Graphical Abstract

PatrIoT: Practical and agile threat research for IoT

Emre Süren, Fredrik Heiding, Johannes Olegård, Robert Lagerström

Four-stage IoT vulnerability research methodology

Planning	**Threat modeling**	**Exploitation**	**Reporting**
Scoping Information gathering **[1]** Enumeration	Decomposition **[2]** Vulnerability analysis **[3]** Risk scoring **[4]**	Known exploits **[5]** Exploit development Post exploitation	Pentest report **[6]** Disclosure CVE

Key contributions

[1] Standardized inquiry template
[2] Attack surface decomposition
[3] Compilation of top 100 weaknesses
[4] Lightweight risk scoring
[5] Step-by-step guidelines
[6] Reporting template

Highlights

PatrIoT: Practical and agile threat research for IoT

Emre Süren, Fredrik Heiding, Johannes Olegård, Robert Lagerström

- This study resulted in the development of an IoT penetration testing methodology containing four key elements:

  Compilation of top 100 IoT weaknesses

  Logical attack surface decomposition

  A lightweight risk scoring approach

  Step-by-step IoT penetration testing guidelines

PatrIoT: Practical and agile threat research for IoT

Emre Süren[a], Fredrik Heiding[a], Johannes Olegård[a], Robert Lagerström[a]

[a]*School of Electrical Engineering and Computer Science, KTH Royal Institute of Technology, Sweden*

Abstract

The prevalence of IoT solutions and their constantly evolving technical complexity is one of the greatest challenges in the modern cyber security landscape. Most IoT devices are resource-constrained and were not built with security in mind; thus, they are often more hacker-friendly than regular computers. Because they are often almost constantly online, which further enhances their attack surfaces, a large number of critical vulnerabilities have been found in the IoT ecosystem until now. Furthermore, the security features of these devices are less often of concern, and fewer methods, standards, and guidelines are available with which to test them. Although a few approaches are available to assess the security posture of IoT products, these approaches 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: compilation of top 100 IoT weaknesses, logical attack surface decomposition, lightweight risk scoring, and step-by-step penetration testing guidelines. Our proposed methodology is evaluated with multiple IoT products. The results indicate that the methodology 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 weaknesses, attack surface, risk scoring, penetration testing guidelines, Mitre CWE, OWASP IoT Top 10
PACS: 0000, 1111
2000 MSC: 0000, 1111

1. Introduction

The IoT ecosystem provides a platform for the connection of everyday objects over the 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, smart cities, healthcare monitoring, energy metering, agriculture, and even military equipment.

The IoT environment includes hardware manufacturing, software development (firmware, web, mobile, communication APIs), radio connectivity, Internet service providers, and IoT platform integrators. Clearly, all these components need to be secured. Over time, the IoT phenomenon has created a completely new and complex set of problems for the security community. The following eight challenges are worth highlighting:

- As they are almost always connected to the Internet, they provide attackers with an unlimited time frame within which to compromise them.
- They have constrained resources (power, processing, and memory) [1], which make it impossible to take advantage of modern defense techniques, and which provide intruders with a wide-open attack surface.
- The lack of security standards (built-in security in IoT protocols) [2] makes it attractive for hackers to exploit vulnerabilities.
- Although connected devices are intended for long-term use, most of these devices eventually stop receiving security updates [3].
- Identification and recovery of compromised physical devices are not as easy as for a standard computer [4], thus they remain infected in the long run.
- Traditional computers are the primary focus in the security industry; in comparison, the security of many IoT devices is neglected [5].
- From the perspective of enterprises, updating and securing large-scale and resource constraint IoT products is a troublesome task.

- From the perspective of individuals, devices are often configured by people with limited experience, who manage a large amount of personal data and have direct control over everyday objects.

If these challenges are not met satisfactorily, exploitation of the vulnerabilities can severely impact the security of systems and consumers' privacy, mostly in terms of surveillance concerns, as these IoT devices transmit various types of sensitive data. Furthermore, it is not surprising to deduce that smart devices are inevitable candidates for compromise, especially for botnet operators. The legendary example is the Mirai botnet [6], 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.

The devastating effects of IoT insecurity 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 products planned 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 should rest on three pillars: people, processes, and technology. In the first step, we decided to use existing tools for pentesting and focused on addressing the problem from the perspective of the process and the people. Our decision was based on the fact that the available tools are sufficient for starting, but that the process and people's perspectives were lacking. **Process.** Upon examining many IoT pentest reports, we identified four significant shortcomings in IoT vulnerability research. 1) The lack of standardization in security tests, even for very similar IoT devices. 2) The vulnerability research is usually based only on web and network pentesting, which offers limited scope for today's IoT. 3) Threat modeling is either too complex or is omitted entirely. 4) Existing IoT pentesting guidelines are very limited or difficult to follow. **People.** Security research labs 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 labs usually have a high personnel turnover, meaning researchers are constantly arriving and leaving for various reasons (e.g., graduation or better offers). In addition, the shortage of IoT security expertise fans the flames [1]. That is why it makes sense to take some precautions to reveal the maximum benefit to be obtained 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 four shortcomings regarding the process perspective of IoT vulnerability research by introducing the following four key elements: The 100 currently prevalent weaknesses found in IoT environments, logical decomposition of attack surfaces, a lightweight risk scoring approach, and step-by-step 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. Our proposed methodology is evaluated with seven real-world IoT products, where each represents a special IoT product category. The empirical results show that this methodology allows researchers to quickly advance vulnerability research activities and it decreases the risk of overlooking critical steps.

The contribution of the research is that: 1) to the best of our knowledge, a comprehensive compilation of weaknesses has not been made public before, 2) compilation of weaknesses is compatible with *"OWASP IoT Top 10"* and *"Mitre CWE"*, which are the state-of-the-art approach, 3) decomposing attack surfaces into seven components is unique to this study and is highly efficient in discovering IoT-specific vulnerabilities, 4) our lightweight risk scoring approach effectively reduces the overhead of threat modeling, 5) our own step-by-step pentesting guidelines fortify the discovery of critical IoT vulnerabilities, 6) our highly detailed IoT pentest report template facilitates the development of high-quality reports. 7) our information inquiry templates makes easy the information gathering phase of Planning stage.

The remainder of this paper is organized as follows. Section 2 discusses the literature and challenges. Section 3 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 our methodology, is provided in Section 4. In Section 5, the methodology is evaluated, and the results are presented. Section 6 discusses the trade-offs and chal-

[1]https://bit.ly/3CSwaVM

lenges of using the proposed methodology, along with opportunities for future work. Finally, the paper is concluded in Section 7 with a summary.

2. Related work

Traditional (IT) pentesting is a well-studied topic. Generally, researchers agree on a 5-step approach [7, 8, 9], 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 particular, post-exploitation has several objectives: persistence, privilege escalation, lateral movement, pivoting, evasion, covering tracks, and password cracking. In addition, the information system security assessment framework (ISSAF) and the open source security testing methodology manual (OSSTMM) are often referenced and used as penetration testing guidelines. Today, experiments with pentesting techniques can be conducted on various cloud-based pentest practice platforms [2],[3], and there are several certification options for demonstrating pentesting skills [4],[5]. Finally, several mature techniques and tools exist for conducting an IT pentest [6].

The traditional pentest has been extensively detailed over several years, yet the situation is not the same for the IoT ecosystem. Although a few important studies have been reported [10, 11], many aspects thereof remain to be investigated. Unlike IT pentesting, IoT includes other types of attack surfaces that need to be explored. An IoT pentest involves the security assessment and exploitation of all components of an IoT product. The next section describes the different components of IoT solutions, various ways of identifying and exploiting vulnerabilities, tools to be used, and execution of the overall pentest.

We have been conducting pentests for IoT devices, including home security sensors, smart home appliances, smart transportation equipment, payment systems, and drones, and our research lab has published several pentest reports [12] in recent years. Over time, we have refined our approach and present our streamlined IoT vulnerability research methodology in this paper.

[2]hackthebox.eu

[3]vulnhub.com

[4]offensive-security.com/pwk-oscp

[5]giac.org/certification/penetration-tester-gpen

[6]six2dez.gitbook.io/pentest-book

3. Design of methodology

This section presents the design of our proposed IoT vulnerability research methodology following the design science research process [13].

3.1. Problem

The problem we faced was to determine the way in which to conduct IoT vulnerability research such that the benefit would be maximized. During the preliminary analysis, we encountered the four shortcomings described in the Introduction section. Briefly, they include the lack of standards in relation to the IoT weaknesses that need to be tested, limited scope in terms of the IoT attack surfaces, complex or no threat modeling, and insufficient technical documentation on carrying out attacks for IoT vulnerability discovery. We also need to emphasize two challenges related to the people's perspective: high personnel turnover and a shortage of security expertise.

3.2. Objective

Systematic. In a lab environment where vulnerability research is performed regularly, it is very important to perform the activities systematically and according to a basic standard.
Faster adoption (efficiency). Working systematically accelerates the adaptation of new researchers.
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 found by introducing common security tests, targeting broader attack surfaces, favoring lightweight risk scoring, and providing step-by-step testing guidelines.

3.3. Design & 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.

3.3.1. Element 1: Compilation of top weaknesses

Because the *"OWASP IoT Top 10"* [7] project is considered to be the most comprehensive and well-known study, we chose it as our primary source,

[7]owasp.org/www-project-internet-of-things

which is a type of guideline containing ten security risks. Although this checklist could assist with IoT vulnerability research, what is actually required is the weaknesses and attacks associated with them. Because the security risks cannot be used alone in pentesting, we first aimed to fortify the process by compiling a list of common weaknesses. We accomplished this by leveraging the *MITRE CWE* [8] project to map each security risk to its corresponding weaknesses. The key point here is that *OWASP IoT* and *MITRE CWE* are not compatible. This is because a specific security threat can arise as a result of various weaknesses. For example, the *OWASP IoT* checklist contains ten items, whereas the *CWE* database contains close to a hundred records for the hardware attack surface alone [9]. This indicates that a risk on the *OWASP IoT* checklist corresponds to multiple weaknesses in the *CWE*. In addition, there could be weaknesses in this database that do not resort under any of the items 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, this weakness compilation is enriched with clear descriptions, risk impact factors, default severity values, and more [14]. The objective here is to determine the minimum but major security tests that need to be performed first to enable us to progress quickly without missing the key parts.

3.3.2. Element 2: Attack surface decomposition

Upon examination of the IoT pentest reports, we discovered that IoT systems are still being tested using traditional techniques (web application or network pentesting). The main reason for this is that more resources are available on these topics, and researchers are experienced in them. However, research in different domains requires different techniques, which also demands a more specialized skill set. More precisely, the IoT infrastructure is built on top of various components; thus, we logically decompose the IoT architecture into seven attack surfaces: *hardware, firmware, radio protocol, network service, web application, cloud service, and mobile app*, as in Table 1. We also categorize the generated weaknesses into attack surfaces to

[8] cwe.mitre.org

[9] cwe.mitre.org/data/definitions/1194.html

link them to each other. The objective here is to divide the tasks to enable researchers who are more specialized in a certain domain to work and thus produce higher quality output.

3.3.3. Element 3: Lightweight risk scoring

We utilize *STRIDE* to identify and categorize threats and leverage *DREAD* to identify the potential impacts of threats by calculating risk scores. Although *CVSS* is becoming the de facto risk scoring method, the challenge is that it is not easy to use for calculations, which is why we prefer a simpler customized routine. The objective here is to simplify the threat modeling and use it to guide the pentests.

3.3.4. 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 begin.

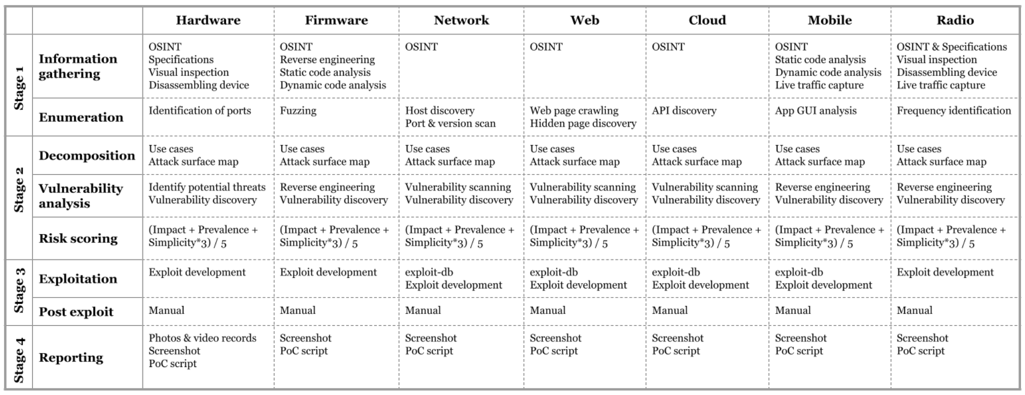
4. Application of the methodology

This section explains the use of our four-stage IoT vulnerability research methodology, as shown in Figure 1, which is built on top of the four key elements.

Table 1: 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

Figure 1: IoT pentest process



		Hardware	Firmware	Network	Web	Cloud	Mobile	Radio
Stage 1	**Information gathering**	OSINT Specifications Visual inspection Disassembling device	OSINT Reverse engineering Static code analysis Dynamic code analysis	OSINT	OSINT	OSINT	OSINT Static code analysis Dynamic code analysis Live traffic capture	OSINT & Specifications Visual inspection Disassembling device Live traffic capture
	Enumeration	Identification of ports	Fuzzing	Host discovery Port & version scan	Web page crawling Hidden page discovery	API discovery	App GUI analysis	Frequency identification
Stage 2	**Decomposition**	Use cases Attack surface map	Use cases Attack surface map	Use cases Attack surface map	Use cases Attack surface map	Use cases Attack surface map	Use cases Attack surface map	Use cases Attack surface map
	Vulnerability analysis	Identify potential threats Vulnerability discovery	Reverse engineering Vulnerability discovery	Vulnerability scanning Vulnerability discovery	Vulnerability scanning Vulnerability discovery	Vulnerability scanning Vulnerability discovery	Reverse engineering Vulnerability discovery	Reverse engineering Vulnerability discovery
	Risk scoring	(Impact + Prevalence + Simplicity*3) / 5	(Impact + Prevalence + Simplicity*3) / 5	(Impact + Prevalence + Simplicity*3) / 5	(Impact + Prevalence + Simplicity*3) / 5	(Impact + Prevalence + Simplicity*3) / 5	(Impact + Prevalence + Simplicity*3) / 5	(Impact + Prevalence + Simplicity*3) / 5
Stage 3	**Exploitation**	Exploit development	Exploit development	exploit-db Exploit development	exploit-db Exploit development	exploit-db Exploit development	exploit-db Exploit development	Exploit development
	Post exploit	Manual	Manual	Manual	Manual	Manual	Manual	Manual
Stage 4	**Reporting**	Photos & video records Screenshot PoC script	Screenshot PoC script	Screenshot PoC script	Screenshot PoC script	Screenshot PoC script	Screenshot PoC script	Screenshot PoC script

4.1. Stage 1 - Planning

The goal in the first stage is to assess the entire infrastructure to obtain an overall idea of the functionality of the product. The word *"product"* refers to the entire IoT infrastructure (*e.g., Smart Home IoT product*), whereas the term *"device"* refers to the individual smart devices that together form the "product" product. Later on, this stage will allow us to estimate the types of potential vulnerabilities that could be present in the attack surface.

4.1.1. Scoping

In fact, 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 are specified, namely *hardware, firmware, radio, etc.* The type of project is defined as a black box, white box, or gray-box test. The test type directly affects the type of vulnerabilities that can be discovered in the research, 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 has limitations due to its nature [15]. 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, usually 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

Figure 2: Architecture map

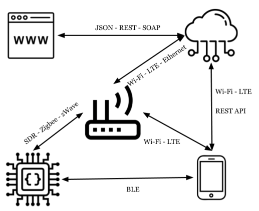
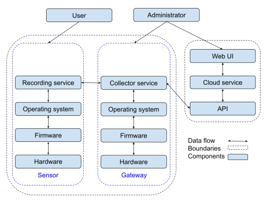


Figure 3: Potential attack surface



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 Section . The physical location of the pentest is also determined in this phase, that is, whether the test will be performed in a customer environment or a controlled research lab.

4.1.2. Information gathering & Enumeration

The next step is reconnaissance in which all assets are identified with relevant information. Initially, every device belonging to the product is physically examined. Then, a comprehensive review of the documentation of the devices is performed. Finally, certain active and passive scanning and enumeration techniques were applied to each attack surface. In light of these three inquiries, all hardware and software components and communication protocols are identified, and their functionality is documented. We share the details of information gathering phase of each attack surface with a separate document [14].

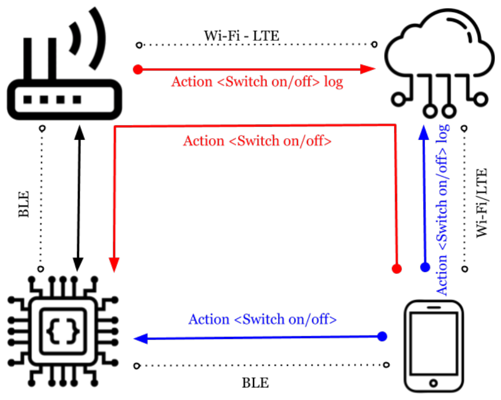
4.2. Stage 2 - Threat modeling

The second stage of the IoT vulnerability research process is *"threat modeling"* [16, 17]. The objective of this stage is to estimate the relevant potential threats for each attack surface, predict the possible impacts of successful exploitation, and prioritize them by calculating risk scores.

4.2.1. Attack surface decomposition

We start modeling by identifying the architecture first, then breaking it down into logical components and attributing the functions of those identified in the information-gathering stage. A general architecture mapping provides a simple overview of the system in its entirety, as shown in Figure 2, whereas

Figure 4: Use case



component decomposition provides a clear view of the potential attack surfaces. Decomposition involves modeling the data flow (communication protocols) between the components, as shown in Figure 3. 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. Recall that, in the previous stage (information gathering) we listed the functions of each component item by item, how the functionalities were triggered and ended, and the effects of each action, because this also makes it easy to identify threats. For repeatable testing, it is essential to detail the steps required up to the start of the test. The number of use case descriptions changes based on three factors: the penetration test scope, the number of different devices, and the user profiles. For complex scenarios, we also developed use-case diagrams, as illustrated in Figure 4. The following is an example of use case description steps.

- Place a device <X> and power it up
- Place a hub for device <X> and power it up
- Register an account in the (Android/iOS/etc.) 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

Table 2: Risk assessment

Rating	High (3)	Medium (2)	Low (1)
Damage	Read/write any content, privilege escalation or granting full access	Sensitive data leakage	Little to no sensitive data can be leaked or lost
Reproducibility	Bypass prevention	Bypass prevention only when certain conditions exist	Difficult to bypass prevention
Exploitability	Little skill	Moderate skill	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	No detection	Insufficient detection	Requires significant manual review

- 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

The task of pentesting IoT devices is complex because these systems have multiple attack surfaces. Thus, information about the examined system usually emerges over time. Therefore, we first create a threat model based on the information we have and start the vulnerability analysis. Then, we update the threat model based on information discovered in the exploitation stage. Briefly, we use an iterative threat modeling approach.

4.2.2. Vulnerability analysis

Our threat modeling approach is completely relies on entry points and vulnerability data, and using this information we develop attack paths. All the information gathered from the previous stage and attack surface mapping is interpreted in terms of potential threats. First, we conduct an online search for previously published vulnerabilities that are specific to individual attack surfaces [10]. Although the main objective of this study is to explore unknown

[10]cve.mitre.org/cve

vulnerabilities, we also check for existing vulnerabilities, as it inspires, and these can potentially be used to find new vulnerabilities as well.

Second, to use resources effectively, we must be able to estimate the weaknesses with high potential for an attack surface. This procedure is relatively difficult to perform and requires prior knowledge, but a part of the presented methodology provides a compilation of these most common weaknesses. We provide 100 common weaknesses by *CWE-IDs*, thus matching *CAPEC-IDs* [11] can be used to lookup associated attack vectors. In addition to the compilation that we introduce, we have developed IoT-specific pentesting guidelines. 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 [12], which contain pre-configured tools, especially for hardware and radio surfaces, are commonly used. We share the details of vulnerability discovery phase of each attack surface with a separate document [14].

4.2.3. Risk scoring

Despite having shortlisted weaknesses in the previous step, we still prioritize weaknesses according to the highest impact and highest probability of success. We customize a combination of known mechanisms for compatibility, but we favor on simplicity in threat modeling. We categorize threats similarly to STRIDE, but in our contemporary terminology, our sole aim is to assess the impact of threats. According to our evaluation; the potential impact of (s)poofing category is authentication bypass, for (t)ampering; it is integrity violation; for (r)epudiation, it is weak authentication/authorization; for (i)nformation disclosure, it is confidentiality violation; for (d)enial of service, it is availability violation, and for the (e)levation of a privilege (privilege escalation) it is unauthorized access.

By default, we assign high severity for remote code execution, middle for authentication bypass/modification, and low for the rest. However, we use the following routine to find the actual risk score. Although *CVSS* is becoming the de facto severity scoring standard, this value is often challenging to calculate [18]. Moreover, devoting too much effort to these calculations may not provide much information, which is why we prefer a simpler and

[11]capec.mitre.org

[12]attify.com/attifyos

lightweight routine. We leverage an approach similarly to *DREAD* to calculate the risk scores, but simpler and clearer, as in Table 2. We perceive the (d)amage as impact, (a)ffected users as prevalence, and the remaining three, (r)eproducibility, (e)xploitability, and (d)iscoverability, as simplicity (probability of success) metrics. Therefore, we assign a value between 1 and 3 to these three metrics, and our risk rating equation becomes *(impact + prevalence + simplicity*3) / 5*. If the average score is greater than 2, the overall severity is high; if the score equals 1, the overall severity is low; otherwise, it is medium. Although a default value is assigned to the severity of each weakness, the value should be changed according to the result of exploitation.

4.3. 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 usually uphold the assumptions. Additionally, it is commonly known that chaining a few low-level vulnerabilities could result in a critical vulnerability. However, this cannot be determined with sole vulnerability analysis. Therefore, realizing exploitation is a valuable exercise.

4.3.1. Known vulnerabilities

After conducting threat modeling, exploits are matched with known vulnerabilities found in public databases [13], [14]. An exploit is usually a snippet of code developed with scripting language and is run manually from the command line. Sometimes, semi-automated tools [15] can be utilized for easy and stable exploitation. In the case of known vulnerabilities that have no public exploits, we develop our own [16].

4.3.2. Exploit development

The goal of hardware hacking is to gain access to the firmware image file. Firmware analysis is conducted to collect helpful information for other attack

[13]exploit-db.com
[14]packetstormsecurity.com/files/tags/exploit
[15]github.com/rapid7/metasploit-framework
[16]github.com/beyefendi/exploit

surfaces, for example, encryption keys (SSH private key), a hash or password for authentication, libraries used by network services, and compiled or source code of applications. 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 or cloud services, the aim is to execute a command on the device (remote code execution) as well as bypass authentication. As vulnerabilities are found with the help of web application scanners, they also provide primitive payloads for proof of concept. These hints are then used to design the real working PoC exploit codes for the purpose. In addition, some types of vulnerabilities can be exploited by using semi-automated tools [17]. In the last case, brute force attacks are performed against login pages.

One of the most challenging problems to solve during pentests is to determine whether the vulnerability is caused by the product/protocol itself or its setup. If it is due to configuration, this is likely not a vulnerability that can be reported to the manufacturer, followed by a request for a CVE. Therefore, we should indicate the cause of the vulnerability in our reports. We observe that security misconfiguration is one of the common weaknesses we encounter. Notifying the manufacturer also serves as a confirmation of the vulnerability, hence this notification is highly important.

4.3.3. Post exploitation

We also perform post-exploitation activities right after compromising smart devices. These usually comprise elevating privileges after gaining user-level access to the operating system.

4.4. Stage 4 - Reporting

Vulnerability research is concluded with a document that is the fourth and final stage of the activities. Using the same template for each pentest makes it easy to generate statistics and conduct searches in previous reports. Particularly, the discovery of a previously unknown vulnerability would require additional steps to be taken.

4.4.1. Template

The IoT pentest report contains eight sections [14]. The first section is "General information," which includes the service recipient, the time period,

[17] github.com/sqlmapproject/sqlmap

the systems under test, the target, 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 which systems were tested, what types of tests failed, what types of tests were successful, 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 systems with their severity. The second is the "known vulnerabilities table," which lists previously known vulnerabilities (e.g., vulnerabilities in the CVE 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 systems. The 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 a traditional penetration test, an IoT pentest report also has its own characteristics; mainly, sections dedicated to hardware and radio components are extremely detailed and include high-quality photographic images and video demonstrations.

4.4.2. Disclosure

Public disclosure is the practice of announcing security vulnerabilities discovered in software or hardware products. As usual, security researchers, users, and product owners have different priorities. Researchers prefer to publish details of vulnerabilities as soon as possible. Users of vulnerable products or services also ask that the systems be patched as quickly as possible. On the other hand, vendors favor the disclosure of vulnerabilities only to them to avoid affecting user loyalty. These conflicts have been resolved to a certain extent by developing bug bounty programs. 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*) [19].

Security researchers have a public responsibility for disclosing vulnerabilities, whereas product owners have the same level of responsibility in terms of fixing flaws. Industry vendors [18] generally agree that a maximum deadline of 90 days is acceptable. However, once the vulnerabilities are actively exploited in the wild, the vulnerability can be publicly released without waiting for the deadline.

A vulnerability disclosure process includes certain typical steps: A researcher discovers the security vulnerability and then documents the findings, including supporting evidence (a PoC script and screenshots) and a full disclosure timeline. The researcher then notifies the vendor directly or via a bug bounty program by submitting the report via secure channels. Once the system providers accept the finding, they have a reasonable time frame within which to patch the vulnerability. Once a patch is available or the timeline has elapsed, product owners inform the public by issuing a security advisory report. Then, the researcher can publish a full disclosure analysis on certain platforms [19],[20],[21]. The disclosure usually contains details of the vulnerability, including a working exploit script. Companies that receive reports of vulnerabilities through bug bounty programs commonly avoid any disclosure.

4.4.3. CVE

Common Vulnerabilities and Exposures is a limited database of known security vulnerabilities. Once a new vulnerability is discovered, a security researcher may want to have a CVE-ID assigned to his brand new finding [22] 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 [20]. For example, although an unauthenticated remote code execution vulnerability is usually described as zero-day, a simple cross-site scripting flaw is not considered to belong to

[18]googleprojectzero.blogspot.com
[19]exploit-db.com
[20]packetstormsecurity.com/files/tags/exploit
[21]github.com
[22]`cve.mitre.org/cve/request_id.html`

this category.

5. 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 [12]. 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 3: Tested devices

Work	**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

Recall that we address the IoT vulnerability research in four stages: 1) planning (Section 4.1), 2) threat modeling (Section 4.2), 3) exploitation (Section 4.3), and 4) reporting (Section 4.4). In short, we start by collecting information related to the device; With this information, we decompose the attack surfaces, identify known weaknesses and vulnerabilities, assess risks, and create a threat model. In the exploitation stage, we check the items in the compilation of weaknesses starting with the highest risk score and following the step-by-step guidelines. Finally, based on the findings, we generate our penetration test report using the given standard template.

The threat models created for the seven chosen devices are shown in Figure 5. Our approach requires no specific tools, and the researchers on each project choose the ones they find the most appropriate [23],[24],[25], providing

[23]https://www.microsoft.com/en-us/securityengineering/sdl/threatmodeling

[24]https://owasp.org/www-project-threat-dragon/

[25]https://foreseeti.com/securicad-professional/

flexibility. Figure 6 shows the attack surfaces tested in each study, and the vulnerabilities discovered by using our compilation of weaknesses.

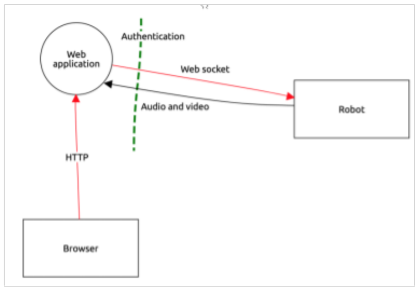
5.1. Robot

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

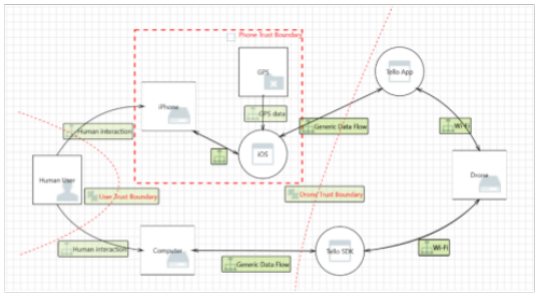
5.2. Drone

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. **Threat model.** 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 succeeded in intercepting the traffic between the drone and controller with *Wireshark.* This was recorded as MitM

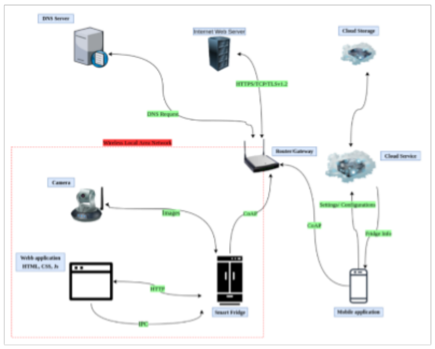
Figure 5: Threat models



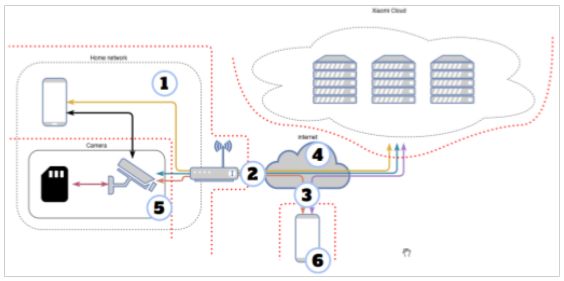
Study 1 - Robot



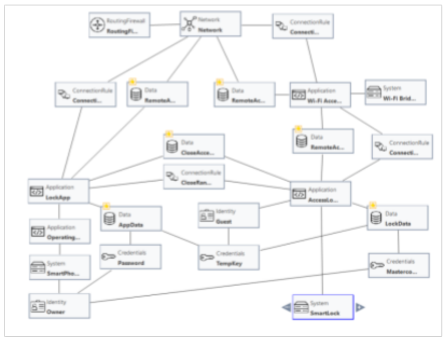
Study 2 - Drone



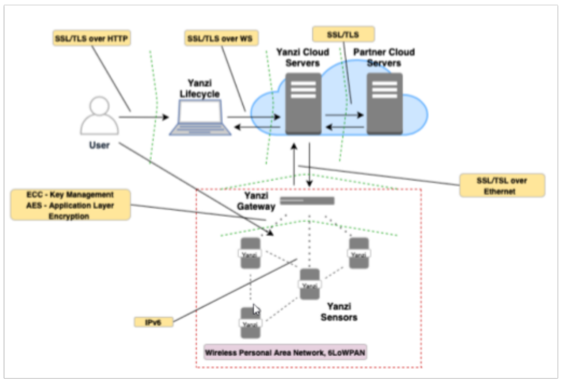
Study 3 - Fridge



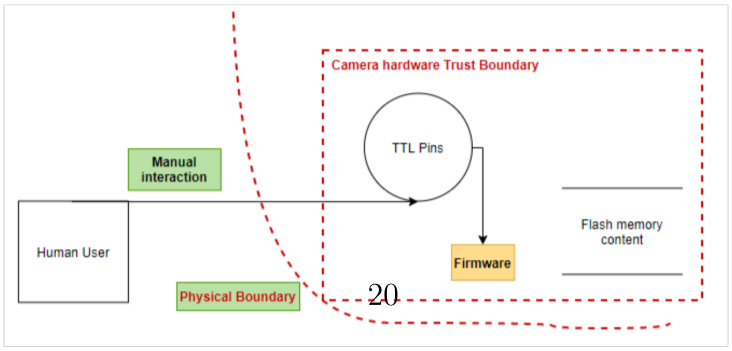
Study 4 - Camera



Study 5 - Door lock

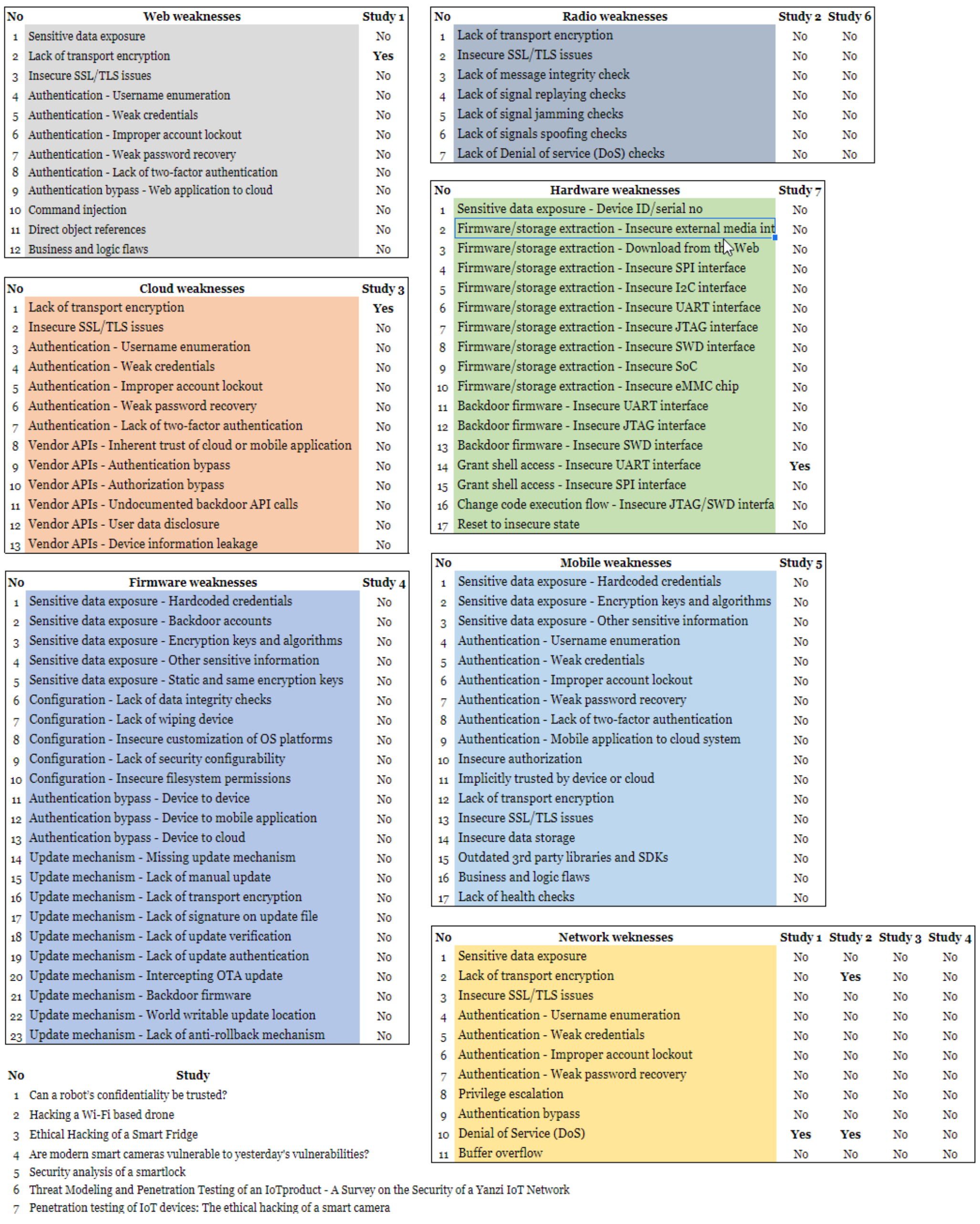


Study 6 - Air quality sensor



Study 7 -Camera 2

Figure 6: Identified vulnerabilities



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 int	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 interfa	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 algorithms	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

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.

5.3. 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. **Threat model.** 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.

5.4. 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. **Threat model.** 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.

5.5. 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. **Threat model.** 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.

5.6. 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*. **Threat model.** 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.

5.7. 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. **Threat model.** 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*.

5.8. Evaluation summary

First, before we had *PatrIoT*, the reported findings didn't 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 a brute-force attack, they did not report it as a weakness. According to *PatrIoT*, it is counted as an "Improper Account Lockout" weakness, as long as the target system allows brute-force attacks. 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., a web application firewall).

6. 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: compilation of the top 100 weaknesses, logical attack surface decomposition, lightweight risk scoring, and step-by-step pentesting guidelines. While addressing these limitations, we also considered the typical research environment: dozens of 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 pentests.

Compilation of top weaknesses. 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 *"Mitre CWE"*, which allowed us to follow a standard in all studies. Defining a baseline also has six benefits: 1) A shortlist motivates researchers from the start because it is short and provides a road map. 2) This approach enables us to easily benefit from the previous report when testing the upgraded version of the device or a similar but new device. 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 using a particular checklist and document 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.

Attack surface decomposition. Examination of the output of many studies published as IoT pentest reports resulted in the discovery that vulnerability research is usually based only on web and network pentesting. According to our experiments, traditional pentesting is not sufficient to ensure that an IoT product is secure because such devices contain unconventional attack surfaces. For example, hardware, firmware, and radio communication are specialized in the IoT landscape. Thus, vulnerability research in the IoT ecosystem indicates that this system is largely characterized by its particular weaknesses, which can only be discovered by applying techniques tailored for IoT. We start threat modeling by decomposing attack surfaces into seven, highlighting IoT-specific threats more than traditional weaknesses. Our decomposition approach ensures that none of the components is overlooked and enables specialized researchers to work in certain domains. It also helps to produce higher-quality outputs and execute specific activities organized and accurately. As threat models are designed using the same technique, they follow a type of template. It accelerates the comprehension of potential threats while evaluating the models.

Lightweight risk scoring. We observed that researchers consider threat modeling to be unnecessarily cumbersome. This may be because they employ complex techniques or do not make use of the model after creating it. We provided default risk impact factors and default severity values for each weakness defined in our compilation. Although our methodology favors prominent attack vectors, starting with high potential weaknesses among them increases motivation once again. This is why we apply risk scoring to prioritize. Because we avoid threat modeling as an overhead, we calculate risk scores more simply than traditional methods. Although we build the threat model before we start pentesting, we can contribute the most by updating this model with the findings at the end of the pentests. As previously mentioned, this is why we utilized an iterative threat modeling approach.

Step-by-step pentesting guidelines. These practical guidelines constitute a very goal-oriented set of information that would enable research activities to be further organized and shortened compared with security risks. Because it touches the complex points of IoT pentesting, it simplifies vulnerability research activities and reduces the time required to conduct IoT pentests. In addition, we report our results with a standard template that familiarizes reviewers and allows authors to reuse previous reports.

Argument 1. Why is the number of weaknesses to be tested sufficiently broad to assist in performing all existing attacks? 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 the pentest with a shortlist of the most common weaknesses known at that moment. Actually, this is why this methodology is agile. In addition, 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. An expert would refer to well-known individual books on hardware hacking, firmware analysis, radio hacking, etc., in the first step. In addition, as previously mentioned, the *CWE* database lists more than a thousand weaknesses, thus this database can be beneficial to understand the remaining attacks while extending this methodology. Moreover, we also suggest documenting the personal knowledge and experience gained from the vulnerabilities discovered during research activities. In reality, times keep changing, and threats evolve accordingly. Eventually, we would need to update this list based on the information harvested from the pentests. Extending the presented methodology will increase the number of attacks and extend the duration of the study accordingly.

Argument 2. Why did we 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, which is why we based our study on the OWASP IoT top 10 project.

Argument 3. Why did we not produce a tool that fully automates the exploitation stage of this study? As mentioned in Introduction, 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 the IoT ecosystem contains seven different attack surfaces, the findings on one surface (e.g., decryption key) could contribute to examinations on other surfaces. In addition, a critical severity vulnerability can be created only by chaining two different vulnerabilities. Many attacks therefore require manual intervention and are conducted in a semi-automated rather than a full automated manner.

Challenges. Even if the methodology was entirely flawless, the quality of work usually depends on the researcher's competence. A qualified IoT vulnerability researcher would focus on the pentest by prioritizing mostly IoT-specific components (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 traditional pentester role to the IoT pentester overnight.

Future work. We have been using *PatrIoT* in our research for some time now, but we will continue to test and validate the methodology in future studies. It will be further improved over time in an agile way. We will present our findings in parallel publications, including information about the discovered vulnerabilities.

7. Conclusion

The devastating effects of IoT insecurity have raised the need for IoT-specific penetration testing capabilities. We identified significant shortcomings regarding the process dimension and challenges regarding the human resources dimension upon examining IoT vulnerability research literature. The overall purpose of this study was to introduce a systematic approach that allows faster adoption (efficiency) 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: compilation of top 100–weaknesses, logical attack surface decomposition, lightweight risk scoring, and step-by-step pentesting guidelines. We also explained how to systematically perform the IoT pentesting activities in these four stages. In addition to this publication, we provide five external publicly available documents via GitHub[14]: 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 inquiry template for IoT information gathering, and an IoT pentest report template.

PatrIoT was evaluated with multiple IoT products in our lab. The empirical results showed 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 allows the maximum benefit to be obtained from the security studies.

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

IoT-specific 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 other fields. The methodology we have shared here is actually a summary of a wealth of information. 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. The minimum requirements to be tested in terms of weaknesses within the scope of the IoT pentesting service can be defined. Likewise, it can be used to measure the quality of IoT penetration test reports.

8. 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. We would also like to thank all the students participating in ethical hacking projects in our lab.

References

[1] J. Deogirikar, A. Vidhate, Security attacks in iot: A survey, in: 2017 International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud)(I-SMAC), IEEE, 2017, pp. 32–37.

[2] D. Minoli, B. Occhiogrosso, Blockchain mechanisms for iot security, Internet of Things 1-2 (2018) 1–13.

[3] E. Ronen, A. Shamir, Extended functionality attacks on iot devices: The case of smart lights, in: 2016 IEEE European Symposium on Security and Privacy (EuroS&P), IEEE, 2016, pp. 3–12.

[4] F. Dang, Z. Li, Y. Liu, E. Zhai, Q. A. Chen, T. Xu, Y. Chen, J. Yang, Understanding fileless attacks on linux-based iot devices with honeycloud, in: Proceedings of the 17th Annual International Conference on Mobile Systems, Applications, and Services, 2019, pp. 482–493.

[5] J. Noorman, J. V. Bulck, J. T. Mühlberg, F. Piessens, P. Maene, B. Preneel, I. Verbauwhede, J. Götzfried, T. Müller, F. Freiling, Sancus 2.0: A low-cost security architecture for iot devices, ACM Transactions on Privacy and Security (TOPS) 20 (3) (2017) 1–33.

[6] M. Antonakakis, T. April, M. Bailey, M. Bernhard, E. Bursztein, J. Cochran, Z. Durumeric, J. A. Halderman, L. Invernizzi, M. Kallitsis, et al., Understanding the mirai botnet, in: 26th USENIX security symposium USENIX Security 17), 2017, pp. 1093–1110.

[7] P. Engebretson, The basics of hacking and penetration testing: ethical hacking and penetration testing made easy, Elsevier (2013).

[8] G. Weidman, Penetration testing: a hands-on introduction to hacking, No Starch Press, 2014.

[9] P. Kim, The Hacker Playbook 3: Practical Guide to Penetration Testing, The Hacker Playbook Series, Independently Published, 2018.

[10] A. Guzman, A. Gupta, IoT Penetration Testing Cookbook: Identify vulnerabilities and secure your smart devices, Packt Publishing Ltd, 2017.

[11] A. Gupta, The IoT Hacker's Handbook: A Practical Guide to Hacking the Internet of Things, Apress, 2019.

[12] Kth pentest reports, `https://bit.ly/38FoGbL`, accessed: 2021-06-12.

[13] K. Peffers, T. Tuunanen, M. A. Rothenberger, S. Chatterjee, A design science research methodology for information systems research, Journal of management information systems 24 (3) (2007) 45–77.

[14] Patriot artifacts, `github.com/beyefendi/penbook/`, accessed: 2021-06-12.

[15] M. Felderer, M. Büchler, M. Johns, A. D. Brucker, R. Breu, A. Pretschner, Security testing: A survey, in: Security Testing: A Survey, Vol. 101 of Advances in Computers, Elsevier, 2016, pp. 1–51. URL `https://www.sciencedirect.com/science/article/pii/S0065245815000649`

[16] W. Xiong, R. Lagerström, Threat modeling–a systematic literature review, Computers & security 84 (2019) 53–69.

[17] W. Xiong, E. Legrand, O. Åberg, R. Lagerström, Cyber security threat modeling based on the mitre enterprise att&ck matrix, Software and Systems Modeling (2021) 1–21.

[18] P. Johnson, R. Lagerström, M. Ekstedt, U. Franke, Can the common vulnerability scoring system be trusted? a bayesian analysis, IEEE Transactions on Dependable and Secure Computing 15 (6) (2016) 1002–1015.

[19] J. Ruohonen, L. Allodi, A bug bounty perspective on the disclosure of web vulnerabilities (2018). `arXiv:1805.09850`.

[20] L. Bilge, T. Dumitraş, 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, 2012, p. 833–844. `doi:10.1145/2382196.2382284`.

Table .4: 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 .5: 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 .6: 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 .7: 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 .8: 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 .9: 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 .10: 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