-Debug techniques are used in order to detect if the malicious code is being debugged and thus slow down the process. The findings are presented to the following table:

DEX	**DETECTIONS**	
classes.dex	Anti-VM Code	Build.FINGERPRINT check Build.MODEL check Build.MANUFACTURER check Build.PRODUCT check Build.BOARD check Possible Build.SERIAL check Network operator name check Subscriber ID check
classes2.dex	Anti-Debug Code	Debug.isDebuggerConnected() check
	Anti-VM Code	Build.FINGERPRINT check Build.MANUFACTURER check Build.BOARD check Possible Build.SERIAL check Subscriber ID check
classes3.dex	Anti-VM Code	Build.FINGERPRINT check Build.MODEL check Build.MANUFACTURER check Build.HARDWARE check Build.BOARD check

		Possible Build.SERIAL check Build.TAGS check SIM operator check Network operator name check Subscriber ID check Possible ro.secure check Emulator file check
lib/arm64-v8a/libJni_wgs2gcj.so lib/armeabi/libJni_wgs2gcj.so	Obfuscator	Obfuscator-LLVM version 3.6.1
lib/armeabi/libAMapSDK_MAP_v6_9_3.so	Packer Found	Sharelib UPX

Table 6. APKiD Analysis

All the domains relevant to the app were checked and are secure as nothing out of the ordinary stood out during the analysis. Several emails were also visible, which are assumingly connected with test operations as well as used in order to send logged errors etc. Furthermore, the following trackers utilized by the application were identified:



TRACKER NAME	URL
Baidu Location	https://reports.exodus-privacy.eu.org/trackers/97
Baidu Map	https://reports.exodus-privacy.eu.org/trackers/99
Facebook Analytics	https://reports.exodus-privacy.eu.org/trackers/66
Facebook Login	https://reports.exodus-privacy.eu.org/trackers/67
Facebook Share	https://reports.exodus-privacy.eu.org/trackers/70

Table 7. List of trackers

4.3 Dynamic Analysis

Initially an attempt was made to utilize MobSF's dynamic analysis capabilities. This required running MobSF outside a virtual machine while also installing an Android Emulator. For this purpose, Genymotion was installed alongside Virtual Box in order to create an android virtual environment. During execution of dynamic analysis, I encountered several crashes and errors preventing me from obtaining any useful information. Therefore, a different approach was preferred (creating a Wi-Fi access point with the utilization of a Raspberry Pi as well as using MITM proxy) which is further explained below.

4.3.1 MITM Proxy

When mentioning Man-In-The-Middle Proxy we refer to a part of software that is running on e.g. a Wi-Fi access point or a router and is used as an intermediate between the client, for example a mobile device, and the server. By doing so the MITM Proxy can intercept and parse web traffic between the two points of the client and the server, as well as modify a certain request and then pass it on to the relevant party (client or server).

In many cases the MITM proxy can be used as a mean of protection and monitoring inside networks, e.g. a company's private network where an MITM proxy can be used in order to avoid passing sensitive data to the Internet.

MITM proxy can be also used in malicious ways in order to help the attacker intercept traffic and manipulate it. An example can be any open network (these can still be found in many public places e.g. cafeteria) where the attacker has set up the MITM proxy to a created Wi-Fi access point. Thus, if a client connects to the access point, the malicious party will be able to intercept and parse traffic that may contain sensitive data.

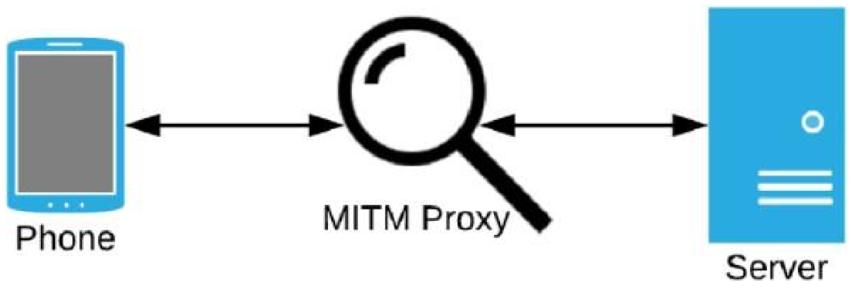


Figure 13. MITM Proxy [48]

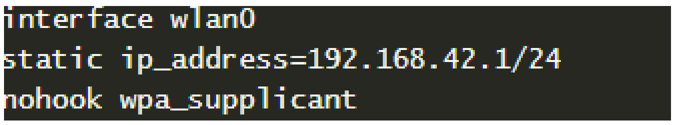
4.3.2 Setting up MITM Proxy on Raspberry Pi 3 A+

4.3.2.1 Creation of Wi-Fi access point

The first step in order to set up MITM Proxy was to utilize the Raspberry Pi as a Wi-Fi access point in which the phone will be connected, and traffic will be visible. Initially, it is important to configure the Dynamic Host Configuration Protocol (DHCP). The DHCP is an automation mechanism that configures the network interface when it received a dynamical address [49]. Another option is to assign a static address to the network interface which is the chosen option in this set up.

Firstly, I configured the Raspberry Pi's DHCP client configuration in order to assign a static address to the wireless network interface (wlan0). This will behave as the Wi-Fi access point's address in which devices can connect. In the case of the DHCP server side, the software is responsible for assigning the address to the devices connecting to the above network.

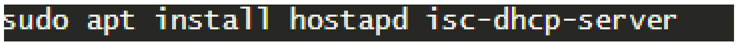
The following lines are added to the *dhcpcd.conf* file:



```
interface wlan0
static ip_address=192.168.42.1/24
nohook wpa_supplicant
```

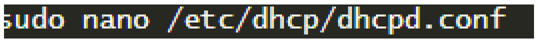
Now the static ip_address is assigned and the wpa_supplicant's hook is disabled in order to prevent creating unwanted connection events [50].

Additionally, *isc-dhcp-server* and *hostapd* need to be installed. Isc-dhcp-server is used in order to assign addresses to the connected devices while hostapd work as an authenticator of devices connecting to the access point. In order to install the dhcp-server:



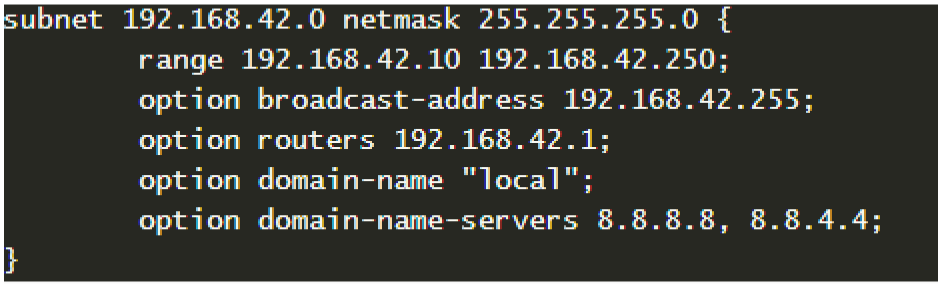
```
sudo apt install hostapd isc-dhcp-server
```

After the installation is complete, the configuration file must be edited,



```
sudo nano /etc/dhcp/dhcpd.conf
```

and authoritative option uncommented. In order to configure the DHCP server to assign addresses in the range of 192.168.42.10 to 192.168.42.250 and a broadcast address of 192.168.42.255 we need to add the following lines to the DHCP server configuration file:



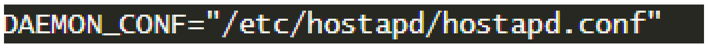
```
subnet 192.168.42.0 netmask 255.255.255.0 {
        range 192.168.42.10 192.168.42.250;
        option broadcast-address 192.168.42.255;
        option routers 192.168.42.1;
        option domain-name "local";
        option domain-name-servers 8.8.8.8, 8.8.4.4;
}
```

Additionally, in the DHCP server file the *interfacev4* needs to be set as the wireless network interface (wlan0) as well as the *interfacev6* need to be commented out.

The next step is the configuration of *hostapd* .config file where we set the following parameters:

- interface=wlan0
- ssid=[Enter you ssid here (id of the Wi-Fi access point)]
- macaddr_acl=0
- auth_algs=1
- wmm_enabled=0
- wpa=2
- wpa_passphrase=[Enter your password here]
- wpa_key_mgmt=WPA-PSK
- wpa_pairwise=TKIP
- rsn_pairwise=CCMP

Also, uncommenting the **DAEMON_CONF** line in the *hostapd* file is required:



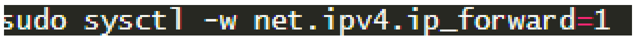
```
DAEMON_CONF="/etc/hostapd/hostapd.conf"
```

Now the created Wi-Fi access point should be visible, and devices should be able to connect to it.

4.3.2.2 Installation and usage of MITM Proxy

In order to install the MITM Proxy a simple ***sudo pip install mitmproxy*** command is used. When the installation is complete, we can verify that *mitmproxy* was correctly installed with: ***mitmproxy --version.***

Furthermore, in order to be able to successfully intercept web traffic and let it pass through the wired and wireless interfaces, the IP tables' rules must be modified and saved. The final step in the setup is to allow the OS to forward traffic between networks and can be achieved by executing,



```
sudo sysctl -w net.ipv4.ip_forward=1
```

as well as uncommenting **net.ipv4.ip_forward=1** option in the sysctl.conf file in order to allow the aforementioned setting to persist when the system is rebooted.

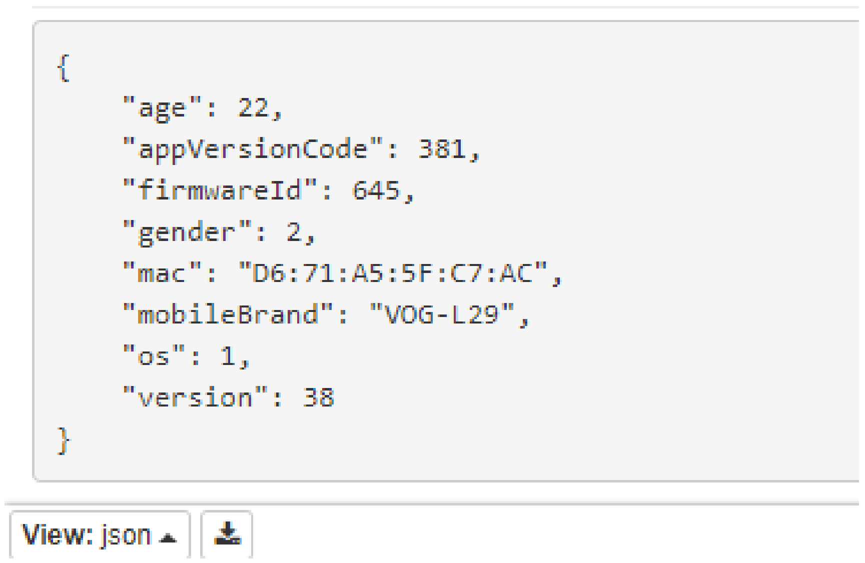
4.3.3 Traffic observation and analysis

The set-up is complete, and we can now intercept traffic and observe it through MITM proxy something that can be achieved conveniently through its web interface.

Initially and after starting the fit band's accompanied mobile application, a POST Request is created: **POST http://***(name of the application's api)***/firmware/getLatestV2 HTTP/1.1**

It is easily observed that the POST request is an unencrypted and unauthenticated http Request. This remains the case in every request the application and the servers send to each other and renders it easy for the attacker to spy and alter traffic between those two points.

Interestingly the application in the aforementioned initial POST request uses data (such as age and gender) which can be considered unnecessary:

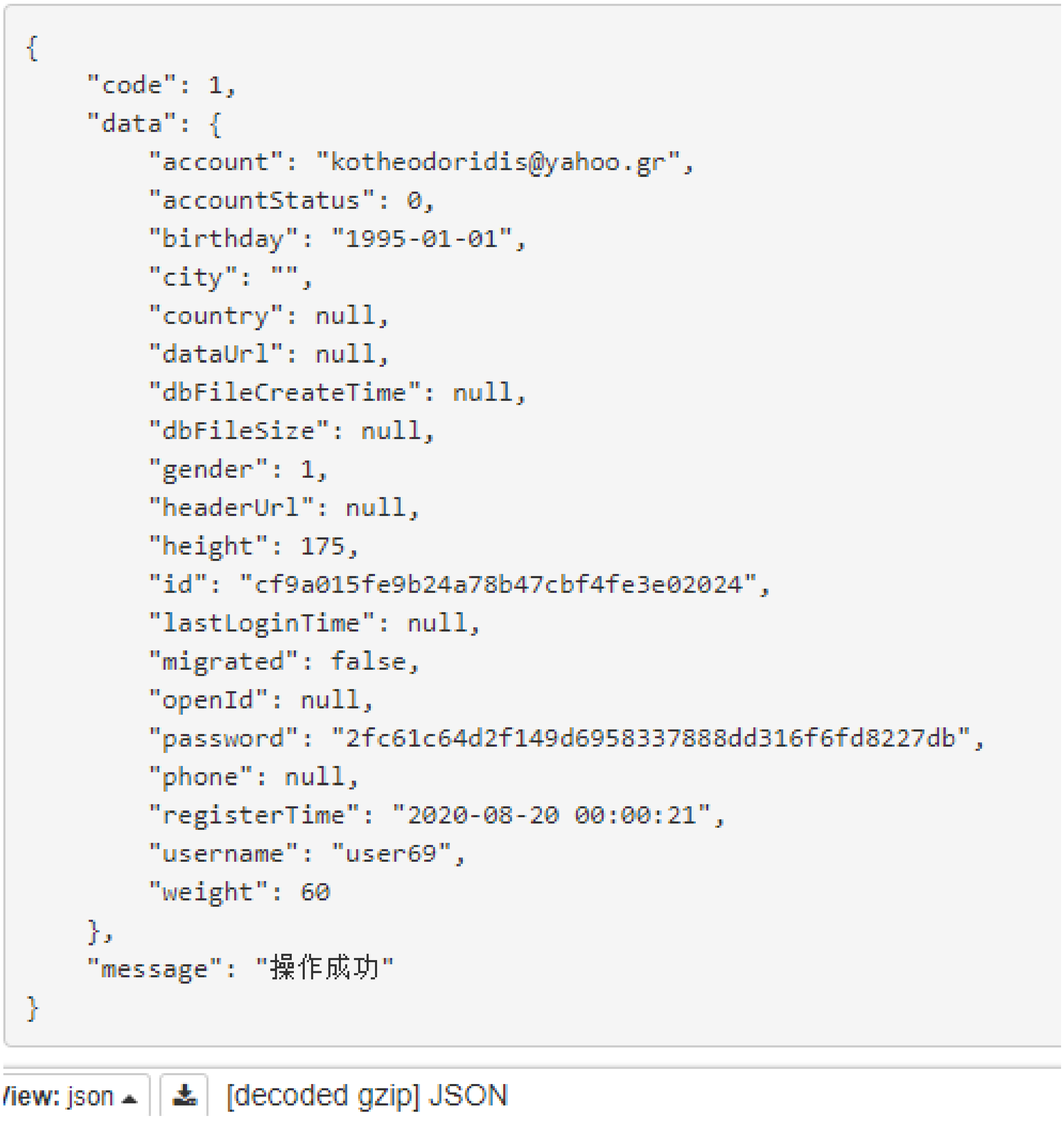


```
{
    "age": 22,
    "appVersionCode": 381,
    "firmwareId": 645,
    "gender": 2,
    "mac": "D6:71:A5:5F:C7:AC",
    "mobileBrand": "VOG-L29",
    "os": 1,
    "version": 38
}
```

Figure 14: Application obtains firmware Id

After the application obtains the response from the server, it attempts to automatically login with a **POST** http://*(name of the application's api)***/user/login HTTP/1.1**

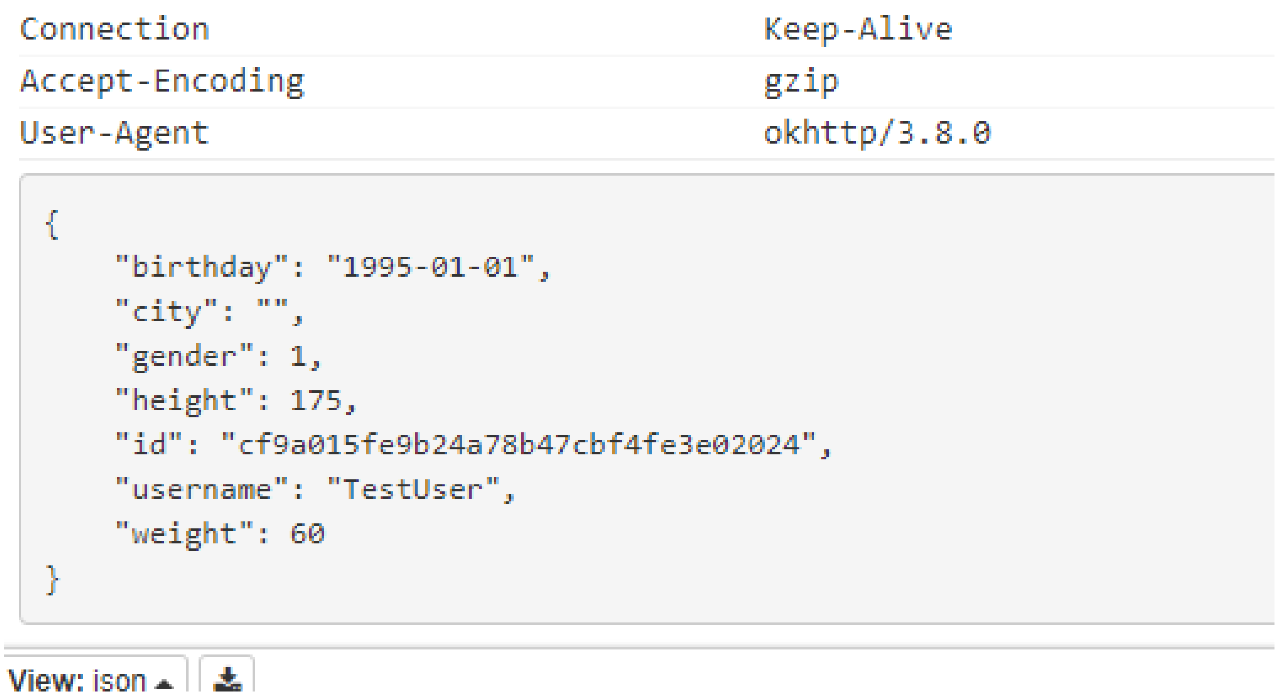
The server's response contains personal information that a malicious party could intercept (personal data which is stored in the server is assumingly used in order to populate the various fields of the application).



```
{
    "code": 1,
    "data": {
        "account": "kotheodoridis@yahoo.gr",
        "accountStatus": 0,
        "birthday": "1995-01-01",
        "city": "",
        "country": null,
        "dataUrl": null,
        "dbFileCreateTime": null,
        "dbFileSize": null,
        "gender": 1,
        "headerUrl": null,
        "height": 175,
        "id": "cf9a015fe9b24a78b47cbf4fe3e02024",
        "lastLoginTime": null,
        "migrated": false,
        "openId": null,
        "password": "2fc61c64d2f149d6958337888dd316f6fd8227db",
        "phone": null,
        "registerTime": "2020-08-20 00:00:21",
        "username": "user69",
        "weight": 60
    },
    "message": "操作成功"
}
```

Figure 15: Server response after login request

Furthermore, when the mobile application user attempts to change her personal information, the POST request http://*(name of the application's api)*/**user/updateUserInfo HTTP/1.1** is an HTTP request with unencrypted data:



Connection	Keep-Alive
Accept-Encoding	gzip
User-Agent	okhttp/3.8.0

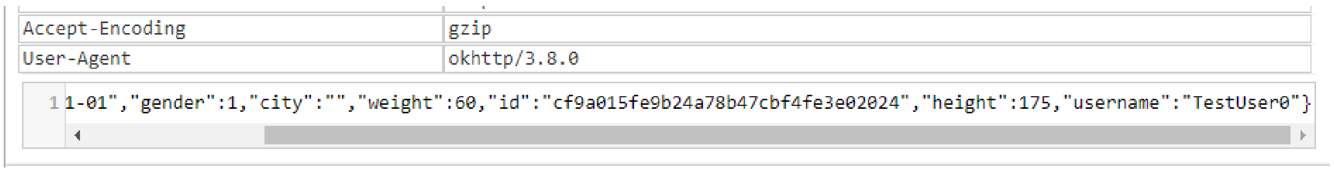
```
{
    "birthday": "1995-01-01",
    "city": "",
    "gender": 1,
    "height": 175,
    "id": "cf9a015fe9b24a78b47cbf4fe3e02024",
    "username": "TestUser",
    "weight": 60
}
```

Figure 16. Request when user edits personal information

In the above request the user id token is visible as well. After logging out of the mobile application and waiting for a considerable amount of time I attempted to investigate whether the token changes or is refreshed (depending either on the session being closed or a specific amount of time has passed). This is not the case, as the token remains the same and is reused every time the mobile application's user is attempting to change e.g. account information as shown above.

This presents a serious security vulnerability since in the instance where the malicious party intercepts the user id token, it can use the token in any request sent to the servers and pretend to be the legitimate user of the mobile application. Additionally, the user id token seems to be the only data the application uses in order to authenticate the logged in user when requests are sent to the server. The fact that all requests are unencrypted and unauthenticated HTTP requests, makes it easy for the attacker to spoof requests.

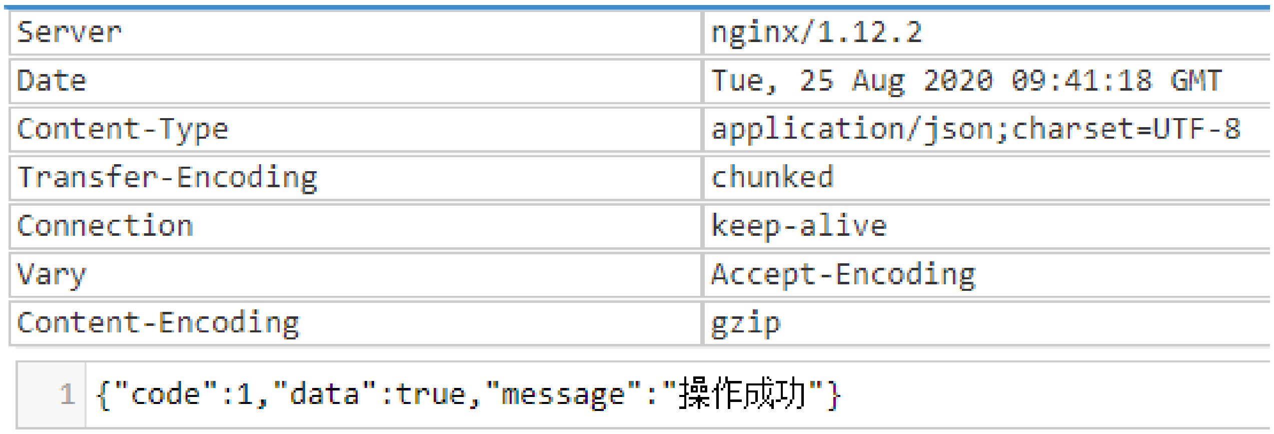
One example is the case where the same request is sent to the server but with the different username (as a result the user name is changed to "**TestUser0**" in the mobile application):



Accept-Encoding	gzip
User-Agent	okhttp/3.8.0

```
1-01","gender":1,"city":"","weight":60,"id":"cf9a015fe9b24a78b47cbf4fe3e02024","height":175,"username":"TestUser0"}
```

Figure 17. Request sent in order to change username

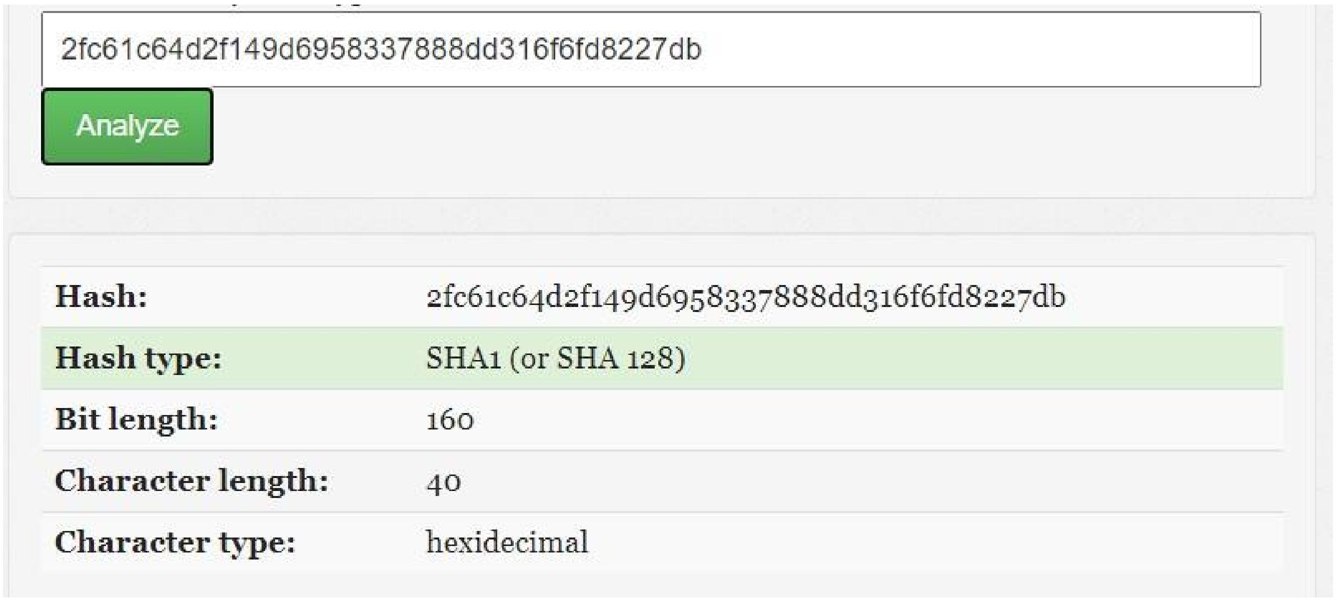


Server	nginx/1.12.2
Date	Tue, 25 Aug 2020 09:41:18 GMT
Content-Type	application/json;charset=UTF-8
Transfer-Encoding	chunked
Connection	keep-alive
Vary	Accept-Encoding
Content-Encoding	gzip

```
{"code":1,"data":true,"message":"操作成功"}
```

Figure 18. Server's successful response

Additionally, when a user logs into the mobile application the password, as seen in Figure 15, is included to the server's response. Using a hash analyzer, it is identified that the password is encrypted with a SHA-1 hash algorithm.

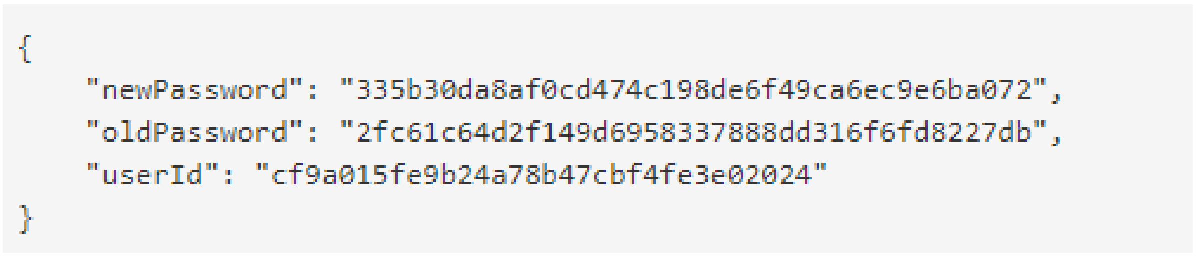


2fc61c64d2f149d6958337888dd316f6fd8227db

Hash:	2fc61c64d2f149d6958337888dd316f6fd8227db
Hash type:	SHA1 (or SHA 128)
Bit length:	160
Character length:	40
Character type:	hexidecimal

Figure 19. Hash analysis of user password

Therefore, the malicious party can create its own password and encrypt it using the SHA-1 hash algorithm. Since the user id token is not refreshed a POST request http://*(name of the application's api)*/**user/updatePasswordByUserId HTTP/1.1** can be spoofed and sent to the server in order to change the user's password:

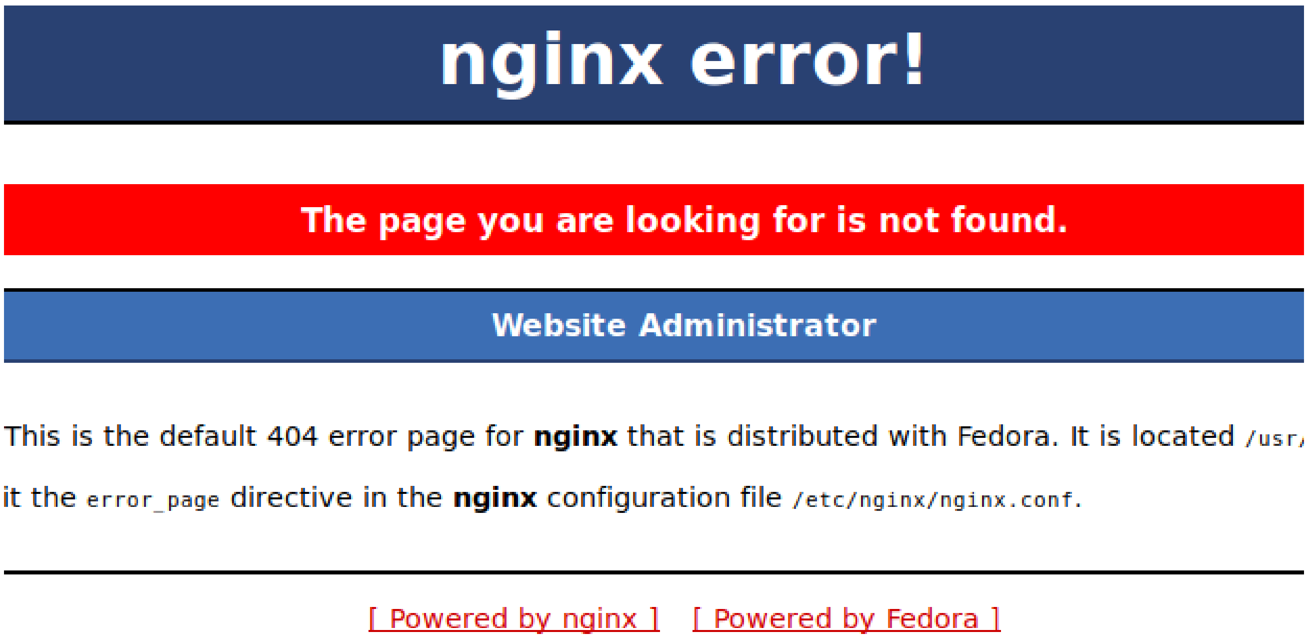


```
{
    "newPassword": "335b30da8af0cd474c198de6f49ca6ec9e6ba072",
    "oldPassword": "2fc61c64d2f149d6958337888dd316f6fd8227db",
    "userId": "cf9a015fe9b24a78b47cbf4fe3e02024"
}
```

Figure 20. POST request to change account's password

The server's respond contains a "successful operation" message indicating that the spoofed request was accepted (thus verifying that only the userId is checked upon). Indeed, when trying to login into the mobile application the password is changed into the one set by the attacker. Therefore, not only does the malicious party gain access to the legitimate user's account but also prevents the user from gaining access into the account.

Lastly, another observation that can be made is when a 404-server response is received the page is the default one which has not been customized accordingly. Therefore, the response includes unnecessary information such as the operating system, which in this case is Fedora and the server (nginx).



nginx error!

The page you are looking for is not found.

Website Administrator

This is the default 404 error page for **nginx** that is distributed with Fedora. It is located `/usr/`

it the `error_page` directive in the **nginx** configuration file `/etc/nginx/nginx.conf`.

Figure 21. Server's 404 response

5 Discussion

Both the static and dynamic analysis which the case study is consisted of revealed interesting results considering the security of the smart device's accompanied mobile application. It is also worth mentioning here that the application has over five million downloads in Google play store at the time of writing, hence raising the relevant security issues' importance even more. Furthermore, the smart toy device itself is easily purchasable in major online stores such as e.g. Amazon.

Firstly, there is a plethora of concerns and security issues after performing the static analysis. The application is vulnerable to the Janus vulnerability as it is signed with v1 signature scheme and this can be avoided by signing the application with a v2 signature scheme. This is only the case in newer versions of Android 7.0 and above as older versions are still vulnerable even with a v2 signature scheme. However, developers should always use the v2 signature scheme.

Furthermore, the mobile application requires multiple permissions in order to function as intended. This is a broader topic of discussion as many mobile applications abuse this by requesting user's permission to access certain functionalities of the mobile device that would be seemingly unnecessary to the application. This is called the Permission Abuse Problem [51]. Many of the aforementioned permissions can be utilized in order to obtain sensitive data such as contact addresses or even the location of the mobile device. One of the most interesting permissions which is utilized also by the case study's investigated mobile application, is android's READ_PHONE_STATE permission. As one of the most abused permissions, it can detect the mobile device's serial number and potentially uncover the owner's personal ID information [52]. Wu et al. [51] propose that a percentage of 77.6% of mobile applications are requesting a larger number of permissions that are actively needed therefore leading to a Permission Abuse Problem. The pitfallof this security issue is more frequently observed in less experienced developers.

The Andoid manifest analysis indicates several weak security points that could potentially be exploited by a malicious party. One of them being the launch mode of several activities. As Ren et al. demonstrated a system task has a certain preferred state by the attacker in which it contains a mix of malicious activities (from the malware) as well as harmless activities from the potential victim's application. The hijacked state transition occurs when a malware takes over and turns the system task into a dangerous one. There are two types of a hijack state transitions as either a malicious activity gets pushed into the victim's task back stack or a benign activity from the application gets pushed into the malware's back stack. The pre-conditions in order for certain hijacked state transitions to happen is that the launch mode of the relevant activity is set to "singleTask" instead of "standard" [53]. Therefore, it is advisable that developers are cautious when setting the launch mode to anything else than the standard mode.

An additional point that can be mentioned is the best practices that should be used when sharing data across multiple applications. The investigated application has several activities and services that can be accessed through other applications on the mobile device. Several actions can be made in order to enforce security when sharing the application's data such as explicitly defining the read only or write only permissions depending on the relevant occasion and needs. Furthermore, certain flags such as e.g. **FLAG_GRANT_READ_URI_PERMISSION** can be utilized in order to provide clients a single time access to relevant information and data. Lastly it is recommended to use "content://" URIs and not "file://" URIs [54].

The application uses the SQLite database while also using raw SQL queries. The source code reveals that string concatenation is used in order to construct the SQL queries, something that is strongly inadvisable. Including data to the initial query string can present problems ranging from bugs and smelly code to severe security risks. The recommended approach in order to avoid the aforementioned issues is to utilize parameterization. Instead of including data to the query, it includes only characters such as "?" whereas data is passed on as parameters in the same order as the special character appears in the query. The advantage of using parameterization when building the SQL query is that there is a clear separation between data and SQL syntax [55].

Another topic worth mentioning is WebView as it is an essential feature in Android development. Moreover, the static analysis revealed certain security risks related to how WebView is implemented in the investigated application. While WebView is one of the most popular tools in Android development and it is used extensively in many applications, it is vulnerable in cases where user's actions trigger the execution of the application's code. This is caused by certain features of WebView that are missing and are not supported in applications based on it. Some of the aforementioned features that exist in traditional web browsers but not in WebView based applications are the address bar which allows the user to be aware about the URL of the page that is being loaded as well as several security notifications such as the padlock that informs about the status of the page's security level (as seen in Figure 11) [45]. Furthermore, during the static analysis and code review of the application, it is observed that there is an improper SSL handling procedure in place, where the app is bypassing SSL certification and ignoring certificate errors. Both of those issues can lead to an increased risk of a malicious party performing a Man-in-the-Middle attack on the application.

In order to minimize the risk of an attack it is advisable that certain practices are being followed. Firstly, in order to address the lack of security cues that could help indicate to the user whether the page rendered is secure or not, Shin *et al*. [45] suggest using a visual security cue system called SSLight (or a similar system to SSLight) which displays cues similar to traffic lights and therefore informing the user of the security level of the rendered page. Moreover, another recommended security practice is the verification of the content being loaded as well as the evaluation in order to ensure that the content is the expected one. This can be performed by overriding the shouldOverrideUrlLoading() and the shouldInterceptRequest() methods and checking the domain the pages are loaded from [56]. Lastly and more importantly the SSL certificate error handling has to be addressed as ignoring error can lead to security holes. The majority of the SSL errors occur when the certificate has not been verified, thus leading in to blocking the unauthorized connection. Therefore, it is strongly recommended that developers avoid utilizing bypassing techniques or generally ignoring SSL errors and instead obtain and install a signed certificate as it is the most secure option [56].

Aside from the static analysis, the dynamic analysis indicates several security concerns as well. The main and most crucial security issue is that communications occur over HTTP as observed from the captured GET and POST requests. Using HTTP makes it considerably easier for an attacker to intercept communications between the application and the server. Additionally, the requests from the application as well the responses received from the server, often include potentially unnecessary personal information and data such as city and country of residence, age, weight, height etc. It is extremely inadvisable to use HTTP as instead a secure communication protocol such as HTTPS should be preferred and used in all communications.

Another critical security flaw that was recognized during the dynamic analysis was the fact that the token used in order to verify the user is never refreshed as it is assumingly stored and used as is without any restrictions or conditions set. This presents with a security issue where if an attacking party attempts and succeeds on intercepting the user id token, it can spoof requests sent to the server in order to pretend being the legitimate user. As shown in the analysis' examples the attacker can then change the user's information ranging from trivial information to crucial data such as password, email etc. It is important that developers address security flaws like the aforementioned one and therefore a recommended solution to the unaltered token is to establish necessary mechanisms in order for it to refresh. Some examples can be a time restriction on the token so that it is periodically being refreshed depending on the set time interval, or a session sensitive token where any time the user logs out of the mobile application the user id token is changed.

Lastly, it is observed that the application seemingly misuses unnecessary information either in requests towards the server or as shown in Figure 21 during a 404-error page server response where no customization has been made and both the operating system and the server are revealed. It is crucial that developers restrict the exposure of unrelated information in order to minimize the attack surface a malicious party can use in order to attack the system (e.g. customize the server's response containing error page in order to conceal the relevant information).

6 Conclusion and Future Work

IoT devices and more specifically Smart Toys have become increasingly popular therefore several questions about how well protected smart toys are, have arisen. A plethora of different security issues have already been identified and certain recommendations have been suggested in order to help developers enforce the security of smart toys. This thesis has analyzed and investigated vulnerabilities and potential security risks presented in a smart toy, as well as the similarity and correlation between already analyzed devices.

As shown during the static and dynamic analysis of the smart toy, there are multiple issues that are present in a commercial smart device, which can potentially be exploited as well as sensitive information be exposed. The inexperience of developers when considering security especially related to smart devices is an important factor that contributes to the lack of security awareness. Furthermore, smart toys and IoT devices in general have restricted resources and limited capabilities which are challenging developers.

The smart device is commercially available as of May 2021 through e-commerce platforms such as Amazon. The investigated smart toy has several security risks that are fundamental in nature. Unencrypted requests using HTTP as well as user identification token that is not refreshed possibly indicate lack of expertise and inexperience of developers, related to following appropriate security practices during code creation.

In the case presented during the dynamic analysis, we simulated an attacker intercepting the GET or POST requests that were transmitted through the mobile application and the server. The prerequisite is that the user connects to an insecure Wi-Fi access point where a man-in-the-middle attack is being performed. Naturally, the fact that all communications are unencrypted makes it easier for the attacker to intercept and benefit from the obtained data. Other methods and approaches of malicious attacks could be further investigated in the future in order to identify and define additional security flaws.

Furthermore, the investigated smart toy does not have Wi-Fi capabilities and instead communicates with the mobile application through a Bluetooth connection. The dynamic and static analysis focuses solely on the smart device's accompanied mobile application. A future topic to investigate and explore further can be the investigation of security flaws focused on the Bluetooth communication between the smart toy and the mobile application.

This thesis focuses on analyzing a single smart device and based on the results it is evident that security practices are often neglected during a smart device's development thus it is necessary developers increase the priority of enforcing security in smart toys. However, further research is required on a variety of smart toys in order to create a concrete view related to smart toys' security infrastructure. Moreover, it would be beneficial to continue investigating the devices after they receive an update in order to evaluate and identify improvements to their security level.

As the number of released smart devices increases, the workload of manually auditing them becomes increasingly larger. Therefore, a topic for future work is to create an automated tool that will help audit the devices and identify security flaws. While there are available tools that perform evaluation of network traffic and source code there is a need for connecting results with a product's privacy policies. Thus, when the automated auditing tool recognizes security issues it can compare them with the defined smart toy's privacy policies and recognize privacy breaches. The development of such a tool would be a challenging task as many companies purposefully define policies vaguely. As such there is a need to map the privacy policies to valid security properties upon which the automated tool can be based and evaluate the related product.

Conclusively, there is still the need to enforce security protocols in smart toys and in IoT devices in general as potentially every smart device can be vulnerable in a variety of attacks. This is troublesome especially in the case of smart toys since because of the nature of the devices, the potentially leaked information can have a greater impact and implications since it is related to children. Therefore, smart toys manufacturers must closely cooperate with security experts and supervising parties in order to ensure robust and strict security protocols are being followed. Future and continued work and auditing is required in order to ensure the privacy policies of the abovementioned products are not breached as well as motivate smart toys manufacturers to continue striving for improved and more secure products which can safely educate and entertain children.

7 Bibliography

[1] R. M. Milteer, K. R. Ginsburg and D. A. Mulligan, "The Importance of Play in Promoting Healthy Child Development and Maintaining Strong Parent-Child Bond: Focus on Children in Poverty," *Journal of American Academy of Pediatrics,* 2011.

[2] L. G. d. Carvalho and M. M. Eler, "Security Tests for Smart Toys," in *Proc. 20th International Conference on Enterprise Information Systems,* 21-24 March 2018, Funchal, Madeira, Portugal, pp. 111-120, SciTePress Digital Library.

[3] J. Haynes, M. Ramirez, T. Hayajneh and M. Z. A. Bhuiyan, "A Framework for Preventing the Exploitation of IoT Smart Toys for Reconnaissance and Exfiltration," in *International Conference on Security, Privacy and Anonymity in Computation, Communication and Storage*. 12-15 Dec 2017, Guangzhou, China, pp. 581-592, Springer, Charm.

[4] G. Chu, N. Apthorpe and N. Feamster, "Security and Privacy Analyses of Internet of Things Children's Toys," *IEEE Internet of Things Journal,* vol. 6, no. 1, pp. 978-985,2019.

[5] C. Maple, "Security and privacy in the internet of things," *Journal of Cyber Policy,* pp.155-184, 2017.

[6] I. Lee and K. Lee, "The Internet of Things (IoT): Applications, investments, and challenges for enterprises," *Business Horizons,* vol. 58, no. 4, pp. 431-440, 2015.

[7] J. Gubbi, R. Buyya, S. Marusic and M. Palaniswami, "Internet of Things (IoT): A vision, architectural elements, and future directions," *Future Generation Computer Systems,* vol.29, no. 7, pp. 1645-1660, 2013.

[8] T. Dillon, C. Wu and E. Chang, "Cloud Computing: Issues and Challenges," in *24th IEEE International Conference on Advanced Information Networking and Applications*, 20-23 April 2010, Perth, WA, Australia, pp. 27-33, IEEE.

[9] D. Craigen, N. Diakun-Thibault and R. Purse, "Defining Cybersecurity," *Technology Innovation Management Review,* vol. 4, no. 10, pp. 13-21, 2014.

[10] R. v. Solms and J. v. Niekerk, "From information security to cyber security," *Computers & Security,* vol. 38, no. Cybercrime in the Digital Economy, pp. 97-102, 2013.

[11] M. Abomhara and G. M. Køien, "Cyber Security and the Internet of Things: Vulnerabilities, Threats, Intruders and Attacks," *Journal of Cyber Security and Mobility,*vol. 4, no. 1, pp. 65-88, 2015.

[12] W. Zhou, Y. Jia, A. Peng, Y. Zhang and P. Liu, "The Effect of IoT New Features on Security and Privacy: New Threats, Existing Solutions, and Challenges Yet to Be Solved,"*IEEE Internet of Things Journal,* vol. 6, no. 2, pp. 1606 - 1616, 2018.

[13] Y. Yang, L. Wu, G. Yin, L. Li and H. Zhao, "A Survey on Security and Privacy Issues in Internet-of-Things," *IEEE Internet of Things Journal,* vol. 4, no. 5, pp. 1250 - 1258, 2017.

[14] W. Trappe, R. Howard and R. S. Moore, "Low-Energy Security: Limits and Opportunitiesin the Internet of Things," *IEEE Security & Privacy,* vol. 13, no. 1, pp. 14-21, 2015.

[15] L. Zhao, G. Li, B. D. Sutter and J. Regehr, "ARMor: fully verified software fault isolation,"in *EMSOFT '11: Proceedings of the ninth ACM international conference on Embedded software*, October 2011 Taipei, Taiwan, pp 289-298, Association for Computing Machinery.

[16] H. Shafagh, A. Hithnawi, A. Droescher, S. Duquennoy and W. Hu, "Poster: Towards Encrypted Query Processing for the Internet of Things," in *MobiCom '15: Proceedings ofthe 21st Annual International Conference on Mobile Computing and Networking*, September 2015, Paris, France, pp 251-253, Association for Computing Machinery.

[17] R. Kotamsetty and M. Govindarasu, "Adaptive Latency-Aware Query Processing on Encrypted Data for the Internet of Things," in *25th International Conference on Computer Communication and Networks (ICCCN)*, 1-4 August 2016, Waikoloa, USA, pp. 1-7, IEEE.

[18] H. Liu, C. Li, X. Jin, J. Li, Y. Zhang and D. Gu, "Smart Solution, Poor Protection: An Empirical Study of Security and Privacy Issues in Developing and Deploying Smart Home Devices," in *IoTS&P '17: Proc. 2017 Workshop on Internet of Things Security and Privacy*, November 2017, Dallas, USA, pp 13-18, Association for Computing Machinery.

[19] J. Zaddach, L. Bruno, A. Francillon and D. Balzarotti, "Avatar: A Framework to Support Dynamic Security. Analysis of Embedded Systems' Firmwares", in *Network and Distributed System Security Symposium,* 23-26 February, San Diego, California, USA, ResearchGate.

[20] D. D. Chen, M. Egele, M. Woo and D. Brumley, "Towards Automated Dynamic Analysisfor Linux-based Embedded Firmware," in *NDSS Symposium*, 21-24 February, San Diego, California, USA, ResearchGate.

[21] S.Sicari, A.Rizzardi, L.A.Grieco and A.Coen-Porisini, "Security, privacy and trust in Internet of Things: The road ahead," *Computer Networks,* vol. 76, pp. 146-164, 2015.

[22] A. Witkovski, A. Santin, V. Abreu and J. Marynowski, "An IdM and Key-Based Authentication Method for Providing Single Sign-On in IoT," in *2015 IEEE Global Communications Conference (GLOBECOM)*, 6-10 December 2015, San Diego, California, USA, pp. 1-6, IEEE.

[23] A. Fongen, "Identity Management and Integrity Protection in the Internet of Things," in *2012 Third International Conference on Emerging Security Technologies*, 5-7 September 2012, Lisbon, Portugal, pp. 111-114, IEEE.

[24] Y. Zhang, S. Kasahara, Y. Shen, X. Jiang and J. Wan, "Smart Contract-Based Access Control for the Internet of Things," *IEEE Internet of Things Journal,* vol. 6, no. 2, pp. 1594- 1605, 2019.

[25] S. Gusmeroli, S. Piccione and D. Rotondi, "IoT Access Control Issues: A Capability Based Approach," in *2012 Sixth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing*, 4-6 July 2012, Palermo, Italy, pp. 787-793, IEEE.

[26] T. Yu, V. Sekar, S. Seshan, Y. Agarwal and C. Xu, "Handling a trillion (unfixable) flaws on a billion devices: Rethinking network security for the Internet-of-Things," in *HotNets-XIV: Proceedings of the 14th ACM Workshop on Hot Topics in Networks*, November, 2015,

Philadelphia PA, USA, pp. 1-7, Association for Computing Machinery

[27] Y. J. Jia, Q. A. Chen, S. Wang, A. Rahmati, E. Fernandes, Z. M. Mao and A. Prakash, "ContexIoT: Towards Providing Contextual Integrity," in *NDSS Symposium*, San Diego,2017.

[28] A. Jacobsson and P. Davidsson, "Towards a model of privacy and security for smart homes," in *2015 IEEE 2nd World Forum on Internet of Things (WF-IoT)*, 14-16 December 2015, Milan, Italy,pp. 727-732, IEEE.

[29] P. P. Gaikwad, J. P. Gabhane and S. S. Golait, "A survey based on Smart Homes system using Internet-of-Things," in *2015 International Conference on Computation of Power, Energy, Information and Communication (ICCPEIC)*, 22-23 April 2015, Chennai, India, pp. 330-335, IEEE.

[30] M. Wang, G. Zhang, C. Zhang, J. Zhang and C. Li, "An IoT-based appliance control system for smart homes," in *2013 Fourth International Conference on Intelligent Controland Information Processing (ICICIP)*, 9-11 June 2013, Beijing, China, pp. 744-747, IEEE.

[31] N. Manworren, J. Letwat and O. Daily, "Why you should care about the Target data breach," *Business Horizons,* vol. 59, no. 3, pp. 257-266, 2016.

[32] J. Valente and A. A. Cardenas, "Security & Privacy in Smart Toys," in *IoTS&P '17: Proc. 2017 Workshop on Internet of Things Security and Privacy*, November 2017, Dallas, Texas, USA, pp. 19-24, Association for Computing Machinery.

[33] L. G. d. Carvalho and M. M. Eler, "Security Requirements for Smart Toys," in *Proc. 19th International Conference on Enterprise Information Systems*, 26-29 April 2017, Porto, Portugal, pp. 144-154, SciTePress Digital Library.

[34] S. Shasha, M. Mahmoud, M. Mannan and A. Youssef, "Playing With Danger: A Taxonomy and Evaluation of Threats to Smart Toys," *IEEE Internet of Things Journal,*vol. 6, no. 2, pp. 2986 - 3002, 2019.

[35] A. Marback, H. D. K. He, S. Kondamarri and D. Xu, "A threat model-based approach to security testing," *Journal of Software: Practice and Experience,* vol. 43, no. 2, pp. 241-258, 2013.

[36] B. Yankson, F. Iqbal and P. C. K. Hung, "Privacy Preservation Framework for Smart Connected Toys," in *Computing in Smart Toys*, Springer, 2017, pp. 149-164.

[37] J.-S. Pleban, R. Band and R. Creutzburg, "Hacking and securing the AR.Drone 2.0 quadcopter: investigations for improving the security of a toy," in *Proceedings Volume 9030, Mobile Devices and Multimedia: Enabling Technologies, Algorithms, and Applications*, 2-6 February 2014, San Francisco, USA, SPIE Digital Library.

[38] N. Kshetri and J. Voas, "Cyberthreats under the Bed," *Computer,* vol. 51, no. 5, pp. 92-95, 2018.

[39] "mitmproxy," 2020. [Online]. Available: https://mitmproxy.org/. [Accessed 20 July 2020].

[40] "Mobile Security Framework," 2020. [Online]. Available: https://github.com/MobSF/Mobile-Security-Framework-MobSF. [Accessed 10 August 2020].

[41] "MobSF Requirements," [Online]. Available: https://mobsf.github.io/docs/#/requirements. [Accessed 10 August 2020].

[42] "New Android vulnerability allows attackers to modify apps without affecting their signatures," 13 November 2017. [Online]. Available: https://www.guardsquare.com/en/blog/new-android-vulnerability-allows-attackers-modify-apps-without-affecting-their-signatures. [Accessed 10 August 2020].

[43] A. l. o. o. panelEricThompson, "MD5 collisions and the impact on computer forensics," *Digital Investigation,* vol. 2, no. 1, pp. 36-40, 2005.

[44] "Bifocals: Analyzing WebView Vulnerabilities in Android Applications," in *Information Security Applications*, International Workshop on Information Security Applications, 19-21 August 2013, Jeju Islands, Korea (Republic of), pp. 138-159, Springer.

[45] D. Shin, H. Yao and U. Rosi, "Supporting visual security cues for WebView-based Android apps," in *SAC '13: Proc. 28th Annual ACM Symposium on AppliedComputing*, March 2013, Coimbra Portugal, pp 1867-1876, Association for Computing Machinery.

[46] T. Luo, H. Hao, W. Du, Y. Wang and H. Yin, "Attacks on WebView in the Android system," in *ACSAC '11: Proceedings of the 27th Annual Computer Security ApplicationsConference*, December 2011, Orlando Florida, USA, pp. 343-352, Association for Computing Machinery.

[47] I. F. Elashry, O. S. F. Allah, A. M. Abbas and S. El-Rabaie, "A new diffusion mechanism for data encryption in the ECB mode," in *2009 International Conference on Computer Engineering & Systems*, 14-16 December, Cairo, Egypt, pp. 288-293, IEEE.

[48] D. Fizzotti, "Running a man-in-the-middle proxy on a Raspberry Pi 3," [Online]. Available: https://www.dinofizzotti.com/blog/2019-01-09-running-a-man-in-the-middle-proxy-on-a-raspberry-pi-3/. [Accessed 20 August 2020].

[49] R. Droms, "Automated configuration of TCP/IP with DHCP," *IEEE Internet Computing,* vol. 3, no. 4, pp. 45 - 53, 1999.

[50] "dhcpcd - ArchWIki," [Online]. Available: https://wiki.archlinux.org/index.php/dhcpcd. [Accessed 20 August 2020].

[51] J. Wu, M. Yang and T. Luo, "PACS: Pemission abuse checking system for android applictions based on review mining," in *2017 IEEE Conference on Dependable and SecureComputing*, 7-10 August 2017, Taipei, Taiwan, pp. 251.258, IEEE.

[52] M. A. Jutail, M. Al-Akhras and A. Albesher, "Associated Risks in Mobile Applications Permissions," *Journal of Information Security,* vol. 10, no. 2, pp. 69-90, 2019.

[53] C. Ren, Y. Zhang, H. Xue and T. Wei, "Towards Discovering and Understanding Task Hijacking in Android," in *24th USENIX Security Symposium*, 12-14 August 2015, Washington, USA, pp. 945-959, USENIX.

[54] "App security best practices," [Online]. Available: https://developer.android.com/topic/security/best-practices. [Accessed 20 August 2020].

[55] J. Burns, "Developing secure mobile applications for Android" iSEC Partners, 2008 [Online]. Available: https://research.nccgroup.com/wp-content/uploads/2020/07/isec_securing_android_apps.pdf [Accessed 02 May 2021].

[56] R. Nuseibeh, "Android WebView: Are Secure Coding Practices Being Followed?," 20 December 2018. [Online]. Available: https://www.checkmarx.com/blog/android-webview-secure-coding-practices/. [Accessed 20 August 2020].