Internet of Things 22 (2023) 100707

Contents lists available at ScienceDirect

Internet of Things

journal homepage: www.elsevier.com/locate/iot

Research article

PETIoT: PEnetration Testing the Internet of Things

Giampaolo Bella [a,*], Pietro Biondi [a], Stefano Bognanni [b], Sergio Esposito [c]

[a] Dipartimento di Matematica e Informatica, Università degli Studi di Catania, Catania, Italy
[b] Cybersecurity Division, Leonardo S.p.A., Catania, Italy
[c] Department of Information Security, Royal Holloway University of London, Egham, UK

ARTICLE INFO

Keywords:
Cyber Kill Chain
Vulnerability assessment
VAPT
Ethical hacking
Attack
Exploitation
Fix
Offensive security

ABSTRACT

Attackers may attempt exploiting Internet of Things (IoT) devices to operate them unduly as well as to gather personal data of the legitimate device owners'. Vulnerability Assessment and Penetration Testing (VAPT) sessions help to verify the effectiveness of the adopted security measures. However, VAPT over IoT devices, namely VAPT targeted at IoT devices, is an open research challenge due to the variety of target technologies and to the creativity it may require. Therefore, this article aims at guiding penetration testers to conduct VAPT sessions over IoT devices by means of a new cyber Kill Chain (KC) termed PETIoT. Several practical applications of PETIoT confirm that it is general, while its main novelty lies in the combination of attack and defence steps. PETIoT is demonstrated on a relevant example, the best-selling IP camera on Amazon Italy, the TAPO C200 by TP-Link, assuming an attacker who sits on the same network as the device's in order to assess all the network interfaces of the device. Additional knowledge is generated in terms of three zero-day vulnerabilities found and practically exploited on the camera, one of these with High severity and the other two with Medium severity by the CVSS standard. These are camera Denial of Service (DoS), motion detection breach and video stream breach. The application of PETIoT culminates with the proof-of-concept of a home-made fix, based on an inexpensive Raspberry Pi 4 Model B device, for the last vulnerability. Ultimately, our responsible disclosure with the camera vendor led to the release of a firmware update that fixes all found vulnerabilities, confirming that PetIoT has valid impact in real-world scenarios.

1. Introduction

The Internet of Things (IoT) has revolutionised the current technological landscape. It has changed people's uses of traditional devices and has, at the same time, sparked off the definition of new ones. It would have been otherwise impossible, for example, to operate kitchen appliances from a smartphone and to conceive a smart plug. The IoT has virtually no boundaries. Notably, it spans over the application domain of *Cyber–Physical Systems* (CPSs) and that of Information-Operation Technology (IT/OT). While CPSs include layman's devices and applications such as those in the automotive and healthcare areas, IT/OT concerns industrial systems, where manufacturing and production are increasingly intertwining and becoming computer-controlled.

Correspondingly with a rise in pervasiveness, the IoT has experienced a rise in terms of cybersecurity and privacy threats. Once the devices are connected to the Internet, criminals gain a potential opportunity to abuse them and deny their services; for example, the Mirai malware built a botnet to compromise thousands of devices [1]. People's data is often the ultimate target of exploitation due to the fact that IoT devices, particularly CPSs, collect a variety of user preferences and generated data. A car will process data

[*] Corresponding author.
E-mail addresses: giamp@dmi.unict.it (G. Bella), pietro.biondi@phd.unict.it (P. Biondi), stefano.bognanni@leonardo.com (S. Bognanni), sergio.esposito.2019@live.rhul.ac.uk (S. Esposito).

https://doi.org/10.1016/j.iot.2023.100707
Received 3 October 2022; Received in revised form 24 January 2023; Accepted 24 January 2023
Available online 10 February 2023

such as cabin preferences and history of visited places and, in healthcare, an ultrasound machine will handle data such as personally identifiable information and health conditions. An additional surge of attacks has been reported during the COVID-19 pandemic [2].

The need to assess IoT devices from a cybersecurity and privacy standpoint is, therefore, strongly justified, hence the relevance of the contents of this article. One of the relevant approaches is to instruct a team of ethical hackers to carry out that assessment empirically, often impersonating criminals, albeit in a controlled environment. Such activities stand under the big umbrella of Vulnerability Assessment and Penetration Testing (VAPT), hence an ethical hacker is also addressed as a tester. There are various incarnations of VAPT, such as red–blue–purple teaming, white–black–grey approaches and social-engineering assessments, but these are not pivotal for the sequel of this manuscript. VAPT sessions happen repeatedly over IoT devices, for example, at device manufacturing time if the manufacturer follows a Secure Software Development Lifecycle. Alternatively, they may be conducted by potential buyers of the device with the aim of operating a market choice on the basis of the security guarantees that the device offers.

Given the importance of VAPT in the IoT domain, this article poses the following research questions: *How to approach a VAPT session with ethical purposes over IoT devices? Do approaches exist that could be leveraged profitably? And what tailoring would these require?* These questions are relevant because VAPT is challenging in general, requiring creativity beyond sheer competence, but is all the more challenging, in particular, over IoT devices due to the domain size and variety of adopted technologies. In its aim to address the research questions stated above, this article makes the research hypotheses that VAPT is not fully unstructured as a process, and that it is possible to distil out and frame its fundamental steps specifically towards the IoT. This is done through the formalisation of a Kill Chain (KC), each KC being a comprehensive description of the various steps forming an attack.

1.1. Contributions

This article demonstrates how to go about VAPT over IoT devices and, at the same time, demonstrates that existing, general approaches are useful but require appreciable rethinking. This article introduces PETIoT, a new cyber Kill Chain (KC) to conduct VAPT sessions over IoT devices. PETIoT is slim, so that testers may begin to apply and experience it quickly. The value of PETIoT is to enable the penetration tester who approaches the IoT domain with a clearly defined roadmap to conduct her experiments. As a limitation, it must be stressed that, while a KC is meant to spell out the steps to take to conduct the activities, it does not prescribe the very tools to use through each step and, most importantly, does not teach how to use them. A few basic tools will be suggested below as examples for use during the demonstration of the KC, but it is firm that many alternatives exist and that the tester may use their favourite ones.

By applying PETIoT using basic freeware tools only, we found three vulnerabilities on the TP-Link TAPO C200, the best-seller domestic camera on Amazon Italy at the time of the research. The vulnerabilities discussed in this article can be briefly outlined and evaluated (according to CVSS [3]) as follows:

- Vulnerability 1 — Improper neutralisation of inbound packets allows complete Denial of Service. It means that the tester can DoS the camera and make it unusable. This vulnerability has severity 6.5, that is, Medium. We exploited it by an intensive scan through a vulnerability scanner.
- Vulnerability 2 — Insufficient entropy in encrypted notifications allows breach of motion detection. It means that the tester can infer whether the camera detects motion despite the fact that the corresponding notifications are encrypted. The tester can also drop the relevant packets, thereby causing a denial of the motion detection service. This vulnerability has severity 5.4, that is, Medium. We exploited it by TLS packet inspection in a dedicated experiment.
- Vulnerability 3 — Cleartext transmission of video stream allows breach by unintended actors. It means that the tester can intercept and interpret the full video stream if the camera users run third-party video players. This vulnerability has severity 8.8, that is, High. We exploited it by traffic eavesdropping in a dedicated experiment. We also prototype a fix for this vulnerability by leveraging an inexpensive Raspberry Pi 4 Model B device [4] to transmit the video stream through a cryptographic tunnel.

The three vulnerabilities were zero-day vulnerabilities when we found them because they had not been reported before. During the responsible disclosure process, the vendor acknowledged all issues discussed in this article and followed our advice on how to thwart them; the current firmware release addresses all three vulnerabilities.

In summary, this manuscript answers its research questions by making the following contributions:

- an account on the relevant existing KCs (Section 2.1);
- an original tailoring of existing KCs into a new KC that combines attack and defence for the first time and is specific for PEnetration Testing the IoT, here termed PETIoT[1] (Section 3);
- a practical demonstration of PETIoT during a VAPT session on a relevant, best-seller device, the TP-Link TAPO C200 with firmware 1.0.16 (Section 5);
- a discovery of three unknown vulnerabilities on the camera (one of High severity and two of Medium severity), their exploitation in an experimental environment, their responsible disclosure with the vendor, the design of their fixes and a proof of concept of the most complicated of the fixes, which requires a cryptographic tunnel (Section 5.4);

[1] The penetration tester's paradox of attacking for the ultimate goal of defending echoes the paradoxical life of Marcel André Henri Félix Petiot, who was a doctor as well as a serial killer. Hence the name of our Cyber Kill Chain.

- ultimately, a compilation of lessons learned and recommendations contributing to strengthen the general cybersecurity posture in the IoT domain (Section 6).

1.2. Generality

PETIoT has taken a long time to shape up as the way it is defined below. It has been gestated over years of experiments with several IoT devices, starting with printers [5] and continuing with VoIP phones [6,7]. It guided VAPT efforts over modern voice personal assistants, notably favouring the discovery of the AvA attack on Amazon Echo Dot [8]. PETIoT was also used over several IP cameras, including Netvue Orb Mini, Netvue Vigil Cam, Ctronics CTIPC-380C-4MP and TP-Link TAPO C200, with preliminary results made public about the TAPO C200 [9].

This article conceptualises and describes PETIoT fully for the first time. To provide a relevant and complete demonstration of the practical application of our KC, the article adopts it during a VAPT session against the current best-seller IP camera on Amazon Italy and discusses the complete findings. PETIoT was born by abstraction and generalisation on top of the innumerable vulnerability assessment and penetration testing experiments conducted so far; also, as a KC, its constituent steps are inherently broad. Therefore, PETIoT is general *by design*.

1.3. Novelty

The contributions described above are novel at least in two ways. One concerns the very findings on the Tapo C200, which were already mentioned above. Not only were the three vulnerabilities unknown but, remarkably, we shall see how they arise thanks to focus of PETIoT on the device with its network environment rather than solely on the device as a target to intrude into.

The second element of novelty concerns the methodological approach focusing on ethical hacking, so that PETIoT deliberately aims at guiding VAPT activities more than it seeks out to represent known attacks. In consequence, our KC distinctively culminates with a step devoted at fixing the vulnerabilities that may have been found and exploited. On one hand, this is innovative in the KC domain, whose focus normally is on the offensive steps and solely on those, as is the case, to just advance an example, with the popular Mitre ATT&CK [10], while Mitre D3FEND [11] is the only relevant KC for the defensive activities. On the other hand, PETIoT combines attack and defence because this is an obvious necessity for the penetration tester, who cannot complete her reports without detailing clear fixing or mitigation activities for the exploited vulnerabilities.

1.4. Article structure

This manuscript continues by presenting the related work (Section 2). It then introduces PETIoT (Section 3) and the basics information and toolkit to apply it in practice (Section 4). It demonstrates PETIoT over the TAPO C200 (Section 5), derives lessons learned and recommendations (Section 6) and concludes (Section 7).

2. Related work

The related work can be conveniently partitioned into two groups. One concerns the KCs and the other one is about the security aspects of IP cameras.

2.1. Cyber kill chains

A historical account on KCs is due to Grant et al. [12]. However, at the time of this writing, the best established KCs can be summarised as follows:

- Lockheed Martin's Cyber Kill Chain [13] is the pioneering representative. It sets the scene in 2011 with the 7 essential steps of Reconnaissance, Weaponization, Delivery, Exploitation, Installation, Command and Control, Actions on Objectives.
- Mitre ATT&CK [10] is a subsequent KC offering a finer-grained description of an attack over 14 steps. Most importantly, it is enhanced with a knowledge base of relevant techniques.
- Unified Kill Chain [14] is rather modern, dating back to 2017. It spells out 18 steps, grouped as the three essential macro-steps Initial Foothold, Network Propagation, Action on Objectives.
- Expanded Cyber Kill Chain [15] extends Lockheed Martin's with an Internal Kill Chain and a Target Manipulation Kill Chain. It thus offers finer and finer detail.
- Mitre D3FEND [11] "*is at an early stage and is an experimental research project*". It aims at expressing the defensive steps against attacks.

Over the years, the trend is clear towards specialising the various steps of Lockheed Martin's CKC further, with the result that KCs become more detailed, hence longer. While detail itself may be useful in general, it may turn out an overkill in certain scenarios, sometimes even reaching redundancy, as it can be argued when KCs are spelled out in the IoT domain. The reason is an architectural one. Precisely, because the VAPT target is the IoT device, the latter must be made reachable to the tester, otherwise the activities would be aiming at a different target, such as penetrating the network that hosts the device. A *physical reach* means that the device is physically given to the tester, who may then interact by all available device interfaces, such as Ethernet, Wi-Fi, Bluetooth, etc. A *logical reach* means that the device is only made logically accessible to the tester by means of some interface, typically by means of an IP address. In general, a physical reach allows for a broader test, covering all interfaces, while this is harder to reproduce by a logical reach only. Also, a physical reach normally implies a logical reach, but the opposite does not hold.

In consequence, there appears to be some redundancy through the application of Lockheed Martin's CKC to the IoT domain. For example, Reconnaissance as such, which is aimed at identifying and selecting targets, is void because the target is given, hence this step reduces to the sheer gathering of information about the target, for example through network scanning. Also, Weaponization seems redundant because a basic toolkit may suffice to tackle innumerable devices, as we shall see below. We shall see that thwarting these sources of redundancy contributes to PETIoT slimness.

Moreover, with VAPT over IoT devices as its goal, PETIoT focuses on the given device with its network environment, which often sees traffic with some supporting server or cloud as well as a companion software app to extend the minimal user interface of the device. Because of this, also the Installation and Command and Control steps of Lockheed Martin's CKC are unnecessary, upon the basis that no software needs to be maliciously installed and operated as in the case of intrusions into the device. Also a reduction of these sources of redundancy will keep PETIoT slim. We do not believe that the more recent and detailed KC listed above would add significant value to VAPT activities over IoT devices.

Finally, two more works are related. One, of 2018, tailors the KC model to multimedia service with its network environments [16], but it seems to lack suitable practical demonstration. A more recent one, of 2020, derives a novel KC from a large number of IoT attacks [17]; while this work is full of insights and relevant information, it seems specifically oriented at the creation of zombie devices towards DDoS.

2.2. Security of IP cameras

The related work concerning IP camera security hosts a number of items, because it has been shown that IP cameras may suffer multiple security issues, including multimedia security, network security and cloud security. However, only a few works are closely related to the contents of this article.

In 2015, Tekeoglu and Tosun looked into the security of cloud-based wireless IP cameras [18]. They studied the traffic generated by a wireless IP camera that is easy to set up for the average home user. Hence, they proved that an attacker can sniff the IP camera's network traffic and would be able to reconstruct the JPEG images from the data stream. Based on their information, their system has some limitations, e.g. their script only reconstructed 253 JPEG images over about 20 h of video track. This work relates to ours in its focus on the device with its network environment, but its aims are more device-specific than ours of a general KC.

In 2017, Liranzo and Hayajneh presented an analysis of the main privacy and security issues affecting thousands of IP camera consumers globally [19]. They proposed a series of recommendations to help protect consumers' IoT devices in the intimacy of their houses, including security the (remote) connection between the smartphone running the video player and the camera. This is coherent with the observations and the findings reported in the present article. Recently, Abdalla and Varol tackled an IP camera with similar aims to ours [20], but only contributed an assessment of basic vulnerabilities such as the use of default passwords. This and similar vulnerabilities would be captured during the central steps of PETIoT.

The Master thesis mentioned above [21] is of 2022 and targets the TAPO C200 too. It experiments with the Heartbleed vulnerability affecting the camera firmware up to version 1.0.10, which is prior to version 1.0.16 of our experiments. It also demonstrates an intrusion into the camera due to CVE-2021-4045 vulnerability, which affects firmware up to version 1.1.15, hence also the version of our experiments. However, that vulnerability was only published in March 2022, while our experiments date back to a year before, hence our automated scanners could not report it because they were lacking the relevant signature. With its focus on the device with its network environment, the sequel of this article demonstrates that the use of PETIoT unveils additional, unknown vulnerabilities.

3. Defining PETIoT, a cyber kill chain for the IoT

KCs can be understood as a comprehensive description of the various steps that must be followed to carry out some offensive activities. Our research started by considering existing KCs and continued by assessing how to leverage them towards answering our research questions. As a result, PETIoT was defined in a way to provide those answers and, therefore, it now guides the penetration tester's VAPT activities in the IoT domain. In consequence:

- **PETIoT spells out VAPT activities.** While we are used to KCs that only describe the steps of an intrusion or, separately, the steps to defend against that, PETIoT combines both for the first time simply because it aims at guiding VAPT activities. These always culminate with reports that not only describe prototypical exploitations of the vulnerabilities that were found, but also explain how to mitigate in practice if not entirely fix them.
- **PETIoT focuses on the IoT device.** Existing KCs bear some redundancy when applied to the IoT domain due to the fact that the target IoT device is made reachable to the tester and most of the needed tools are consolidated. Therefore, redundancy is appreciable in the early steps of Reconnaissance, which reduces to information gathering about the device, and Weaponization, which merely leverages a consolidated, basic toolkit.
- **PETIoT also focuses on the network environment.** While most KCs spell out intrusions into the target,
none pay sufficient attention to the environment, which is particularly relevant when the target is an IoT device, hence has traffic with supporting server or cloud and companion app. So, no offensive steps are needed after Exploitation because there is no software to operate remotely, while Actions on Objectives happens during Exploitation.

PETIoT prescribes the following 6 steps:

1. **Experiment Setup.** The tester is granted *physical reach* to the IoT device to test, namely is given the device itself, or merely *logical reach* to it, namely is provided with the device address such as an IP or a Bluetooth access. Reach is a plausible assumption because the target of the test is the device itself with its network environment, coherently with the scope of PETIoT. Appropriate setups follow, depending on the specific network interface to use to reach the device. The tester has to install the device in a realistic environment to enable herself to probe it by the relevant toolkit. The setup is termed *non-invasive* when the device is installed and not tampered with. It is *invasive* when the tester disassembles parts of the device with the aim of getting hold of relevant information that the device would never mean to transmit, such as a private key or a whitelist.
2. **Information Gathering.** Similarly to other KCs, the tester attempts to acquire as much information as possible about the functioning of the device, also to unveil potentially hidden features. In a *black-box session*, that is, a testing session in which the tester has no information on the target environment at the beginning, this is normally achieved by probing the device repeatedly through the setup prepared before. For example, crucial tests aim at understanding the use of cryptography, its type and working assumptions such as the protection of a cryptographic key into some Hardware Security Module.
   A further capability, although exclusive for *white-box sessions*, is that the tester may also assess source codes to derive additional information geared, as always, at identifying possible vulnerabilities. In fact, during a white-box session, the tester has usually access to all details regarding network, coding, policies, etc. The tester uses a standard toolkit (recalled below) to intercept, interpret and, if needed, decipher traffic in a passive (*eavesdropping*) or in an active (*man-in-the-middle)* fashion.
3. **Traffic Analysis.** It is always worthwhile to attempt to distill out the entire network functioning of the device. This effort may be interpreted as a reiteration of the previous step through structured experiments aimed at combining into the full picture the various insights gained about the network traffic. For example, a successful Traffic Analysis over the networking activity yields a sequence diagram detailing the exchanges that occur among all relevant participants. However, it is clear that this step may not always succeed depending on the intricacies of the functioning of the target device as well as on its implementation of dedicated protection measures such as the use of randomised response mechanisms and of advanced cryptographic primitives.
4. **Vulnerability Assessment.** Once information is available about the target device, the tester can move on to study, mechanically or manually, whether vulnerabilities exist. It means that, while standard scanners such as OpenVAS, Nessus Pro and Nmap with scripting offer valid help, the tester may need to formulate parallel conjectures or develop special scripts depending on the type of target. For example, standard scanners may not come with software connectors for all possible target architectures. Vulnerabilities should be formulated as scenarios that would break at least the essential security properties of confidentiality, integrity and availability. The tester should be prepared to hunt for vulnerabilities at any architectural level, including hardware and software, and, in particular, at any level of the communication protocols.
5. **Exploitation.** Not all vulnerabilities can be practically exploited, and this step aims at verifying whether that is the case. Exploitation perhaps is the step that requires most of the tester's creativity and stubbornness. It unfolds through repeated trials whereby the tester takes a rather empirical approach aimed at verifying whether a trial succeeds. For example, only specific payloads may lead to success, while all others may not. Exploitation is also the core and main target of the entire VAPT activity. The white-box approach or a successful Traffic Analysis following the black-box one may favour this step because the tester would be more easily driven through precise trials. Following the stated focus on the IoT device environment, all offensive activities terminate in this step.
6. **Fixing.** Once the penetration tester successfully attacks the target device by the exploitation of a vulnerability, she must continue with an assessment on how to fix the attack. Fixes could be of *preventative nature* so that, once they get deployed over all similar devices on the large-scale production level, the vulnerability will no longer affect neither the future devices, nor the updated ones. When preventative fixes are not possible, the tester might revert to engineering fixes of *notification nature*, which aim at informing the device user when a scenario that may suffer the attack occurs — in the hope that the user will know how to find secure ways to face that scenario. For example, if a browser cannot fully verify the certification of the displayed URL up to a trusted root authority, it notifies its users of that because it was impossible to fully prevent the risk of a man-in-the-middle attack (since the certificate does not chain back to a valid root certificate).

4. Basic toolkit and information

This Section recalls a well-known, basic toolkit in support of any VAPT activity, then continues by outlining the basic information that is useful to follow the sequel of the manuscript. We consider a basic toolkit to include the following software tools:

- Ettercap [22] is a suite of tools for Man-in-the-Middle attacks. It can be used to capture network packets, filter contents, dissecting protocols, etc. In this work, we will use it to eavesdrop between the Tapo application and the Tapo camera;
- Nessus [23] is a vulnerability scanner. It is generally used to assess the level of security of a node, or of an entire network, against known attacks and vulnerabilities. In this work, we will use it to perform a vulnerability assessment on the TAPO camera.
- Nmap [24] is primarily an utility for network discovery. It can rapidly scan large network to discover running hosts and services. In this work, we will use it to gather information about the services exposed by the camera;

- SSL packet capture [25] is a strong debugging tool that allows to intercept encrypted traffic for subsequent analysis. In this work, we will use it to interpret SSL traffic to and from the Tapo application;
- IPtables [26] is an administration tool used for packet filtering and NAT. It allows an administrator to manage firewall rules in the Linux kernel, so that packets with certain characteristics are all handled correctly. In this work, we will use it to actively interpose between the Tapo application and the Tapo camera;
- Wireshark [27] is network protocol analyser that allows to capture and analyse network packets, following streams such as TCP or HTTP, and decrypt protocols such as IPSec and TLS. In this work, we will use it to convert network traffic into playable video via the H264 extractor extension.

Having seen the relevant tools in support of VAPT, further information is necessary to understand the activities conducted below to demonstrate PETIoT. As a start, the ability to appeal to the video streaming services may be gained by using the open source, multi-platform media player VLC [28] and the iSpy DVR [29]. Consolidated protocols exist to make the streaming work. An essential protocol to integrate the camera with third-party infrastructures and systems, such as the Real Time Streaming Protocol (RTSP) [30]. IT is a network protocol used for video streaming and to orchestrate the exchange of media, such as audio and video, across the network. RTSP extends the RTP [31] and RTCP [32] protocols by adding the necessary directives for streaming, such as:

- Describe, used by the client to obtain information about the desired resource.
- Setup, used to specify how a single media stream is to be transported.
- Play/Pause, used to start or pause video playback.
- Record, used to store a stream.

Further relevant information comes from the Open Network Video Interface Forum (ONVIF), whose aim is to promote compatibility of video surveillance equipment so that devices made by different companies can interoperate. ONVIF provides an interface for communication with different devices through the use of various protocols. It also provides a few configuration profiles to make the best use of the different technical features. In particular, profiles G, Q, S and T, are dedicated to video systems, however, the Tapo C200 is only compatible with Profile S [33]. Profile S is designed for IP-based video systems and involves the use of a compliant device (e.g. Tapo C200) and a compliant client (e.g. iSpy [29]) that can together configure, request and control video streaming. The most relevant features of Profile S concern user authentication, NTP support and H264 [34] audio and video streaming using RTSP.

One of the modern technologies to strengthen security protocols that are based on public-key cryptography is Certificate Pinning, which "*leverages knowledge of the pre-existing relationship between the user and an organisation or service to help make better security related decisions*" [35]. For example, this is useful for a mobile application meant to connect to its supporting server or cloud. Because such a relationship is known since design time, the server's public key certificate is preinstalled in the application so that the latter will not follow up to encrypted traffic from a different peer even if that traffic is correctly certified.

Once an attack is found, measuring its severity is useful for several reasons, notably to promptly bootstrap the applicable processes during the incident management and response and, with less urgency, to inform the periodical cybersecurity and data protection risk assessment. A widely established approach to measure attack severity is the Common Vulnerability Scoring System (CVSS) [3]. While a few incarnations of CVSS exist, the widely established 3.0 version requires the tester to assign parameters Attack Vector (AV), Scope (S), Attack Complexity (AC), Confidentiality (C), Privileges Required (PR) & Integrity (I), User Interaction (UI) and Availability (A). The system then returns a score ranging between 0 and 10, with the highest number indicating highest severity.

Finally, the Raspberry Pi 4 Model B device must be mentioned, a rather inexpensive device boasting a Broadcom BCM2711, Quad core Cortex-A72 (ARM v8) 64-bit SoC at 1.5 GHz and several network interfaces. In particular, its IEEE 802.11ac Wi-Fi and Gigabit Ethernet features will be used below. When needed, its network interface card will be turned into an access point by means of Hostapd [36].

5. Demonstrating PETIoT

PETIoT needs practical demonstration. This is already available through the previous experiments that were instrumental to define and sharpen the very steps of our KC. For example, Experiment Setup and Information Gathering were proven essential for the first time over printers and phones [6], while Traffic Analysis and Vulnerability Assessment turned out much larger in the application described below. Exploitation, in turn, was problematic over Amazon Echo Dot, a case in which the Fixing step inspired a new breed of preventative security measures that are still being tested [8].

To offer a complete demonstration of PETIoT in the present manuscript, we chose a relevant IoT device that supports a paradigmatic application of the kill chain. Of course, the list of possible candidates is never ending, but we decided that the best balance between inexpensiveness and popularity lies in the area of small, portable cameras such as those used for home surveillance, often IP cameras [37], and those typically used on vehicles, namely dash cameras [38]. With an estimate of around 770 million IP cameras used for CCTV in 2019 [39], we decided to search through Amazon Italy and chose the no.1 best seller in category "Dom camera", which includes domestic cameras. As shown in Fig. 1, it is the Tapo C200 by TP-Link, which costs approximately 30€ and totals over 38.000 rather positive reviews at the time of this writing.

The C200 is an entry-level IP camera released in 2020. Its technical specifications support several functionalities, such as night vision, resolution up to 3MP, motion detection, acoustic alarm, data storage, voice control, use with third-party software and webcam mode. We argue that such a popular target is ideal to demonstrate our kill chain, so the sequel of this article discusses how the

Bestseller in Dom camera
#1
TP-Link Telecamera Wi-Fi Interno, Videocamera sorveglianza 1080P, Visione Notturna, Audio Bidirezionale, Notifiche in…
38.122
29,98 €

Fig. 1. The Tapo C200 on Amazon Italy.

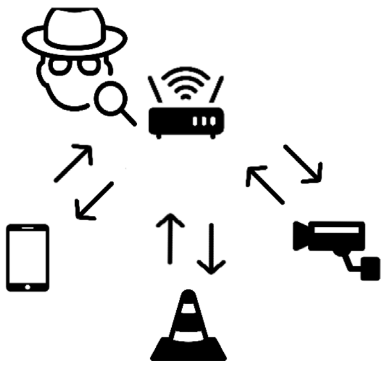


Fig. 2. The testbed for the TAPO C200.

main steps of PETIoT can be followed to conduct a VAPT session in practice on a real device. In this example, the tester is granted physical reach to the camera, conducts a black-box session of VAPT on it and concludes with a fix of preventative nature for one of the reported vulnerabilities.

5.1. *Experiment setup*

We begin by setting up our experiment over the C200. To operate the camera, a TP-Link account is required to enable access to the various cloud services available. Access to the TP-Link account is via the Tapo app, which is available free of charge on Google Play for devices running Android 4.4+ [40] and App Store [41] for those running IOS 9+. The application has a simple and intuitive user interface, from which it is possible to use and manage the devices of the Tapo family, including the Tapo C200. The range of services offered for the C200 includes: remote access and camera control; sharing the camera with other accounts; synchronisation of settings; integration of the camera with smart home systems; receipt of motion notifications.

Fig. 2 shows the testbed on which the experiments were conducted and how the various devices, including the attacker's machines are inter-connected via a Wi-Fi access point also operating as a switch. The testbed consists of:

- A Wi-Fi switch to which the various devices are connected.
- The Tapo C200 IP camera.
- A smartphone Oneplus 8T (Android 11, 8 Core, 8 GB RAM, 128 GB storage) running SSL Packet Capture and the Tapo application.
- A Linux machine on which the attacker runs Ettercap and Wireshark.
- A Linux machine on which the attacker runs the third-party players iSpy and VLC.

While the Tapo C200 works normally, our testbed enables us to monitor its network environment by means of the various tools with the aim of extracting useful information for operational analysis.

5.2. Information gathering

Once the experiment is setup and ready, we move on to information gathering. The camera is now integrated with third-party infrastructures and systems, hence we can probe it through repeated sessions of experiments to gather our initial knowledge about its functioning. The basic information and toolkit outlined above guide us to conduct such experiments through this step.

Generally, an Nmap scan is the first tool that is used as it gathers information on open ports and exposed services. The Nmap manual offers a large range of scanning options, but simple ones are suitable in this case, mostly because stealthiness is not a requirement. Ettercap enables the tester's machine to interpose between the Tapo application and the Tapo camera, then Wireshark can be used to derive a dynamic view of the exposed services, namely the traffic they generate and receive. Appeals follow to SSL packet capture to interpret traffic oriented at the external TAPO server, and to Nessus to learn about known vulnerabilities that are in place.

In consequence of all the above trials, we learn about the services exposed by the camera and about the specific protocols that are used to make them available. Moreover, we find out that when the Tapo application and the Tapo camera are on different networks, a classic client–server architecture is used in which the Tapo Server is used to distribute information, such as the device configuration. Conversely, when both are on the same network, no external servers are used to control the camera and access the video stream.

No known vulnerabilities are found by the automated Nessus scans conducted through this step. However, it is remarkable that three unknown ones are manually found later by combining our experience with the gathered information.

5.3. Traffic analysis

Our work on this step tailors information gathering to an understanding of the precise message sequences that flow between the user, the app and the camera. The Traffic Analysis effort is now partitioned depending on the software app that is used to stream from the camera, in particular on whether that software is the proprietary or third-party's.

5.3.1. Using the Tapo C200 through proprietary software

The sequence diagram in Fig. 3 describes the various exchanges that allow the user to access the video stream through the Tapo application. After choosing one of the initialised devices, the Tapo application logs into the Tapo C200, obtaining the *stok* token, and this is needed to perform the control and modification operations.

More precisely, the *stok* is a token generated during authentication of the application to the camera. First, the Tapo C200 authenticates the user by a username and password, then assigns the *stok* to the Tapo application. The purpose of the token is to create a session without having to re-authenticate the application every time the camera settings are changed. The various requests generated by the application carry the *stok* obtained in the last authentication phase. In this way the Tapo C200 only accepts requests from authenticated users.

Following the user's request to access the video stream, the Tapo application and the Tapo C200 coordinate to serve that. In particular, the Tapo C200 sends to the application a *nonce* that is necessary for the construction of a symmetric key. At this point, both parties calculate the initialisation vector and an AES key in order to later encrypt the video stream using AES in CBC mode. Once the key is calculated, the Tapo application authenticates itself to the Tapo C200 by providing a "response" tag. Finally, the Tapo C200, after verifying the validity of the response, starts the video streaming session by encrypting it with the agreed key. This completes the Traffic Analysis of the camera functioning when its proprietary app is used.

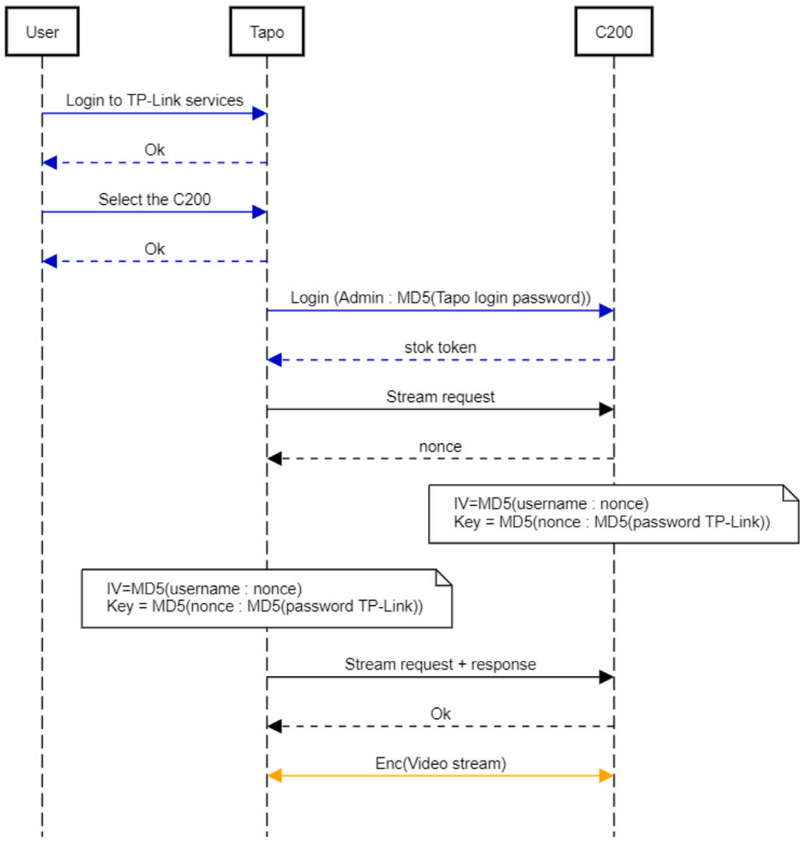
5.3.2. Using the Tapo C200 through third-party software

Fig. 4 describes the outcome of our Traffic Analysis efforts targeting the camera use via third-party software. Using the Tapo application, we first need to access the device and change its settings so that the camera can create a new user for third-party software. After that, the Tapo C200 works with the software pair mentioned above, namely iSpy and VLC. These are used to carry out a login on the camera so that it initiates a free-to-air media streaming session. In particular, the diagram shows that the *stok* token is forwarded again to the camera, due to the necessary creation of the dedicated user for the third-party app to access the camera.

This diagram is derived by mounting a man-in-the-middle attack to the camera and analyse the traffic while the other machine running VLC and iSpy are accessing the camera and receiving the video stream. More precisely:

- On VLC, select the "network resources" section of the player and enter:
  `rtsp://username:password@`
  `<tapoc200address>/stream/1`
- On iSpy, configure the ONVIF S profile to URI:
  `http://username:password@`
  `<tapoc200address>/onvif/`
  `device_service`

Since ONVIF's S profile does not add any functionality to the video stream, no differences emerged from the analysis of the traffic generated for the video stream of the two players. In particular, the packets analysed showed that, after access through the specific URI for ONVIF devices, iSpy uses RTSP in the exact way used by VLC. However, a remarkable difference is apparent by comparing the two diagrams discussed above, namely that, when third-party software engages with the camera, this starts an unencrypted video streaming session. It follows that such contents are intelligible to a man-in-the-middle attacker.


User
Tapo
C200
Login to TP-Link services
Ok
Select the C200
Ok
Login (Admin : MD5(Tapo login password))
stok token
Stream request
nonce
IV=MD5(username : nonce)
Key = MD5(nonce : MD5(password TP-Link))
IV=MD5(username : nonce)
Key = MD5(nonce : MD5(password TP-Link))
Stream request + response
Ok
Enc(Video stream)

⬛ Communication in plaintext
🟦 Communication over TLS channel
🟧 Stream video Encrypted with AES key

Fig. 3. Sequence diagram for using the IP camera with the proprietary Tapo application.

5.4. *Vulnerability assessment*

The sequence diagrams discussed above explain in great detail how the camera works, making a few significant features apparent. In particular, we note that no traffic filtering mechanism is there, hence what follows.

Vulnerability 1 – Improper neutralisation of inbound packets allows complete Denial of Service. The camera uses no mechanisms to filter traffic, hence it would honour any message request as soon as possible, without checking the genuineness of the caller and of the content. It means that an attacker could maliciously attempt to flood it with traffic and make it unavailable. This vulnerability is a specific instance of weakness "CWE-707: Improper Neutralisation" [42].

5.4.1. *Using the Tapo C200 through proprietary software*

Our vulnerability assessment then continues by tackling motion detection. This important feature works as follows:

1. The Tapo C200 detects some movement and sends a notification to the TP-Link server.
2. The server receives the notification and sends an alert to all devices connected to the account associated with the camera.
3. The application receives the message and generates a notification for the human.

When the Tapo application and Tapo C200 are not on the same network, security in terms of the confidentiality of the video stream is entrusted to the channel itself on which the stream flows. Using SSL/TLS on port 443, all data exchanged between the Tapo C200, the TP-Link server and the Tapo application is properly encrypted. In addition, the Tapo application and the Tapo C200 use certificate Pinning. It follows that, if information needs to be shared with the TP-Link server, no SSL/TLS session is established towards any entity other than a TP-Link server whose SSL/TLS certificate they already have by default. We find no vulnerabilities about the implementation of these cryptographic protocols.

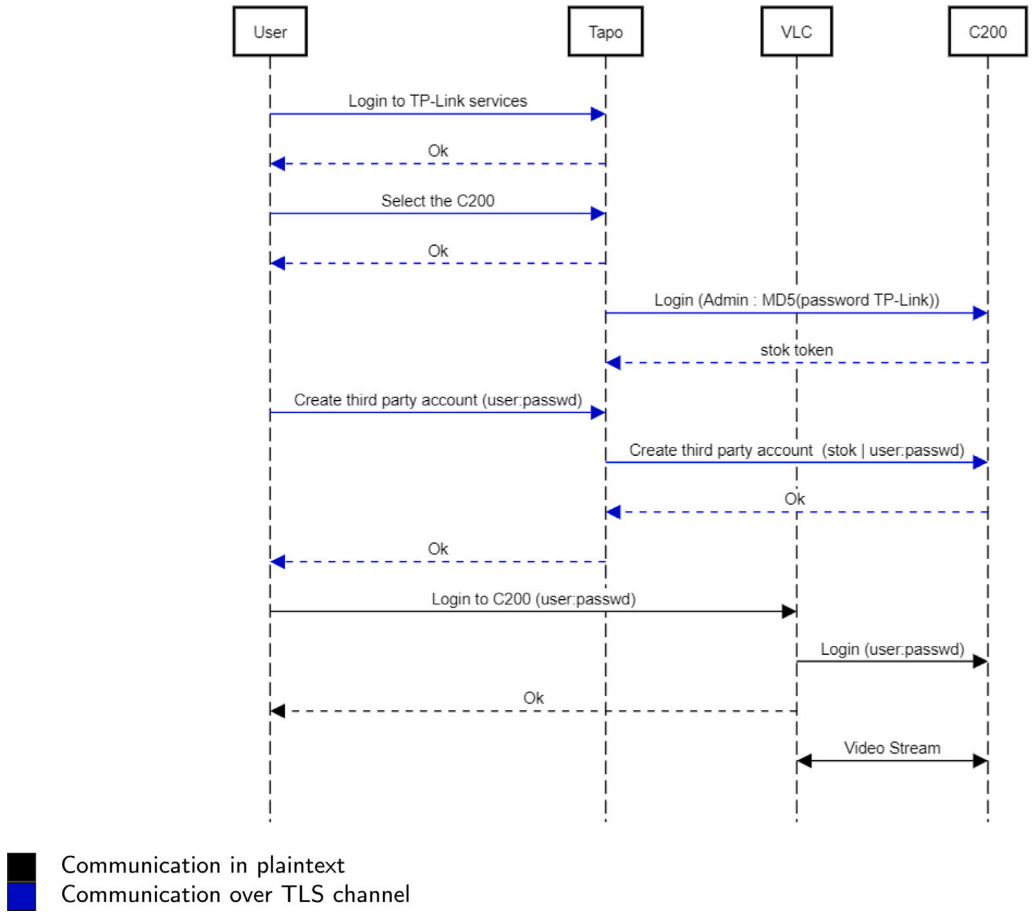

User
Tapo
VLC
C200
Login to TP-Link services
Ok
Select the C200
Ok
Login (Admin : MD5(password TP-Link))
stok token
Create third party account (user:passwd)
Create third party account (stok | user:passwd)
Ok
Ok
Login to C200 (user:passwd)
Login (user:passwd)
Ok
Video Stream
Communication in plaintext
Communication over TLS channel

Fig. 4. Sequence diagram for using the IP camera with third party software.

Our manual assessment returns on the way motion detection is implemented, concluding that also the detection notifications are cryptographically protected hence should be unintelligible to the attacker. However, we observe a specific feature leading to a vulnerability.

> *Vulnerability 2 – Insufficient entropy in encrypted notifications allows breach of motion detection.* The size of the motion detection notifications, despite the fact that these are encrypted, is always the same, precisely 523 bytes. We therefore report that, by using the Tapo C200 as an oracle, the attacker may be able to discern movement notifications based on the size of the messages, without intervening in the cryptographic scheme, which remains secure. This vulnerability is a specific instance of weakness "CWE-331: Insufficient Entropy" [43].

Following this style of Traffic Analysis, the attacker may deduce, for example, the precise times when someone is in the house that the camera monitors or when the house is empty. We shall see below in what scenario this vulnerability can be exploited. Additionally, by filtering and blocking only the messages carrying out a motion detection alert, the attacker could effectively deny the relevant notification to the legitimate user and convince the victim that no movement occurs in the area covered by the camera.

5.4.2. *Using the Tapo C200 through third-party software*

The use of third-party players for video streaming is common because it is convenient for the end-user: it supports concurrent streaming from multiple cameras while the native app does not. With the increasing popularity of these devices, perhaps due to their low price and to the widespread need for surveillance, third-party players are becoming increasingly popular. Still, we found a feature that causes a vulnerability.

> *Vulnerability 3 – Cleartext transmission of video stream allows breach by unintended actors.* Because the ONVIF profile S is limited to providing an interface for the use of RTSP, both streaming through ONVIF on port 2020 and RTSP on port 554 do not support encryption, hence there is no security measure to ensure confidentiality of the video stream. In fact, after configuring

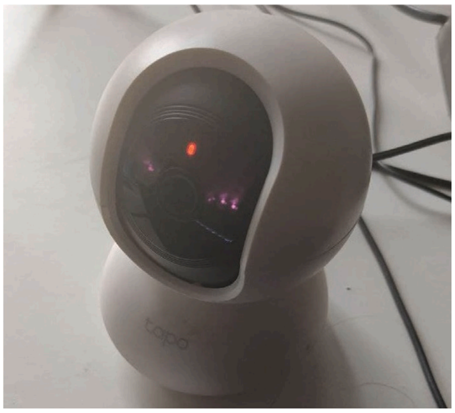
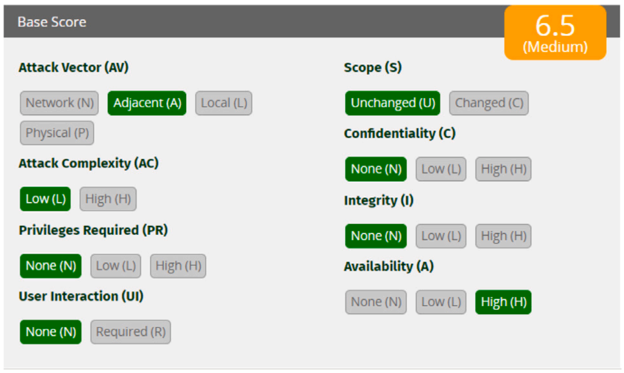


Fig. 5. Crash of the Tapo C200 camera.



Base Score 6.5 (Medium)

Attack Vector (AV): Network (N) [✓ Adjacent (A)] Local (L) Physical (P)

Attack Complexity (AC): [✓ Low (L)] High (H)

Privileges Required (PR): [✓ None (N)] Low (L) High (H)

User Interaction (UI): [✓ None (N)] Required (R)

Scope (S): [✓ Unchanged (U)] Changed (C)

Confidentiality (C): [✓ None (N)] Low (L) High (H)

Integrity (I): [✓ None (N)] Low (L) High (H)

Availability (A): None (N) Low (L) [✓ High (H)]

Fig. 6. CVSS score calculated for Vulnerability 1 – Denial of Service.

the Tapo C200 to use these systems, whenever a request is made, the video stream is sent in the clear by H264 encoding. This is clearly insecure, as the attacker could intercept the stream and decode it so that it would become fully intelligible. Also for this vulnerability, to what extent it can be exploited will be seen below. This vulnerability is a specific instance of weakness "CWE-319: Cleartext Transmission of Sensitive Information" [44].

5.5. *Exploitation*

This Section reports on the exploitation experiments corresponding to the vulnerabilities discussed above. It is possible to exploit all three vulnerabilities reported above at least for an attacker who has access to the same network on which the TAPO C200 operates. This is the setup that is used through our exploitation efforts reported below.

5.5.1. *Denial of Service*

A Denial of Service (DoS) is a drastic malfunction of a device due to some form of resource exhaustion. Such attacks tend to undermine quality and, eventually, availability of a service by overloading it with an arbitrarily number of requests.

We succeeded in exploiting the vulnerability that allows DoS on the Tapo C200 camera, that is, Vulnerability 1. In fact, an intensive scan by Nessus makes the device crash, as it can be seen from the red led in Fig. 5. After the crash, the device reboots. As also occurred elsewhere, this may be due to "*insufficient memory specific to TCP/IP stack leading to a small amount of probes consuming it all and causing DoS*" but we are "*unable to determine what the root cause is in this situation because we do not possess information about the TCP/IP implementation on the remote host*" [45].

Fig. 6 shows the CVSS score that we calculated for Vulnerability 1. As there is a complete loss of availability for every service offered by the camera, the Availability parameter is set to High, leading to the 6.5 CVSS score, which evaluates to Medium.

5.5.2. *Breach of motion detection*

To exploit Vulnerability 2 and demonstrate that it is practically feasible, we devise a special experiment consisting of two stages. A capture stage, namely to use the Tapo C200 to record a public place; a processing stage, namely to verify whether recorded events can be predicted correctly by intercepting *encrypted* motion detection notifications. We conducted the capture part of the experiment

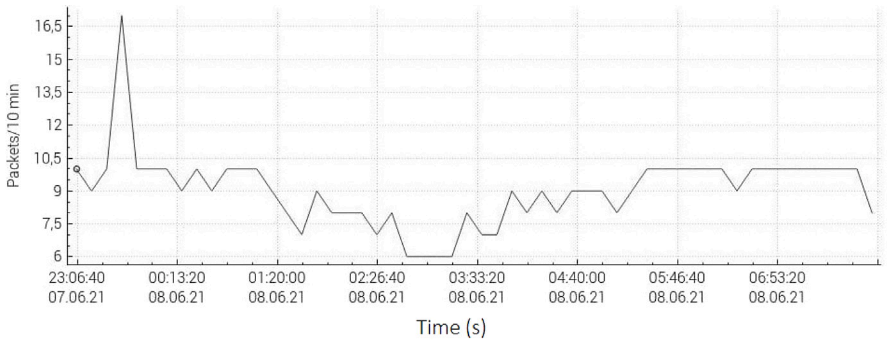


Fig. 7. Packets carrying notifications of motion detection from an overnight capture.

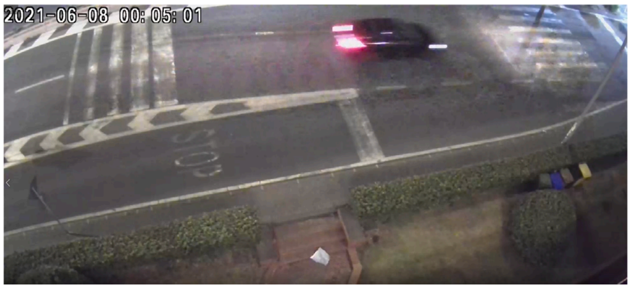


Fig. 8. One of the frames at 00:05 recording a moving car.

by recording a suburban street of our hometown overnight back in June 2021. For the processing part, we intercepted packets using Ettercap then selected SSL/TLS packets of length 523 every 10 min. The outcome is represented in the chart in Fig. 7.

It can be seen that starting at 23:00, the amount of packets begins to decrease until about 3:00. From 3:30 it is possible to see an increase in packets reaching a maximum after 5:00. We remark that the chart is solely built out of encrypted data. The trend of the curve is consistent with what could be expected from overnight traffic, but the experiment continued to confirm that by comparing the chart with the actual recorded video. As a result, the charted packets match recorded movements precisely. For example, Fig. 8 shows a frame taken at about 00:05, when a car can be seen in transit, while the chart shows 10 motion detection packets at the subsequent sampling time.

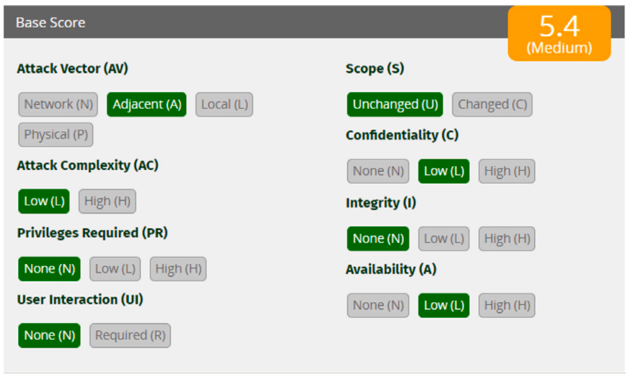
If on one hand our experiment can be interpreted as a confirmation of the reliability of the motion detection functionality, it becomes apparent, on the other hand, that such a function leaks relevant information due to the strictly deterministic use of encryption. As mentioned, an attacker could profitably take advantage of such an exploit. If the camera is used to protect a household, the adversary could infer whether the legitimate user is at home or not, based on the captured motion detection notifications. Furthermore, Ettercap can be used to intercept motion detection messages and appropriate Iptables rules defined to prevent their delivery, thereby denying the motion detection service.

Having verified how to exploit Vulnerability 2, we are ready to score it by CVSS. Fig. 9 shows the resulting score, of 5.4, thus Medium relevance.

5.5.3. Breach of video stream

To verify whether Vulnerability 3 can be exploited, we need to demonstrate how an attacker could obtain the video stream transmitted using third-party software. We conduct an experiment on the machine running the multimedia players by starting a multimedia session with the Tapo C200. At the same time, the other machine is acting as a MITM to intercept the traffic exchanged between the other two devices. Traffic is first interpreted by using Wireshark. It is then possible to use the H264 extractor tool to reconstruct the entire video stream from its frames, which are shown in Fig. 10. It is clear that lack of encryption makes the success of this exploit relatively easy to achieve.

Fig. 11 shows the CVSS score we calculate for this vulnerability: there is a High loss of Confidentiality as the adversary is able to read the whole video stream; Integrity loss is High as well, as the adversary could replace every H264 frame with an entirely different one, or could tamper with the sent frames; finally, Availability loss is High as the adversary could drop all H264 frames, thereby practically discarding the entire video stream. This leads to a CVSS score of 8.8 for this vulnerability, which evaluates to High severity.



Base Score 5.4 (Medium)

Attack Vector (AV)
Network (N) [✓ Adjacent (A)] Local (L) Physical (P)

Attack Complexity (AC)
[✓ Low (L)] High (H)

Privileges Required (PR)
[✓ None (N)] Low (L) High (H)

User Interaction (UI)
[✓ None (N)] Required (R)

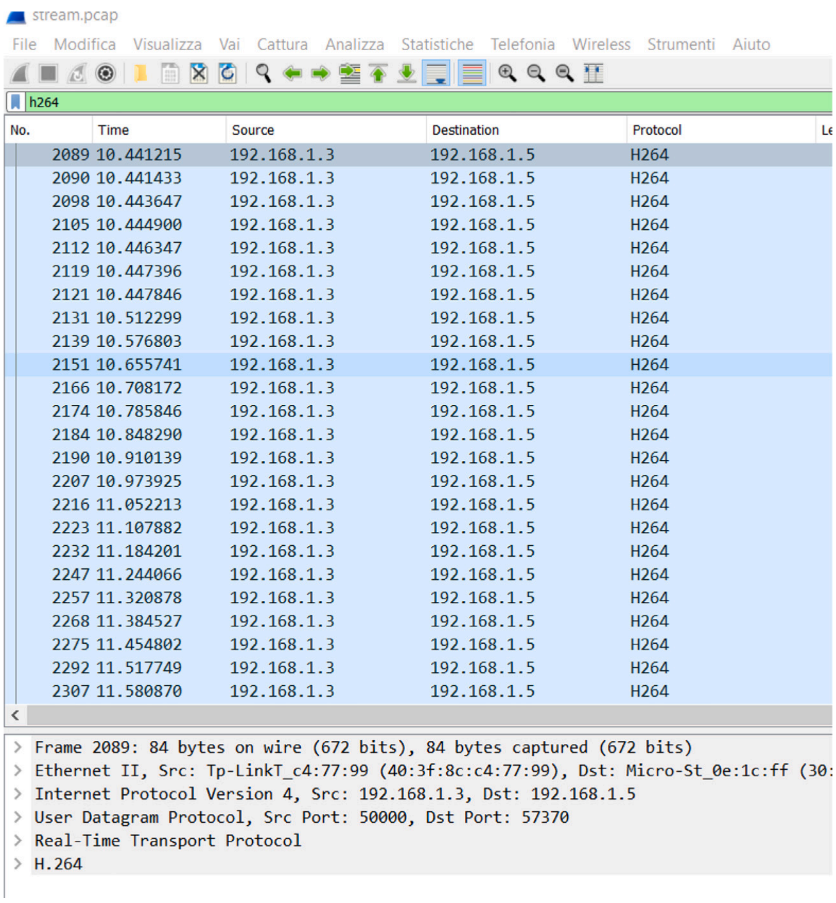
Scope (S)
[✓ Unchanged (U)] Changed (C)

Confidentiality (C)
None (N) [✓ Low (L)] High (H)

Integrity (I)
[✓ None (N)] Low (L) High (H)

Availability (A)
None (N) [✓ Low (L)] High (H)

Fig. 9. CVSS score calculated for Vulnerability 2 – Breach of motion detection.


stream.pcap
File Modifica Visualizza Vai Cattura Analizza Statistiche Telefonia Wireless Strumenti Aiuto

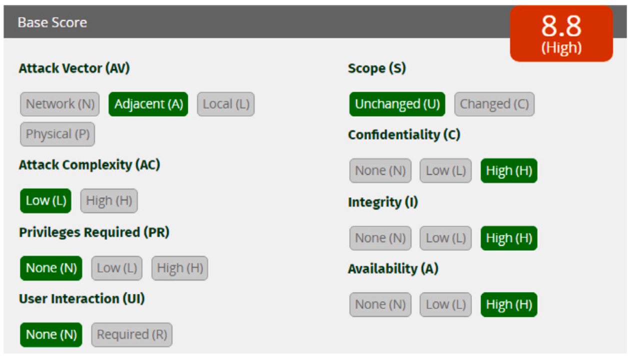
```
h264
No.   Time       Source        Destination   Protocol
2089 10.441215   192.168.1.3   192.168.1.5   H264
2090 10.441433   192.168.1.3   192.168.1.5   H264
2098 10.443647   192.168.1.3   192.168.1.5   H264
2105 10.444900   192.168.1.3   192.168.1.5   H264
2112 10.446347   192.168.1.3   192.168.1.5   H264
2119 10.447396   192.168.1.3   192.168.1.5   H264
2121 10.447846   192.168.1.3   192.168.1.5   H264
2131 10.512299   192.168.1.3   192.168.1.5   H264
2139 10.576803   192.168.1.3   192.168.1.5   H264
2151 10.655741   192.168.1.3   192.168.1.5   H264
2166 10.708172   192.168.1.3   192.168.1.5   H264
2174 10.785846   192.168.1.3   192.168.1.5   H264
2184 10.848290   192.168.1.3   192.168.1.5   H264
2190 10.910139   192.168.1.3   192.168.1.5   H264
2207 10.973925   192.168.1.3   192.168.1.5   H264
2216 11.052213   192.168.1.3   192.168.1.5   H264
2223 11.107882   192.168.1.3   192.168.1.5   H264
2232 11.184201   192.168.1.3   192.168.1.5   H264
2247 11.244066   192.168.1.3   192.168.1.5   H264
2257 11.320878   192.168.1.3   192.168.1.5   H264
2268 11.384527   192.168.1.3   192.168.1.5   H264
2275 11.454802   192.168.1.3   192.168.1.5   H264
2292 11.517749   192.168.1.3   192.168.1.5   H264
2307 11.580870   192.168.1.3   192.168.1.5   H264

> Frame 2089: 84 bytes on wire (672 bits), 84 bytes captured (672 bits)
> Ethernet II, Src: Tp-LinkT_c4:77:99 (40:3f:8c:c4:77:99), Dst: Micro-St_0e:1c:ff (30:
> Internet Protocol Version 4, Src: 192.168.1.3, Dst: 192.168.1.5
> User Datagram Protocol, Src Port: 50000, Dst Port: 57370
> Real-Time Transport Protocol
> H.264
```

Fig. 10. Example intercepted H264 frames.

5.6. *Fixing*

The last step of PETIoT prescribes the definition of countermeasures for the vulnerabilities that may have been found and exploited. In the particular case of the Tapo C200, the first vulnerability could be addressed by standard filtering strategies to mitigate denial-of-service attacks at kernel level and at firewall level. Vulnerability 2 could be addressed by randomising the size of the frame carrying the motion detection notification, and Vulnerability 3 by adding a cryptographic tunnel to the video stream.

The implementation of such fixes clearly requires upgrades to the camera proprietary software, which we cannot do. However, we investigated various home-made setups as alternative fixes, and the sequel of this Section demonstrates them against Vulnerability


Base Score 8.8 (High)

Attack Vector (AV): Network (N) [✓ Adjacent (A)] Local (L) Physical (P)

Attack Complexity (AC): [✓ Low (L)] High (H)

Privileges Required (PR): [✓ None (N)] Low (L) High (H)

User Interaction (UI): [✓ None (N)] Required (R)

Scope (S): [✓ Unchanged (U)] Changed (C)

Confidentiality (C): None (N) Low (L) [✓ High (H)]

Integrity (I): None (N) Low (L) [✓ High (H)]

Availability (A): None (N) Low (L) [✓ High (H)]

Fig. 11. CVSS score calculated for Vulnerability 3 – Breach of video stream.

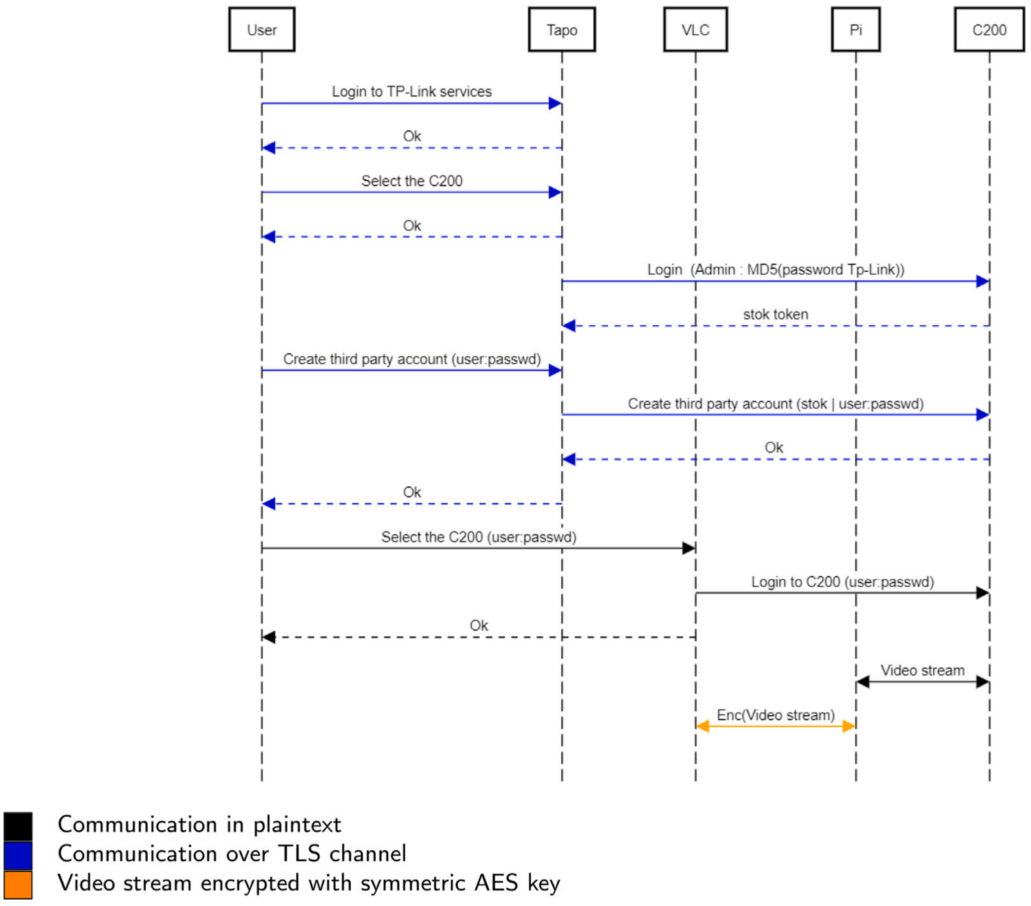

User
Tapo
VLC
Pi
C200
Login to TP-Link services
Ok
Select the C200
Ok
Login (Admin : MD5(password Tp-Link))
stok token
Create third party account (user:passwd)
Create third party account (stok | user:passwd)
Ok
Ok
Select the C200 (user:passwd)
Login to C200 (user:passwd)
Ok
Video stream
Enc(Video stream)
Communication in plaintext
Communication over TLS channel
Video stream encrypted with symmetric AES key

Fig. 12. Sequence diagram for tunnelling the Tapo C200 video stream through a Raspberry Pi 4 Model B.

3. The idea behind our own solution to protect the video stream in the third party scenario is to use the Raspberry Pi 4 Model B as an access point of the Tapo C200 in order to modify the traffic in transit by applying a layer of encryption. One Pi suffices to secure one camera but it must be remarked that this is only meant as a proof of concept of our fix, while the cryptographic tunnel that the Pi contributes to establish between the camera and the video player should be implemented by the vendor in the camera firmware instead.

5.6.1. Upgrading the sequence diagram

A sequence diagram representing the insertion of the Pi device when the Tapo C200 works with third-party software is shown in Fig. 12. It can be seen that the presence of the new device is totally transparent to the user, so that she can use the service exactly

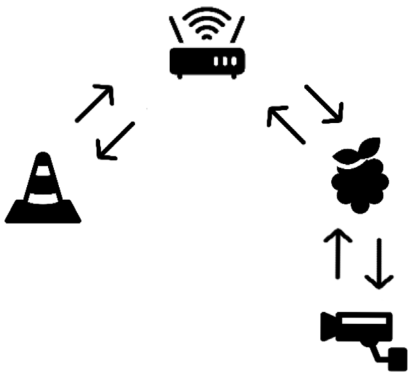


Fig. 13. The testbed for the TAPO C200 with the Raspberry Pi 4 Model B.

as she used to in the original scenario. The essential difference is that the Pi relays the cleartext video stream to the player but adds the desired encryption layer. It is clear that the new diagram makes sense against the same threat model considered before, which sees malicious activity within the same network that interconnects the devices. However, the upgraded diagram assumes that the attacker cannot sneak in through the connection between the camera and the Pi, an assumption that can be realistically enforced in practice by connecting solely the Pi and the camera on a dedicated SSID. This is only a proof-of-concept of the fix, while the mentioned cryptographic tunnel should be implemented by the vendor in the camera firmware, so as to thwart the odds that the attacker interposes.

5.6.2. *Upgrading the testbed*

The Raspberry Pi 4 Model B has a network card that exposes a wired and a wireless interface. It was then possible to connect the device to the testbed switch by ethernet cable, while the Pi also exposes a separate Wi-Fi network to which the Tapo C200 can connect. Fig. 13 shows the resulting network architecture. More precisely, the Pi can be configured to act as an access point for the camera by Hostapd and some DHCP tweaks.

5.6.3. *Attempting exploitation of Vulnerability 3 again*

In order to demonstrate the effectiveness of our home-made fix, the same experiment to exploit Vulnerability 3 was carried out again, but this time the execution of the H264 had no luck. Fig. 14 shows that the intercepted traffic is unintelligible, hence the execution of the H264 extractor tool could not produce valid results.

5.6.4. *A glimpse at the implementation of encryption*

A fundamental prerequisite was to configure IPtables on the Raspberry Pi with the policy in Listing 1:

Code 1: Configuration of IPtables for encryption

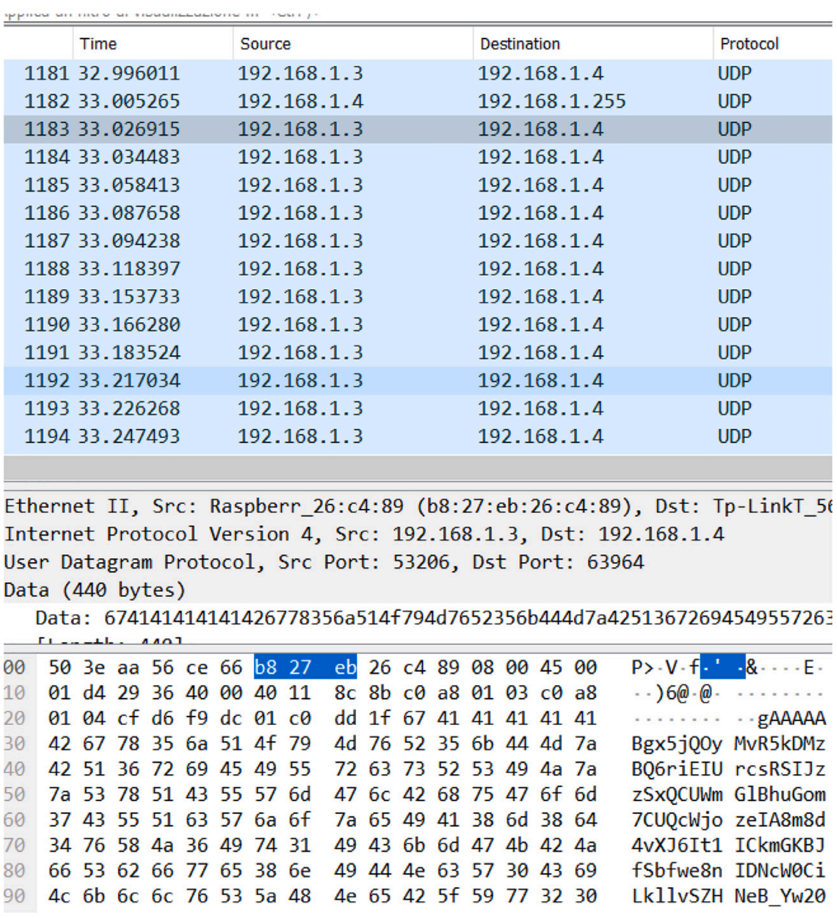
```
iptables — A FORWARD — m
——physdev—is—in wlan0 —p udp
—s TAPOC200_ADDRESS_IPV4
—j NFQUEUE ——queue num 1

iptables —A INPUT—A FORWARD —m
——physdev—is—in wlan0 —f —j DROP
```

The first rule makes it possible to intercept packets in transit (FORWARD chain) from the network interface (-m –physdev-is-in wlan0) transported with UDP (-p udp) coming from the Tapo C200 (-s TAPOC200_ADDRESS_IPV4) and then to send them on queue 1. After that, the second rule eliminates all possible fragments of the packets in transit, in order to avoid sending partial packets.

Once the firewall queues up the packets as discussed, the Python script shown in Listing 2, encrypts the packets in the queue. The script retrieves packets from the queue, extracts the payload (by skipping 28 bytes of headers), encrypts it and sends it back to the recipient's address.

Code 2: Encryption script

```
def encrypt(packet):
  cipher_suite = Fernet(key)
  encoded_text = cipher_suite.encrypt
  (packet.get_payload()[28:])
```



```
     Time        Source         Destination      Protocol
1181 32.996011   192.168.1.3    192.168.1.4      UDP
1182 33.005265   192.168.1.4    192.168.1.255    UDP
1183 33.026915   192.168.1.3    192.168.1.4      UDP
1184 33.034483   192.168.1.3    192.168.1.4      UDP
1185 33.058413   192.168.1.3    192.168.1.4      UDP
1186 33.087658   192.168.1.3    192.168.1.4      UDP
1187 33.094238   192.168.1.3    192.168.1.4      UDP
1188 33.118397   192.168.1.3    192.168.1.4      UDP
1189 33.153733   192.168.1.3    192.168.1.4      UDP
1190 33.166280   192.168.1.3    192.168.1.4      UDP
1191 33.183524   192.168.1.3    192.168.1.4      UDP
1192 33.217034   192.168.1.3    192.168.1.4      UDP
1193 33.226268   192.168.1.3    192.168.1.4      UDP
1194 33.247493   192.168.1.3    192.168.1.4      UDP

Ethernet II, Src: Raspberr_26:c4:89 (b8:27:eb:26:c4:89), Dst: Tp-LinkT_5
Internet Protocol Version 4, Src: 192.168.1.3, Dst: 192.168.1.4
User Datagram Protocol, Src Port: 53206, Dst Port: 63964
Data (440 bytes)
   Data: 674141414141426778356a514f794d7652356b444d7a42513672694549557 26

00  50 3e aa 56 ce 66 b8 27  eb 26 c4 89 08 00 45 00   P>·V·f·' ·&····E·
10  01 d4 29 36 40 00 40 11  8c 8b c0 a8 01 03 c0 a8   ··)6@·@· ········
20  01 04 cf d6 f9 dc 01 c0  dd 1f 67 41 41 41 41 41   ········ ··gAAAAA
30  42 67 78 35 6a 51 4f 79  4d 76 52 35 6b 44 4d 7a   Bgx5jQOy MvR5kDMz
40  42 51 36 72 69 45 49 55  72 63 73 52 53 49 4a 7a   BQ6riEIU rcsRSIJz
50  7a 53 78 51 43 55 57 6d  47 6c 42 68 75 47 6f 6d   zSxQCUWm GlBhuGom
60  37 43 55 51 63 57 6a 6f  7a 65 49 41 38 6d 38 64   7CUQcWjo zeIA8m8d
70  34 76 58 4a 36 49 74 31  49 43 6b 6d 47 4b 42 4a   4vXJ6It1 ICkmGKBJ
80  66 53 62 66 77 65 38 6e  49 44 4e 63 57 30 43 69   fSbfwe8n IDNcW0Ci
90  4c 6b 6c 6c 76 53 5a 48  4e 65 42 5f 59 77 32 30   LkllvSZH NeB_Yw20
```

Fig. 14. A sample of encrypted network traffic.

```
pkt = IP(packet.get_payload())
UDP_IP = pkt[IP].dst
UDP_PORT = pkt[UDP].dport
MESSAGE = encoded_text

sock = socket.socket(socket.AF_INET,
socket.SOCK_DGRAM)
sock.sendto(MESSAGE,
(UDP_IP, UDP_PORT))

packet.drop()
```

5.6.5. *A glimpse at the implementation of decryption*

A configuration of IPtables was also necessary for the client machine running third-party video player in order to enable it to decrypt the encrypted stream originating with the Pi. Listing 3 shows the policy.

Code 3: Configuration of IPtables for decryption

```
iptables —A INPUT —p udp
—s RASPBERRY_ADDRESS_IPV4
—j NFQUEUE ——queue num 2
```

This policy intercepts incoming packets (INPUT chain) transported with UDP (-p udp) coming from the Pi (-s RASPBERRY ADDRESS _ IPV4) and then sends them on queue 2.

The corresponding script for decrypting the packets is shown in Listing 4. It fetches packets from the queue, extracts the payload and decrypts it before accepting it.

Code 4: Decryption script

```
def decrypt(packet):
  cipher_suite = Fernet(key)
  decoded_text = cipher_suite.decrypt
  (packet.get_payload()[28:])

  pkt = IP(packet.get_payload())
  UDP_IP = pkt[IP].dst
  UDP_PORT = pkt[UDP].dport
  MESSAGE = decoded_text

  sock = socket.socket(socket.AF_INET,
  socket.SOCK_DGRAM)
  sock.sendto(MESSAGE,
  (UDP_IP, UDP_PORT))

  packet.drop()
```

6. Lessons learned and recommendations

The research and developments discussed above taught us that the notion of a Kill Chain is used rather broadly in the ethical hacking community. Therefore, this article aimed, first and foremost, at preciseness. The first limitation arising from the state of the art was the separation between KCs for attack or for defence, whereas both are indispensable during VAPT activities, hence PETIoT combines them. Moreover, we learned and experienced that each KC comes with a precise focus, which, by necessity arising from the (im)maturity of the state of the art, is not always taken strictly by practitioners. For example, Lockheed Martin's CKC concerns intrusions into a target device, other KCs detail the intrusion process further, while others still concern operations that are internal to the target. Then, we saw how steps that are established within other KCs become redundant in PETIoT due to the focus on the IoT domain, or unnecessary due to the focus on the device environment.

What is noted above allowed PETIoT to stay very slim hence, arguably, easily applicable in practice, so that it is the recommended choice for the penetration tester who is tasked with an IoT device, providing the most effective steps to focus on the device environment. Even as a result of this recommendation, which has been demonstrated throughout the present article, it is firm that the state of the art remains useful. For example, a simplification of Lockheed Martin's CKC as explained above is recommended to guide actual intrusion activities in an IoT device because the KC is general about intrusions into *any* device. Notably, a recent Master's thesis [21] may be interpreted in support of this recommendation despite the fact that the thesis itself does not make it explicit that it is, in fact, applying a KC. Conversely, PETIoT could be applied to any device beyond the IoT to instruct VAPT activities focused on the environment of the device.

The very demonstrator that was tackled, the TAPO C200, also taught us general lessons. Because flooding this device (and similar ones, as noted over other devices discussed elsewhere) is rather simple, not just household chores but also sensitive monitoring such as over newborns or house perimeters are at risk. Therefore, it is recommended that this and every IoT device implements appropriate traffic filtering measures. Also, a robber could profitably exploit the breach of motion detection to infer when a target household is very likely to be unmanned. The consequent recommendation is to randomise the motion detection signal on this and similar devices to minimise its predictability. Finally, the video stream breach is clearly intrusive, hence a natural recommendation is that no unencrypted stream ever leaves any IP camera, independently of the use case. The camera vendor acknowledged all three vulnerabilities through the responsible disclosure process that we followed with them, so that the current firmware version of the TAPO C200 fixes all vulnerabilities discussed in this article.

7. Conclusions

This article aimed at facilitating the adoption of the KC approach for the sake of ethical hacking in the IoT application domain. By defining the PETIoT KC for IoT devices, the article showed how penetration testers can follow a KC during VAPT sessions. PETIoT is slim and readily applicable. Its focus is on how the tester can violate a device she can reach, either physically or logically. Our KC insists on a fixing step for devising preventative or notification measures and thwart the vulnerabilities that may have been found and exploited. In consequence, our research questions can be answered by noting that, in general, there do exist approaches that can be leveraged towards VAPT in the IoT domain albeit with appreciable tailoring and that, in particular, PETIoT can be applied beside a simplified version of Lockheed Martin's CKC.

To demonstrate PETIoT, we followed it to engage into a VAPT session on the best-selling IP camera on Amazon Italy, the TAPO C200 by TP-Link. Under specific circumstances, three vulnerabilities arose, one in terms of denial of service, the other two in terms of privacy of the camera users and overall security of the services that the camera exposes to them. It is found that the tester can, respectively, understand motion detection signals from cryptographically protected notifications, and breach the captured video when it is streamed unencrypted towards a third-party player. With a CVSS score of 8.8, the last vulnerability, in particular, is of High severity, and a proof-of-concept fix for it was provided. Responsible disclosure with the vendor led to fixing all reported vulnerabilities in the latest firmware released for the camera.

PETIoT has been gestated out of years of experiments with several types of IoT devices. We have been evaluating its benefits empirically over the many devices our research group has been besieging. The outcomes have always been very positive, confirming that PETIoT effectively drives the testers through the operations, saving time, although any KC may require several sub-steps such as scripting, fuzzing, probing, physical tampering, etc, which depend on the precise architecture and functions of the target device as well as on the tester's skillset. Future work includes further applications of our KC to more devices, and this will be facilitated by the popularity of the techniques experienced over the TAPO C200, including ONVIF profiles and RTSP. It seems fair to conclude that our demonstration of PETIoT gained us an in-depth understanding of the target camera from a cybersecurity and privacy standpoint, hence it is well worth employing on more IoT devices.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

Acknowledgements

We are grateful to Angelo Zinna for contributing to the experiments. This research was supported by "Linea di intervento 2, progetto MEGABIT, Italy" of funding programme "PIAno di inCEntivi per la RIcerca di Ateneo 2020/2022 (PIACERI)" of University of Catania, and open access was supported by "Linea di Intervento 4, Italy" of the same funding programme.

References

[1] M. Antonakakis, et al., Understanding the Mirai botnet, in: 26th USENIX Security Symposium, USENIX Security 17, USENIX Association, Vancouver, BC, 2017, pp. 1093–1110.
[2] Helpnetsecurity, IoT malware attacks rose 700% during the pandemic, 2021, https://www.helpnetsecurity.com/2021/07/20/iot-malware-attacks-rose/.
[3] Common Vulnerability Scoring System, CVSS, 2022, https://www.first.org/cvss/.
[4] Raspberry Pi foundation, Raspberry Pi 4 Model B, 2021, https://www.raspberrypi.org/products/raspberry-pi-4-model-b/.
[5] G. Bella, P. Biondi, You overtrust your printer, in: A. Romanovsky, E. Troubitsyna, I. Gashi, E. Schoitsch, F. Bitsch (Eds.), Computer Safety, Reliability, and Security, Springer International Publishing, Cham, 2019, pp. 264–274, http://dx.doi.org/10.1109/FMEC49853.2020.9144875.
[6] G. Bella, P. Biondi, S. Bognanni, Multi-service threats: Attacking and protecting network printers and VoIP phones alike, Elsevier Internet Things 18 (2022) http://dx.doi.org/10.1016/j.iot.2022.100507.
[7] P. Biondi, S. Bognanni, G. Bella, VoIP can still be exploited - badly, in: 2020 Fifth International Conference on Fog and Mobile Edge Computing, FMEC, 2020, pp. 237–243, http://dx.doi.org/10.1109/FMEC49853.2020.9144875.
[8] S. Esposito, D. Sgandurra, G. Bella, Alexa versus alexa: Controlling smart speakers by self-issuing voice commands, in: Proceedings of the 2022 ACM on Asia Conference on Computer and Communications Security, ASIA CCS '22, Association for Computing Machinery, New York, NY, USA, 2022, pp. 1064–1078, http://dx.doi.org/10.1145/3488932.3497766.
[9] P. Biondi, S. Bognanni, G. Bella, Vulnerability assessment and penetration testing on IP camera, in: 2021 8th International Conference on Internet of Things: Systems, Management and Security, IOTSMS, 2021, pp. 1–8, http://dx.doi.org/10.1109/IOTSMS53705.2021.9704890.
[10] MITRE, Mitre att&ck, 2022, https://attack.mitre.org/.
[11] MITRE, Mitre d3fend, 2022, https://d3fend.mitre.org/.
[12] T. Grant, I. Burke, R. van Heerden, Comparing models of offensive cyber operations, in: 7th International Conference on Information Warfare and Security, 2012, p. 108.
[13] E. Hutchins, M. Cloppert, R. Amin, Intelligence-driven computer network defense informed by analysis of adversary campaigns and intrusion kill chains, in: Leading Issues in Information Warfare & Security Research, Vol. 1, 2011.
[14] P. Pols, The unified kill chain, 2021, https://www.unifiedkillchain.com/.
[15] S. Malone, The expanded cyber kill chain model, 2016, https://www.blackhat.com/docs/us-16/materials/us-16-Malone-Using-An-Expanded-Cyber-Kill-Chain-Model-To-Increase-Attack-Resiliency.pdf.
[16] H. Kim, H. Kwon, K.K. Kim, Modified cyber kill chain model for multimedia service environments, Multimedia Tools Appl. 78 (3) (2019) 3153–3170, http://dx.doi.org/10.1007/s11042-018-5897-5.
[17] J. Haseeb, M. Mansoori, I. Welch, A measurement study of IoT-based attacks using IoT kill chain, in: 19th IEEE International Conference on Trust, Security and Privacy in Computing and Communications, TrustCom, 2020, pp. 557–567, http://dx.doi.org/10.1109/TrustCom50675.2020.00080.
[18] A. Tekeoglu, A.S. Tosun, Investigating security and privacy of a cloud-based wireless IP camera: NetCam, in: 2015 24th International Conference on Computer Communication and Networks, ICCCN, 2015, pp. 1–6, http://dx.doi.org/10.1109/ICCCN.2015.7288421.
[19] J. Liranzo, T. Hayajneh, Security and privacy issues affecting cloud-based IP camera, in: 2017 IEEE 8th Annual Ubiquitous Computing, Electronics and Mobile Communication Conference, UEMCON, 2017, pp. 458–465, http://dx.doi.org/10.1109/UEMCON.2017.8249043.
[20] P. Abdalla, C. Varol, Testing IoT security: The case study of an IP camera, in: 8th International Symposium on Digital Forensics and Security, ISDFS, 2020, pp. 1–5, http://dx.doi.org/10.1109/ISDFS49300.2020.9116392.
[21] H. Gustafsson, H. Kvist, Cyber security demonstrations using penetration testing on Wi-Fi cameras, 2022, https://www.diva-portal.org/smash/record.jsf?pid=diva2:1679623.

[22] Ettercap, Ettercap project, 2021, https://www.ettercap-project.org/.
[23] Tenable Network Security, Nessus, 2022, https://www.tenable.com/products/nessus.
[24] Insecure, Nmap, 2022, https://nmap.org/book/man.html.
[25] Grey Shirts, Packet capture, 2022, https://play.google.com/store/apps/details?id=app.greyshirts.sslcapture.
[26] N.C. Team, iptables(8) - Linux man page, 2022, https://linux.die.net/man/8/iptables.
[27] Wireshark, Wireshark project, 2022, https://www.wireshark.org/.
[28] Videolan, VLC, 2022, https://www.videolan.org/vlc/index.it.html.
[29] iSpyConnect, iSpy, 2022, https://www.ispyconnect.com/.
[30] A. Rao, R. Lanphier, H. Schulzrinne, Real Time Streaming Protocol (RTSP), RFC 2326, RFC Editor, 1998, http://dx.doi.org/10.17487/RFC2326.
[31] RFC, RTP: A transport protocol for real-time applications, 2003, https://tools.ietf.org/html/rfc3550.
[32] E. Schooler, J. Ott, J. Chesterfield, RTP Control Protocol (RTCP) Extensions for Single-Source Multicast Sessions with Unicast Feedback, RFC 5760, RFC Editor, 2010, http://dx.doi.org/10.17487/RFC5760.
[33] ONVIF, Conformant products, 2022, https://www.onvif.org/conformant-products/.
[34] ONVIF, ONVIF profile s specification, 2019, https://www.onvif.org/wp-content/uploads/2019/12/ONVIF_Profile_-S_Specification_v1-3.pdf.
[35] OWASP, Certificate and public key pinning, 2022, https://owasp.org/www-community/controls/Certificate_and_Public_Key_Pinning.
[36] Gentoo, Hostapd, 2022, https://wiki.gentoo.org/wiki/Hostapd.
[37] Asmag security and IoT, Smart home camera confirms popularity in the market, 2019, https://www.asmag.com/showpost/28003.aspx.
[38] ProClip, The rising popularity of dash cameras, 2020, https://blog.proclipusa.com/the-rising-popularity-of-dashcams.
[39] E. Cosgrove, One billion surveillance cameras will be watching around the world in 2021, a new study says, 2019, https://www.cnbc.com/2019/12/06/one-billion-surveillance-cameras-will-be-watching-globally-in-2021.html.
[40] Google Play Store, TP-link Tapo, 2022, https://play.google.com/store/apps/details?id=com.tplink.iot&hl=it&gl=US.
[41] Apple Store, TP-link Tapo app, 2022, https://apps.apple.com/us/app/tp-link-tapo/id1472718009.
[42] MITRE, CWE - CWE-707: Improper neutralization (4.9), 2008, https://cwe.mitre.org/data/definitions/707.html.
[43] MITRE, CWE - CWE-331: Insufficient entropy (4.9), 2006, https://cwe.mitre.org/data/definitions/331.html.
[44] MITRE, CWE - CWE-319: Cleartext transmission of sensitive information (4.9), 2006, https://cwe.mitre.org/data/definitions/319.html.
[45] Tenable Community, Applications on a host being scanned crash while Nessus is scanning the host, 2021, https://community.tenable.com/s/article/Applications-on-a-host-being-scanned-crash-while-Nessus-is-scanning-the-host.