International Journal of Information Security
https://doi.org/10.1007/s10207-022-00633-3

REGULAR CONTRIBUTION

PatrIoT: practical and agile threat research for IoT

Emre Süren[1] · Fredrik Heiding[1] · Johannes Olegård[1] · Robert Lagerström[1]

Abstract
The Internet of things (IoT) products, which have been widely adopted, still pose challenges in the modern cybersecurity landscape. Many IoT devices are resource-constrained and almost constantly online. Furthermore, the security features of these devices are less often of concern, and fewer methods, standards, and guidelines are available for testing them. Although a few approaches are available to assess the security posture of IoT products, the ones in use are mostly based on traditional non-IoT-focused techniques and generally lack the attackers' perspective. This study provides a four-stage IoT vulnerability research methodology built on top of four key elements: logical attack surface decomposition, compilation of top 100 weaknesses, lightweight risk scoring, and step-by-step penetration testing guidelines. Our proposed methodology is evaluated with multiple IoT products. The results indicate that *PatrIoT* allows cyber security practitioners without much experience to advance vulnerability research activities quickly and reduces the risk of critical IoT penetration testing steps being overlooked.

Keywords IoT attack surfaces · Weaknesses · Risk scoring · Penetration testing guidelines · MITRE CWE · OWASP IoT Top 10

1 Introduction

The IoT ecosystem provides a platform for connecting everyday objects over a network (e.g., Internet). Over the last few years, IoT-enabled solutions have become significantly popular with both consumers and industries, and the number of connected things has already exceeded the world population. IoT devices are leveraged to build solutions for personal assistants, home automation, building management, healthcare monitoring, and energy metering.

The IoT environment includes hardware manufacturing, software development [firmware, web, mobile, network service, cloud application programming interfaces (API)], radio connectivity, Internet service providers, and IoT platform integrators. Clearly, all these components need to be secure.

If IoT challenges are not met, exploitation of vulnerabilities can impact both the security of systems and consumers' privacy. Furthermore, smart devices are inevitable candidates for large-scale compromise, especially for botnet operators. The legendary example is the *Mirai* botnet [1], which compromised millions of physical devices that were turned into a botnet controlled by criminal groups. It was utilized to attack organizations across many nations, resulting in one of the highest recorded DDoS volumes of all time. It is worth recalling that while the same weakness being exploited in IoT systems can lead to massive results, the impact may not be the same when it happens in Information Technology (IT) systems. As in the example above, it is not common to create a botnet after exploiting a default password vulnerability in an IT system.

The proliferation and the increasing demand for smart devices make it necessary to prioritize their security. Such concerns have raised the need for IoT-specific capabilities.

We consider part of the solution to this challenging problem to be vulnerability research, as it is vital to discover vulnerabilities before threat actors. For example, had the company behind these compromised IoT products planned

✉ Emre Süren
emsuren@kth.se

Fredrik Heiding
fheiding@kth.se

Johannes Olegård
jolegard@kth.se

Robert Lagerström
robertl@kth.se

[1] School of Electrical Engineering and Computer Science, KTH Royal Institute of Technology, Stockholm, Sweden

Published online: 18 November 2022

a vulnerability research activity, it would have been possible to identify and patch weaknesses in time before attackers exploited them. More precisely, we believe that there should be practical IoT-specific penetration testing (pentest) capabilities in addition to traditional testing skills.

The development of security testing capabilities rests on three pillars: people, processes, and technology. In the first step, we used existing tools for IoT pentesting and addressed the problem from the process and people perspective. Our decision was based on the fact that the available tools are sufficient to get started and that the process and people perspectives were lacking. **Process.** Upon executing several pentest projects and examining many IoT pentest reports, we identified four significant shortcomings in IoT vulnerability research. (1) Vulnerability research is usually based (only) on web and network pentesting, which offers a limited attack surface for IoT.[1] (2) The lack of security test standardization leads to subjective assessments (tested weaknesses differ between very similar IoT devices). (3) Threat modeling is either too complex or is omitted entirely (difficulties in calculating risk scores). (4) Existing IoT pentesting guidelines are very limited or difficult to follow [2]. **People.** Offensive security research laboratories continuously pentest devices and report the results. This means that large-scale studies offer great potential from which to gain substantial knowledge. On the other hand, both academic and industry laboratories usually have a high personnel turnover, meaning researchers are constantly arriving and leaving for various reasons (e.g., graduation or better offers). This period sometimes falls below six months and the projects started are left unfinished. Another researcher should be able to take the unfinished work and only complete the remaining part. In addition, since we usually work with novice researchers, the shortage of IoT security expertise fans the flame.[2] That is why it makes sense to take some precautions to obtain the maximum benefit from these studies at a given period.

Therefore, we seek an answer to the question: *What should a methodology contain to provide the maximum benefit from vulnerability research studies of various IoT devices performed by dynamic and fairly novice security teams?*

This study addresses the four shortcomings regarding the process perspective of IoT vulnerability research by introducing the following four key elements: The logical decomposition of attack surfaces, 100 currently prevalent weaknesses found in IoT environments, a lightweight risk scoring approach, and step-by-step IoT pentesting guidelines. Then, we incorporate these four key elements into our IoT vulnerability research methodology that contains four stages of activities: planning, threat modeling, exploitation, and reporting, as shown in Table 1.

To evaluate the proposed methodology, we selected seven real-world IoT products, each representing a common IoT product category. The empirical results show that *PatrIoT* allows researchers to conduct vulnerability research activities quickly, and it indicates a decreased risk of overlooking critical steps.

The contribution of the research is that:

(a) information inquiry templates make the IoT information gathering phase easier,
(b) decomposing IoT infrastructure into seven logical components is an efficient way for IoT attack surfaces mapping,
(c) compilation of weaknesses compatible with *"Open Web Application Security Project (OWASP) IoT Top 10"*[3] and *"MITRE Common Weakness Enumeration (CWE),"*[4] which are the state-of-the-art approaches, facilitates threat modeling (to the best of our knowledge, such a comprehensive list for IoT has not been made public before),
(d) lightweight risk scoring approach effectively reduces the overhead of IoT threat modeling,
(e) step-by-step pentesting guidelines fortify the discovery of critical IoT vulnerabilities,
(f) detailed report templates for IoT pentest and vulnerability disclosure assist in developing high-quality content,
(g) all artifacts of *PatrIoT* [3] are kept in a public repository and open for collaboration.

The remainder of this paper is organized as follows. Section 2 describes the approach we follow to address the four shortcomings. A comprehensive explanation of the methodology, including the incorporation of these four key elements into the stages, is provided in Sect. 3. Section 4 evaluates the methodology, and presents the results. Section 5 discusses the trade-offs and challenges of using the proposed methodology, along with opportunities for future work. Section 6 introduces the literature and compares similar studies. Finally, the paper is concluded in Sect. 7 with the main takeaway.

2 Design of methodology

Although we prefer the term *"vulnerability research,"* we use *"pentesting"* interchangeably when it is proper due to its widespread use. Similarly, this paper uses the term *"researcher"* to abbreviate *"vulnerability researcher,"* meaning *"penetration tester."* Penetration testing is the simulation of an attack on a system, network, piece of equipment,

[1] https://bit.ly/3Eu6GkU.
[2] https://bit.ly/3CSwaVM.
[3] https://owasp.org/www-project-internet-of-things/.
[4] https://cwe.mitre.org.

Table 1 Four-stage IoT vulnerability research methodology

Planning	Threat modeling	Exploitation	Reporting
Scoping	Attack surface decomposition [b]	Known vulnerabilities [e]	Template [f]
Information gathering [a]	Vulnerability analysis [c]	Exploit development	Vulnerability disclosure
Enumeration	Risk scoring [d]	Post-exploitation	CVE

or other facilities to prove how vulnerable that system or target would be to a real attack. Information technology (IT) pentesting is a well-studied topic, covering one or two attack surfaces, mostly testing networks and web applications. This study also uses the *"traditional pentesting"* term when referring to the well-known *"IT pentesting"* [4,5] or *"network, OS, and web app pentesting."* Unlike IT systems, the IoT also includes additional attack surfaces (hardware, firmware, and radio) that need to be explored. That is why we use the *"IoT pentesting"* term to refer to *PatrIoT*, which involves the exploitation of all components of an IoT product. We consider *"weakness"* as a generic concept and *"vulnerability"* as an instance of a weakness in a product. In other words, when a weakness is discovered in a product, we call it a vulnerability.

We use the word *"product"* to refer to the entire IoT infrastructure (*e.g., Smart Home IoT product*), whereas the term *"device"* refers to the individual IoT devices that together form the *"product."* We have gained IoT vulnerability research experience mostly by pentesting consumer IoT devices widely used globally, IoT devices from local manufacturers, or IoT devices used for public services. For example; smart things (e.g., TVs, watches, speakers, IP cameras), home security sensors (e.g., fire, water, presence), smart home appliances (e.g., doors, refrigerators, vacuum cleaners), smart transportation equipment (e.g., Bluetooth-activated rental e-scooters, ticket readers, drones), and payment systems. Many of these devices have already become an indispensable part of our lives. Our research laboratory has published several IoT pentest reports [6] in recent years.

Over time, we have refined our approach and presented our streamlined IoT vulnerability research methodology in this paper, following the design science research process [7].

2.1 Problem

The problem we faced was to determine the way in which to conduct IoT vulnerability research such that the benefit would be maximized. The review of many studies published as IoT pentest reports revealed the four shortcomings described in Sect. 1. Briefly, limited scope in terms of the IoT attack surfaces, the lack of standards in relation to the IoT weaknesses that need to be tested, complex or no threat modeling and no risk score calculation, and insufficient technical documentation on carrying out attacks for IoT vulnerability discovery and exploitation. We also need to emphasize two challenges related to the people's perspective: high personnel turnover and a shortage of security expertise.

2.2 Objective

Systematic. It is critical to perform activities systematically and according to a basic standard in a laboratory environment where vulnerability research is conducted on a regular basis.
Faster adoption (efficiency). Working systematically accelerates the adaptation of new researchers to the research method and team.
Higher quality (effectiveness). Doing tasks according to a certain minimum standard maintains the quality of the research output above a certain level.

To achieve our overall objective, we address the four shortcomings observed by targeting broader IoT attack surfaces, introducing common IoT weaknesses compilation, favoring lightweight risk scoring, and providing step-by-step IoT testing guidelines.

2.3 Design and development

The approach we introduced to address the four shortcomings is described below. We then build our systematic four-stage IoT vulnerability research methodology on top of these four key elements.

2.3.1 Element 1: Attack surface decomposition

We observed that IoT products are being tested for limited attack surfaces (network and web app pentesting).

The main reasons for this are that more publications are available on IT pentesting topics. Researchers are experienced in IT pentesting, and they are performing vulnerability research in the IoT with the knowledge of IT pentesting. However, research in different domains requires a different specialized skill set. According to our experience, IT pentesting is not sufficient to properly assess the security posture of an IoT product because such devices contain unconventional attack surfaces. Hardware, firmware, and radio communication are essential for IoT devices, which is why vulnerability research should also comprise weaknesses on these surfaces.

The IoT infrastructure is built on top of various components. We choose to decompose it from the attacker's point of view. The proper approach is logical decomposition based on technology. That is why we used the terminology in the offensive security domain instead of the layer-based approach (i.e., perception, transport, and business layers) used in the IoT architecture reference models. Thus, we logically decompose the IoT infrastructure into seven attack surfaces: *hardware, firmware, radio protocol, network service, web application, cloud API, and mobile app*, as in Table 2.

Table 2 IoT infrastructure

Attack surface	Functionality
Hardware	Smart devices
Firmware	OS + utilities + configs
Radio	Local network
Network	Network services
Web	Management app
Cloud	Communication API
Mobile	Controlling app

2.3.2 Element 2: Compilation of top weaknesses

As the *"OWASP IoT Top 10"* project is considered to be the most comprehensive and well-known study, we chose it as our primary source, which is a type of guideline containing ten security risks. Although this checklist could assist with IoT vulnerability research, what is actually required are the weaknesses and associated attacks. Therefore, we first aimed to fortify the process by compiling a list of common weaknesses. We accomplished this by leveraging the *"CWE"* project to map each security risk to its corresponding weaknesses. The key point here is that *OWASP IoT Top 10* and *CWE* are not compatible. Additionally, a specific security risk can arise as a result of various weaknesses. For example, the *OWASP IoT Top 10* checklist contains ten items, whereas the *CWE* database contains close to a hundred records for the hardware attack surface alone.[5]

Moreover, there could be weaknesses in this database that are not related to any item on the checklist.

The challenge here is to determine which weaknesses correspond to each particular risk. Our mapping technique is based on the wealth of knowledge we have gathered as a result of the pentest projects we have conducted thus far. Finally, the *OWASP IoT Top 10* checklist was transformed into 100 security weaknesses with *CWE-IDs*. Additionally, the weaknesses compilation is enriched with clear descriptions, risk impacts, default severity values, and more [3]. The objective here is to determine the concise yet major security tests that need to be performed first to enable researchers to progress quickly without missing the key parts. The compilation of weaknesses is listed in "Appendix A."

2.3.3 Element 3: Lightweight risk scoring

We shortlisted potential weaknesses in the previous step; now, we prioritize them based on the risk score. Quantitatively ranking potential weaknesses or vulnerabilities is the key in decision making, especially in the third stage, exploiting vulnerabilities, which is a relatively more resource-consuming part.

The Common Vulnerability Scoring System *(CVSS)* with 8 base parameters is the de facto risk scoring standard [8]. In addition, *OWASP* also provides a risk rating methodology with 16 parameters [9]. It is a known fact that it is challenging to determine the values of parameters [10]. Additionally, devoting much effort to these calculations may not yield much information or may not pay off. That is why we ended up customizing a commonly known method, *DREAD*, containing five parameters and introduced an alternative lightweight routine with three parameters.

Here, the goal is to give researchers the ability to choose parameters by reasoning rather than by randomly choosing values. We believe that using a simpler method properly for risk calculation would be more useful than using mature techniques incorrectly.

2.3.4 Element 4: Step-by-step pentesting guidelines

It is possible to find guidelines on *network, web, cloud, and mobile pentesting*, but it is not easy to find the equivalent work for IoT-specific attack surfaces. Therefore, we prepared step-by-step guidelines for *hardware, firmware, and radio* attack surfaces. The objective here is to simplify the exploitation stage and encourage novice researchers to work from the attacker's perspective.

3 Application of the methodology

This section explains the use of our four-stage IoT vulnerability research methodology, as illustrated in Fig. 1, which is built on top of the four key elements. It also describes the seven attack surfaces of the IoT infrastructure, various ways of identifying and exploiting vulnerabilities, and execution of the overall pentest.

3.1 Stage 1: Planning

The goal in the first stage is to examine the entire infrastructure to obtain an overall idea of the functionality of the

[5] https://cwe.mitre.org/data/definitions/1194.html.

		Hardware	Firmware	Network	Web	Cloud	Mobile	Radio
Planning	Scoping	Black/Gray/White box & Lab settings & Rules of engagement & NDA						
	Information gathering	OSINT						
		Specifications Visual inspection						Specifications Frequency identification
	Enumeration	Disassembling device Identification of modules	Obtaining firmware Static code analysis	Host discovery Port & version scan	Web page crawling Hidden page discovery	API discovery	App GUI analysis Live traffic capturing	Disassembling device Live traffic capturing
Threat modeling	Attack surface decomposition	Use cases development, Attack surface mapping, Threat classification						
	Vulnerability analysis	Identify potential threats Vulnerability discovery	Reverse engineering Dynamic code analysis	Vulnerability scanning Fuzzing	Vulnerability scanning Vulnerability discovery		Reverse engineering Dynamic code analysis	Fuzzing Vulnerability discovery
		Reuse findings from one surface on another attack surface & Identify the relationships between vulnerabilities & Find ways to chain vulnerabilities & Develop attack paths						
	Risk scoring	(Impact + Coverage + Simplicity*3) / 5						
Exploitation	Known vulnerabilities	Public exploit databases (exploit-db)						
	Exploit development	Manual exploit development						
	Post exploitation	Running post exploitation scripts						
Reporting	Report template	Screenshot & PoC script & Video recordings						
	Vulnerability disclosure	Bug bounty programs						
	CVE	CVE Authorities (e.g., MITRE)						

Fig. 1 Steps of IoT vulnerability research methodology

IoT product. Later on, this stage will allow us to estimate the types of potential weaknesses that could be present in specific attack surfaces.

3.1.1 Scoping

Before starting the vulnerability research, boundaries must be drawn, which we refer to here as scoping. The attack surfaces that will be included in the pentest study are specified, namely *hardware, firmware, radio, etc.* The type of test is defined as a black-box, white-box, or gray-box. The test type directly affects the type of weaknesses that can be discovered, thus it is important to make a well-planned decision. As an example, web application testing is mostly performed using a black-box approach. This type of test for web attack surface has specific limitations due to its nature [11]. The most well-known example is the authentication bypass (e.g., password forgot), which cannot be fully tested. The forgot password function usually sends a password reset URL address, containing a random value via email. The attacker aims to reset the password of a targeted user (e.g., administrator), but one has to guess the random value in the URL address. If we knew how this random value was generated, we might predict it. This does not mean that the white-box approach should always be applied. For example, the device may be damaged in hardware tests while trying to obtain firmware. Briefly, although we favor a white-box approach, the selection usually depends on the context, and we explain how we chose which approach in what context in Sect. 3.2.2. The physical location of the pentest is also stated in this phase, that is, whether the test will be performed in a customer environment or a controlled research laboratory. If we do not purchase IoT products but obtain them from an organization as part of a service, we sign a contract containing initial rules of engagement and a non-disclosure agreement (NDA) to declare the protection of confidential information concerning the parties.

3.1.2 Information gathering

This step is sometimes called open-source intelligence (OSINT), where we usually do not communicate directly with the devices. A comprehensive review of the documentation of the devices is performed, where all assets with functionalities are identified with relevant information. As an example, we identify the type of hardware board (e.g., Arduino, Raspberry Pi, or custom), available ports, slots, and buttons; the firmware version (e.g., proprietary or open-source software); and the operating radio frequency of the device.

3.1.3 Enumeration

In this step, active and passive scanning techniques are applied to each attack surface. Every device belonging to the product is physically examined and interacted with. The main purpose of this step is to collect data that cannot be obtained at the end of a superficial examination. As an example, we disassemble the hardware and identify non-default modules on the board. We obtain the firmware image to externally examine especially the customized features. We

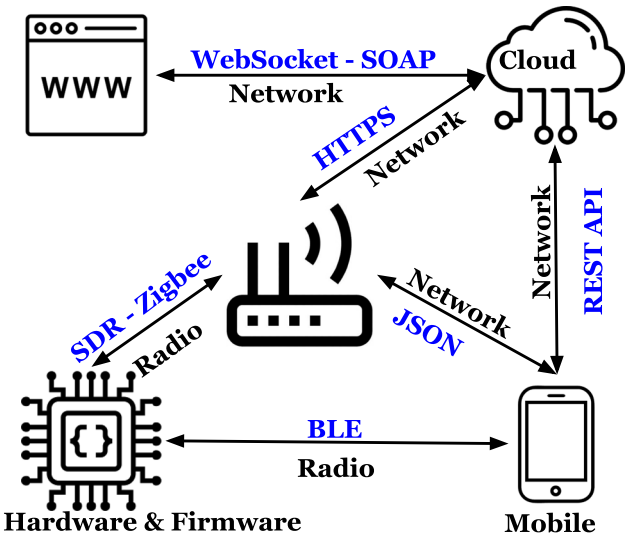

Fig. 2 Architecture mapping

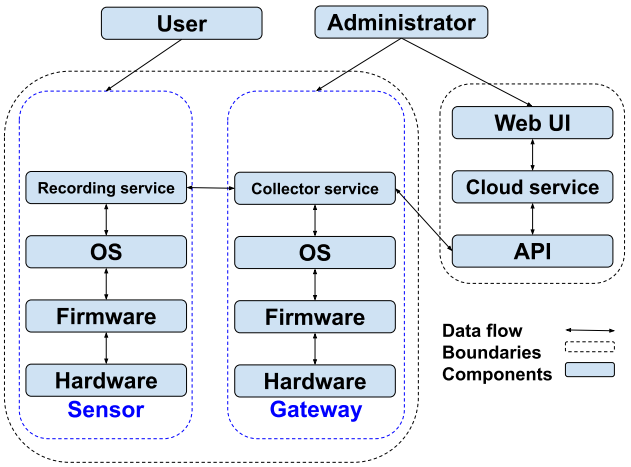

Fig. 3 Component decomposition

investigate whether similar equipment operate in the same frequency range as these devices or whether devices also have an undocumented channel for communication. In light of these inquiries, all hardware and software components and communication protocols are identified. We document the functions of each component item by item, how the functionalities are triggered and ended, and the effects of each action. We share the details of information gathering and enumeration steps of each attack surface and information inquiry templates in our public repository [3].

3.2 Stage 2: Threat modeling

The objective of the second stage, threat modeling [12,13], is to predict the relevant potential threats for each attack surface, estimate the possible impacts of successful exploitation, and prioritize them by calculating risk scores. Our approach requires no specific tools, and the researchers on each project choose the ones they find the most appropriate[6,7,8] providing flexibility.

3.2.1 Attack surface decomposition

We start modeling by identifying the architecture of the IoT product. A general architecture mapping provides a simple overview of the system in its entirety, a high-level interaction of all hardware and software components, and communication protocols, as shown in Fig. 2. Then we break the architecture model down into seven logical components. Component decomposition involves modeling the data flow (communication protocols) between the IoT components, as shown in Fig. 3, and provides a clear view of the potential IoT attack surfaces. The main point to focus on here is which boundaries are crossed when communicating with which component, as it will help us identify the exact entry points.

Secondly, we use the information from the previous stage to derive use cases from the original purpose of a particular function. The number of use case descriptions changes based on the penetration test scope, the number of different devices, and the user profiles. For repeatable testing, it is essential to detail the use case steps required up to the start of the test. The following is an example of use case description steps. We also develop use case diagrams for complex use cases, as illustrated in Fig. 4.

- Place a device <X> and power it up
- Place a hub for device <X> and power it up
- Register an account in the mobile app
- Scan the environment via the app to identify the device <X>
- Add the identified device <X> via the mobile app
- Scan the environment via the app to identify the hub for the device X
- Add the identified hub via the mobile app
- Pair the device with the hub via the mobile app
- The system is now installed and the user has owner privileges

Thirdly, threats related to each function are estimated. To use resources effectively, we must be able to estimate the weaknesses that are most likely to be observed on the basis of attack surfaces. This procedure is relatively difficult to perform and requires prior knowledge. The presented methodology provides a compilation of the most common IoT weaknesses per attack surface.

[6] www.microsoft.com/en-us/securityengineering/sdl/threatmodeling.
[7] https://owasp.org/www-project-threat-dragon/.
[8] https://foreseeti.com/securicad-professional/.

Table 3 Risk assessment

Rating	High (3)	Medium (2)	Low (1)
Damage	Code execution	Authentication bypass, Weak authentication/authorization, Tampering, and Privilege escalation	Sensitive data disclosure and Denial of service
Reproducibility	Bypass prevention—Stable	Bypass prevention only when certain conditions exist	Difficult to bypass prevention—Unstable
Exploitability	Exploit publicly available—Little skill	Need to modify exploit code—Moderate skill	Need to write exploit from scratch—High skill
Affected Users	All users or critical processes are significantly affected	Some users are affected or critical processes are interrupted	Little or no impact on users, nor critical processes
Discoverability	Automated tools—Black-box	Manual intervention—Gray-box	Significant manual review—White-box

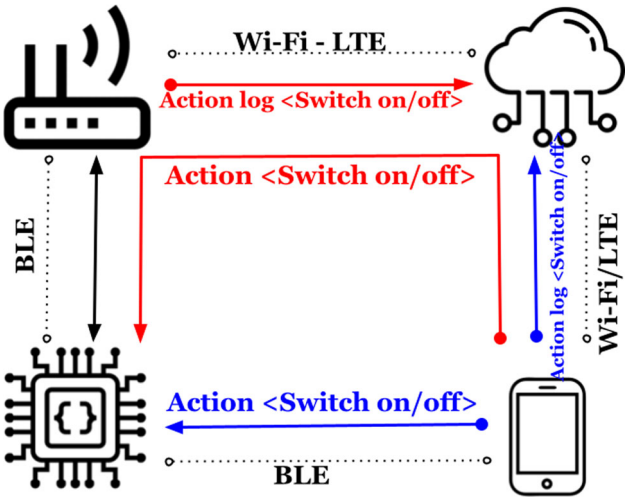

Fig. 4 Use case diagram

We share the details of how to step-by-step estimate the potential threats to an IoT product in our public repository [3].

Finally, we categorize threats at this stage to estimate impact values since we need these values while calculating risk scores. We classify threats similar to *STRIDE* which is one of the most widely known threat classification methods.

There are six categories in *STRIDE*; (S)poofing, (T)ampering, (R)epudiation, (I)nformation disclosure, (D)enial of service, and (E)levation of privilege. We add one more category, which is code execution. According to the experiences we have obtained from our IoT pentests so far, each category has a specific impact. The potential impact of the spoofing category is authentication bypass, for tampering; it is integrity violation; for repudiation, it is weak authentication / authorization; for information disclosure, it is confidentiality violation; for denial of service, it is availability violation, for the elevation of a privilege (privilege escalation) it is unauthorized access, and for the code execution it is the combination of integrity and confidentiality violations.

It is worth remembering that although we assign default values for impacts to make the threat modeling easier, we determine the real impact value after performing the exploitation stage. At the end of this step, we deduce specific attack surfaces, functions (entry points), associated threats, and their categories and impacts.

As pentesting IoT products involves multiple attack surfaces, information about the examined devices is gradually revealed over time. Therefore, we first create a threat model based on the information we have initially collected. We update the threat model based on findings discovered in the exploitation stage. That is why we utilize an iterative threat modeling approach.

3.2.2 Vulnerability analysis

All the information gathered from the attack surface mapping is interpreted in terms of potential vulnerabilities. Firstly, we try to detect vulnerabilities automatically using tools called vulnerability scanners specific to each attack surface (e.g., firmware security scanner). Such tools can identify known vulnerabilities and also point out new ones. On the other hand, in addition to a high rate of false-positive findings, using these tools without fine-tuning can miss vulnerabilities; even vulnerabilities found in public databases cannot be detected. We configure tools according to our needs, which are the estimated threats in the previous step.

Secondly, we conduct a search in public databases[9] for previously published vulnerabilities specific to individual functionalities identified. According to our experience, it is possible to find a working exploit code for a known vulnerability missed by automated tools. That is why a manual examination is vitally important. Beyond that, exploring known vulnerabilities online also inspires vulnerability researchers.

[9] https://cve.mitre.org/cve.

The assumption is that a specific product is developed by a specific group of people who tend to cause similar flaws. It is also known that many vulnerabilities are discovered in specific features of a technology one after another. Even security patches have led to new vulnerabilities. Therefore, examining previously vulnerable components helps gain insight into the target and understand common flaws, eventually it is used to discover new vulnerabilities.

Finally, we put the findings together. We focus on identifying the relationships between vulnerabilities to chain them. We also deal with reusing findings from one surface on another attack surface. We finalize this step by developing attack paths according to the prerequisite relationship of vulnerabilities with each other. For example, identifying the user name by sniffing the network, brute-forcing passwords for authentication bypass, command injection for code execution, enumerating the OS for privilege escalation, and exfiltrating sensitive data, respectively. Therefore, our threat modeling approach relies on entry points, vulnerability data, and attack paths but not only predicted threats. We share the details of the vulnerability analysis steps of each attack surface in our public repository [3].

3.2.3 Risk scoring

We leverage an approach similar to *DREAD* to calculate the risk scores but using fewer parameters. We perceive the (d)amage as impact, the (a)ffected users as coverage, and the (r)eproducibility, (e)xploitability, and (d)iscoverability altogether as simplicity (probability of success) as in Table 3. We assign a value between 1 and 3 to these three metrics, and our risk rating equation becomes $(\text{impact} + \text{coverage} + \text{simplicity} * 3)/5$.

We provide default values for the impact parameter; we set it to 3 for remote code execution, 2 for authentication bypass, weak authentication/authorization, tampering, and privilege escalation, and 1 for information disclosure and denial of service. We determine the impact coverage value based on the rate of the affected user base. We set it to 3 if all users are affected by the threat; we set it to 2 if only a group of users (e.g., privileged users) are affected; otherwise, we set it to 1. For the simplicity parameter, as an example, if we can find exploit code and successfully execute it without making any changes, we set it to 3; if we need to modify the code, we set it to 2; if we need to develop the exploit code from scratch, we set it to 1. We multiply the simplicity parameter by three because it represents three *DREAD* parameters.

If the risk score is greater than 2.5, the severity is high; if the score is less than 2, the severity is low; otherwise, it is medium. Although we also provide a default severity value for each weakness in the compilation, all values are updated based on the previous step, and the final value is set according to the result of the exploitation stage.

3.3 Stage 3: Exploitation

The third stage is exploitation, which is a widely accepted practice to prove whether a system is really vulnerable, and to establish what an attacker could accomplish by taking advantage of this exploitation. If we perform a vulnerability assessment and do not execute this stage, we may not go further than making assumptions about potential threats. Additionally, it is commonly known that chaining a few low-level vulnerabilities could result in a critical vulnerability. However, this cannot be achieved solely with vulnerability analysis. Therefore, realizing exploitation is a valuable exercise.

PatrIoT provides *CWE-IDs* for the IoT weaknesses in the compilation. Since the CWE project is linked to the *Common Attack Pattern Enumeration and Classification (CAPEC)* project, associated attack vectors can be found using CAPEC-IDs[10] can be used to lookup associated attack vectors. In addition, we have developed IoT-specific pentesting guidelines [3]. These consist of instructions to conduct step-by-step attacks and the tools that can be used for exploitation. A comprehensive toolkit is valuable when pentesting IoT devices; thus, customized virtual machines,[11] which contain pre-configured tools, especially for hardware and radio surfaces, are commonly used.

3.3.1 Known vulnerabilities

An exploit is usually a snippet of code written with a scripting language and is run manually from the command line. We usually download exploit code for known vulnerabilities from public databases[12,13]. Sometimes, we may need to modify downloaded exploit code to successfully execute. Additionally, semi-automated tools[14] that contain exploit code built-in can be utilized for easy and stable exploitation. For the known vulnerabilities with no public exploits, we develop our own[15] by leveraging disclosed data regarding the vulnerability.

3.3.2 Exploit development

Automated tools (e.g., vulnerability scanners or fuzzers) help find the signs of vulnerabilities; they may also provide primitive payloads for proof of concept (PoC). These hints are then used to design the real working exploit code for the purpose.

[10] https://capec.mitre.org.

[11] https://attify.com/attifyos.

[12] https://exploit-db.com.

[13] https://packetstormsecurity.com/files/tags/exploit.

[14] https://github.com/rapid7/metasploit-framework.

[15] https://github.com/beyefendi/exploit.

In this step, we explain the common goals when exploiting the vulnerabilities on the basis of each IoT attack surface. The goal of hardware hacking is to gain shell access or access to the firmware image file. Firmware analysis and mobile app testing are conducted to collect helpful information for other attack surfaces, for example, encryption keys (e.g., SSH private key), a hash or password for authentication, libraries used by network services, and compiled or source code of applications. Similarly, radio signal analysis is performed to capture sensitive data or replay commands over the air.

Hundreds of weakness types can be observed in IT systems, enterprise web applications, or networks. However, web apps or network utilities served on IoT devices are developed for certain purposes (e.g., remote administration), so they are prone to certain types of weaknesses. In principle, the purpose of network service exploitation is to gain user-level access to the device and then escalate the privileges to enable full compromise. For web applications, the aim is to execute a command on the device (remote code execution) as well as bypass authentication. The objective of cloud service exploitation is intercepting or modifying the data transmitted by the IoT device and affecting the data analytics.

We observe that security misconfiguration is one of the common weaknesses we encounter. One of the most challenging problems to solve is to determine whether a vulnerability is caused by the product/protocol itself or its setup. For example, during IoT device setup, if encryption is not set for a protocol, the network/radio traffic can be sniffed, and sensitive information can be captured. This finding is related to the configuration of the device and not a vulnerability that can be reported to the manufacturer. In rare cases, researchers do not purchase the device but receive it from an intermediary organizations (e.g., drone from military). In these cases, the misconfiguration issue is reported to the organization (e.g., military) as well. However, if this setting is due to the preconfiguration of the device in factory settings or if there is no option to set encryption, the manufacturer is notified about the vulnerability. That is why we should indicate the cause of the vulnerability in the reports. While a penetration test report is provided to the third parties, a vulnerability disclosure report is authored for manufacturers.

3.3.3 Post-exploitation

Post-exploitation activities are performed right after gaining user-level access to the firmware of the devices. The common activities are persistence, privilege escalation, lateral movement, pivoting, evasion, data exfiltration, covering tracks, and password cracking. In IoT pentesting our major focus is elevating privileges and password cracking.

Recall that we can reuse findings discovered from the analysis of another attack surface. Therefore, once gaining access to the target device, we can have multiple options for privilege escalation. For example, we can discover a credential while reversing the mobile app; then, we can use it to elevate to admin privileges without much effort while pentesting the network services.

Additionally, after gaining full access, we extract credentials from devices and crack them to build an IoT-specific hash and password database. Particularly, hard-coded passwords in firmware are useful when performing tests for devices from the same vendors. Moreover, researchers can go further depending on multiple criteria (e.g., the importance of the device for national security). As an example, a drone's radio communication is jammed to disconnect from the command control center. It is forced to land at a desired location, so the device is hijacked. Then a backdoor firmware is flashed onto the device, and it is dispatched to its owner to spy on.

Some devices may have various security mechanisms, albeit primitive (e.g., blacklisting and rate limiting); in such cases, it is necessary to bypass the prevention mechanisms. Additionally, while testing an IoT environment containing multiple devices (such as sensors and gateways), the key activities become pivoting, lateral movement, and persistence. Such post-exploitation activities are worth underlining as they practically mean turning a particular IoT network into a botnet.

3.4 Stage 4: Reporting

Vulnerability research is concluded with a document, pentest report, that is the fourth and final stage of the activities. Particularly, the discovery of a previously unknown vulnerability would require additional steps to be taken, for those, we author vulnerability disclosure report. We share both report templates in our public repository [3]. As we report our results with a standard template that familiarizes reviewers and allows authors to reuse previous reports. Using the same template for each pentest makes it easy to generate statistics and conduct searches in previous pentest reports.

3.4.1 Report template

An IoT pentest report contains eight sections. The first section is "General information," which includes the service recipient, the time period, the devices under test, the objective, and what information is included in the report. The second section is "Scope," which identifies the assets and type of test (black-box or white-box). The third section is the "General testing methodology," which describes the steps followed to perform the tests. The fourth section is "Risk assessment," which defines the meanings of the vulnerability severity levels. The fifth section presents the "Executive summary," one page of information on the devices tested, what types of tests failed, what types of tests were success-

ful, and what actions were taken. The sixth section is the "Technical summary," which explains the "Executive summary" in detail. The seventh section contains three tables. The first is the "Summarized vulnerabilities table," which shows the vulnerabilities of devices with their severity. The second is the "Known vulnerabilities table," which lists previously known vulnerabilities (e.g., vulnerabilities in the public exploit database). The third is the "All attacks table," which contains all the weaknesses tested and the attacks performed. The eighth section is "Findings," which details which vulnerabilities have been discovered on which devices. This section contains the following information for each vulnerability: a brief description of the vulnerability, attack surface, attacker profile, the techniques used to find the vulnerability, references to the identified vulnerability, impact and severity of the vulnerability, PoC scripts for exploitation, screenshots, steps to reproduce exploitation, and references for remediation. Unlike an IT penetration test, an IoT pentest report also has its own characteristics; mainly, sections dedicated to hardware and radio components contain high-quality photographic images and video demonstrations.

3.4.2 Vulnerability disclosure

Public disclosure is the practice of announcing security vulnerabilities discovered in software or hardware products. As usual, stakeholders have different priorities. Security researchers prefer to publish their vulnerability discoveries as soon as possible and receive recognition. Beyond that, notifying the manufacturer also confirms the vulnerability and provides credibility, so this notification is highly important for researchers. Customers of vulnerable products or services also ask that the systems be patched as quickly as possible. On the other hand, product owners favor the disclosure of vulnerabilities only to themselves to have enough room to fix flaws and avoid affecting user loyalty. The conflicts of priorities have been resolved to some extent by the appropriate implementation of the disclosure policies (responsible, coordinated, self, 3rd party, vendor, full, etc.).

A typical responsible disclosure policy includes the steps below. A researcher discovers the security vulnerability and then documents the findings, including supporting evidence (a PoC script and screenshots) and a disclosure timeline. The researcher then notifies the vendor directly or via a bug bounty program by submitting the report through secure channels. Once the system provider accepts the finding, they have a reasonable time frame within which to patch the vulnerability. Once a patch is available, product owner informs the public by issuing a security advisory report. Industry vendors[16] generally agree that a maximum deadline of 90 days is acceptable.

This approach helps people or organizations using vulnerable products take their precautions.

Companies determine how findings will be reported to them. Bug bounty programs, which act as an intermediary between vulnerability researchers and product owners, are popular. They assist in protecting both parties from potential conflicts (e.g., subjective interpretations) and provide a professional process for companies unfamiliar with vulnerability disclosures. The discoveries are submitted to these platforms that enable researchers to receive recognition and earn moral rewards (*e.g., appearing in the hall of fame*) [14]. Companies that receive bug reports can prevent detailed information about the vulnerability from being published publicly.

Several IoT device producers collaborate with bug bounty programs [17,18]

In the full vulnerability disclosure model [15], researchers also publish a working exploit script in addition to vulnerability details. There are well-known outlets[19,20,21] where multiple exploits are published daily. This practice has also been regularly performed live at prestigious conferences (e.g., Blackhat and Defcon). These are not illegal platforms but are supported by vulnerability researchers leading the industry. The objective here is not to back up the potential attackers but to share offensive security knowledge with the community to allow vulnerability researchers to keep advancing.

3.4.3 CVE

MITRE Common Vulnerabilities and Exposures (CVE)[22] is a limited database of known security vulnerabilities of systems. Once a new vulnerability is discovered, a security researcher may want to have a *CVE-ID* assigned to his new finding[23] for attribution. If the vendor is allowed to provide it, the researcher first contacts the vendor to request a *CVE-ID*; otherwise, the researcher applies to *MITRE* to obtain one.

Zero-day. The term 0day refers to a product vulnerability that is unknown to its owner or manufacturer. The widespread use of the product and the impact of the vulnerability play a critical role in determining whether a vulnerability could be categorized as zero-day [16]. They usually deserve a monetary value, and both the manufacturer and the adversaries are willing to pay for them. For example, although an unauthenticated code execution vulnerability is usually referred

[16] https://googleprojectzero.blogspot.com.

[17] https://bugcrowd.com/solutions/iot-and-web3/.

[18] https://hackerone.com.

[19] https://exploit-db.com.

[20] https://packetstormsecurity.com/files/tags/exploit.

[21] https://github.com.

[22] https://cve.mitre.org.

[23] https://cve.mitre.org/cve/request_id.html.

Table 4 Tested devices

Study	Smart device	Attack surface
1	AI robot	Web
2	Ryze tello drone	Radio—Network
3	Samsung smart fridge	Network—Cloud
4	Xiaomi Mi home security camera	Network—Firmware
5	Yale L3 smart door lock	Mobile
6	Yanzi air quality sensor	Radio
7	Xiaomi Mi home security camera	Hardware

Table 5 Steps to execute IoT vulnerability research

Step
Decompose the device into seven attack surfaces
Conduct information gathering with the provided template to identify the functionalities of the device
Derive different use cases from the original purpose of the functions
Match each function to the items in the weaknesses compilation provided based on attack surfaces
Update the default impact values provided, if necessary, taking into account the functions of the product
Run automated tools to identify potential vulnerabilities and eliminate false positives
Check for known vulnerabilities in the public databases
Identify the relationships between vulnerabilities to chain them
Reuse findings from one surface on another attack surface
Develop attack paths according to the prerequisite relationship of vulnerabilities with each other
Calculate risk scores by determining the simplicity and reusing the default impact, coverage, and severity values
Start analysis with high-risk scores or high-impact vulnerabilities
Employ provided IoT-specific pentesting guidelines
Develop exploit code or use semi-automated tools
Get screen shots
Generate the penetration test report using the provided template

as zero-day, an information disclosure flaw is not considered to belong to this category.

4 Evaluation

In the last two years, we have conducted pentests on more than 30 IoT devices, identified dozens of vulnerabilities, and presented thorough documentation of our findings [6]. We have studied and reported the systematic application of this methodology from start to end. In this section, we present the results of our vulnerability research on seven IoT devices to demonstrate the practical application of *PatrIoT* (Table 4).

The threat models created for the seven chosen devices are accessible as a separate document [3]. How these threat models are generated is found in the individual pentest reports. Figure 5 shows the attack surfaces tested in each study, and the vulnerabilities discovered by using our compilation of weaknesses.

4.1 Robot

The first study explored the web and network attack surfaces of an *AI robot*. **Information gathering.** First, *nmap* identified an open HTTP port using network scanning. Then, *dirbuster* discovered the website map, the web pages it actually contains, and *nikto* allowed us to determine the development technologies along with their versions. After examining these outputs, it was observed that the HTTP service provides a web service to authenticate users with password-based authentication. **Vulnerability analysis.** Given the information from the enumeration phase, the threat model is based on authentication bypass and DoS scenarios. **Exploitation.** Examination of the authentication process with *Burp Suite* revealed that the communication was not encrypted. The password input entered by the user was hashed on the client side and then transmitted. Even though this provides a layer of security, the hash value could still be cracked by rainbow table attacks. As these problems relate to transport security, the weakness was recorded as involving a lack of transport encryption.

No	Web weaknesses	Study 1
1	Sensitive data exposure	No
2	Lack of transport encryption	**Yes**
3	Insecure SSL/TLS issues	No
4	Authentication - Username enumeration	No
5	Authentication - Weak credentials	No
6	Authentication - Improper account lockout	No
7	Authentication - Weak password recovery	No
8	Authentication - Lack of two-factor authentication	No
9	Authentication bypass - Web application to cloud	No
10	Command injection	No
11	Direct object references	No
12	Business and logic flaws	No

No	Cloud weaknesses	Study 3
1	Lack of transport encryption	**Yes**
2	Insecure SSL/TLS issues	No
3	Authentication - Username enumeration	No
4	Authentication - Weak credentials	No
5	Authentication - Improper account lockout	No
6	Authentication - Weak password recovery	No
7	Authentication - Lack of two-factor authentication	No
8	Vendor APIs - Inherent trust of cloud or mobile application	No
9	Vendor APIs - Authentication bypass	No
10	Vendor APIs - Authorization bypass	No
11	Vendor APIs - Undocumented backdoor API calls	No
12	Vendor APIs - User data disclosure	No
13	Vendor APIs - Device information leakage	No

No	Firmware weaknesses	Study 4
1	Sensitive data exposure - Hardcoded credentials	No
2	Sensitive data exposure - Backdoor accounts	No
3	Sensitive data exposure - Encryption keys and algorithms	No
4	Sensitive data exposure - Other sensitive information	No
5	Sensitive data exposure - Static and same encryption keys	No
6	Configuration - Lack of data integrity checks	No
7	Configuration - Lack of wiping device	No
8	Configuration - Insecure customization of OS platforms	No
9	Configuration - Lack of security configurability	No
10	Configuration - Insecure filesystem permissions	No
11	Authentication bypass - Device to device	No
12	Authentication bypass - Device to mobile application	No
13	Authentication bypass - Device to cloud	No
14	Update mechanism - Missing update mechanism	No
15	Update mechanism - Lack of manual update	No
16	Update mechanism - Lack of transport encryption	No
17	Update mechanism - Lack of signature on update file	No
18	Update mechanism - Lack of update verification	No
19	Update mechanism - Lack of update authentication	No
20	Update mechanism - Intercepting OTA update	No
21	Update mechanism - Backdoor firmware	No
22	Update mechanism - World writable update location	No
23	Update mechanism - Lack of anti-rollback mechanism	No

No	Study
1	Can a robot's confidentiality be trusted?
2	Hacking a Wi-Fi based drone
3	Ethical Hacking of a Smart Fridge
4	Are modern smart cameras vulnerable to yesterday's vulnerabilities?
5	Security analysis of a smartlock
6	Threat Modeling and Penetration Testing of an IoTproduct - A Survey on the Security of a Yanzi IoT Network
7	Penetration testing of IoT devices: The ethical hacking of a smart camera

No	Radio weaknesses	Study 2	Study 6
1	Lack of transport encryption	No	No
2	Insecure SSL/TLS issues	No	No
3	Lack of message integrity check	No	No
4	Lack of signal replaying checks	No	No
5	Lack of signal jamming checks	No	No
6	Lack of signals spoofing checks	No	No
7	Lack of Denial of service (DoS) checks	No	No

No	Hardware weaknesses	Study 7
1	Sensitive data exposure - Device ID/serial no	No
2	Firmware/storage extraction - Insecure external media interface	No
3	Firmware/storage extraction - Download from the Web	No
4	Firmware/storage extraction - Insecure SPI interface	No
5	Firmware/storage extraction - Insecure I2C interface	No
6	Firmware/storage extraction - Insecure UART interface	No
7	Firmware/storage extraction - Insecure JTAG interface	No
8	Firmware/storage extraction - Insecure SWD interface	No
9	Firmware/storage extraction - Insecure SoC	No
10	Firmware/storage extraction - Insecure eMMC chip	No
11	Backdoor firmware - Insecure UART interface	No
12	Backdoor firmware - Insecure JTAG interface	No
13	Backdoor firmware - Insecure SWD interface	No
14	Grant shell access - Insecure UART interface	**Yes**
15	Grant shell access - Insecure SPI interface	No
16	Change code execution flow - Insecure JTAG/SWD interface	No
17	Reset to insecure state	No

No	Mobile weaknesses	Study 5
1	Sensitive data exposure - Hardcoded credentials	No
2	Sensitive data exposure - Encryption keys and algorithr	No
3	Sensitive data exposure - Other sensitive information	No
4	Authentication - Username enumeration	No
5	Authentication - Weak credentials	No
6	Authentication - Improper account lockout	No
7	Authentication - Weak password recovery	No
8	Authentication - Lack of two-factor authentication	No
9	Authentication - Mobile application to cloud system	No
10	Insecure authorization	No
11	Implicitly trusted by device or cloud	No
12	Lack of transport encryption	No
13	Insecure SSL/TLS issues	No
14	Insecure data storage	No
15	Outdated 3rd party libraries and SDKs	No
16	Business and logic flaws	No
17	Lack of health checks	No

No	Network weknesses	Study 1	Study 2	Study 3	Study 4
1	Sensitive data exposure	No	No	No	No
2	Lack of transport encryption	No	**Yes**	No	No
3	Insecure SSL/TLS issues	No	No	No	No
4	Authentication - Username enumeration	No	No	No	No
5	Authentication - Weak credentials	No	No	No	No
6	Authentication - Improper account lockout	No	No	No	No
7	Authentication - Weak password recovery	No	No	No	No
8	Privilege escalation	No	No	No	No
9	Authentication bypass	No	No	No	No
10	Denial of Service (DoS)	**Yes**	**Yes**	No	No
11	Buffer overflow	No	No	No	No

Fig. 5 Identified vulnerabilities

Additionally, checking 11 network weaknesses contained in our compilation indicated that the robot was running an old Apache webserver with known DoS vulnerability.

4.2 Drone

The second study explored the network attack surface of a Wi-Fi-based drone, *Ryze Tello* from *DJI*. **Information gathering.** The *airmon-ng* identified that the Wi-Fi network protocol is WPA2. A review of the documentation revealed that the Wi-Fi network by default requires no passwords to connect. However, the user has the option to configure the use of a password. This means that any device with access to the Wi-Fi network can send commands to the drone. **Vulnerability analysis.** Given the information from the previous phase, the threat model highlights the risk of authentication bypass, tampering with communication, and DoS scenarios. **Exploitation.** First, *airodump-ng* and *aircrack-ng* can perform brute-force attacks on devices with WPA2 authentication. Second, an SYN flood with *hping* could have the result that the drone and controllers lose communication with each other. Therefore, this was recorded as a DoS vulnerability. Finally, an ARP spoofing attack using the *arpspoof* tool allowed traffic to be rerouted to the attacker's system, resulting in intercepting the traffic between the drone and controller using the *Wireshark* tool. This was recorded as MitM vulnerability. Testing the device for the 11 network weaknesses contained in our compilation indicated that the communication was also unencrypted; hence, it was recorded as a lack of transport encryption vulnerability.

4.3 Fridge

The third study was conducted to evaluate the network and cloud attack surfaces of a smart fridge, *Family hub*, from *Samsung*. **Information gathering.** Passive network sniffing with *Wireshark* identified outgoing traffic from the fridge to a cloud-based web service. **Vulnerability analysis.** Given the information from the previous phase, the threat model was based on tampering with communication. **Exploitation.** The communication between the fridge and cloud components is not only transmitted via HTTPS but also via HTTP, thus the traffic can be intercepted in plaintext. This was recorded as a lack of transport encryption vulnerability.

4.4 Camera

The fourth study evaluated the firmware and network attack surfaces of a web camera, *Home security camera* 360° *& cloud* from *Xiaomi Mi*. **Information gathering.** A documentation review revealed that webcams serve a webpage for management purposes. In addition, they provide authentication with a username and password, can be controlled remotely, and offer live broadcasting over the RTP protocol. **Vulnerability analysis.** Given the information from the previous phase, the threat model was based on authentication bypass and communication tampering. **Exploitation.** The previous version of this camera solution had various high-severity security vulnerabilities. All of these vulnerabilities seem to have been addressed in the latest version of the camera, hence this product is secure according to our baseline.

4.5 Door lock

The fifth study evaluated the network and mobile attack surfaces of a smart lock, *L3*, from *Yale*. **Information gathering.** A documentation review revealed that the smart lock requires authentication via a mobile app. **Vulnerability analysis.** Given the information from the previous phase, the threat model was based on authentication bypass and communication tampering. **Exploitation.** Network sniffing indicated that the communication is encrypted, and reverse engineering of the mobile app does not yield sensitive information. The previous version of this door lock device had several vulnerabilities, but they were patched in the final version. Therefore, this product is secure according to our baseline.

4.6 Air quality sensor

The sixth study evaluated the radio attack surface of an air quality sensor, *Comfort*, from *Yanzi*. **Information gathering** Sniffing the radio signals via *HackRF* identified the radio protocol as *6lowpan*. **Vulnerability analysis.** Given the information from the previous phase, the threat model was based on authentication bypass, communication tampering, and DoS scenarios. **Exploitation.** Network sniffing led to the discovery that the communication is encrypted. Testing the device against the 7 radio weaknesses contained in our compilation indicated that the device passed all security checks.

4.7 Camera 2

The seventh study evaluated the hardware attack surface of a web camera, *Home security camera* 360° *& cloud* from *Xiaomi Mi*. **Information gathering.** A physical examination of the device board revealed that TTL pins (TX and RX) are identifiable with a multimeter. **Vulnerability analysis.** Given the information from the previous phase, the threat model was based on granting shell access. **Exploitation.** Connecting to the TTL pins using a *buspirate* allowed to gain root shell form the Linux console. With this access, the entire firmware was extracted from flash memory. Although a part of the firmware is encrypted, the Wi-Fi password was observable in clear-text with *binwalk*.

4.8 Evaluation summary

First, before we had *PatrIoT*, the reported findings did not follow standard weakness naming conventions (e.g., Interception and modification vs. Man-in-the-Middle). Therefore, even if the same vulnerability was discovered by different researchers, we were not able to correlate the finding. *PatrIoT* fundamentally solved this problem. Second, we observed that some specific vulnerabilities were never reported by researchers. For example, when researchers failed to find a valid password after performing an uninterrupted brute-force attack, they did not report it as a weakness. If there is no limit for failed attempts, *PatrIoT* considers it an "Improper Account Lockout" weakness. Third, some vulnerabilities (e.g., DoS and ARP spoofing) were sometimes reported as protocol vulnerabilities, causing inconsistencies. *PatrIoT* explicitly includes such weaknesses in its compilation of top weaknesses to ensure researchers that the discovered weaknesses are related to the device itself and not the protocol. Finally, our pentest report template document guides researchers in documenting why an attack fails. In this way, we can make sure that an attack has been performed in a proper way but failed due to some type of prevention mechanism (e.g., intrusion prevention system).

5 Discussion

In this section, we discuss the benefits and limitations of using the presented methodology. Recall that we observed four shortcomings in IoT vulnerability research process, which we addressed by introducing four key elements: logical attack surface decomposition, compilation of the top 100 weaknesses, lightweight risk scoring, and step-by-step pentesting guidelines. While addressing these limitations, we also considered the typical research environment: dozens of IoT devices being subjected to the pentest, high personnel turnover, and a shortage of IoT security expertise. We then developed a four-stage systematic methodology on top of the four key elements to reveal the maximum benefit to be obtained from pentesting studies.

5.1 Compilation of top weaknesses

Standardization. As we could not find standard guidelines for testing IoT weaknesses, the purpose was to compile a list that contains the minimum but major security tests that need to be performed first. We developed a compilation for 100 weaknesses by matching *OWASP IoT Top 10* to the *CWE*, which allowed us to follow a standard in all studies. Defining a baseline also brings in six benefits: (1) A shortlist motivates researchers from the start because it is short and provides a road map. (2) It enables us to easily benefit from the previous reports when pentesting the upgraded version of the device or a similar device from a different vendor. Reusing previous knowledge allows faster progress. (3) In addition, researchers do not tend to document failed attacks; to a certain extent, using a standard template also prevents work that has been carried out from being wasted and avoids potential duplicate effort. (4) Even if we do not discover any vulnerabilities during a pentest, we can claim that the system is secure against such specific attacks because we perform the pentest following particular stages and document all the results. (5) Moreover, performing the same tests enables the security level of two similar devices to be compared (e.g., we can compare all home security cameras that were tested and identify the weaknesses that are common to all the cameras). (6) Performing and documenting the same security tests also facilitate the generation of statistics to visualize which vulnerabilities are more common in total and which vulnerabilities are starting to trend.

Argument 1. Is the number of weaknesses to be tested sufficiently broad to assist in performing all existing attacks for IoT? Because our goal is to identify a high potential vulnerability in a limited period, it is not reasonable to perform all the tests. It would make more sense to start with a shortlist of the most common IoT weaknesses. Actually, this is why this methodology is agile. On the other hand, suppose the result of a pentest is not satisfactory (e.g., no vulnerabilities found). In that case, the researcher extends the coverage by including new potential weaknesses to test, and thus new attack vectors. In the first step, it makes sense to refer to well-known individual books on hardware hacking, firmware analysis, radio hacking, etc. In addition, as previously mentioned, the *CWE* database lists more than a thousand weaknesses and *CAPEC* database contains more than a half-thousand attacks. Thus, these databases can be beneficial to understand the remaining weaknesses and attacks while extending the weaknesses compilation.

Moreover, we strongly suggest documenting the personal knowledge and experience, preferably as a pentesting guideline, gained from the vulnerabilities discovered during research activities. In reality, times keep changing, and threats evolve accordingly. Eventually, we would need to update this compilation based on the information harvested from the performed work. Extending the presented weaknesses will increase the number of attacks and extend the duration of the study accordingly.

Argument 2. New measures are constantly being taken against security vulnerabilities. How can *PatrIoT* keep up with this continuous evolution? There are many attack techniques against just one weakness. While attack techniques evolve as a result of countermeasures against vulnerabilities (e.g., Authentication bypass in a specific product), the weaknesses (e.g., Authentication bypass via weak password reset) usually remain the same to a certain extent. Therefore, we can

assume that the emergence of new weaknesses is slower than the development of new attack techniques. That is why the compilation of weaknesses is affected over time. In other words, the emergence of new attacks does not directly affect the compilation of weaknesses. However, the situation differs when new types of weaknesses emerge and spread, and they enter the top 100 list after a while. In this case, the weaknesses compilation must be updated. The precaution for this is that we publicly share all the artifacts of *PatrIoT*, including the weaknesses compilation and penetration testing guidelines, on GitHub [3]. In this way, we hope *PatrIoT* will be a live project, and it will be updated by its users.

Argument 3. Why did *PatrIoT* not rank the vulnerabilities in the entire *CWE* database in the order of priority rather than selecting the top 100 weaknesses? The main reason for this is that there is neither the time nor the budget to apply all of these tests, even if we were able to sort them. Sorting the entire corpus would be an unnecessarily and unimaginably heavy workload, with the high potential of low payback. Thus, we decided to distill the most common weaknesses without sorting this large database, relying on the *OWASP IoT Top 10* project. This list provides the top 10 security risks, which we mapped to the corresponding weaknesses in *CWE* database. This is literally why we based our study on the *OWASP IoT Top 10* project.

5.2 Threat modeling

Decomposition. Logical decomposition makes it easy to start building threat models for IoT products. First, *PatrIoT* recommends seven attack surfaces that simplify conceptualization and ensure none of the components are overlooked. Second, technology-based decomposition allows estimating and categorizing threats with the help of our weaknesses compilation. Third, it also enables dividing the tasks to enable more specialized researchers on certain attack surfaces to work. Correspondingly, it helps execute specific activities efficiently and produce higher-quality outputs.

Recall that this study is based on *OWASP IoT Top 10*, which has its limitations. Although we could extend the coverage with our initiative, we did not consider that for the following reasons. Firstly, humans may represent the weakest link in the cybersecurity chain, and consumer IoT devices are usually configured by people who have less experience in security. Therefore, it is reasonable to have an attack surface related to social engineering. But as we mentioned, contrary to IT pentesting, IoT pentesting is often performed in a laboratory setting, even if researchers obtain IoT products from organizations. These environments contain tools specific to IoT analysis, such as hardware and radio analysis kits. Additionally, since we do not choose to serve organizations directly, we cannot perform social engineering attacks against real users. We believe it would also be not be credible to perform social engineering attacks with artificial users. Therefore, social engineering attacks on IoT product users are beyond our scope.

Secondly, this study describes the hardware as an IoT-specific attack surface and handles it meticulously. However, it does not cover some of the threats in a cyber and physical context. It does not consider, for example, what risks would arise if the device (a physical entity) interacted with the physical environment. As a concrete example, an attacker compromises a speaker (actuator) and microphone (sensor) to send voice commands to another sensor (e.g., an air conditioner).

Last but not least, although PatrIoT does introduce a separate privacy attack surface, it includes sensitive data disclosure weaknesses in all attack surfaces. Therefore, it covers personally identifiable information (PII) issues.

Categorization. The weaknesses compilation already divides potential weaknesses into seven categories based on their attack surfaces. Why is there another classification for threats with a *STRIDE*-like approach?

As we previously mentioned, each STRIDE category has a specific impact. Impact value is important for researchers as it allows them to filter specific impact categories. Recall that the impact value is used in the calculation of the risk score as well. Naturally, researchers' priority is to start with vulnerabilities with high-risk scores and progress toward those with low-risk scores. Impact categorization gives researchers the flexibility to start with high-impact vulnerabilities. For instance, we prefer spending more time on vulnerabilities with an authentication bypass or code execution impact, even if they have low-risk scores, than on vulnerabilities with a denial-of-service impact, even if they have high-risk scores.

There are some advantages of designing threat models using *STRIDE*-like categories. Threat models can be found for many technologies which can be partially reused. Additionally, when a threat model is built, categories are checked to see if anything is missing. On the other hand, the information to be used during the exploitation stage is the attack paths. It is hard to get accurate information from a *STRIDE*-based model. For example, it is not easy to see where to start an attack and which path to follow to exploit a potential vulnerability that prerequisites another exploit to be successful. That is why we use *STRIDE* only for categorization but rely on vulnerability data and attack paths in threat modeling.

Vulnerability analysis. Vulnerability data complements our IoT attack surface decomposition. It helps identify the relationships between vulnerabilities, reuse findings from one surface on another attack surface, find ways to chain vulnerabilities, and develop attack paths.

Reuse and familiarity. As threat models are designed using the same technique, they follow a type of template. In this way, we can easily benefit from the previous threat model when we test a similar but new device. Reusing previous

threat models allows for faster progress as well. Additionally, it accelerates the comprehension of potential weaknesses for those who review the threat models.

5.3 Lightweight risk scoring

Although our methodology promotes the most common threats, starting pentesting with high potential weaknesses increases motivation once again. There are some advantages of using our risk scoring method. As we simplify the commonly known method, *DREAD*, we avoid risk scoring being an overhead. Our equation contains only three parameters, and it is easy to calculate. Additionally, we provide default values for two parameters, impact and coverage. Naturally, we cannot provide a default value for the simplicity parameter, so we cannot know the exact risk score. However, based on our IoT vulnerability research experience, we also provide default severity values for each weakness defined in the compilation. Default values were assigned for convenience only.

On the other hand, it is reasonable for a risk scoring formula to have a parameter related to the possibility of attack detection by security systems. Compared to the other three parameters, we have experienced that predicting such a parameter is much more difficult and prone to error. In addition, such prevention systems are still not common in IoT products. Furthermore, since we built our model to simplify the existing *DREAD*, we did not find it appropriate to add a new parameter that does not exist in *DREAD*.

It is worth highlighting that our priority is to use *CVSS* scores as long as researchers can calculate them or we obtain them from *CVE* authorities. We recommend our alternative method for researchers who experience difficulty in calculating *CVSS*, *OWASP*, or *DREAD* scores. As usual, any approach may have its flaws, but as long as we apply the same technique to all, we can maintain some consistency among the severity ratings. So we can correctly rank the estimated threats first and then the vulnerabilities identified.

5.4 Step-by-step pentesting guidelines.

These practical documentations constitute a very goal-oriented set of information compared with security risks or hundreds of pages of books. As they directly support IoT pentesting, they simplify the activities and reduce the time required to conduct the study.

Although the penetration testing guidelines are one of the extra contributions, it would be more beneficial to update them with the new attack techniques in the long run. *PatrIoT* is compatible with *MITRE*, so the compilation of weaknesses also refers to the *CWE-IDs*, which is the ID of the weaknesses provided by *MITRE CWE*. As *CWE-IDs* are mapped to *CAPEC-IDs*, they can be utilized to keep up with new attack techniques as well.

0day discovery. In light of the 0day definition in Sect. 3.4.3, how can the methodology support zero-day discovery? *PatrIoT* decomposes attack surfaces into seven logical components. It highlights the findings on one surface can contribute to another attack surface. It also emphasizes that chained vulnerabilities can have higher impacts. In summary, when the information obtained from one attack surface is reused on another attack surface, and different vulnerabilities are associated, it can lead to 0day vulnerabilities. The step-by-step IoT pentesting guidelines [3] contain information about interpreting findings and providing them as input each other. For example, a 0day finding can be discovered by obtaining information that will enable the authentication bypass process and combining this vulnerability with a code execution vulnerability. The compilation of the top 100 weaknesses contains records causing these types of vulnerabilities.

5.5 Automation

Why did *PatrIoT* not introduce a tool that fully automates the exploitation stage of this study? As mentioned in Sect. 1, we need three resources for vulnerability research: people, processes, and technology. In this study, our focus was on developing a methodology rather than on developing a tool. In addition, as mentioned in the exploitation stage, we use multiple mature tools known by the community. Although we prefer the use of automated attacks, this does not mean hitting the start button and obtaining results. As *PatrIoT* introduces seven attack surfaces for IoT ecosystem, the findings (e.g., decryption key) on one surface could be reused for other surfaces.

Moreover, a critical severity vulnerability can be discovered only by chaining two or more vulnerabilities. It is worth remembering that reusing a finding and linking vulnerabilities requires serious manual intervention since it is not an approach that automated tools can achieve.

5.6 Lessons learned

Initially, we planned to carry out this study in a different way. That is, we would develop an automated tool, sort of a webpage crawler. It would search for the keywords (e.g., smartwatch) in the CVE database parse and extract the information from the retrieved webpage. We would extract the information we provided within the scope of this study (name of the vulnerability, type of the weakness, impact, severity, CVSS score, *CWE* and *CAPEC* category matching with the CVE, etc.). With the information we collected, we could make a top 100 list. This way, the study would be both a large-scale analysis and the top 100 list would be updated frequently. Although the CVE database looks like a struc-

tured structure, unfortunately, these fields are not mandatory, so they are not filled. That is why we could not make it.

5.7 Challenges

The notable limitation of the present study is that the number of weaknesses to be investigated is limited to 100. Recall that the CWE database, containing thousands of types of weaknesses and constantly expanding, can be further examined if more extensive tests are needed. However, this is a very demanding task. For this reason, we suggest expanding the weakness set in line with the findings obtained as a result of the application of the present method.

Even if the methodology was entirely flawless, the quality of work usually depends on the researcher's competence. An IoT vulnerability researcher would focus on the pentest by prioritizing mostly IoT-specific attack surfaces (hardware, firmware, and radio protocols). However, this requires a background and skills specific to IoT, which means it is not easy to shift from the IT pentester role to the IoT pentester.

5.8 Future work

One of the future tasks is keeping this weakness compilation up-to-date. We have been employing *PatrIoT* in our research and will continue to validate it in future studies. But it will be the most productive contribution if additional IoT vulnerability research teams utilize this method and share their results. That is why we share all artifacts in a public repository so that other researchers can use this method more easily and new ideas can be reported. With community support, this list can be kept up-to-date over time.

6 Related work

Information technology (IT) pentesting, or traditional pentesting, is a well-studied topic covering mostly testing networks, operating systems, and web applications. Generally, researchers [21–23] agree on a 5-step approach for pentesting, which comprises (1) information gathering (e.g., open-source intelligence (OSINT)); (2) network mapping (system reconnaissance); (3) vulnerability scanning; (4) exploitation; and (5) post-exploitation. In addition, the *"Information System Security Assessment Framework (ISSAF)"* [4] and the *Open Source Security Testing Methodology Manual (OSSTMM)* [5] are often referenced and employed as penetration testing guidelines. Today, experiments with pentesting techniques can be conducted on various cloud-based pentest practice platforms[24,25] and there are several certification options for demonstrating pentesting skills[26,27] Finally, several mature techniques and tools exist for conducting an IT pentest.[28]

The traditional pentesting has been extensively detailed over several years, yet the situation is not the same for the IoT ecosystem. A methodical comparison of similar studies having an attacker's perspective and calling themselves IoT penetration testing or vulnerability research is shown in Table 6. Two studies [2,24] are the closest to the philosophy of *PatrIoT*. The major shortcoming is their lack of a methodological approach, more precisely, how to perform an IoT pentest systematically from start to finish. Although the authors [18,19] introduce an *OWASP*-compatible approach, it only contains a handful of weaknesses. *PENTOS* [17] combines existing security tools, and the major focus is attacks, but not weaknesses. The approach used in studies that develop virtual simulations instead of testing real-world devices [25–27] is basically contrary to our approach, so we could not evaluate such studies. The *Common Criteria (CC)* [20] has a systematic method for IoT device evaluation. However, its perspective is on security requirements and protections rather than vulnerabilities and attack paths.

To date, some reports have been published with a perspective of the standardization of security measures in IoT devices. Those studies mostly appeal to manufacturers. The *Baseline Security Recommendations for IoT* document from *European Union Agency for Cybersecurity (ENISA)* reports the security requirements of IoT, mapping critical assets and relevant threats, assessing possible attacks, and identifying potential good practices and security measures to apply in order to protect IoT systems [28]. The *IoT Security Controls Framework* introduces the base-level security controls required to mitigate many of the risks associated with an IoT system [29]. The *Code of Practice for Consumer IoT Security* aims to support all parties involved in the development, manufacturing and retail of consumer IoT with a set of guidelines to ensure that products are secure by design and to make it easier for consumers to stay secure in a digital life [30]. The *National Institute of Standards and Technology Interagency or Internal Report (NISTIR 8200)* document evaluates the international cybersecurity standards development for IoT [31]. The *Cyber Security for Consumer IoT: Baseline Requirements (ETSI EN 303 645)* report aims to support parties involved in the development and manufacturing of consumer IoT with guidance on securing their products [32]. The *Common Criteria (CC)* provides IoT protection profile aiming to standardize the security requirements of an IoT secure element to be used in an IoT device [20].

[24] https://hackthebox.eu.

[25] https://vulnhub.com.

[26] https://offensive-security.com/pwk-oscp.

[27] https://giac.org/certification/penetration-tester-gpen.

[28] https://six2dez.gitbook.io/pentest-book.

Table 6 Comparison of similar studies

Study	PatrIoT	[2]	[17]	[18]	[19]	[20]
Information inquiry templates	✓	✓	✗	✗	✗	✓
Attack surface decomposition+	7	5	3	4	3	6
Compilation of top weaknesses	✓	✗	✗	✓	✓	✗
Compatibility*	✓	✗	✚	✚	✚	✗
Penetration testing guidelines	✓	✓	✗	✗	✓	✗
Risk scoring	✓	✗	✗	✗	✗	✓
Pentesting report template	✓	✗	✗	✗	✗	✗
Vulnerability disclosure report	✓	✗	✗	✗	✗	✗
Evaluation	✓	✓	✓	✓	✓	✓
Tools developed	✗	✓	✓	✗	✗	✗

*: ✚=OWASP ✓=OWASP + MITRE
+: Number of attack surfaces covered

7 Conclusion

The effects of insecurity in the IoT landscape have raised the need for IoT-specific vulnerability research capabilities. We identified significant shortcomings regarding the process dimension and challenges regarding the human resources dimension upon examining IoT penetration test reports.

The overall purpose of this study was to introduce an approach that allows faster adoption (efficiency) of new researchers to the research method and team and delivers higher quality (effectiveness) to IoT pentesting.

To achieve our overall objective, we developed a systematic and agile methodology, *PatrIoT*, for IoT vulnerability research that incorporates four key elements: logical attack surface decomposition, compilation of top 100 weaknesses, lightweight risk scoring, and step-by-step pentesting guidelines. We also explained how to systematically perform the IoT pentesting activities in four stages. While we rely on existing guidelines for web, network, cloud, and mobile attack surfaces, we developed guidelines for hardware, firmware, and radio attack surfaces. Additionally, we suggest identifying the relationships between vulnerabilities, reusing findings from one surface on another attack surface, finding ways to chain vulnerabilities, and accordingly developing attack paths.

In addition to this publication, we provide many external publicly available documents via GitHub [3]: a comprehensive spreadsheet with a compilation of the 100 common weaknesses, a sample document explaining how we decompose attack surfaces, step-by-step penetration testing guidelines, a standardized template for IoT information gathering, an IoT pentest report template, and a vulnerability disclosure report template.

PatrIoT has been evaluated with multiple IoT products in our laboratory. The empirical results show that it allows rapid advancement in vulnerability research activities and eliminates the risk of overlooking critical steps.

PatrIoT standardizes the way to work, making newcomers' orientation easy, which results in increased efficiency. It also defines a baseline that maintains the quality of the research output, which ensures effectiveness. Overall, it simplifies such a complex research process and enables the work to be completed in a standardized manner, quickly and accurately. Therefore, it has the potential to allow the maximum benefit to be obtained from the security studies.

Such a systematic solution is intended to help vulnerability researchers who periodically pentest multiple IoT devices by allowing them to discover new vulnerabilities, including zero-day, in time or ensure that the system is secure against certain attacks; it is also assumed to raise the bar for threat actors.

IoT penetration testing is perceived as unattainable to many novice researchers. In fact, the main reasons for this are the lack of widespread resources, as in IT pentesting domain. The methodology we have shared here is actually a summary of a wealth of information. It's easy to read as it's just a handful of pages, not the size of a book. We hope it will remove barriers and encourage people who may want to enter this research area.

The compilation of weaknesses will also help organizations that require IoT pentesting services. It can be used to define minimum requirements to be tested in terms of weaknesses within the scope of the IoT pentesting service. Likewise, it can be used to measure the quality of IoT penetration test reports.

Acknowledgements This project has received funding from the European Union's H2020 research and innovation programme under Grant Agreement No. 832907, Swedish Governmental Agency for Innovation Systems (Vinnova), and the Swedish Energy Agency.

Funding Open access funding provided by Royal Institute of Technology.

Declarations

Conflict of interest Authors Süren, Heiding, Olegård, and Lagerström declare that they have no conflict of interest.

Ethical approval This article does not contain any studies with human participants or animals performed by any of the authors.

Research data policy and data availability Data supporting the findings of this study are available on our public Github [3] repository.

Appendix A: Appendix I

See Tables 7, 8, 9, 10, 11, 12, and 13.

Table 7 Weaknesses of IoT physical interfaces

ID	Hardware weakness
T001	Sensitive data exposure—Device ID/serial no
T002	Firmware/storage extraction—Insecure external media interfaces
T003	Firmware/storage extraction—Download from the Web
T004	Firmware/storage extraction—Insecure SPI interface
T005	Firmware/storage extraction—Insecure I2C interface
T006	Firmware/storage extraction—Insecure UART interface
T007	Firmware/storage extraction—Insecure JTAG interface
T008	Firmware/storage extraction—Insecure SWD interface
T009	Firmware/storage extraction—Insecure SoC
T010	Firmware/storage extraction—Insecure eMMC chip
T011	Backdoor firmware—Insecure UART interface
T012	Backdoor firmware—Insecure JTAG interface
T013	Backdoor firmware—Insecure SWD interface
T014	Grant shell access—Insecure UART interface
T015	Grant shell access—Insecure SPI interface
T016	Change code execution flow—Insecure JTAG/SWD interface
T017	Reset to insecure state

Table 8 Weaknesses of IoT firmware

ID	Firmware weakness
T018	Sensitive data exposure—Hardcoded credentials
T019	Sensitive data exposure—Backdoor accounts
T020	Sensitive data exposure—Encryption keys and algorithms
T021	Sensitive data exposure—Other sensitive information
T022	Sensitive data exposure—Static and same encryption keys
T023	Configuration—Lack of data integrity checks
T024	Configuration—Lack of wiping device
T025	Configuration—Insecure customization of OS platforms
T026	Configuration—Lack of security configurability
T027	Configuration—Insecure filesystem permissions
T028	Authentication bypass—Device to device
T029	Authentication bypass—Device to mobile application
T030	Authentication bypass—Device to cloud
T031	Update mechanism—Missing update mechanism
T032	Update mechanism—Lack of manual update
T033	Update mechanism—Lack of transport encryption
T034	Update mechanism—Lack of signature on update file
T035	Update mechanism—Lack of update verification
T036	Update mechanism—Lack of update authentication
T037	Update mechanism—Intercepting OTA update
T038	Update mechanism—Backdoor firmware
T039	Update mechanism—World writable update location
T040	Update mechanism—Lack of anti-rollback mechanism

Table 9 Weaknesses of IoT network services

ID	Network weakness
T041	Sensitive data exposure
T042	Lack of transport encryption
T043	Insecure SSL/TLS issues
T044	Authentication—Username enumeration
T045	Authentication—Weak credentials
T046	Authentication—Improper account lockout
T047	Authentication—Weak password recovery
T048	Privilege escalation
T049	Authentication bypass
T050	Denial of Service (DoS)
T051	Buffer overflow

Table 10 Weaknesses of IoT web application

ID	Web weakness
T052	Sensitive data exposure
T053	Lack of transport encryption
T054	Insecure SSL/TLS issues
T055	Authentication—Username enumeration
T056	Authentication—Weak credentials
T057	Authentication—Improper account lockout
T058	Authentication—Weak password recovery
T059	Authentication—Lack of two-factor authentication
T060	Authentication bypass—Web application to cloud
T061	Command injection
T062	Insecure direct object references (IDOR)
T063	Business and logic flaws

Table 11 Weaknesses of IoT cloud services

ID	Cloud weakness
T064	Lack of transport encryption
T065	Insecure SSL/TLS issues
T066	Authentication—Username enumeration
T067	Authentication—Weak credentials
T068	Authentication—Improper account lockout
T069	Authentication—Weak password recovery
T070	Authentication—Lack of two-factor authentication
T071	Vendor APIs—Inherent trust of cloud or mobile application
T072	Vendor APIs—Authentication bypass
T073	Vendor APIs—Authorization bypass
T074	Vendor APIs—Undocumented backdoor API calls
T075	Vendor APIs—User data disclosure
T076	Vendor APIs—Device information leakage

Table 12 Weaknesses of IoT mobile application

ID	Mobile weakness
T077	Sensitive data exposure—Hardcoded credentials
T078	Sensitive data exposure—Encryption keys and algorithms
T079	Sensitive data exposure—Other sensitive information
T080	Authentication—Username enumeration
T081	Authentication—Weak credentials
T082	Authentication—Improper account lockout
T083	Authentication—Weak password recovery
T084	Authentication—Lack of two-factor authentication
T085	Authentication—Mobile application to cloud system
T086	Insecure authorization
T087	Implicitly trusted by device or cloud
T088	Lack of transport encryption
T089	Insecure SSL/TLS issues
T090	Insecure data storage
T091	Outdated 3rd party libraries and SDKs
T092	Business and logic flaws
T093	Lack of health checks

Table 13 Weaknesses of IoT radio communication

ID	Radio weakness
T094	Lack of transport encryption
T095	Insecure SSL/TLS issues
T096	Lack of message integrity check
T097	Lack of signal replaying checks
T098	Lack of signal jamming checks
T099	Lack of signals spoofing checks
T100	Lack of Denial-of-service (DoS) checks

References

1. Antonakakis, M., April, T., Bailey, M., Bernhard, M., Bursztein, E., Cochran, J., Durumeric, Z., Halderman, J. A., Invernizzi, L., Kallitsis, M., et al.: Understanding the Mirai botnet. In: 26th USENIX security symposium USENIX Security 17), pp. 1093–1110 (2017)
2. Gupta, A.P: The IoT Hacker's Handbook: A Practical Guide to Hacking the Internet of Things. Apress (2019)
3. PatrIoT artifacts. https://github.com/beyefendi/penbook/. Accessed: 06 Dec 2021 (2021)
4. Rathore, B., Brunner, M., Dilaj, M., Herrera, O., Brunati, P., Subramaniam, R., Raman, S., Chavan, U.: Information systems security assessment framework (ISSAF), Draft 0.2 B 1 2006 (2006)
5. Herzog, P.: Osstmm 3 the open source security testing methodology manual. Contemporary security testing and analysis, ISECOM-Institute for Security and Open Methodologies
6. Kth pentest reports. https://bit.ly/38FoGbL. Accessed 06 Dec 2021 (2020)
7. Peffers, K., Tuunanen, T., Rothenberger, M.A., Chatterjee, S.: A design science research methodology for information systems research. J. Manag. Inf. Syst. **24**(3), 45–77 (2007)
8. Cvss - common vulnerability scoring system. https://nvd.nist.gov/vuln-metrics/cvss/v3-calculator. Accessed 06 Dec 2021 (2021)

9. OWASP risk rating methodology, https://owasp.org/www-community/OWASP_Risk_Rating_Methodology. Accessed 06 Dec 2021 (2021)
10. Johnson, P., Lagerström, R., Ekstedt, M., Franke, U.: Can the common vulnerability scoring system be trusted? A Bayesian analysis. IEEE Trans. Dependable Secure Comput. **15**(6), 1002–1015 (2016)
11. Felderer, M., Büchler, M., Johns, M., Brucker, A. D., Breu, R., Pretschner, A.: Security testing: a survey. In: Advances in Computers, Vol. 101, pp. 1–51. Elsevier (2016)
12. Xiong, W., Lagerström, R.: Threat modeling—a systematic literature review. Comput. Secur. **84**, 53–69 (2019)
13. Xiong, W., Legrand, E., Åberg, O., Lagerström, R.: Cyber security threat modeling based on the MITRE enterprise ATT&CK matrix. In: Software and Systems Modeling, pp. 1–21 (2021)
14. Ruohonen, J., Allodi, L.: A bug bounty perspective on the disclosure of web vulnerabilities. arXiv:1805.09850 (2018)
15. Nmap—full disclosure. https://nmap.org/mailman/listinfo/fulldisclosure. Accessed 06 Dec 2021 (2021)
16. Bilge, L., Dumitraş, T.: Before we knew it: an empirical study of zero-day attacks in the real world. In: Proceedings of the 2012 ACM Conference on Computer and Communications Security, CCS '12, pp. 833–844 (2012). https://doi.org/10.1145/2382196.2382284
17. Visoottiviseth, V., Akarasiriwong, P., Chaiyasart, S., Chotivatunyu, S.: Pentos: Penetration testing tool for internet of thing devices. In: TENCON 2017-2017 IEEE Region 10 Conference, pp. 2279–2284 (2017)
18. Lally, G., Sgandurra, D.: Towards a framework for testing the security of IoT devices consistently. In: International workshop on emerging technologies for authorization and authentication, pp. 88–102. Springer (2018)
19. Nadir, I., Ahmad, Z., Mahmood, H., Shah, G.A., Shahzad, F., Umair, M., Khan, H., Gulzar, U.: An auditing framework for vulnerability analysis of IoT system. In: IEEE European Symposium on Security and Privacy Workshops (EuroS&PW), 2019, pp. 39–47 (2019)
20. Common Criteria: IoT Secure Element Protection Profile. https://www.commoncriteriaportal.org/files/ppfiles/pp0109b_pdf.pdf. Accessed 06 Dec 2021 (2019)
21. Engebretson, P.: The Basics of Hacking and Penetration Testing: Ethical Hacking and Penetration Testing Made Easy. Elsevier (2013)
22. Weidman, G.: Penetration Testing: A Hands-On Introduction to Hacking. No Starch Press, San Francisco, California, US (2014). https://nostarch.com
23. Kim, P.: The Hacker Playbook 3: Practical Guide to Penetration Testing. Independently Published, The Hacker Playbook Series (2018)
24. Guzman, A., Gupta, A.: IoT Penetration Testing Cookbook: Identify Vulnerabilities and Secure Your Smart Devices. Packt Publishing Ltd, San Francisco, California, US (2017). https://nostarch.com
25. Chu, G., Lisitsa, A.: Penetration testing for internet of things and its automation, in: 2018 IEEE 20th International Conference on High Performance Computing and Communications; IEEE 16th International Conference on Smart City; IEEE 4th International Conference on Data Science and Systems (HPCC/SmartCity/DSS), pp. 1479–1484 (2018)
26. Mahmoodi, Y., Reiter, S., Viehl, A., Bringmann, O., Rosenstiel, W.: Attack surface modeling and assessment for penetration testing of IoT system designs. In: 2018 21st Euromicro Conference on Digital System Design (DSD), pp. 177–181 (2018)
27. Yadav, G., Paul, K., Allakany, A., Okamura, K.: Iot-pen: A penetration testing framework for IoT. In: International Conference on Information Networking (ICOIN), 2020, pp. 196–201 (2020)
28. ENISA Baseline Security Recommendations for IoT. https://www.enisa.europa.eu/publications/baseline-security-recommendations-for-iot. Accessed 06 Dec 2021 (2017)
29. CSA IoT Security Controls Framework. https://cloudsecurityalliance.org/artifacts/iot-security-controls-framework/. Accessed 06 Dec 2021 (2019)
30. Code of Practice for Consumer IoT Security. https://www.gov.uk/government/publications/code-of-practice-for-consumer-iot-security. Accessed 06 Dec 2021 (2018)
31. NISTIR 8200: International Cybersecurity Standardization for the IoT. https://csrc.nist.gov/publications/detail/nistir/8200/final. Accessed 06 Dec 2021 (2018)
32. ETSI EN 303 645: Cyber Security for Consumer IoT: Baseline Requirements. https://www.etsi.org/technologies/consumer-iot-security. Accessed 06 Dec 2021 (2020)