NRC·CNRC

NRC Publications Archive
Archives des publications du CNRC

A comprehensive survey on IoT attacks: taxonomy, detection mechanisms and challenges

Sasi, Tinshu; Lashkari, Arash Habibi; Lu, Rongxing; Xiong, Pulei; Iqbal, Shahrear

This publication could be one of several versions: author's original, accepted manuscript or the publisher's version. / La version de cette publication peut être l'une des suivantes : la version prépublication de l'auteur, la version acceptée du manuscrit ou la version de l'éditeur.
For the publisher's version, please access the DOI link below./ Pour consulter la version de l'éditeur, utilisez le lien DOI ci-dessous.

Publisher's version / Version de l'éditeur:
https://doi.org/10.1016/j.jiixd.2023.12.001
Journal of Information and Intelligence, 2, 6, pp. 455-513, 2023-12-22

NRC Publications Archive Record / Notice des Archives des publications du CNRC :
https://nrc-publications.canada.ca/eng/view/object/?id=cac39ec4-3edc-4871-856a-a22de42cf3a1
https://publications-cnrc.canada.ca/fra/voir/objet/?id=cac39ec4-3edc-4871-856a-a22de42cf3a1

Access and use of this website and the material on it are subject to the Terms and Conditions set forth at
https://nrc-publications.canada.ca/eng/copyright
READ THESE TERMS AND CONDITIONS CAREFULLY BEFORE USING THIS WEBSITE.

L'accès à ce site Web et l'utilisation de son contenu sont assujettis aux conditions présentées dans le site
https://publications-cnrc.canada.ca/fra/droits
LISEZ CES CONDITIONS ATTENTIVEMENT AVANT D'UTILISER CE SITE WEB.

Questions? Contact the NRC Publications Archive team at PublicationsArchive-ArchivesPublications@nrc-cnrc.gc.ca. If you wish to email the authors directly, please see the first page of the publication for their contact information.

Vous avez des questions? Nous pouvons vous aider. Pour communiquer directement avec un auteur, consultez la première page de la revue dans laquelle son article a été publié afin de trouver ses coordonnées. Si vous n'arrivez pas à les repérer, communiquez avec nous à PublicationsArchive-ArchivesPublications@nrc-cnrc.gc.ca.

National Research Council Canada | Conseil national de recherches Canada

Canada

Journal of Information and Intelligence xxx (xxxx) xxx

Contents lists available at ScienceDirect

Journal of Information and Intelligence

journal homepage: www.journals.elsevier.com/journal-of-information-and-intelligence

A comprehensive survey on IoT attacks: Taxonomy, detection mechanisms and challenges

Tinshu Sasi [a], Arash Habibi Lashkari [a,b], Rongxing Lu [a,*], Pulei Xiong [c], Shahrear Iqbal [c]

[a] Canadian Institute for Cybersecurity (CIC), Faculty of Computer Science, University of New Brunswick (UNB), Fredericton E3B 5A3, Canada
[b] Behaviour-Centric Cybersecurity Center (BCCC), School of Information Technology, York University, Toronto M3J 1P3, Canada
[c] National Research Council, Ottawa K1A 0R6, Canada

ARTICLE INFO

Keywords:
Internet of Things
IoT attacks
Taxonomy
Vulnerabilities
Detection methods
Challenges

ABSTRACT

The Internet of Things (IoT) has set the way for the continuing digitalization of society in various manners during the past decade. The IoT is a vast network of intelligent devices exchanging data online. The security component of IoT is crucial given its rapid expansion as a new technology paradigm since it may entail safety-critical procedures and the online storage of sensitive data. Unfortunately, security is the primary challenge when adopting Internet of Things (IoT) technologies. As a result, manufacturers' and academics' top priority now is improving the security of IoT devices. A substantial body of literature on the subject encompasses several issues and potential remedies. However, most existing research fails to offer a comprehensive perspective on attacks inside the IoT. Hence, this survey aims to establish a structure to guide researchers by categorizing attacks in the taxonomy according to various factors such as attack domains, attack threat type, attack executions, software surfaces, IoT protocols, attacks based on device property, attacks based on adversary location and attacks based on information damage level. This is followed by a comprehensive analysis of the countermeasures offered in academic literature. In this discourse, the countermeasures proposed for the most significant security attacks in the IoT are investigated. Following this, a comprehensive classification system for the various domains of security research in the IoT and Industrial Internet of Things (IIoT) is developed, accompanied by their respective remedies. In conclusion, the study has revealed several open research areas pertinent to the subject matter.

1. Introduction

The IoT is a network of smart assets deployed in various locations, characterized by its openness and comprehensiveness. In particular, it can utilize a range of sensing devices to accomplish the real-time acquisition of diverse monitored, interconnected, interactive entities and the corresponding necessary data. Consequently, these gadgets amalgamate with the Internet to establish an expansive network. Furthermore, the IoT can autonomously arrange and distribute information resources in response to environmental circumstances. The gadgets are utilized for diverse functions within an open network, enhancing user accessibility to each device. Nevertheless, a significant drawback of the IoT is its lack of security procedures comparable to those on servers, personal computers

* Corresponding author.
E-mail addresses: tinshu.sasi@unb.ca (T. Sasi), ahabibil@yorku.ca (A.H. Lashkari), rlu1@unb.ca (R. Lu), Pulei.Xiong@nrc-cnrc.gc.ca (P. Xiong), Shahrear.Iqbal@nrc-cnrc.gc.ca (S. Iqbal).

https://doi.org/10.1016/j.jiixd.2023.12.001
Received 30 November 2023; Received in revised form 13 December 2023; Accepted 14 December 2023
Available online xxxx
2949-7159/© 2023 The Authors. Published by Elsevier B.V. on behalf of Xidian University. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

Please cite this article as: T. Sasi et al., A comprehensive survey on IoT attacks: Taxonomy, detection mechanisms and challenges, Journal of Information and Intelligence, https://doi.org/10.1016/j.jiixd.2023.12.001

(PCs), and laptops. Most embedded devices lack the computational capabilities to implement sophisticated security rules and encryption protocols effectively [1]. In recent years, a significant endeavour has been made to address security concerns within the IoT paradigm. Specific techniques in the realm of IoT security focus on addressing security concerns at a particular layer, while alternative approaches strive to ensure comprehensive end-to-end security for IoT systems [2].

Fig. 1 depicts a comprehensive representation of the IoT and IIoT. The IoT architecture may be categorized into seven levels: perception, transport, edge, processing, application, business, and security layer. The system operates as a closed loop, facilitating the production of customized goods tailored to meet each end customer's particular requirements. The rapid growth of the IoT necessitates implementing robust security and privacy protocols to mitigate potential system vulnerabilities and threats. Moreover, within the IIoT realm, factors like dependability, scalability, and power consumption emerge as crucial considerations. In the present setting, conventional security measures may not always be suitable.

Hence, the primary objective of this survey is to examine the diverse security attacks and develop a taxonomy that can assist researchers in categorizing these attacks based on factors such as attack domains, threat types, execution methods, software surfaces, IoT protocols, device properties, adversary locations, and information damage levels. This categorization will benefit researchers, practitioners, and industry professionals identify the attacks pertinent to their specific application domains. Moreover, individuals in this field can thoroughly analyze the current solutions and generate novel ones based on the nature of the applications they are engaged with. Certain attacks pertinent to the IoT might result in much more severe consequences in an IIoT setting. An instance of a replay attack targeting the commands directed towards an industrial robot responsible for transporting raw materials from the inventory to the production floor has the potential to result in the cessation of operations on the assembly line situated at the production floor.

Overall, the contributions of the paper include:

- A comprehensive taxonomy on IoT attacks on attack domains, attack threat type, attack executions, software surfaces, IoT protocols, attacks based on device property, attacks based on adversary location, and attacks based on information damage level.
- A comprehensive analysis of IoT vulnerabilities and vulnerability databases.
- A study on works related to IoT attack case studies, surveys, and detection methods.
- Presents the existing challenges and future research scopes in IoT attacks.

The overall objective of the paper is to offer a comprehensive reference to the IoT research community for understanding the growth and consequences of IoT attacks. The remainder of the paper is structured into ten sections. Section 2 provides foundational knowledge about IoT, its architecture, and the various attacks it faces. Section 3 offers a comprehensive overview of the research methodology employed in the study. Section 4 delves into an extensive review of related works in the field, providing a background for the research. Section 5 focuses on the specific vulnerabilities associated with IoT systems, shedding light on their weaknesses. In Section 6, the paper discusses the motivation for IoT attacks, IoT security, and IoT privacy. Section 7 introduces a taxonomy of attacks targeting IoT devices and networks, categorizing them for clarity.

Section 8 specifically explores datasets related to IoT attacks, providing valuable resources for researchers. Section 9 discusses the various methods and techniques for detecting and mitigating IoT-related attacks, emphasizing the importance of security measures. Section 10 addresses the challenges encountered during the research and outlines potential areas for further investigation, highlighting the complexity of IoT security. Finally, in Section 11, the paper concludes by summarizing key findings and their implications for the field of IoT security, bringing the research full circle.

2. Overview of IoT, IoT architecture, and IoT attacks

By 2023, more than 43 billion IoT devices will be used worldwide, predicts McKinsey [3]. These billions of devices are already revolutionizing our society, from enabling remote medical monitoring to assisting oil corporations in spill prevention, and they will continue to do so over the following years. Making sure that such a diverse range of devices function seamlessly together is the most challenging aspect, though. This is where the layers, systems, and devices of the IoT architecture come into play [4].

Before delving further into IoT attacks, it is necessary to have a foundational understanding of the topic. Gaining a comprehensive understanding of the many technical terminologies associated with the subject matter is of utmost significance.

- *Vulnerability*: Vulnerabilities refer to inherent weaknesses present in a system or its design, which enable unauthorized individuals to execute instructions, get access to data without proper authorization, and potentially engage in denial-of-service attacks. Vulnerabilities may be identified within several IoT systems domains. Specifically, vulnerabilities might manifest in system hardware or software, as well as in the rules and procedures employed within the systems and even in the behaviours and actions of the system users. IoT systems typically have two primary components: hardware and software. It is not uncommon for both of these components to exhibit design errors. Identifying hardware vulnerabilities poses significant challenges since they are inherently complex and elusive. Furthermore, rectifying these vulnerabilities, even after identification, is difficult due to compatibility, interoperability issues, and the substantial work required for remediation.

  Software vulnerabilities can be identified in several components, including operating systems, application software, and control software, such as communication protocols and device drivers. Several elements contribute to the occurrence of software design errors, encompassing both human factors and the complexity of the software. Technical vulnerabilities often arise as a result of human frailties. The consequences of inadequate comprehension of the requirements encompass commencing the project without a

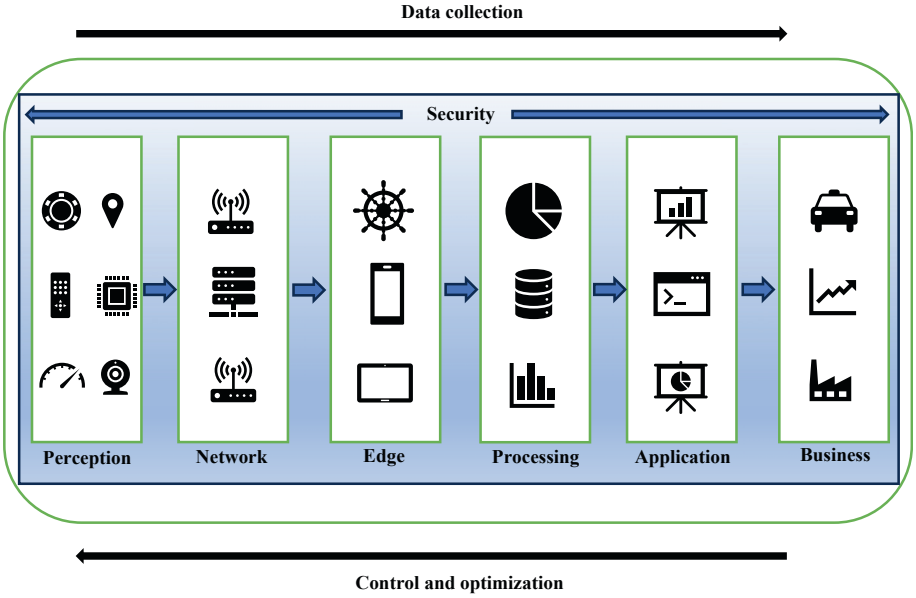

Fig. 1. IoT architecture levels.

well-defined strategy, ineffective communication between developers and users, insufficiency of resources, skills, and expertise, and the inability to effectively manage and govern the system [5].

- *Exposure*: Exposure refers to a problem or mistake in the system setup that permits an unauthorized individual to engage in actions to acquire information. Resilience against physical attacks is a significant challenge within the IoT realm. In most IoT applications, it is common for devices to be unattended and positioned in widely accessible locations, increasing their vulnerability to potential attackers. This level of vulnerability presents the potential for an attacker to get possession of the device, steal cryptographic secrets, manipulate their programming, or substitute them with a hostile device under the attacker's authority [5].
- *Threats*: A threat refers to an intentional or unintentional action that exploits vulnerabilities inside a system, resulting in adverse consequences. Threats may be classified into two main categories: human and natural. Natural hazards, including earthquakes, hurricanes, floods, and fires, can harm computer systems significantly. Limited measures may be taken to mitigate the impact of natural catastrophes, and it is impossible to prevent their occurrence completely. Disaster recovery plans, such as backup and contingency plans, are the most effective strategies for safeguarding systems from natural hazards. Human threats are risks that people create, including internal actors with authorized access and external entities operating outside of the network. These threats are characterized by their malevolent intent, seeking to inflict harm and disrupt the functioning of a system [5].
  Human dangers can be classified into the following categories: (i) Unstructured risks mainly involve people lacking expertise who rely on readily accessible hacking tools. (ii) Structured threats refer to those who possess knowledge of system vulnerabilities and possess the ability to comprehend, create, and utilize programs and scripts for exploitation. One illustration of a structured threat is the phenomenon known as advanced persistent threats (APT). It is a complex network attack that targets explicitly valuable information within commercial and government entities, including manufacturing, finance, and national defence, to acquire data illicitly [5].
- *Attacks*: Attacks refer to deliberate acts undertaken to inflict harm on a system or interrupt its regular operations by capitalizing on weaknesses through diverse tactics and tools. Attackers initiate offensive actions to attain particular objectives driven by personal gratification or the need for compensation. Attack cost refers to quantifying the exertion required by an attack, encompassing factors such as their level of skill, available resources, and level of motivation [5].
  Attack actors are those who pose a significant risk to the realm of digital technology. Potential actors in this context include individuals with expertise in hacking, individuals engaging in illicit activities, and even governmental entities. Some of the most common types of attacks are active network attacks, which involve watching unencrypted traffic to find sensitive data. Passive network attacks involve watching communications that are not adequately protected in order to decrypt weakly encrypted traffic and get authentication information, close-in attacks, and exploitation by insiders.

2.1. IoT architecture

The IoT architecture pertains to the arrangement and configuration of IoT devices to meet users' specific needs and demands. The components of an IoT system are categorized into a range of three to seven layers based on their level of complexity, with each layer

serving a distinct role. The absence of standardized protocols in the architecture of the IoT poses more challenges regarding interoperability, security, and several other concerns. The architecture of the IoT has the potential to encompass a maximum of seven layers [4].

- *Perception Layer*: The perception layer, often called the device layer, comprises various types of sensors, RFID scanners, security cameras, GPS modules, etc. These devices may be used with industrial machinery, such as conveyor systems, industrial robots, automated guided vehicles (AGVs), etc. These devices gather sensory data, keep an eye on the factory floor and the surrounding environment, transfer raw materials, etc. [2].
- *Transport Layer/Network Layer*: The transport/network layer is responsible for transmitting information to the processing systems of the next layer [2]. IoT gateways must first convert the incoming input from analogue to digital format. After that, the gateway can send the data to a local or cloud data centre using several data transfer protocols (DTPs) [4].
  IoT protocols are a collection of rules and regulations that govern the communication between devices in systems that can detect, gather, and share data in real-time. IoT protocols come in a wide variety, each with its characteristics and functionalities. While some IoT protocols are more general-purpose and may be used in a variety of different situations, others are more particular and are developed for specific purposes, such as wireless communication, data transmission, or device management [6]. The proper protocol can ensure the ability of devices to connect and share data properly, which is essential for the IoT system's seamless operations. Some of the leading IoT protocols are Bluetooth, Wi-Fi, Zigbee, Z-Wave, 6LoWPAN (IPv6 over Low-Power Wireless Personal Area Networks), MQTT (Message Queue Telemetry Transport), CoAP (Constrained Application Protocol), DDS (Data Distribution Service), AMQP (Advanced Message Queuing Protocol).
- *Edge Layer*: The edge layer in IoT workloads encompasses the physical hardware of devices, their embedded operating systems for process management, and device firmware, which includes the software and instructions embedded in IoT devices [7]. As IoT networks expand and more devices connect to a central hub, latency emerges as a key performance challenge. Edge computing, implemented through the IoT system's edge layer, addresses this by facilitating data processing and analysis proximate to the data source. Latency, the time delay between detection and response, is a critical issue for connected devices. By situating computing power near the sensors, or essentially at the company's edge, latency can be significantly reduced, enabling devices to communicate and exchange data instantaneously.
  For specific application scenarios, this may be crucial. For instance, any delay in detecting an impending accident and changing direction (steering, braking) might have disastrous results in an autonomous car moving at highway speeds. For industrial safety systems like fire alarms, seconds wasted in data transmission might be crucial. The ability to broadcast what they observe in the form of data packets to nodes that further process the data is a characteristic shared by all IoT edge devices [8]. Specific “smart” edge devices are configured to stop the target function or start a damage control operation when a significant abnormality is detected. Edge layer can also provide a path to accelerate and simplify data processing and provide needed insights where and when required.
- *Processing Layer*: The processing layer, also known as the middleware layer, consists of servers and databases and carries out various activities such as decision-making, computing optimization algorithms, and storing vast quantities of data [2]. This layer comprises cloud computing platforms that can analyze and interpret data from the real world. It accepts raw sensor data and converts it into meaningful information using cloud services and extensive data modules. In addition, the processing layer enables the system to act on inputs and responses in real-time. It can make judgments and perform on the data it receives. Based on the data obtained in the perception stage, this layer is also utilized to develop predictions and give insights [9].
- *Application Layer*: The application layer of IoT infrastructure is where data is analyzed to address business problems or accomplish objectives. The application layer delivers application-specific needs to cater to end users. This layer is made up of applications and services that are built on top of the processing layer. Software tools translate data from the processing layer into relevant information for humans or automated operations [10].
  For users, it is the layer that is most obvious. Users get access to the data gathered and processed in the previous layers through this layer. It enables users to observe, examine, and respond to network data. Users may monitor IoT devices and analyze data through tools like dashboards and mobile applications while interacting directly with the application layer. Integration of IoT software with middleware is achieved using application programming interfaces (APIs) [2,9,10].
- *Business Layer*: The business layer is where the application layer's decisions based on data and solutions are derived. It might include many instances of the application layer. At the business layer level, patterns that may be decoded from the application layer are used to elucidate business insights further, forecast future trends, and guide operational decisions that enhance productivity, security, cost-effectiveness, customer satisfaction, and other crucial business functioning factors. The business layer aims to manage business transactions and models tied to connected devices. Business process management, business analytics, and business rules are all included in this layer. It is the layer responsible for managing the business logic and establishing business procedures to guarantee that all the IoT system's business objectives are achieved [4,9,10].
- *Security Layer*: The security layer extends throughout all IoT architecture layers and is critical to the success of an IoT solution. An IoT system often exchanges sensitive data. There are three primary components to the IoT security layer [4,10].
  - *Device Security*: It starts with devices with hardware and firmware protections at the perception layer. It entails guarding against viruses and hijacks on real IoT endpoints. To keep your devices safe, you should use cryptographic keys for correct authentication, hardened exteriors to keep them safe, stop unauthorized code from running on connected devices, and fix problems with firmware updates and security patches [10].
  - *Connection Security*: It typically employs encryption to secure data transported across networks. The baseline for the security of IoT connections is the transport layer security (TLS) protocol. End-to-end encryption eliminates the risk of unauthorized users intercepting and misusing data [10].

– *Cloud Security*: Encrypting stored data is a critical component of cloud security since it lowers the possibility of sensitive data being exposed through data breaches. Strong authentication and permission controls must be used to restrict access to IoT applications. To protect against malicious devices, devices must be authorized before connecting to the cloud or an IoT system [10].

2.2. IoT attacks

This section addresses the background of IoT attacks in a lucid manner. IoT attacks are cyber-attacks that employ any IoT device to access consumers' sensitive data. Attackers typically install malware on the device, damage it, or get access to further organizations' data. Since IoT devices are not designed with adequate security mechanisms. As a result, they are one of an organization's weakest links and offer a significant security risk [11].

A basic IoT device often lacks the necessary built-in security solution to combat cyber attacks. Most IoT devices have limited applications and purposes since they aim to perform simple activities. As a result, their security posture is frequently overlooked, exposing them to cyber attacks. Common vulnerabilities and zero-day exploits might aid hackers or organizations in breaching IoT devices and utilizing them in various ways to carry out comprehensive cyber attacks [12].

2.2.1. IoT attacks vs IT attacks

Compared to conventional IT (Information Technology) attacks, IoT attacks bring new problems that need specialized security solutions to guard against these dangers fully [13]. Some of the different ways are:

- *Attack Surface*: IoT devices frequently have low processing speeds and resources. So they may lack security features to protect against attacks, leaving them more vulnerable to attacks than IT.
- *Diversity of Devices*: IoT device types differ significantly in form factor, operating systems, and network connection. Standardized security procedures are, therefore, more complicated, making some devices more susceptible to attacks.
- *Physical Impact*: IoT devices are frequently employed in crucial infrastructure or life-sustaining systems, such as medical equipment; therefore, a cyberattack on these devices might have very harmful physical repercussions. Conversely, most IT attacks aim to steal data or interfere with services.
- *Legacy Devices*: Usually IoT devices have a longer lifespan. Thus, there will be many old devices connected and in use. Older devices cannot get software upgrades or security patches, making them more vulnerable to attacks or compromises.

Table 2 summarizes the difference between IT and IoT attacks. Following are some common IoT vulnerabilities that attract IoT devices to IoT attacks [14].

- *Weak/Default Passwords*: Lack of robust password recovery mechanism; weak or default password; enforcement of stronger password rules; inability to modify the default username and password.

Table 1
Top 10 OWASP IoT vulnerabilities.

Rank	Vulnerability	Description
1	Weak, guessable, or hardcoded passwords	The utilization of credentials that are susceptible to brute force attacks, openly accessible to the public, or unalterable, which may include the presence of backdoors in firmware or client software that enable illegal entry into deployed systems.
2	Insecure network services	The presence of unnecessary or vulnerable network services operating on the device itself, particularly those accessible via the internet, poses a threat to the confidentiality, integrity, and availability of information, as well as the potential for unwanted remote control.
3	Insecure ecosystem interfaces	The presence of insecure online, backend API, cloud, or mobile interfaces in the external ecosystem might potentially compromise the device or its associated components. Frequent concerns encompass the absence of authentication or authorization, inadequate or insufficient encryption, and insufficiency in input and output filtering.
4	Lack of secure update mechanism	The device's inability to undergo secure updates. The absence of firmware validation on the device, inadequate secure delivery during transit (lack of encryption), insufficiency of anti-rollback procedures, and the absence of alerts on security changes resulting from upgrades.
5	Use of insecure or outdated components	The utilization of outdated or vulnerable software components or libraries that may potentially expose the device to unauthorized access. This encompasses the unsafe customization of operating system platforms and utilizing third-party software or hardware components obtained from a hacked supply chain.
6	Insufficient privacy protection	The user's personal information is kept on the device or within the ecosystem in a manner that lacks security, is utilized inappropriately, or is accessed without sufficient authorization.
7	Insecure data transfer and storage	The absence of encryption or access control measures for sensitive data throughout the ecosystem encompassing data at rest, in transit, and during processing.
8	Lack of device management	One of the key issues observed in devices deployed in production is the absence of adequate security support. This includes deficiencies in asset management, update management, secure decommissioning, systems monitoring, and response capabilities.
9	Insecure default settings	Devices or systems are sometimes distributed with unsecured default settings or cannot enhance security by limiting operators from altering configurations.
10	Lack of physical hardening	The absence of physical hardening measures enables potential attackers to acquire crucial information that may facilitate a subsequent remote attack or enable them to assume local control of the device.

Table 2
IoT attacks vs IT attacks.

Category	IoT attacks	IT attacks
Attack surface	Limited resources, higher vulnerability	Robust security, lower vulnerability
Diversity of devices	Varied types, complex security	Standardized, simplified security
Impact	Harmful physical consequences	Data theft, service disruption
Legacy devices	Older devices, no updates, higher risk	Regular updates, lower risk

- *Insecure Network Services*: Adversaries use flaws in IoT devices' communication protocols and services to compromise and breach confidential or sensitive information sent between the device and a server. For instance, man-in-the-middle (MitM) attacks seek to utilize these flaws to steal the endpoint authentication credentials required to launch larger attacks.
- *Insecure Ecosystem Interfaces*: Insecure web, backend API, cloud, or mobile interfaces in the ecosystem outside of the device that allows compromise of the device or its related components. E.g., weak encryption, a lack of input and output filtering, and a lack of authentication/authorization
- *Lack of Secure Update Mechanism*: Lack of secure device update capability. This includes a lack of device-based firmware validation, insecure delivery (transferred data is not encrypted), anti-rollback processes, and alerts of security changes brought on by upgrades.
- *Use of Insecure or Outdated Components*: Use of outdated or unsafe software components or libraries that might make the device vulnerable to attack. This involves using third-party software or hardware from a compromised supply chain and insecurely customizing system platforms. Vulnerabilities in software dependencies or outdated systems may jeopardize the security of the IoT ecosystem. Manufacturers who design their IoT devices with open-source modules have a convoluted supply chain that is challenging to keep track of. These components might inherit vulnerabilities known to the attackers, developing the potential risk environment available for exploitation.
- *Insufficient Privacy Protection*: Users' private data that is unintentionally, incorrectly, or unlawfully utilized and stored on the device or in the ecosystem. These personal details might include medical data, energy usage, and driving habits. A lack of appropriate controls will jeopardize privacy, and there may be legal repercussions if the proper measures are not in place.
- *Insecure Data Transfer and Storage*: Lack of encryption or access control of sensitive data anywhere within the ecosystem, whether at rest, in transit, or during processing. Since data are utilized in automated controls and decision-making procedures, it is crucial to the dependability and integrity of IoT applications. If unauthorized access or usage occurs, then it will have detrimental effects.
- *Lack of Device Management*: Lack of security support for production-ready devices, including asset management, update management, secure decommissioning, systems monitoring, and response capabilities. Unauthorized devices can access business networks, monitor activity, and intercept data if exposed to the IoT ecosystem.
- *Insecure Default Settings*: Systems or devices that cannot improve system security by preventing users from changing configurations or that arrive with insecure default settings. Attackers can target hardcoded default passwords, hidden backdoors, and vulnerabilities in the device firmware once these settings have been obtained. The user finds it hard to change these settings at the same time.
- *Lack of Physical Hardening*: Lack of physical defences enables potential attackers to get hold of confidential information that might be used in a future remote attack or take over the device locally. IoT devices are set up in remote and scattered settings. By obtaining access to the physical layer and making changes, an attacker may interfere with the services provided by IoT devices.

Among all these shortcomings, weak/default passwords are identified as one of the top vulnerabilities in IoT devices [15,16]. To access the device easily, users skip changing the passwords or update poor passwords that Brute Force attacks can crack. The attackers perform further infection once the device is accessed based on their objectives [17].

3. Research methodology

This section presents the methodology of selecting relevant articles, different data sources, search criteria, research questions, taxonomy creation, and the scope of this work. In this paper, we have proposed a taxonomy developed based on the study of many research articles and resources on IoT attacks. To have meaningful research, most of the papers were selected from various databases, including IEEE Xplore, ScienceDirect, SpringerLink, ACM Digital Library, and Google Scholar. On each database, search keywords like “IoT attacks”, “IoT malware”, “Smart home attack”, “IoT attack”, “IoT malware detection”, “IoT attack detection”, “IoT attack survey”, “IoT attack machine learning”, and “IoT attack review” were considered which led to finding relevant papers.

Moreover, we have included a backward approach for finding new articles and identifying new ones based on the list of references in the article under study. Furthermore, we narrowed the analysis by considering the following questions:

(1) What is the current research state of IoT attacks as new attacks are emerging daily?
(2) What are the causes of IoT attacks?
(3) What are the different aspects of IoT attacks?
(4) What are the different aspects of IoT vulnerabilities, and what are popular IoT vulnerability databases?
(5) Which techniques are used to perform IoT attack detection?
(6) How can the research gaps identified in the existing IoT attack detection studies be filled?

All articles from non-English journals or conferences were excluded.

Following the abovementioned criteria, we have selected articles and identified different aspects of IoT attacks. On identifying each attribute, we briefly described one or two related works on that attribute. These associated works are considered from online resources like Medium, Bleeping Computer, Malwarebytes, etc.

4. Related works

Despite various studies on IoT attacks, comprehensive literature on the taxonomy of IoT attacks is currently lacking. Hence, we put up a novel taxonomy in this part that examines prevailing surveys and taxonomies.

Numerous scholarly publications extensively examine the diverse facets of threats and attacks within the IoT domain. The aforementioned literary works [18–22] comprehensively discuss the categorization of threats and attacks associated with the IoT. The primary emphasis of these publications mainly revolves around two overarching categories, namely the architectural aspects of the IoT and the protocols and standards employed within the IoT domain. While the literature on threats and attack taxonomy provides extensive coverage, it is worth noting that only a limited number of studies address potential countermeasures.

Nevertheless, it is worth noting that the works above do not provide any remedies against dangers and attacks that arise from the widespread use of ubiquitous technologies such as blockchain (BC), fog computing (FC), edge computing (EC), and machine learning (ML). The authors have conducted surveys on various ubiquitous technologies for analyzing threats and attacks in a fragmented manner, as documented in the following sources: [23,30–34].

In their article, [35] conducted a comprehensive analysis of security attacks in the IoT and provided an overview of machine learning approaches that have been employed to address these attacks. The authors examined 78 publications published up to 2017, highlighting the solutions, problems, and areas of study that remain unexplored in this domain. In a survey conducted in 2018, the authors [36] examined various attack models targeting the IoT. These models encompass spoofing attacks, denial-of-service attacks, jamming, and eavesdropping. The authors also proposed potential security measures to mitigate these threats, which included IoT authentication, access control, malware detection, and secure offloading. Notably, the suggested security solutions incorporated machine learning techniques [24].

5. IoT vulnerabilities

Based on the author [47], we can categorize IoT vulnerabilities in 9 categories. Also, Table 6, Table 7, Table 8, Table 9, Table 10, Table 11 show the mapping between IoT attacks and IoT vulnerabilities.

- *Deficient Physical Security*: Most IoT devices operate independently in unsupervised environments. Adversaries can easily get physical access to these devices and take control. As a result, attackers may cause physical damage to devices, expose cryptographic algorithms, duplicate firmware, or compromise control or cyber data.
- *Insufficient Energy Harvesting*: IoT devices usually have a finite energy source and may not be able to recharge themselves. An attacker can drain stored energy by flooding devices with valid or corrupted messages, impeding lawful use.
- *Inadequate Authentication*: The IoT paradigm's energy and processing constraints make complex authentication methods difficult to deploy. An attacker may exploit weak authentication to install fake hostile nodes or compromise data integrity to infiltrate IoT

Table 3
Comparison of IoT attack surveys.

Year	Title	# Attacks discussed	Taxonomy	Attack to vulnerability mapping	Detection methods	Challenges in detection methods
2020	A survey on privacy and security of Internet of Things [19]	12	Yes	No	Yes	Yes
2017	A survey of intrusion detection in Internet of Things [20]	0	Yes	No	Yes	Yes
2019	Intrusion detection systems in the Internet of Things: A comprehensive investigation [21]	9	Yes	No	Yes	Yes
2018	Internet of things security: A top-down survey [22]	0	Yes	No	Yes	Yes
2021	State-of-the-art review on IoT threats and attacks: Taxonomy, challenges and solutions [23]	59	Yes	No	Yes	Yes
2020	Machine learning based solutions for security of Internet of Things (IoT): A survey [24]	31	Yes	No	Yes	Yes
2018	IoT security: Review, blockchain solutions, and open challenges [25]	19	Yes	Yes	Yes	Yes
2018	A comprehensive IoT attacks survey based on a building-blocked reference model [26]	51	Yes	No	No	No
2020	A comprehensive survey on attacks, security issues and blockchain solutions for IoT and IIoT [2]	22	Yes	No	No	No
2022	A survey on IoT security: Attacks, challenges and countermeasures [27]	17	Yes	No	Yes	Yes
2021	A survey on security attacks and solutions in the IoT network [28]	17	Yes	No	No	No
2021	A survey on classification of cyber-attacks on IoT and IIoT devices [29]	32	Yes	No	Yes	Yes
2023 (Our survey)	A comprehensive survey on IoT attacks: Taxonomy, detection mechanisms and challenges	149	Yes	Yes	Yes	Yes

Table 4
List of IoT vulnerabilities databases.

IoT databases	Total vulnerabilities count	Description
National Vulnerability Database (NVD)	226,654	The United States government maintains the National Vulnerability Database (NVD), a vast repository of vulnerability management data that adheres to established standards. The provided data facilitates automating vulnerability management, security measurement, and compliance. The National Vulnerability Database (NVD) comprises a collection of databases, including references to security checklists, software vulnerabilities connected to security, misconfigurations, product terminology, and metrics measuring the effect of these vulnerabilities [37].
Chinese National Vulnerability Database (CNVD)	117,454	The Chinese National Vulnerability Database (CNNVD) serve as the official national vulnerability databases for the People's Republic of China. The China Information Technology Evaluation Center, which is a division of the Chinese Ministry of State Security (MSS), maintains and manages the website at https://www.cnvd.org.cn [38].
Chinese National Vulnerability Database of Information Security (CNNVD)	117,454	Chinese National Vulnerability Database of Information Security, a second database from China. Usually follows data found in NVD [38].
Japan Vulnerability Notes Database (JVNDB)	143,807	JVNDB is a database and information resource providing information about software and hardware product vulnerabilities. JVNDB is operated and maintained by the Japan Cybersecurity Emergency Response Team (JPCERT/CC), which is a government organization in Japan responsible for responding to and mitigating cybersecurity threats [39].
ICS Vulnerability Database (IVD)	1342	Winicssec, a Chinese company specialising in ICS security, offers the ICS Vulnerability Database as a comprehensive resource. This dataset incorporates information obtained from other sources, including the National Vulnerability Database (NVD), the China National Vulnerability Database (CNVD), and the China National Vulnerability Database (CNNVD) [39].
Industrial Control Systems Cyber Emergency Response Team (ICS-CERT-CN)	1342	ICS-CERT-CN, or the Industrial Control Systems Cyber Emergency Response Team for China, is an organization that focuses on cybersecurity for critical infrastructure and industrial control systems within China. ICS-CERT-CN operates under the auspices of the Chinese government and works to enhance the security and resilience of critical infrastructure sectors, such as energy, water, transportation, and manufacturing, by providing cybersecurity guidance, vulnerability assessments, incident response support, and threat intelligence [40].
United States Computer Emergency Readiness Team (US-CERT)	1012	The United States Computer Emergency Readiness Team (US-CERT) is responsible for evaluating and mitigating cyber threats and vulnerabilities and distributing pertinent information on cyber threat warnings. Additionally, US-CERT coordinates activities related to incident response. The division utilizes its extensive network and digital media analysis capabilities to effectively address and combat hostile activities targeting domestic and international networks [41].

Table 5
List of IoT vulnerabilities databases.

IoT databases	Total vulnerabilities count	Description
Other CERTs	Multiple sources	The webpages of other CERTs, deemed less valuable, were consolidated into a unified entry.
ZDI	1493 in 2023 (As of Oct 4, 2023)	The Zero Day Initiative (ZDI) was established to incentivize researchers to confidentially disclose zero-day vulnerabilities to the companies impacted by offering financial rewards. ZDI is recognized as the most extensive bug bounty program globally, catering to several vendors without bias. The Zero Day Initiative (ZDI) distinguishes itself from other projects regarding its approach to acquiring vulnerability information. The disclosure of technical specifics regarding the vulnerability is withheld from the public until the vendor officially issues a fix [42].
Bugtraq	16,431 in 2023 (As of Oct 4, 2023)	Bugtraq was an electronic mailing list that focused on matters related to computer security. On-topic matters encompass recent deliberations concerning vulnerabilities, security-related announcements from vendors, techniques of exploitation, and remedial measures [43].
Vulners	Multiple sources	Vulners is widely regarded as a reliable and authoritative resource that provides extensive and regularly updated information about vulnerabilities. Vulners is a platform that converts fragmented information into comprehensive vulnerability and threat data specifically tailored for human and machine consumption [44].
Exploitee.rs	Multiple sources	Exploitee.rs was a website and community known for its focus on hacking, security research, and exploitation of various technology platforms, including game consoles, smart TVs, and other embedded devices. The community was active in the early to mid-2010s and gained recognition for finding vulnerabilities and exploits in consumer electronics [45].
Common vulnerabilities and exposures (CVE)	213,341	CVE is a widely recognized system for identifying and tracking vulnerabilities in various software and hardware products, including IoT devices [46].

devices and network connections. Transferred and used authentication keys are always at risk of loss, destruction, or corruption. Complex authentication systems are less effective when keys are not securely held or transferred.

- *Improper Encryption*: Data preservation is vital in IoT domains, especially in critical cyber-physical systems (CPS) like electricity utilities, industrial facilities, and building automation. Data is securely stored and sent via encryption, which ensures that only authorized users can access and utilize it. Cryptographic systems depend on their algorithms, but IoT resource restrictions affect their

Table 6
IoT attacks to vulnerability mapping [74–78].

IoT attack	Vulnerabilities
Tampering	CVE-2019-17662, CVE-2020-12345, CVE-2018-7654, CVE-2021-4321, CVE-2017-8901
Hardware fault injection	CVE-2018-3621, CVE-2019-9213, CVE-2020-8833, CVE-2021-3156, CVE-2022-1234
Software fault injection	CVE-2019-15894, CVE-2022-42961
Sleep denial attack	CVE-2023-29761, CVE-2023-38905, CVE-2021-37183, CVE-2020-13128, CVE-2019-6190, CVE-2015-5310, CVE-2014-2672, CVE-2007-1337
RF spoofing/jamming	CVE-2019-18659
Fake node injection	CVE-2023-33566
Social engineering	CVE-2023-37901, CVE-2023-37203, CVE-2023-2746, CVE-2023-2706, CVE-2023-25815, CVE-2023-23618, CVE-2023-1552, CVE-2022-4261, CVE-2022-36633,CVE-2022-35228, CVE-2022-30280, CVE-2022-24919, CVE-2022-24918, CVE-2022-24917, CVE-2022-24441, CVE-2022-24349, CVE-2021-45099, CVE-2021-41380, CVE-2021-41157, CVE-2021-41156, CVE-2021-41139, CVE-2021-38150, CVE-2021-37624, CVE-2021-34683, CVE-2021-34419, CVE-2021-30650, CVE-2021-29436, CVE-2021-22098, CVE-2021-1585, CVE-2020-8994, CVE-2020-6244, CVE-2020-6195, CVE-2020-5629, CVE-2020-5628, CVE-2020-24847, CVE-2020-15959, CVE-2020-10263, CVE-2019-9807, CVE-2019-6795, CVE-2019-12773, CVE-2019-0062, CVE-2018-6353, CVE-2018-5411, CVE-2018-1432, CVE-2018-12246, CVE-2018-12241, CVE-2018-11633, CVE-2018-11632, CVE-2018-0819, CVE-2017-9963, CVE-2017-7969, CVE-2017-7840, CVE-2017-7839, CVE-2017-6955, CVE-2017-5858, CVE-2017-5606, CVE-2017-5605, CVE-2017-5604, CVE-2017-5603, CVE-2017-5602, CVE-2017-5593, CVE-2017-5592, CVE-2017-5591, CVE-2017-5590, CVE-2017-5589, CVE-2016-9092, CVE-2016-5124, CVE-2016-4048,CVE-2016-2840, CVE-2015-3439, CVE-2010-1240, CVE-2009-2982, CVE-2004-1778, CVE-2001-1410, CVE-1999-0181
Side channel attacks	CVE-2023-37479, CVE-2023-25000, CVE-2022-48251, CVE-2022-35888, CVE-2022-26382, CVE-2022-23304, CVE-2022-23303, CVE-2021-31829, CVE-2021-29155, CVE-2020-4699, CVE-2020-4661, CVE-2020-4660, CVE-2020-27171, CVE-2020-27170, CVE-2020-12912, CVE-2020-12402, CVE-2020-11713, CVE-2019-9495, CVE-2019-9494, CVE-2019-7308, CVE-2019-19960, CVE-2019-18222, CVE-2019-16910, CVE-2019-15703, CVE-2019-13377, CVE-2019-11743, CVE-2018-5407, CVE-2018-12433, CVE-2018-11846, CVE-2018-0737, CVE-2017-13218, CVE-017-12872, CVE-2017-0379, CVE-2016-9074, CVE-2016-6489, CVE-2014-5270, CVE-2013-4576, CVE-2013-1624, CVE-2013-1623, CVE-2013-1620, CVE-2013-1619, CVE-2013-1618, CVE-2013-0169
Permanent denial of service	CVE-2023-41349, CVE-2023-32470, CVE-2023-28071, CVE-2023-21280, CVE-2023-20910, CVE-2022-40227, CVE-2022-36848, CVE-2022-24925, CVE-2022-23968, CVE-2022-20731, CVE-2022-20661, CVE-2022-20425, CVE-2022-20143, CVE-2021-41546, CVE-2021-40127, CVE-2021-39670, CVE-2021-39624, CVE-2021-32623, CVE-2021-25474, CVE-2021-25473, CVE-2021-25452, CVE-2021-25334, CVE-2021-0338, CVE-2020-3567, CVE-2020-28393, CVE-2020-1700, CVE-2020-0318, CVE-2020-0021, CVE-2019-1947, CVE-2019-15855, CVE-2018-3834, CVE-2018-15453, CVE-2017-9538, CVE-2016-8467, CVE-2016-8395, CVE-2016-6763,
Malicious code injection	CVE-2023-40221, CVE-2023-3006, CVE-2023-28849, CVE-2023-27474, CVE-2023-25719, CVE-2023-20887, CVE-2022-42009, CVE-2022-40294, CVE-2022-3853, CVE-2022-37317, CVE-2022-27665, CVE-2022-24915, CVE-2022-24881, CVE-2022-23808, CVE-2022-23068, CVE-2022-22985, CVE-2021-41128, CVE-2021-22439, CVE-2020-8966, CVE-2020-7493, CVE-2020-7489, CVE-2020-7475, CVE-2020-3207, CVE-2020-24036, CVE-2020-12782, CVE-2020-10803, CVE-2019-3652, CVE-2019-3427, CVE-2019-19676, CVE-2019-17661, CVE-2019-11872, CVE-2017-8188, CVE-2017-6782, CVE-2003-0394, CVE-2002-1135

resilience, efficiency, and efficacy. A potential attacker may be able to circumvent encryption to reveal private data or alter actions with little effort.

- *Unnecessary Open Ports*: Numerous IoT devices exhibit the presence of open ports that are not essential for their operation, exposing vulnerable services. This allows malicious actors to establish connections and exploit various vulnerabilities.
- *Insufficient Access Control*: Secure IoT devices and data require good credential management. Many IoT devices lack complicated passwords in conjunction with their cloud management systems. Many devices don't prompt users to change their default passwords after installation. Most customers have high permissions. Consequently, an attacker can get unauthorized access to the device, threatening data security and Internet integrity.
- *Improper Patch Management Capabilities*: It's crucial to periodically patch IoT operating systems and embedded firmware and software. This method reduces vulnerabilities and improves functionality. Many reports show manufacturers routinely fail to apply security fixes or lack automated patch updates. Existing update systems lack integrity guarantees, rendering them vulnerable to malicious changes and broad use.
- *Weak Programming Practices*: Numerous researchers have shown that many firmware updates include well-documented vulnerabilities despite rigorous development techniques and security components. These vulnerabilities include backdoors, root users as major access points, and SSL absence. Thus, an attacker can use well-known security flaws to perform buffer overflows, modify data, or get device access.
- *Insufficient Audit Mechanisms*: Numerous IoT devices exhibit a deficiency in comprehensive logging protocols, enabling malicious activity to be concealed from IoT sources.

The top 10 OWASP IoT vulnerabilities in IoT are highlighted in Table 1. The paper [47] also presents a classification of IoT vulnerabilities, which are categorized as indicated in Fig. 2.

- *Layers*: The layers category investigates the impact of the various elements within the IoT domain on the vulnerabilities in IoT systems. The class is broken into three subclasses, which are device-based, network-based, and software-based. This division is based on intuitive categorization. Device-based addresses the vulnerabilities linked to the IoT hardware components. On the one hand, network-based vulnerabilities pertain to gaps in communication protocols that give rise to IoT vulnerabilities. On the other hand, software-based vulnerabilities encompass those associated with the firmware and/or software of IoT devices [47].

Table 7
IoT attacks to vulnerability mapping [74–78].

IoT attack	Vulnerabilities
SQL injection	CVE-2023-5374, CVE-2023-5373, CVE-2023-5350, CVE-2023-5322, CVE-2023-5300, CVE-2023-5298, CVE-2023-5294, CVE-2023-5293, CVE-2023-5285, CVE-2023-5283, CVE-2023-5282, CVE-2023-5281, CVE-2023-5280, CVE-2023-5279, CVE-2023-5278, CVE-2023-5276, CVE-2023-5272, CVE-2023-5271, CVE-2023-5270, CVE-2023-5269, CVE-2023-5268, CVE-2023-5267, CVE-2023-5266, CVE-2023-5265, CVE-2023-5264, CVE-2023-5261, CVE-2023-5260, CVE-2023-5258, CVE-2023-5153, CVE-2023-5152, CVE-2023-5151, CVE-2023-5033, CVE-2023-5032, CVE-2023-5031, CVE-2023-5030, CVE-2023-5029, CVE-2023-5027, CVE-2023-5023, CVE-2023-5020, CVE-2023-5019, CVE-2023-5018, CVE-2023-5017, CVE-2023-5014, CVE-2023-4987, CVE-2023-4974, CVE-2023-4928, CVE-2023-4899, CVE-2023-4873, CVE-2023-4872, CVE-2023-4871, CVE-2023-4867, CVE-2023-4866, CVE-2023-4852, CVE-2023-4851, CVE-2023-4850, CVE-2023-4849, CVE-2023-4848
Command injection	CVE-2023-5301, CVE-2023-4873, CVE-2023-4835, CVE-2023-4833, CVE-2023-4832,CVE-2023-4831, CVE-2023-4830, CVE-2023-4766, CVE-2023-4737, CVE-2023-4711, CVE-2023-4673, CVE-2023-4670, CVE-2023-4661, CVE-2023-4542, CVE-2023-4531, CVE-2023-4414, CVE-2023-4412, CVE-2023-4411, CVE-2023-4410, CVE-2023-43893, CVE-2023-43892, CVE-2023-43891, CVE-2023-43890, CVE-2023-43477, CVE-2023-43207, CVE-2023-43206, CVE-2023-43204, CVE-2023-43202, CVE-2023-43138, CVE-2023-43137, CVE-2023-43130, CVE-2023-43129, CVE-2023-43128, CVE-2023-4310, CVE-2023-42810, CVE-2023-4212, CVE-2023-41738, CVE-2023-41636, CVE-2023-41331, CVE-2023-41303, CVE-2023-4120, CVE-2023-41149, CVE-2023-41109, CVE-2023-41031, CVE-2023-41029, CVE-2023-40934, CVE-2023-40933, CVE-2023-40931, CVE-2023-4092, CVE-2023-40796, CVE-2023-40582, CVE-2023-4034, CVE-2023-4033, CVE-2023-40293, CVE-2023-40144, CVE-2023-40072, CVE-2023-40069, CVE-2023-39944, CVE-2023-39834, CVE-2023-39809, CVE-2023-39780, CVE-2023-3975, CVE-2023-3974, CVE-2023-39638, CVE-2023-39637, CVE-2023-39523, CVE-2023-39455, CVE-2023-39378, CVE-2023-39362, CVE-2023-39293, CVE-2023-39288, CVE-2023-39287
RFID attacks	CVE-2013-0656, CVE-2019-9861, CVE-2019-13531,CVE-2018-5303, CVE-2019-13535, CVE-2019-6568, CVE-2018-4833, CVE-2018-5304, CVE-2019-11523
Selective forwarding	CVE-2015-8087
Sinkhole attack	CVE-2020-1660
Blackhole attack	CVE-2012-4681, CVE-2012-1723
Routing table poisoning	CVE-2017-8147, CVE-2017-6770
Sybil attack	CVE-2020-12821, CVE-2018-7170
Man-in-the-middle attack	CVE-2017-1000209, CVE-2023-4885, CVE-2023-4801, CVE-2023-4586, CVE-2023-4420, CVE-2023-40251,CVE-2023-39982, CVE-2023-39953, CVE-2023-38686, CVE-2023-38356, CVE-2023-38355, CVE-2023-38354, CVE-2023-38353, CVE-2023-38352, CVE-2023-38351, CVE-2023-37948,CVE-2023-36749, CVE-2023-36748, CVE-2023-35947, CVE-2023-34143, CVE-2023-33983,CVE-2023-33849, CVE-2023-33620, CVE-2023-3347, CVE-2023-32994, CVE-2023-32993,CVE-2023-32955, CVE-2023-32548, CVE-2023-32464, CVE-2023-31410, CVE-2023-31195, CVE-2023-31190, CVE-2023-31151, CVE-2023-31136, CVE-2023-30399, CVE-2023-29501, CVE-2023-29175, CVE-2023-29054, CVE-2023-28348, CVE-2023-2820, CVE-2023-27861, CVE-2023-26979, CVE-2023-26467, CVE-2023-26084, CVE-2023-25758, CVE-2023-2538, CVE-2023-23690, CVE-2023-23588, CVE-2023-23546, CVE-2023-23120, CVE-2023-23119, CVE-2023-2310, CVE-2023-22870, CVE-2023-22863, CVE-2023-22812,CVE-2023-22742, CVE-2023-22642, CVE-2023-22367, CVE-2023-22334, CVE-2023-20081

- *Security Impact*: The evaluation of vulnerabilities in security impact is conducted by assessing the extent to which they jeopardize fundamental security objectives, including confidentiality, integrity, and availability. The vulnerabilities inside the IoT that enable unauthorized access to IoT resources and data are directly associated with the breach of confidentiality. Integrity concerns encompass vulnerabilities that allow undetected unauthorized alterations of IoT data and settings. The hindrances to continuous IoT access mainly stem from availability-associated vulnerabilities. It is evident that considering the interdependencies among the many security needs, each found vulnerability in the IoT has the potential to impact several security objectives [47].
- *Attacks*: Attacks refer to identifying and classifying security vulnerabilities in the IoT context based on the methods to exploit these vulnerabilities. The category is structured into three distinct subcategories, each providing detailed explanations and analyses of attacks targeting the principles of confidentiality and authentication, data integrity, and availability [47].
- *Remediation Methods*: Remediation methods/countermeasures refer to categorizing the existing remedial strategies to mitigate vulnerabilities in the IoT context. The category comprises three main components: access and authentication controls, software assurance, and security protocols. Access and authentication controls encompass many security measures, such as firewalls, algorithms, authentication methods, biometric-based models, and context-aware permissions. Moreover, software assurance provides a more detailed explanation of the existing capabilities to enforce integrity restrictions, whereas security protocols focus on implementing lightweight security strategies for effective repair [47].
- *Situation Awareness Capabilities*: The categorization of approaches for gathering accurate and adequate information on created hostile behaviours in the IoT context is known as situation awareness capabilities. This category provides an in-depth examination of vulnerability assessment, honeypots, network discovery, and intrusion detection. The field of vulnerability assessment encompasses several methodologies and approaches utilized by the research and cyber security communities to evaluate the vulnerabilities present in IoT devices, including those previously unknown (known as 0-day vulnerabilities). Potential methodologies that might be employed encompass testbeds, attack simulation methods, and fuzzing techniques [47].
  Moreover, honeypots offer functionalities to capture IoT-specific harmful actions for subsequent examination. At the same time, network discovery focuses on techniques for identifying susceptible and compromised IoT devices on a large scale throughout the Internet. In conclusion, intrusion detection encompasses examining many methodologies that may be utilized to identify and analyze hostile behaviors specifically related to the IoT [47].

Additionally, Table 4 and Table 5 highlight well-known databases that document vulnerabilities in IoT systems.

Table 8
IoT attacks to vulnerability mapping [74–78].

IoT attack	Vulnerabilities
Replay attack	CVE-2023-4299, CVE-2023-39373, CVE-2023-34625, CVE-2023-34553, CVE-2023-33621, CVE-2023-33281, CVE-2023-31763, CVE-2023-31762, CVE-2023-31761, CVE-2023-31759, CVE-2023-31200, CVE-2023-29158, CVE-2023-2846, CVE-2023-26442, CVE-2023-20123, CVE-2023-1886, CVE-2023-1537, CVE-2023-0014, CVE-2022-47930, CVE-2022-45914, CVE-2022-45789, CVE-2022-44457, CVE-2022-44419, CVE-2022-43704, CVE-2022-42731, CVE-2022-41541, CVE-2022-41209, CVE-2022-39064, CVE-2022-37011, CVE-2022-35961, CVE-2022-33971, CVE-2022-33208, CVE-2022-31277, CVE-2022-31265, CVE-2022-30466, CVE-2022-30111, CVE-2022-29593, CVE-2022-29334, CVE-2022-27254, CVE-2022-25180, CVE-2022-25159, CVE-2022-22936, CVE-2022-22806, CVE-2022-2226, CVE-2022-1973, CVE-2022-1521, CVE-2021-46145, CVE-2021-41138, CVE-2021-41030, CVE-2021-41025, CVE-2021-40170, CVE-2021-39363, CVE-2021-39124, CVE-2021-37604, CVE-2021-37586, CVE-2021-36983, CVE-2021-35067, CVE-2021-34739, CVE-2021-34572, CVE-2021-27662, CVE-2021-27572, CVE-2021-27195, CVE-2021-26936, CVE-2021-26824, CVE-2021-25835, CVE-2021-25834, CVE-2021-25480, CVE-2021-25471,CVE-2021-22267, CVE-2020-9058, CVE-2020-9057, CVE-2020-6972, CVE-2020-5300, CVE-2020-5261
Denial/distributed denial of service	CVE-2023-28840, CVE-2023-27077, CVE-2023-22397, CVE-2023-1916, CVE-2022-31006, CVE-2022-30317, CVE-2022-24797, CVE-2021-34697, CVE-2021-3172, CVE-2021-26264,CVE-2021-0280, CVE-2021-0234, CVE-2021-0228, CVE-2021-0214, CVE-2020-1665, CVE-2020-16599, CVE-2020-16593, CVE-2020-14312, CVE-2019-9750, CVE-2019-17450, CVE-2018-8945, CVE-2018-8009, CVE-2018-7642, CVE-2018-7570, CVE-2018-7569, CVE-2018-7568, CVE-2018-7208, CVE-2018-6872,CVE-2018-6759, CVE-2018-6323, CVE-2018-20712, CVE-2018-20657, CVE-2018-20651, CVE-2018-20002, CVE-2018-18701, CVE-2018-18700,CVE-2018-18607, CVE-2018-18606, CVE-2018-18605, CVE-2018-18483, CVE-2018-18309, CVE-2018-17360, CVE-2018-17359, CVE-2018-17358, CVE-2018-13033, CVE-2018-10535, CVE-2018-10373, CVE-2018-0369, CVE-2018-0239, CVE-2018-0117, CVE-2018-0007, CVE-2017-9955, CVE-2017-9954, CVE-2017-9754, CVE-2017-9753, CVE-2017-9752, CVE-2017-9748, CVE-2017-9747, CVE-2017-9745, CVE-2017-9744, CVE-2017-8806, CVE-2017-7614, CVE-2017-6678, CVE-2017-5949, CVE-2017-17821, CVE-2017-17124, CVE-2017-17123, CVE-2017-17121, CVE-2017-17080, CVE-2017-16832, CVE-2017-16831, CVE-2017-16827,CVE-2017-16826, CVE-2017-15939,CVE-2017-15938, CVE-2017-15225, CVE-2017-15025, CVE-2017-15024, CVE-2017-15023,CVE-2017-15022
Hello-flooding	CVE-2023-44466, CVE-2023-3748, CVE-2023-36471, CVE-2023-32018, CVE-2023-31128,CVE-2023-30628, CVE-2023-27479, CVE-2022-39173, CVE-2022-35797, CVE-2022-34836, CVE-2022-31019, CVE-2019-15522, CVE-2018-5400, CVE-2017-5872, CVE-2015-3197, CVE-2015-2281
Clone attack	CVE-2021-21300, CVE-2023-27532
Routing diversion/misdirection attacks	CVE-2021-26928
Routing loop attacks	CVE-2002-2053
RPL exploit	CVE-2023-21674, CVE-2023-2156, CVE-2022-35927, CVE-2021-32771, CVE-2021-28362, CVE-2021-27698, CVE-2021-27697, CVE-2021-27357, CVE-2021-21282, CVE-2021-21257, CVE-2020-13986,CVE-2020-13985, CVE-2015-7715, CVE-2015-7714, CVE-2014-3405, CVE-2007-6450, CVE-2007-4447, CVE-2003-1450
DNS tunnelling attack	CVE-2021-25681, CVE-2006-4983
DNS amplification attack	CVE-2023-26249, CVE-2021-23937, CVE-2020-10995, CVE-2020-12667, CVE-2019-6015, CVE-2018-12571, CVE-2017-6520, CVE-2015-7794, CVE-2015-2809, CVE-2015-1892, CVE-2013-4769, CVE-2013-0198, CVE-2012-3411, CVE-2006-0988
DNS flood attack	CVE-2022-2795, CVE-2015-6287

6. Motivation for IoT attacks, IoT security and IoT privacy

6.1. Motivations for IoT attacks

IoT devices handle vast amounts of data that may be utilized for many purposes, making them an attractive target for a wide range of attackers, including occasional hackers, hacktivists, and cybercriminals. By infiltrating IoT devices through hacking, potential attackers may be incentivized to steal valuable data, such as location information, credit card details, and bank account passwords. Moreover, they could attempt to exploit IoT components, such as edge nodes, to attack a third-party company [48].

Furthermore, the rapid advancement of machines and technology has resulted in several dangers and privacy concerns. Smart devices engage in communication and data exchange inside a network. The entire infrastructure is vulnerable if any device becomes compromised. Hence, the significance of security and privacy has dramatically increased in recent years. It is essential to create security criteria to safeguard against potential threats. For example, a compromised system might jeopardize production and compromise critical data [48].

6.2. Motivations for IoT privacy

With the notable rise in the use and effectiveness of electronic data processing, information privacy has become a crucial concern in modern times. The concept of privacy within the realm of the IoT may be categorized into three distinct classifications [48]:

- Recognizing the potential privacy risks associated with smart devices and services connected to an individual's data.
- Exercising personal control over the gathering and handling personal information by these connected smart devices.
- Being aware of and controlling how personal information is further used and shared by these entities with any external entity beyond the individual's control.

Table 9
IoT attacks to vulnerability mapping [74–78].

IoT attack	Vulnerabilities
DNS cache poisoning	CVE-2023-41045, CVE-2022-33989, CVE-2022-34294, CVE-2022-32983, CVE-2022-30295, CVE-2021-43105, CVE-2020-25926, VE-2008-3280, CVE-2021-3448, CVE-2020-25685, CVE-2020-25684, CVE-2020-17470, CVE-2020-17439, CVE-2013-5661, CVE-2019-3978, CVE-2017-1773, CVE-2017-10874, CVE-2016-10152, CVE-2016-3725, CVE-2010-0290
NXDOMAIN attack	CVE-2020-13960, CVE-2020-12244, CVE-2019-10190, CVE-2016-9778, CVE-2016-1284, CVE-2010-0097
Domain hijacking	CVE-2023-28109, CVE-2021-3672, CVE-2021-43523, CVE-2021-22931, CVE-2016-6851,CVE-2010-1911, CVE-2007-4431
Distributed reflection denial of service	CVE-2013-5211, CVE-2020-2100
Random subdomain attack	CVE-2020-10995, CVE-2020-12667
Network eavesdropping	CVE-2023-33982, CVE-2020-9526, CVE-2020-9525, CVE-2019-10926, CVE-2018-1340, CVE-2013-0570
0-Day	CVE-2023-22515, CVE-2023-42824 CVE-2023-33063, CVE-2023-33107,CVE-2023-33106, CVE-2023-5217, CVE-2023-20109, CVE-2023-41992, CVE-2023-41179, CVE-2023-4211, CVE-2023-36761,CVE-2023-26369, CVE-2023-4863,CVE-2023-41061,CVE-2023-41064, CVE-2023-20269,CVE-2023-35674, CVE-2023-38035,CVE-2023-38180, CVE-2023-38831, CVE-2023-3508, CVE-2023-35078, CVE-2023-41990, CVE-2023-38606 CVE-2023-38205, CVE-2023-3519,CVE-2023-37580, CVE-2023-3595, CVE-2023-36884, CVE-2023-35311, CVE-2023-36874,CVE-2023-32049,CVE-2023-32046, CVE-2023-37450, CVE-2023-3460, CVE-2023-32435, CVE-2023-32439, CVE-2023-32434, CVE-2023-21237, CVE-2023-20867, CVE-2023-3079, CVE-2023-34362, CVE-2023-2868, CVE-2023-32373, CVE-2023-28204, CVE-2023-24932
Worm	CVE-2018-1000087, CVE-2010-2743, CVE-2010-3889, CVE-2010-3888, CVE-2010-2772, CVE-2008-0708, CVE-2007-0059, CVE-2005-4066, CVE-2005-2852, CVE-2005-1983, CVE-2004-1762, CVE-2004-1909, CVE-2004-2348, CVE-2004-2539, CVE-2004-1315,CVE-2003-0533
Trojan	CVE-2023-31468, CVE-2023-26918, CVE-2022-48422, CVE-2023-22368, CVE-2021-36631, CVE-2022-41796, CVE-2022-39959, CVE-2022-36403, CVE-2021-43430, CVE-2022-28128, CVE-2022-25348, CVE-2021-44226, CVE-2022-26526, CVE-2021-44049,CVE-2021-40981, CVE-2021-20793, CVE-2021-36770, CVE-2021-33393, CVE-2021-20726, CVE-2021-20722
Backdoor	CVE-2022-47558, CVE-2022-47555, CVE-2023-23771, CVE-2023-23770, CVE-2023-26243, CVE-2023-27748, CVE-2023-24108, CVE-2023-24107, CVE-2022-32747, CVE-2022-47767, CVE-2022-47209, CVE-2022-46997, CVE-2022-46996, CVE-2022-46609, CVE-2022-44039, CVE-2022-45562, CVE-2020-23591, CVE-2022-4093, CVE-2022-3703, CVE-2022-44054
Spyware	CVE-2021-33971, CVE-2021-33974, CVE-2022-36336
Ransomware	CVE-2023-30024, CVE-2022-23714, CVE-2021-42258, CVE-2020-9452,CVE-2018-19589, CVE-2017-18362, CVE-2018-6318
Directory traversal	CVE-2023-5399, CVE-2023-26152, CVE-2023-42819, CVE-2023-42657, CVE-2023-42487,CVE-2022-4244, CVE-2023-43382, CVE-2023-38346, CVE-2023-42280, CVE-2023-42456, CVE-2023-40930,CVE-2022-45447, CVE-2023-41599, CVE-2022-28357, CVE-2023-39916, CVE-2023-40924,CVE-2023-39912
HTML5 injection	CVE-2023-33439, CVE-2023-25833, CVE-2020-3956

The perception and expectations of privacy differ among individuals, resulting in an ambiguous understanding of personal information. Hence, throughout the design process of new systems and services, it is imperative to conduct a meticulous evaluation of the sensitivity of the information at hand and the corresponding user demands [48].

6.3. Motivations for IoT security

The primary goal of IoT security is to safeguard privacy, maintain confidentiality, protect the security of users, infrastructure, data, and devices inside the IoT, and assure the availability of services provided by an IoT ecosystem. Implementing security measures in an IoT system has more significant difficulties than in a conventional network. Given the wide range of devices and several communication protocols in IoT systems, together with their varied interfaces and services, it is not appropriate to rely on standard IT network solutions for implementing security measures. Indeed, the existing security measures used in a traditional network may be inadequate owing to the diverse range of devices and protocols and the extensive scale or number of nodes inside the system [49].

Although it is observed that authentication is the prevailing method for ensuring security, trust management is increasingly popular owing to its effectiveness in preventing or identifying malicious nodes. Conversely, current research in encryption is centred on developing lightweight and low-cost encryption methods suitable for low-power and constrained devices [49].

7. IoT attack taxonomy

The increasing popularity of the IoT is seeing significant growth, facilitating the interconnection of a progressively larger array of devices and systems. The presence of interconnectivity yields many advantages, although it concurrently presents susceptibilities and complexities regarding security. To adequately confront these problems, it is imperative to classify and comprehend the diverse array of attacks that have the potential to target IoT ecosystems. Table 3 provides a comprehensive view of comparison between various IoT attack surveys.

The IoT attacks taxonomy referred to in Figs. 3 and 4 is a systematic framework that organizes and classifies various attacks against IoT devices, networks, and infrastructure. The presented taxonomy offers a methodical approach to examining and addressing security concerns inside the IoT domain.

Table 10
IoT attacks to vulnerability mapping [74–78].

IoT attack	Vulnerabilities
XML injection	CVE-2023-4037, CVE-2022-4245, CVE-2023-38343, CVE-2023-39643, CVE-2021-36023, CVE-2023-35892, CVE-2022-46751, CVE-2023-0871, CVE-2023-38207, CVE-2020-26710, CVE-2020-26709, CVE-2020-26708, CVE-2023-3113, CVE-2023-29289, CVE-2023-24470, CVE-2015-20108, CVE-2023-27554, CVE-2023-31126, CVE-2023-28009, CVE-2023-28008
In-band SQL injection	CVE-2019-18662
Out-of-band interaction	CVE-2021-43969
Cross-site request forgery	CVE-2015-10125, CVE-2023-40559, CVE-2023-40561, CVE-2023-27433, CVE-2023-25,025, CVE-2023-37995, CVE-2023-25980, CVE-2023-25788, CVE-2023-25489, CVE-2023-41693, CVE-2023-41244, CVE-2023-40558, CVE-2023-39158, CVE-2023-32091, CVE-2023-27435, CVE-2023-40212, CVE-2023-40202, CVE-2023-40201, CVE-2023-40199, CVE-2023-40198
Format string attacks	CVE-2022-26393, CVE-2022-26392, CVE-2022-31129, CVE-2016-9586, CVE-2017-11424, CVE-2016-6537, CVE-2014-9277, CVE-2012-0864, CVE-2007-2027, CVE-2007-0347, CVE-2001-0570
Object injection attacks	CVE-2023-3343, CVE-2023-35810, CVE-2020-36726, CVE-2020-36718, CVE-2023-2500,CVE-2023-1549, CVE-2023-1650, CVE-2023-1347, CVE-2023-1196, CVE-2023-1669, CVE-2023-28667, CVE-2022-4265, CVE-2023-0232, CVE-2023-0669, CVE-2022-4489, CVE-2022-4680
Firmware code injection attacks	CVE-2023-33239, CVE-2023-33238, VE-2023-31986, CVE-2023-31983, CVE-2023-31985, CVE-2023-2131, CVE-2023-1097, CVE-2023-0776, CVE-2022-45768,CVE-2023-24508, CVE-2022-29843, CVE-2021-26731, CVE-2021-26729, CVE-2021-26728, CVE-2021-26727, CVE-2022-40785, CVE-2021-45876, CVE-2020-25197, CVE-2020-29664, CVE-2020-35576
Log poisoning attacks	CVE-2023-41045, CVE-2023-3997, CVE-2023-26125, CVE-2021-40323, CVE-2021-24453,CVE-2019-11642
Hibernate query language injection attacks	CVE-2023-26093, CVE-2016-1595
Indirect prompt injection attacks	CVE-2023-29374, CVE-2023-32786, CVE-2023-32785
Cross-site scripting attacks	CVE-2023-44075, CVE-2023-5113, CVE-2023-44012, CVE-2023-5305, CVE-2023-5304, CVE-2023-5303, CVE-2023-5302, CVE-2023-5287, CVE-2023-5286, CVE-2023-26218, CVE-2023-5273, CVE-2023-43944, CVE-2023-43657, CVE-2023-43874, CVE-2023-43873, CVE-2023-43872, CVE-2023-43871, CVE-2023-41447, CVE-2023-41446, CVE-2023-41453
LDAP injection	CVE-2023-41580, CVE-2023-33201, CVE-2023-3447, CVE-2023-27866, CVE-2022-45801, CVE-2023-28853, CVE-2022-46303, CVE-2023-25613, CVE-2023-0476, CVE-2023-23749, CVE-2015-10027, CVE-2022-40145, CVE-2022-45910, CVE-2022-22360,CVE-2022-22975, CVE-2022-29155, CVE-2021-39031, CVE-2021-41232, CVE-2021-37933, CVE-2020-23148
Remote code execution	CVE-2022-47893, CVE-2023-3656, CVE-2023-43835, CVE-2023-5201, CVE-2023-43655, CVE-2023-44466, CVE-2023-5185, CVE-2023-43740, CVE-2023-38874, CVE-2023-43651, CVE-2023-43234, CVE-2023-43187, CVE-2023-40400, CVE-2021-38243,CVE-2023-40581, CVE-2023-4300, CVE-2023-43764, CVE-2023-43762
Buffer overflow	CVE-2023-41175, CVE-2023-40745, CVE-2023-44839, CVE-2023-44838, CVE-2023-44837, CVE-2023-44836, CVE-2023-44835, CVE-2023-44834, CVE-2023-44833, CVE-2023-44832, CVE-2023-44831, CVE-2023-44830, CVE-2023-44829, CVE-2023-44828, CVE-2023-35803, CVE-2023-3428, CVE-2023-4494, CVE-2023-4491, CVE-2023-30733, CVE-2023-40830
Packet injection	CVE-2023-4513, CVE-2023-4512, CVE-2023-4511, CVE-2023-5371, CVE-2023-3649, CVE-2023-3648, CVE-2023-2952, CVE-2023-2879
Session hijacking	CVE-2023-42446, CVE-2023-40732, CVE-2023-40024, CVE-2023-26450, CVE-2023-26449, CVE-2023-26448, CVE-2023-26447, CVE-2023-26446, CVE-2023-26445,CVE-2023-28809, CVE-2023-33962, CVE-2023-29196, CVE-2023-20866, CVE-2023-26260, CVE-2023-23326, CVE-2023-24975, CVE-2022-36775, CVE-2022-34362, CVE-2022-45789, CVE-2021-38997

7.1. IoT domains

IoT attacks can be divided into six sections based on domain: physical attacks, network attacks, software attacks, encryption attacks, data attacks, and side channel attacks.

7.1.1. Physical attacks

Physical attacks can be launched if the attacker remains physically close to the network or devices of the system [2]. New IoT vulnerabilities are frequently found through physical attacks. The attacker will attempt to physically access the device before launching the attack by purchasing a duplicate of the targeted IoT device from the market. They would then develop a false attack “test” using reverse engineering to determine what kind of outputs might be acquired from it. These physical attacks expose the system's vulnerabilities [50].

- *Tampering*: It describes the process of physically altering a device (such as an RFID) or communication link [2].
- *Fault Injection*: The act of forcing a running device to behave differently to unearth new security details or methods in the system is known as fault injection. It is a technique for evaluating IoT systems and devices to determine their dependability and resilience. It entails purposefully introducing defects or flaws into a system to watch how it responds and find possible weaknesses. Researchers and security experts frequently employ this method to comprehend how a product or system performs under various circumstances and spot vulnerabilities an attacker may exploit [51]. Some of the examples are:
  - *Hardware Fault Injection*: It involves inducing faults directly into the physical components of the IoT device, such as memory corruption or voltage manipulation [51].

Table 11
IoT attacks to vulnerability mapping [74–78].

IoT attack	Vulnerabilities
Credential stuffing	CVE-2016-6650
SSL stripping	CVE-2022-24044
Phishing & vishing	CVE-2022-4145, CVE-2023-41888, CVE-2023-3612, CVE-2023-39364, CVE-2023-38574, CVE-2023-24514, CVE-2023-30952, CVE-2023-30949, CVE-2023-30433, CVE-2023-37561, CVE-2023-37947, CVE-2023-35120, CVE-2023-36462, CVE-2023-36471, CVE-2019-25150, CVE-2023-2183, CVE-2023-28705, CVE-2023-32689, CVE-2023-28370, CVE-2023-29031
Token replay	CVE-2020-5261
Password spraying	CVE-2022-45860, CVE-2022-24044
Reverse engineering attacks	CVE-2017-18642
Logic bombs	CVE-2023-37481, CVE-2022-36114
Data tampering	CVE-2023-25534, CVE-2023-25529, CVE-2023-25528, CVE-2023-25527, CVE-2023-41387, CVE-2023-25522, CVE-2023-25521,CVE-2023-25517, CVE-2023-25515, CVE-2023-34343, CVE-2023-34342, CVE-2023-34334, CVE-2023-34341, CVE-2023-25508, CVE-2023-25507, CVE-2023-0209, CVE-2023-0199,CVE-2023-0184
Device impersonation	CVE-2023-22814, CVE-2022-36331, CVE-2022-39254, CVE-2022-39252, CVE-2020-36128, CVE-2020-16226, CVE-2020-9435, CVE-2019-9742, CVE-2019-7651, CVE-2015-1644
Memory corruption attacks	CVE-2023-30738, CVE-2023-33039, CVE-2023-33035, CVE-2023-33034, CVE-2023-33029, CVE-2023-33028, CVE-2023-28539, CVE-2023-24855, CVE-2023-24853, CVE-2023-24850, CVE-2023-24844, CVE-2023-22385, CVE-2023-22384, CVE-2023-21673, CVE-2023-32823, CVE-2023-5176, CVE-2023-40163, CVE-2023-32614, CVE-2023-32284, CVE-2023-28393
Integer overflow	CVE-2023-41175, CVE-2023-40745, CVE-2023-32829,CVE-2023-32828, CVE-2023-44466, CVE-2023-5173, CVE-2023-40218,CVE-2023-28831, CVE-2023-35684, CVE-2023-35681, CVE-2023-35673, CVE-2023-4576, CVE-2023-40353, CVE-2023-21644, CVE-2023-4734, CVE-2023-4722, CVE-2023-36328, CVE-2023-36327, CVE-2023-36326, CVE-2023-40022
Stack overflow	CVE-2023-4494, CVE-2023-30733, CVE-2023-44023, CVE-2023-44022, CVE-2023-44021, CVE-2023-44020, CVE-2023-44019, CVE-2023-44018, CVE-2023-44017, CVE-2023-44016, CVE-2023-44015, CVE-2023-44014,CVE-2023-44013
Heap overflow	CVE-2023-41175,CVE-2023-40745, CVE-2023-3428,CVE-2023-5344,CVE-2023-5217,CVE-2023-35002,CVE-2023-32614,CVE-2023-23567,CVE-2023-4504,CVE-2023-32643,CVE-2023-3935,CVE-2023-4863,CVE-2023-38076,CVE-2023-38071,CVE-2020-19323,CVE-2023-4576,CVE-2023-4781,CVE-2023-4751,CVE-2023-4738,CVE-2023-4682
Privilege escalation	CVE-2023-26236, CVE-2023-44209,CVE-2023-5402, CVE-2023-4997, CVE-2023-0506,CVE-2023-44218,CVE-2023-44217,CVE-2023-5345,CVE-2023-36628,CVE-2023-32830,CVE-2023-32829,CVE-2023-32828,CVE-2023-32827,CVE-2023-32826,CVE-2023-32824,CVE-2023-32823,CVE-2023-32822,CVE-2023-32821,CVE-2023-20819,CVE-2023-5256
Kernel exploit	CVE-2023-32233, CVE-2022-47939, CVE-2022-0185,CVE-2021-34866, CVE-2017-11176, CVE-2023-3269, CVE-2017-7308, CVE-2016-0728, CVE-2021-33909

  - *Software Fault Injection*: It involves injecting faults into the software of the IoT device, such as modifying data values, altering control flow, or injecting exceptions [51].
  - *Time-Based Fault Injection*: It involves manipulating the timing of events to create faults, such as introducing delays or modifying clock frequencies [52].
- *RF Spoofing/Jamming*: An attacker launches DoS attacks against RFID tags by creating and transmitting noise signals instead of radio frequency (RF) signals to obstruct connectivity [53].
- *Fake Node Injection*: A fake node is dropped by an attacker between two actual network nodes to manipulate data flow between them.
- *Social Engineering*: Social engineering obtains sensitive data by deceiving an IoT system's users.
- *Sleep Denial Attack*: By providing the battery-powered devices with incorrect inputs, the attacker keeps them awake. This wears out their batteries, resulting in a shutdown. The attack's goal is to keep the target devices busy by overtaxing the network with routing traffic due to fictitious requests for actual or imagined destination devices, which prevents them from responding to legitimate requests. This attack prevents the targeted device from entering sleep mode to extend battery life [54].
- *Side Channel Attacks*: As mentioned in the Section 7.1.6.
- *Permanent Denial of Service (PDoS)*: A PDoS attack, also known as a permanent denial-of-service attack, is similar to a DoS attack in that hardware is destroyed or sabotaged instead of resources. Rebooting the device won't help recover the IoT device [17]. Permanent DoS attacks were concisely described in work [55], which identifies the different IoT attacks in IoT wireless protocols such as BLE, LoRaWAN, Z-Wave, and ZigBee along with their vulnerabilities.
- *Malicious Code Injection*: It is an attack strategy that adversaries use to undermine the security of IoT devices. It involves injecting malicious code or commands into a system to gain unauthorized access, steal data, disrupt operations, or achieve other malicious objectives. Code injection attacks are typically executed through vulnerabilities in the software or firmware of the IoT devices [56]. Common types of malicious code injection attacks include:
  - *SQL Injection*: It exploits vulnerabilities in web applications to insert malicious SQL statements into the database, potentially leading to data theft or unauthorized access [57].
  - *Command Injection*: It involves injecting malicious commands into the system's command-line interface, allowing attackers to execute arbitrary code [57].

7.1.2. Network attacks

IoT devices use the network layer to send data to a server or other devices for processing after receiving it from the physical layer. To damage IoT network systems, network attacks are carried out by manipulating them. Without being near the network, it may be deployed with ease. Network attacks are a subset of cyberattacks that target the communication infrastructure of IoT devices. The attacks concentrate on taking advantage of flaws in communication channels, network protocols, and device connections to threaten the

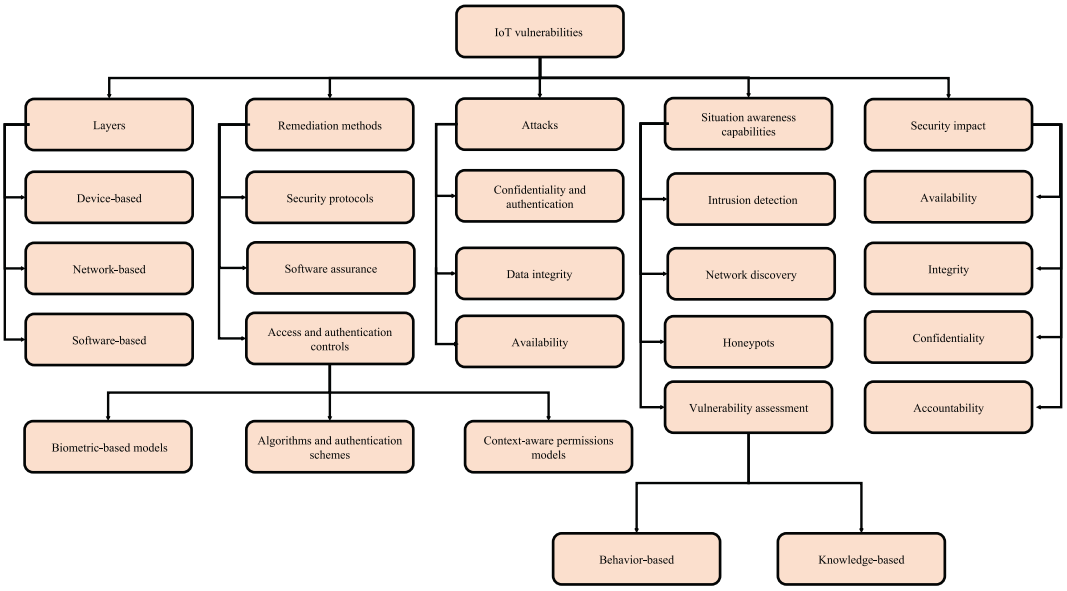

Fig. 2. IoT vulnerability taxonomy [47].

security, availability, and integrity of the IoT ecosystem. IoT network attacks may result in catastrophic outcomes, including unauthorized access, data breaches, service interruptions, and, in certain circumstances, even physical harm [2,28].

Some of the network attacks are:

- *Traffic Analysis Attack*: It is a type of cyberattack that focuses on analyzing the patterns, volume, and behaviour of data traffic between IoT devices and the network infrastructure. Such attacks aim to extract sensitive information or infer valuable insights from the communication patterns without directly intercepting the content of the data packets. By understanding the data flow, an attacker can deduce sensitive information about the IoT devices, users, or the network [28].
- *RFID Spoofing*: RFID spoofing, also known as RFID cloning or RFID emulation, is a technique used to deceive radio frequency identification (RFID) systems by mimicking the signals of legitimate RFID tags or devices. In an RFID spoofing attack, an adversary replicates or manipulates RFID signals to impersonate a valid RFID tag or device. This allows the attacker to gain unauthorized access, bypass security measures, or perform fraudulent activities. RFID spoofing aims to trick the RFID reader into recognizing the attacker's spoofed signal as a genuine RFID tag, enabling them to exploit the system for malicious purposes [58].
- *RFID Unauthorized Access*: RFID unauthorized access refers to gaining entry to a system, facility, or sensitive information by exploiting vulnerabilities in radio frequency identification (RFID) technology. Unauthorized access occurs when an individual or attacker, without proper authorization or credentials, finds a way to exploit weaknesses in the RFID system to gain entry or access sensitive information. This can have profound security implications, allowing unauthorized parties to bypass physical barriers and security checkpoints or gain control over valuable assets [56].
- *Routing Attacks/Routing Information Attacks*: IoT routing attacks are a subset of attacks that target the routing architecture and protocols of IoT networks. These attacks are designed to prevent data packets from being appropriately routed between IoT devices and other network elements, which might result in data interception, manipulation, or denial of service. Attackers who get access to the routing systems can reroute, delay, or discard data packets, seriously impairing the IoT system's ability to communicate and function [59].
  - *Selective Forwarding*: In a selective forwarding attack, the attacker selects some data packets to reject or forward while letting others pass. By controlling the flow of data, the attacker can selectively block critical information or let malicious data through [59].
  - *Hole Attacks*: “Hole-196” attacks are a specific kind of cyberattack called “hole attacks.” The vulnerability in the RPL (IPv6 Routing Protocol for Low-Power and Lossy Networks) routing protocol, which is used in many IoT networks, gave rise to the name “Hole-196.”

    The RPL protocol, intended to facilitate effective routing on low-power and lossy networks, such as those frequently present in IoT environments, has a weakness exploited by “Hole-196” attacks. The attack uses the protocol's capacity to form a network “black hole”, seriously disrupting communication [60].
    * *Sinkhole Attacks*: In a sinkhole attack, the attacker advertises that it has the fastest or shortest route to the target to draw data flow. IoT devices route their data through the attacker's node, unknowing its malicious purpose, allowing it to intercept, modify, or discard the data [56].
    * *Blackhole Attacks*: Like a sinkhole attack, it involves the attacker dropping all incoming data packets rather than forging them. The connectivity between IoT devices and the legitimate network infrastructure is interfered with as a result [56].

Fig. 3. IoT attack taxonomy part 1.

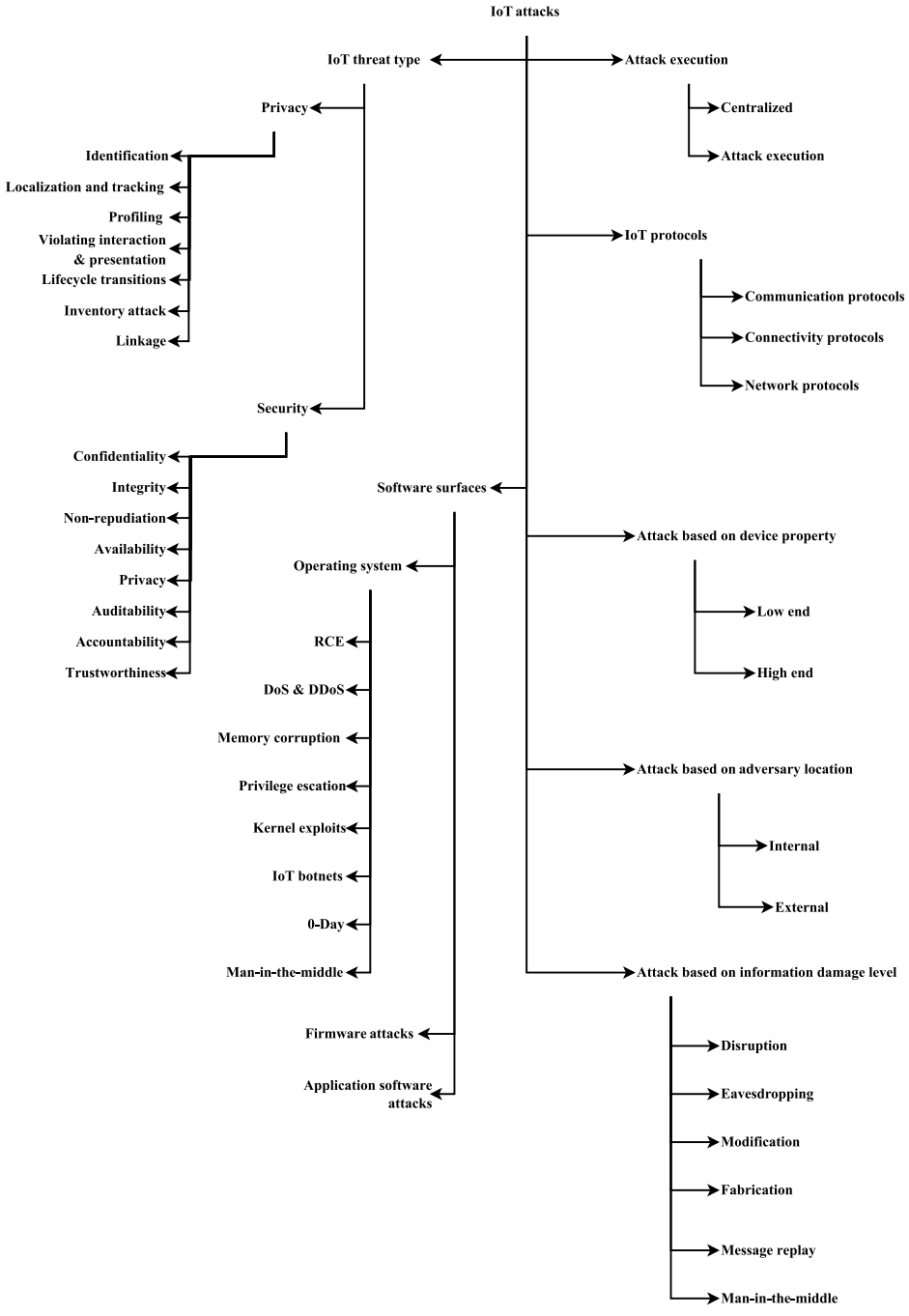

Fig. 4. IoT attack taxonomy part 2.

- *Routing Table Poisoning (RTP)*: One of the most successful attacks involves installing a fake router or sending malicious routing table updates to change the routing table. The attacker may launch a MiTM attack or just divert the traffic to get through FW or IPS in this situation [61].
- *Sybil Attack*: In a peer-to-peer network, a Sybil attack occurs when an adversary generates fake or stolen identities to act as numerous nodes. An adversary can get excessive control to lower the network's efficiency. Selfish behaviour inside a network can impact data integrity, resource usage, and overall network performance [61].

• *Man-in-the-Middle Attack (MiTM)*: A man-in-the-middle attack involves the attacker discreetly putting themselves in the user's and the IoT's path. Whenever someone requests information from the Internet, it passes through several routers before getting to the

right place. If the data is not encrypted, the attackers can read anything that travels through them by putting themselves in the midst. The information can be altered if the data is encrypted and the attackers get hold of the encryption key. Another variation of this attack involves the attacker not modifying any data but instead storing the information they learn about their target. Since there is no method for the user to check the accuracy of the information, these attacks are challenging to spot [29].

- *Replay Attack*: A replay attack involves the attacker capturing and subsequently retransmitting genuine data packets sent back and forth between IoT devices. The attacker can use this method to obtain unauthorized access or carry out malicious activity if the data is not sufficiently safeguarded. Attackers utilize replay attacks when they want to delay information delivery to machines. In this attack, a packet carrying a machine's future instructions is intercepted and transmitted laterally. When an attacker intercepts a packet, they also have the power to alter the instructions and trick the system into carrying out the wrong operations [29].
- *Denial/Distributed Denial of Service (DoS/DDoS) Attack*: Multiple devices are utilized to launch a DDoS attack, a kind of DoS attack. These types of attacks are carried out by bombarding the desired target with an excessive number of access requests until they are overwhelmed. A denial-of-service (DoS) attack involves a malicious attacker trying to saturate the system with rogue and amplified traffic to consume network resources and target legitimate users' CPU time and/or bandwidth. Botnets are networks of Internet-connected devices that are contaminated or under control and used in effective DDoS attacks [62].
- *Hello-flooding*: To introduce themselves to their neighbours, nodes are required by some WSN routing protocols to broadcast hello messages. When a node receives one of these messages, it may believe that the sender is within radio range of it. This assumption, however, may not always be accurate; on occasion, a laptop-class attacker might deceive every other node in the network into believing the attacker is its neighbour by broadcasting routing or additional information with significant transmission power. For example, if an adversary advertises an extremely high-quality route to the base station, many network nodes may seek to use it. But nodes sufficiently remote from the adversary would send the packets into oblivion. The network is consequently left in a state of chaos. This attack mainly affects protocols that rely on localized information sharing between nearby nodes for topology maintenance or flow management. An attacker can use the hello flood attack without creating actual traffic. It can just broadcast overhead packets again with sufficient strength for all other network nodes to receive them [63].
- *Clone (Node-Replication)*: An attack on IoT devices called a clone node explicitly targets the authentication system. Device replication or device cloning attacks are other names for clone node attacks. A cloned node behaves like a genuine node. It engages in malicious network-related activities such as selective forwarding attacks, wormhole attacks, and blackhole attacks since it has access to all secret information (authentication codes and keys) [64].
  In a clone node attack, the attacker can seize physical devices from the IoT network by obtaining their secret credentials, such as IDs and public and private keys. There are several steps to exploiting this vulnerability, which involve capturing the physical device, obtaining the privileged credentials, modifying its function, and placing it back in the network at some desired location. IoT devices frequently lack updated security software and certifications and are developed and manufactured by unreliable security partners, which might contribute to the clone node attack [64].
- *Routing Diversion/Misdirection Attacks*: In a route diversion attack, the enemy succeeds in diverting some current routes but does not stop the formation of new routes. This indicates that the protocol establishes routes different from those it would construct if the adversary did not interfere with the protocol's execution due to the adversary's presence. Route diversion may give attackers more control over communications between select victim nodes. In this scenario, the adversary seeks to ensure that the redirected routes comprise a node under its control or a connection under its observation. The attacker will thus have a simpler time intercepting or changing data exchanged between the victim nodes [65].
  Establishing a tunnel is a particularly effective approach to rerouting routes through nodes controlled by an adversary. Many pairs of communicating nodes might pick the tunnelled paths because they seem shorter, making it more straightforward for the adversary to access their conversations. Increased resource consumption by some nodes may be another goal of route diversion. For instance, the nodes near the two ends of the tunnel will get a higher volume of transit traffic due to the above-described route diversion, necessitating the employment of extra resources at those nodes to forward that traffic. Alternately, the attacker can direct numerous routes straight via a victim node by altering or falsifying routing messages. Last but not least, route diversion may seek to lengthen the already known routes, increasing specific nodes' end-to-end delays and perhaps degrading their quality of service [65].
- *Routing Loop Attacks*: The routing loop attack relies on the fact that a router does not know whether there is an endpoint that can be reached via its tunnel with the packet's source or destination address. Routing Loop attacks exploit inconsistencies between the native IPv6 routing state and a tunnel's overlay IPv6 routing state. They mainly use that each endpoint in an autonomous tunnel is blind to the other nodes currently a part of it. The attacker takes advantage of this by creating a packet sent over a tunnel to a node not part of that tunnel. This node sends the packet over the tunnel onto a real IPv6 network. The entrance point in that network sends the packet back into the tunnel before routing it again. As a result, the packet will enter and exit the tunnel repeatedly. The nodes that forward packets into and out of the tunnel are called the attack's victims. Only when the hop limit field in the packet's IPv6 header is zeroed out can a loop end. 255 is the highest possible value that may be entered into this field [66].
  It should be noted that the hop limit does not reduce when the packet is tunnelled through IPv4 routers. Each step along the attack packet's path will be travelled through $255/N$ times, where N is the number of IPv6 routers that make up the loop. The loops can, therefore, be utilized as $255/N$ traffic amplification methods. The type of attack and the positions of the two victims define how many IPv6 routers are in the loop. The amplification ratio will increase the closer the two victims are to one another [66].
- *Rushing Attacks*: This attack aims to add the malicious device to the routing path. So, by swiftly sending a route discovery packet to the forwarding group, the attacker can abuse the network. The attacker can tamper with the packet by using this technique to include himself in the routing table. A Rushing attack, for instance, can only be successful if the attacker can get its altered packets to other

genuine devices before these devices get the genuine ones. To do this, the attacker can speed up packet transmission by reducing delays at the MAC or routing layer [54].

- *RPL Exploit*: It is a simple exploit used in IoT today and does not include all of the capabilities of the routing protocols in use. It was primarily designed to be used with data sinks that have multiple-point communications [67].
- *DNS Attacks*: Attacks on the DNS (Domain Name System) in the IoT involve taking advantage of vulnerabilities in the DNS infrastructure or protocols to sabotage connectivity between IoT devices or reroute traffic to malicious destinations. A crucial part of the internet is the DNS, which converts human-readable domain names (such as www.example.com) into IP addresses that computers and IoT devices can comprehend. DNS plays a crucial role in facilitating communication between devices and servers on the Internet. Some DNS attacks are:
  - *DNS Tunnelling Attack*: Encoding data from other programs or protocols within DNS queries and responses is known as DNS tunnelling. Typically, it contains data payloads that can hijack a DNS server and give attackers control over the remote server and its applications. DNS tunnelling frequently relies on a hacked system's external network connectivity as a backdoor into an internal DNS server with network access. Controlling a domain, which serves as an authoritative server and executes server-side tunnelling and data payload executable programs, is also necessary [68].
  - *DNS Amplification Attack*: DNS amplification exploits cause a targeted server to experience distributed denial of service (DDoS). This involves exploiting open DNS servers that are publicly available to overwhelm a target with DNS response traffic. Typically, the threat actor sends a DNS query request to the open DNS server as the first step in an attack, faking the source address to become the target address. Once the DNS server returns the DNS record response, it is passed to the new target, which the attacker controls [68].
  - *DNS Flood Attack*: DNS flood attacks employ the DNS protocol to conduct a user datagram protocol (UDP) flood. Threat actors launch valid (but fake) DNS request packets at a very high packet rate, generating a massive group of source IP addresses. The target's DNS servers respond to all queries since they appear genuine. The enormous volume of queries may then cause the DNS server to crash. A DNS attack uses many network resources, wearing down the targeted DNS infrastructure until it is shut down. As a result, the target's ability to access the internet is likewise interrupted [68].
  - *DNS Spoofing or DNS Cache Poisoning*: DNS spoofing, also known as DNS cache poisoning, is a method of utilizing updated DNS records to reroute online traffic to a malicious website that seems to be the intended destination. Users are prompted to enter their accounts at the fake site. They effectively allow the threat actor to steal access credentials and any sensitive information entered into the fraudulent login form after they submit the information. Furthermore, these malicious websites are frequently used to download viruses or worms onto end users' computers, giving the threat actor ongoing access to the device and any data it holds [68].
  - *NXDOMAIN Attack*: A DNS NXDOMAIN flood DDoS attack uses many requests for incorrect or nonexistent records to overload the DNS server. A DNS proxy server frequently handles such attacks by querying the DNS authoritative server with most (or all) of its resources. Both the DNS authoritative server and the DNS proxy server spend time processing invalid queries. As a result, the response time for valid queries gradually increases until it ceases [68].
  - *Domain Hijacking*: This attack may entail alterations to your DNS servers and domain registrar, which might send your traffic to alternative locations instead of the original servers. However, domain hijacking may also occur at the DNS level when attackers seize control of your DNS records [69].
    Domain hijacking is frequently triggered by various factors connected to exploiting a weakness in the domain name registrar's system. Once the threat actors have taken control of your domain name, they'll likely utilize it to carry out malicious actions like creating a phony website for payment systems like PayPal, Visa, or financial institutions. Attackers will exact copies of the legitimate website that stores crucial personal information like email addresses, usernames, and passwords [69].
  - *Fast-flux DNS*: The idea behind a fast-flux network is to provide botnets with the ability to quickly switch from one Internet protocol (IP) address to the next while exploiting a host that has been completely compromised. A fast-flux network employs various IP addresses and swiftly switches between them. One malicious domain name will have all the IP addresses pointing at it, but there will be many ways for users to access it. The IP address changes often, even though the domain remains constant regarding the website each user accesses [70].
    Botnets will use a range of IP addresses with malicious domains. Each will only be active for a short period before cycling in a new one, and as soon as people connect, the domain will collect their login details and other private data. Because the IP address changes frequently, it is pretty challenging to locate the source and stop it [70].
  - *Distributed Reflection Denial of Service (DRDoS)*: A DRDoS attack aims to send requests from its servers. The secret is to spoof the source address assigned to the victim so that all machines respond and flood the target. As witnessed when KrebsOnSecurity was struck by DRDoS in 2016, this type of attack frequently includes and is created by botnets that operate hacked systems or services that will ultimately be utilized to generate the amplification effect and attack the victim [69].
  - *Random Subdomain Attack*: Random subdomain attacks are frequently categorized as DoS attacks since they have the same objective as standard DoS. In this instance, attackers submit several DNS requests to a legitimate, active domain name. The primary domain name will not be the focus of the requests, but rather a large number of invalid subdomains. To stop all DNS record lookups, the attackers must first generate a DoS that overwhelms the authoritative DNS server that hosts the primary domain name. It's an attack that's hard to detect, as the queries will come from botnets from infected users who don't even know they're sending these types of queries from what are ultimately legitimate computers [69].
  - *DoS and DDoS*: A denial-of-service (DoS) attack occurs when an attacker utilizes their resources to make connection requests to a server that provides services to various users, eventually jamming and shutting down the server that provides services. In a DDoS

attack, an attacker (botmaster) requests a server that allows service providers to access various users using bots or computers under the attacker's control that may or may not be known to the computer's owner. As a result, the server is inevitably overwhelmed with requests [71].

In February 2020, Amazon's AWS Shield service thwarted the most significant DDoS attack ever (2.3 Tbps), which was carried out using hijacked Connection-less Lightweight Directory Access Protocol web servers [72].

- *Network Eavesdropping or Sniffing*: Cybercriminals or attackers listening in on network traffic passing through PCs, servers, mobile devices, and IoT devices are said to be conducting eavesdropping attacks. The act of reading or stealing data as it passes between two devices is known as network eavesdropping, often referred to as network sniffing or snooping, and it happens when malicious parties take advantage of weak or unsecured networks. Wireless communication is the most popular kind of eavesdropping [11].
- *Zero Day*: In general, "zero-day" refers to freshly identified security flaws that hackers may exploit to attack systems. Since the vendor or developer has only recently become aware of the problem, they have "zero days" to repair it. This situation is referred to as a "zero-day" defect. When hackers take advantage of the vulnerability before developers can fix it, it is known as a zero-day attack [73]. Zero-day is sometimes known as 0-day. The terms vulnerability, exploit, and attack are frequently used in conjunction with zero-day, and it's essential to know the distinctions between them. A software vulnerability the vendor has not yet patched is a zero-day vulnerability. Since there is no fix for zero-day vulnerabilities due to the vendor's unawareness, attacks are more likely to succeed. A zero-day exploit is a method hackers use to attack systems with a previously unidentified vulnerability. A zero-day attack is when a system is vulnerable, and a zero-day exploit is used to harm the system or steal data from it [73].

7.1.3. Software/application attacks

IoT (Internet of Things) software attacks involve exploiting vulnerabilities in IoT devices, systems, or network software components to compromise security, steal data, disrupt operations, or gain unwanted access. These attacks are directed at the software layer of IoT devices, which includes their operating systems, applications, firmware, and any software interfaces with which they communicate. Some of the software attacks are:

- *Malware Attacks*: Malware is malicious software designed to exploit or attack devices through their hardware or software. Malware is classified into several types: viruses, Trojans, rootkits, backdoors, etc. During the 1980s, malware was file infectors or boot sectors conveyed via floppy disks placed into the machine. However, as technology and electronic devices became more standardized, malware to target such systems progressed. IoT, a set of devices linked to the Internet without human involvement, is one such new technology being abused by malware. Personal computers began to target IoT devices by increasing their capabilities [17].

  Unlike typical malware, IoT malware crawls the Internet for susceptible devices. It hosts their initial payload, which is a stager script, in the devices to download the architecture-specific binary sample. After downloading, the script runs the sample, which talks with the C&C server. Because the malware contains some scanner modules, it infects more devices by sharing the sample. Most malware used to attack personal computers, such as Gamut, Necurs, and Skeeyah, began attacking IoT devices by enhancing their capabilities [17]. Some of the malware categories are:
  - *Worm*: This type of IoT malware spreads and propagates automatically in IoT devices. Juniper Threat classifies the worm as annoying malware due to its propagation mode. Mirai, Darlloz, Brickerbot, and Gitpaste-12 are some of the worms in IoT devices [17].
  - *Trojan*: A Trojan, often known as a Trojan horse or Trojan virus, is another type of IoT malware that seems innocent to users despite having hidden malicious functionality. Indeed, the functionality of a Virus and a Trojan is completely different because the Trojan cannot replicate itself, but the Virus can. ProxyM is an IoT virus that does email spamming and attacks involving DDoS [17].
  - *Virus*: Although the term "virus" is often used in computer science, the term "virus" in IoT devices appears to be confusing. IoT viruses behave similarly to others, except they infect IoT devices via self-replicating malicious code. As a result, the virus is difficult to remove and attacks the device in a complicated way. Silex, for example, is an IoT virus that enters the device and bricks it, commonly known as a permanent DoS attack [17].
  - *Backdoor*: Backdoor, an IoT malware category, is a type of malware where manufacturers make several hidden access mechanisms. Although these mechanisms make the customer fulfil requirements, these pave the way for making the device poor in security aspects. As a result, backdoors are also known as the front doors of attackers. Tsunami and Bashlite are backdoor IoT malware with a few resources that address them as Trojans [17].
  - *Spyware*: The IoT malware category spyware allows attackers to listen in on or spy on a target's data via an infected device. The type of IoT malware spyware allows attackers to listen in on or spy on a target's data via an infected device. Spybot, Skeeyah, and HNS are examples of IoT spyware that monitors users. As the use of IoT devices grows, so does the number of attacks perpetrated by this malware [17].
  - *Ransomware*: IoT ransomware is a type of malware that encrypts IoT devices and demands a ransom from the victim to exchange the device. Once infected, the attacker encrypts the data and prevents users from viewing it. The attacker delivers the decryption key and releases the device after receiving the ransom. Necurs is an IoT virus that performs ransomware attacks and other forms of digital extortion [17].
- *Denial-of-Service (DoS) and Distributed Denial-of-Service (DDoS) Attacks*: As mentioned in Section 7.1.2.
- *Code Injection*: Code injection is a technique threat actors use to enter or inject malicious code that exploits a software validation weakness. The malicious code is often "injected" into the targeted application and then executed by the server. Applications that employ unvalidated input data are often vulnerable to code injection. Code injection attacks often target flaws in data validation.

Data validation concerns may include data formats, the amounts of data expected, and the types of characters allowed [79]. Some of the different kinds of code injection are:

- *Directory Traversal*: Directory traversal is a web vulnerability that allows attackers to access files and directories not meant to be accessible by a UI interface. It occurs due to insufficient validation and sanitization of user input or file paths. In a directory traversal attack, attackers modify the file path using specially crafted input such as "../" or "../../" to access sensitive files or folders. Successful attacks may result in unauthorized access to sensitive information, file modification or deletion, or even execute arbitrary code on a device [80]. Some directory traversal attacks are null byte injection, local file inclusion attack, and remote file inclusion attack.
- *Hypertext Markup Language 5 (HTML5) Injection*: HTML5 injection is a web application vulnerability form allowing an attacker to embed malicious code into a web page, especially within the victim's browser environment. The attack has the potential to do harm and provide the attacker with the capacity to launch more attacks. Some of the HTML5 injection attacks are content injection, script injection, attribute injection, template injection, SVG injection, CSS injection, etc. [80].
- *Extensible Markup Language (XML) Injection*: XML injection attacks are a type of security vulnerability in environments that use XML to transfer and store data. These attacks happen when an attacker modifies the XML data being communicated or saved by altering the elements, attributes, or entities of an XML document, which may then be executed on the victim's application. As a result, this might have a variety of ramifications, allowing the attacker to carry out other harmful acts. Some of the XML injection attacks are XPath injection, XML entity expansion, XML external entity (XXE) injection, extensible stylesheet language transformations (XSLT) injection, etc. [80].
- *SQL Injection*: SQL injection is a type of application security vulnerability when an attacker inserts malicious SQL queries into application input fields to modify the underlying database and extract sensitive information. This attack can potentially target various technologies that rely on a database for storage, including online apps, desktop applications, and mobile applications. An attacker may transmit specially designed information that the application does not properly check, allowing the attacker to execute arbitrary SQL code.

  Furthermore, the attack may change the parameters of a SQL query to include new instructions or data. Depending on the database type of the vulnerable application, the attacker may additionally utilize particular syntax in the input to manipulate the structure or behaviour of the SQL query. Some of the SQL injection attacks are:

  * *In-band SQL Injection*: In-band SQL injection, also known as traditional SQL injection, refers to an injection attack in which the attacker exploits the same communication channel to execute the attack and get the outcomes. In technical terms, the attack entails inserting a malicious SQL query into a vulnerable application. This query is then communicated to the attacker using the same communication channel and the acquired results. There exist two subtypes.

    In the context of SQL injection, union-based SQL injection refers to the technique employed by an attacker to insert a UNION clause into a query. This UNION clause merges the outcomes of many SELECT queries, allowing the attacker to obtain unauthorized access to confidential information.

    In the context of error-based SQL injection, the attacker deliberately transmits SQL queries that are improperly formatted to the application. This action prompts the database to generate error messages, which inadvertently disclose details about the underlying structure of the database. The displayed mistakes typically serve as feedback to the individual carrying out the attack, allowing them to manipulate the injected SQL query to achieve a successful outcome [80].
  * *Inferential SQL Injection*: The attack, often known as blind SQL injection, involves the transmission of data payloads to the targeted device to observe and analyze its reactions and behaviour to get knowledge about its underlying architecture. This methodology is sometimes called blind SQL injection (SQLi) as it involves the absence of data transmission from the targeted website's database to the perpetrator. Consequently, the attacker cannot get in-band information about the attack [80].

    Blind SQL injections may be categorized into two distinct types: The Boolean kind of attack involves the transmission of a SQL query to the database, which then prompts the application to respond. The outcome of this result will be contingent upon the integrity of the enquiry. Consequently, the content of the Hypertext Transfer Protocol (HTTP) response has the potential to either undergo modifications or remain unaltered. Based on the modifications above, the attacker can ascertain if the message generated a correct or incorrect response [80]. The Time-based attack involves submitting a SQL query to the database, resulting in a deliberate delay of the database's response for a predetermined duration of time. The attacker can ascertain a query's veracity by analysing the database's response time. Depending on the result, an HTTP response will be generated promptly or after a certain period. This enables the attacker to ascertain the veracity of their message's outcome without necessitating direct access to the database's data [80].
  * *Out-of-band SQL Injection*: In this particular form of SQL injection attack, the attacker inserts a malicious SQL query into a vulnerable application. However, rather than passively awaiting the application's response for data recovery, the attacker proactively initiates an independent request to obtain the data over another route. As an illustration, the attacker may use a domain name system (DNS) inquiry to acquire the data or utilize an HTTP request to a device within their jurisdiction to collect the info [80].
- *Command Injection*: Command injection is a type of vulnerability that enables a threat actor to insert operating system commands directly into a program and subsequently execute them. These instructions are akin to the ones often entered into a Bash or Powershell terminal [81]. Command injection attacks refer to exploiting vulnerabilities present in programs that execute system commands. A potential threat actor can introduce malicious commands into the program, which are later executed by the underlying operating system. This enables the attacker to execute any intended operations on the system being targeted, potentially providing them with complete control.

Command injection attacks may be categorized into two main types: standard command injections and blind command injections. In the context of standard command injection, the attacker can view the outcomes of the injected instructions immediately. In contrast, in the context of blind command injection, the attacker cannot directly monitor the outcomes of the executed instructions. Instead, individuals must depend on alternate results, such as alterations in application behaviour, to assess the attack's success. Hence, conducting a blind command injection attack is typically more challenging than the above-mentioned method. The rationale for this is that the perpetrator does not receive prompt feedback on the efficacy or ineffectiveness of their injection attempt.

In an alternate scenario, the attacker needs to employ either guesswork or supplementary techniques to ascertain the outcomes of their injection. Some of the command injection attacks are powershell injection, command prompt injection, shell injection, script injection, remote command injection, dynamic link library (DLL) injection, library injection in unix-like operating systems, cron injection, process injection, memory injection attack, environment variable injection attacks, registry injection attacks, lightweight directory access protocol (LDAP) injections and Java logging framework injection (Log4j) [80].

– *Cross-Site Request Forgery (CSRF)*: Cross-site request forging (CSRF), also known as session riding, cross-site reference forging, and hostile linking, is a security vulnerability that arises when a malicious attacker deceives the target into unintentionally executing an action on a web application while they are currently authorized. The attacker commonly does this by introducing malicious code into a target individual's web browser via a malicious website or email, resulting in the user's browser requesting the vulnerable application without the user's awareness or authorization [80].

  The main concern associated with cross-site request forgery (CSRF) is the web application's incapacity to distinguish between a legitimate request launched by the user and a malicious request orchestrated by an attacker. This is because the web application depends exclusively on the authentication credentials saved in the user's browser to recognize the user without further procedures to confirm the user's intention to carry out the requested operation. Consequently, an attacker can exploit cross-site request forgery (CSRF) to execute activities on behalf of the user, including unauthorized financial transfers or altering the user's password, without their awareness or authorization. The subsequent examples illustrate several cross-site request forgery (CSRF) attacks [80].

  In the context of GET-based cross-site request forgery (CSRF), an attack transpires when an adversary entices a target to click on a hyperlink that incorporates a malicious request within the URL. The execution of the request occurs when the recipient interacts with the hyperlink, prompting their web browser to transmit the request to the vulnerable program. Due to the use of the victim's login credentials, the vulnerable application cannot differentiate between a valid request and a malicious one.

  The POST-based cross-site request forgery (CSRF) attack resembles the GET-based CSRF attack, but it exploits the victim's act of submitting a form that includes a malicious request. The attacker can potentially deceive the target by employing social engineering strategies, such as camouflaging the form as an authentic login or registration form. The login CSRF attack focuses explicitly on exploiting the login procedure of a vulnerable application. The potential attacker can include malicious code into a login form or request, so enabling unauthorized access to the victim's account without their awareness or permission [80].

  The logout CSRF attack resembles the login CSRF attack since it focuses on exploiting the logout mechanism of a vulnerable application. The potential offender can include harmful code into a logout request, resulting in the victim being logged out of the application without their awareness or consent. An Ajax-based cross-site request forgery (CSRF) attack transpires when a malicious actor takes advantage of a vulnerable AJAX request present on a webpage. The potential attacker possesses the capability to alter the Ajax request to incorporate a malicious request that is performed upon the victim's visit to the webpage [80].

– *Format String Attacks*: Format string attacks are a type of security vulnerability that may be exploited in many software programs. Format string attacks occur when an attacker can inject format string input into software that is vulnerable to such exploitations. A format string may be seen as a series of characters that establishes the arrangement and presentation of data when outputted or shown on a computer screen. The format string input can be exploited by an attacker by manipulating its content, resulting in the extraction of private information from a program's memory or the execution of arbitrary code. The attacker can utilize specific format string conversion specifiers, such as %n, %s, %x, or %p, to get access to memory regions and modify the data stored within those places. The potential consequences of a successful format string attack are substantial since it allows an attacker to gain control over a susceptible machine and execute malicious code [80].

  Some forms of format string attacks are Format string vulnerabilities exploited by information disclosure attacks to acquire sensitive information from a program's memory. By utilizing precise format string conversion specifiers, malicious actors can get data from memory locations following the desired input's memory address. This could expose system settings, passwords, or user data. Denial of service (DoS) attacks using formatted strings refer to instances where an attacker deliberately delivers input containing format string conversion specifiers, resulting in software crashes or halts. This may be achieved by using the %n specifier to write data to a memory location that is not valid, resulting in program failure [80].

  Arbitrary code execution attacks include the exploitation of format string vulnerabilities to execute arbitrary code on a system susceptible to such attacks. Malicious actors can write specific values to a designated address using format string conversion specifiers. This action can result in executing code loaded into such an address, enabling the attackers to gain control over the system. The abovementioned kind is often regarded as the most severe and dangerous manifestation of format string attacks [80].

– *Object Injection*: Object injection attacks leverage a security vulnerability present in web applications through serialization. Serialization is converting data structures or objects into a format that can be stored or sent. The act of object injection involves the intentional manipulation of the serialization process. This manipulation allows an adversary to inject objects that include malicious code [80].

Once injected, these objects can be run on the application server hosted on a specific IoT device. The consequences of such an attack may be severe, possibly leading to the entire compromise of the system. The vulnerability arises because serialization not only captures the state of an object but also includes information about the object's class and its related methods [80].

Object injection can manifest in several forms, including deserialization vulnerabilities, which refer to a specific form of object injection when an adversary manipulates the serialized data to incorporate harmful objects that can be executed during the deserialization process [80].

Prototype pollution vulnerabilities refer to a class of security vulnerabilities that arise from manipulating JavaScript prototypes. These vulnerabilities might allow an attacker to modify the behaviour of objects and functions. This form of object injection happens when a malicious entity changes an object's prototype to add arbitrary attributes to global object prototypes. As a result, objects created by users can inherit these newly added characteristics. Expression language (EL) injection refers to a specific sort of security vulnerability when an unauthorized individual inserts arbitrary code or expressions into the data processing expressions of an application. This vulnerability commonly impacts web applications that utilize expression languages such as JavaServer Pages (JSP) expression language or AngularJS expressions, among other similar technologies [80].

– *Firmware Code Injection*: Firmware refers to the fundamental software that operates at a low level and is tasked with managing the hardware components of a specific device. The storage of data in non-volatile memory, such as flash memory or read-only memory (ROM), is a common practice. This storage is of utmost importance as it plays a crucial role in the proper functioning of a device [80].

  Nevertheless, the development process may occasionally neglect the firmware of IoT devices, resulting in potential security risks. The attack vector known as firmware code injection involves the exploitation of vulnerabilities to implant malicious code into the firmware of a device. The attacks may manifest in several ways, such as inserting a backdoor into the initial firmware or altering the bootloader to initiate the execution of malicious scripts during the booting process [80].

  Furthermore, the attacker can activate unprotected protocols inside the firmware, rendering it vulnerable to exploitation. Moreover, the attacker can alter or append scripts inside the firmware, rendering it susceptible to further forms of code injection attacks, like SQL injection and XSS. Upon executing the inserted code, the attacker acquires unauthorized access to the device and its functionalities [80].

  The potential outcomes of this situation encompass a wide range of effects, including but not limited to the unauthorized acquisition of data, malfunctioning of the device, and the occurrence of more advanced forms of attacks, such as the utilization of the compromised device as a component of a botnet or the initiation of distributed denial of service (DDoS) operations.

  Several techniques exist that malicious actors might employ to introduce malicious programming into the firmware of IoT devices. In the case of direct physical access, an adversary can exploit hardware debugging tools or influence the firmware update procedure to introduce malicious code into the firmware directly. Remote exploitation refers to attackers using weaknesses in a device's firmware update process or communication protocols to introduce malicious code remotely. The process of attaining this objective frequently involves the practice of reverse engineering the firmware of the device and doing a thorough analysis to identify any vulnerabilities [80].

  Supply chain attacks include firmware compromise during the production phase before the device is delivered to the end user, enabling attackers to introduce malicious code. This issue is of particular significance due to its elusive nature, making it challenging to identify, and its potential to impact several devices concurrently. The breach of firmware can occur when attackers manipulate the files accessible for download on the manufacturer's website. Upon acquiring the firmware, the potential attacker may analyze its structure and functionality using reverse engineering techniques. Subsequently, they might embed a malicious code inside the firmware and reprogram the device with the updated, compromised version [80].

– *Log Poisoning Attacks*: Log poisoning refers to a method employed by malicious actors to inject a code into a log file. This attack exploits misconfigurations in log settings, allowing the injected code to be run. As a result, the attacker can avoid detection and carry out malicious actions on the targeted system. Log files are crucial for system administrators and security experts due to their inherent value in documenting system events, user activity, and potential security vulnerabilities [80].

  Various types of logs can be vulnerable to log poisoning attacks. Web server logs are a type of record that document various aspects of client requests, server answers, and any issues that may arise while processing these requests. The attacker can introduce malicious code into HTTP requests or change URL parameters using code. Some examples of log files often used in web servers include Apache access logs, nginx access logs, and IIS logs. Many software programs, including websites, mobile apps, and desktop software, create application logs. An attack occurs when an attacker inserts malicious code into fields designated for user input, influencing the log entries produced by the program [80].

  System logs encompass several sorts of logs that offer valuable insights into the events taking place within an operating system and its services and components. These logs serve the purpose of providing information and documentation on the activities mentioned above. Some examples of log management systems include Linux Syslog and Windows Event Logs [80].

  Authentication logs are responsible for recording user authentication occurrences, including instances of successful login, unsuccessful login attempts, and instances when an account has been locked. The attack entails the insertion of harmful code into fields designated for usernames or passwords. Examples of log files often used in computer systems are SSH logs in Linux-based operating systems and Windows Event Logs for login events.

  The mail server logs store comprehensive data on email transactions, including sent and received messages, encountered issues, and other pertinent occurrences. An attack occurs when an attacker inserts malicious code into the headers or body of emails. Some examples of logs are those generated by the Postfix, Sendmail, and Exim mail transfer agents [80].

Database logs are responsible for recording events that are associated with database operations. These events include performed queries, data updates, and failures. Attackers can insert harmful SQL code into queries, using weaknesses in log parsing tools or log management systems. Illustrative instances encompass log files derived from several database management systems, such as MySQL, PostgreSQL, Oracle, and SQL Server [80].

- *Hibernate Query Language (HQL) Injection*: HQL is an acronym for hibernate query language. Hibernate is an object-relational mapping (ORM) framework that facilitates mapping class definitions inside source code to corresponding SQL databases. The HQL (Hibernate Query Language) is a programming language similar to SQL (Structured Query Language). However, unlike SQL, HQL operates on persistent objects rather than directly interacting with tables and columns. The Hibernate framework facilitates the translation of HQL queries into SQL queries, which are then executed within the database [82].
  Hibernate query language (HQL) injection is a type of injection attack in which an adversary manipulates the HQL query to execute harmful SQL statements that can manipulate the database server of a web application [83].
- *Indirect Prompt Injection Attacks*: The indirect prompt injection refers to an attack that capitalizes on a vulnerability, allowing an attacker to insert harmful code into an application by manipulating prompts or messages presented to the user. This injection occurs when an application requests input from a user, and the attacker exploits this request to insert harmful code into the program.
  Recent research [84] has investigated the occurrence of injection attacks on application-integrated large language models (LLMs). The authors noted that integrating LLMs, such as ChatGPT, with other applications might render them vulnerable to the intake of untrusted data, potentially including malicious prompts. The authors demonstrated the potential use of such injections in delivering precise payloads. The methodology potentially enables malicious actors to assume control of LLMs by breaching critical security limits by executing a solitary search query [80].
- *Cross-Site Scripting (XSS)*: Cross-site scripting (XSS) is a form of cyberattack wherein the malevolent actor inserts malevolent scripts into websites and web applications. The objective is to ensure the execution of these scripts on the end-point devices of users, enabling threat actors to circumvent safeguards and assume the identity of users. Cross-site scripting (XSS) attacks include exploiting a benign website or online application, transforming it into a means to transmit harmful scripts to the web browsers of those unaware of the impending danger. The objective of the attack is to illicitly acquire cookies, session IDs, names, and passwords [79].
  Numerous programming environments and languages, including Flash, ActiveX, JavaScript, and VBScript, are susceptible to potential risks. JavaScript is often executed on web pages within web browsers, and cyber attackers often focus their efforts on exploiting vulnerabilities in JavaScript due to its strong integration with most browsers [79].
  XSS attacks may be categorized into three primary categories. The security vulnerability known as Reflected XSS occurs when an attacker can insert malicious code into a webpage. This code is then shown on a user's browser and executed inside the context of the website. This vulnerability stems from the web application's inability to adequately authenticate user input, enabling malicious actors to incorporate malicious code into the application's response. To carry out this attack, the attacker commonly entices the target to interact with a deceitfully constructed hyperlink or to provide input through a specifically tailored form that incorporates the malicious code. When the user interacts with the vulnerable web application, the malicious code is transmitted back to the user's browser and executed. This action grants the attacker the ability to initiate more attacks.
  In the context of Stored XSS, an attacker with malicious intent inserts and retains harmful code within a web application's database or alternative storage medium. The malicious code is sent to all individuals who visit the impacted webpage, possibly jeopardizing their security and enabling the attacker to execute various malicious activities.
  DOM-based XSS is a form of online application vulnerability that arises when an adversary can insert malicious code into a web page's document object model (DOM). This injected code is then executed inside the context of the victim's browser. DOM-based cross-site scripting (XSS) attacks commonly include the manipulation of URL parameters, form input fields, or other web page elements that contribute to the dynamic generation of content inside the document object model (DOM) [80].
  In contrast to reflected and stored XSS attacks, which arise from inadequate server-side input validation, DOM XSS vulnerabilities stem from executing client-side scripts. Identifying and mitigating such attacks would become more complex, as traditional techniques of server-side input validation may not be enough to reduce these risks [80].
- *LDAP Injection*: An LDAP injection refers to a type of attack that takes the use of web-based apps that are susceptible to vulnerabilities, wherein these programs create LDAP statements using user input. If software neglects to sanitize user input correctly, it becomes susceptible to potential exploitation by attackers who can manipulate LDAP statements by utilizing a local proxy. This vulnerability enables the execution of arbitrary actions, including the unlawful granting of permissions to queries and the change of material inside the LDAP tree. An LDAP injection attack frequently employs similar exploitation techniques as those utilized in SQL injection attacks [85].

- *Remote Code Execution*: Remote code execution (RCE) refers to a security vulnerability that enables unauthorized individuals to execute arbitrary code within an application's programming language, as per the language chosen by the developer. The word “remote” refers to the capability of an attacker to perform actions from a place distinct from the system where the application is being executed. RCE vulnerabilities have the potential to manifest in several forms of computer software across a wide range of programming languages and on diverse platforms. RCE vulnerabilities may be found in several software systems, including standalone Windows apps developed using C#, online applications and APIs implemented in PHP, mobile applications created using Java, and even within operating systems [86].
  Certain remote code execution (RCE) attacks have the potential to occur after a period of latency. As an illustration, the program can first store the remote code execution (RCE) payload within a configuration file, deferring its execution until later, even on several

occasions. The vulnerability is a stored remote code execution (RCE) vulnerability. It is essential to acknowledge that there is a common misconception between RCE (Remote Code Execution) and OS (Operating System) command injection [86].

In the context of remote code execution (RCE), the executed code is written in the same programming language as the application and operates within the application's environment. In the context of OS command injection, the perpetrator executes an operating system command. It is essential to acknowledge that although the phrase "code injection" is favoured by OWASP and expressly specified in CWE-94, the term "remote code execution" has more prevalence in usage across many contexts [86].

A well-known case of remote code execution (RCE) is the CVE-2021-44228 (often referred to as Log4Shell) found in Apache Log4j 2.x. This vulnerability was then accompanied by CVE-2021-45046 and CVE-2021-45105, which introduced a denial of service vulnerability. These cases exemplify instances of RCE occurring within non-web applications. The vulnerability mentioned above in Log4j, which does not need authentication from the attacker, impacts several versions of Log4j and is mainly located inside the JndiLookup class of this widely used open-source logging library [86].

Several widely used applications and services, such as Steam, Apple iCloud, and Minecraft, were first discovered to have vulnerabilities. CVE-2021-1844, observed in Apple's iOS, macOS, watchOS, and Safari, further illustrates remote code execution (RCE) vulnerability within a module of an operating system. When a victim employs a susceptible device to access a URL under the control of an attacker, the operating system will proceed to execute a harmful payload on said device [86].

The occurrence of CVE-2020-17051 within the Microsoft Windows NFSv3 framework illustrates a remote code execution (RCE) vulnerability found within a module of an operating system. A potential attacker can connect with a susceptible NFS server and transmit a payload, causing the targeted endpoint to execute such a payload.CVE-2019-8942 within WordPress 5.0.0 serves as an instance of a remote code execution (RCE) vulnerability found in a widely used online application. The potential for an attacker to run arbitrary code within the WordPress platform arises when a specifically manipulated picture file is uploaded, whereby the Exif metadata of said file contains PHP code [86].

- *Buffer Overflow*: A buffer overflow is a security vulnerability that arises when a computer program attempts to exceed the capacity of a buffer, a designated space for temporary data storage, by writing excessive data. This phenomenon can potentially result in program failure or, in certain instances, enable an unauthorized individual to execute malevolent instructions on the system.

  Buffer overflows can manifest when the software fails to adequately verify the size or structure of the input it receives, enabling a malicious actor to transmit a substantial volume of data that surpasses the buffer's allotted limit. In such an occurrence, the surplus data can overwrite additional segments of the program's memory, allowing the attacker to execute arbitrary code or gain control over the system [87].

  Buffer overflows are a prevalent security risk, particularly in outdated or inadequately engineered software. Preventing such occurrences might pose challenges due to their inherent nature of including unanticipated or malicious input, which deviates from the planned scope of the program's functionality. To mitigate the risks associated with buffer overflows, developers must exercise caution in validating information and implementing robust error-handling mechanisms inside their programs. This entails verifying the integrity and appropriateness of incoming data while fortifying the program's resilience against unforeseen inputs to prevent system crashes and potential security breaches [87].

  Buffer overflow attacks provide a significant security risk because they enable the unauthorized execution of arbitrary code on a system. This unauthorized execution can grant attackers complete control over the system, facilitating the possible theft of sensitive information. Buffer overflow frequently attains a prominent position inside the SANS Top 20 Most Dangerous Software Errors [87].

  The common weakness enumeration (CWE) is a comprehensive compendium of software security vulnerabilities encompassing a range of problems. Among these issues, there are several that pertain to the occurrence of a buffer overflow. CWE-120, sometimes referred to as "Buffer Copy without Checking Size of Input," delineates a specific circumstance in which a computer replicates data from one buffer to another without sufficiently verifying the input size, creating the possibility of a buffer overflow vulnerability. Additional vulnerabilities under the common weakness enumeration (CWE) framework that are associated with buffer overflows encompass: CWE-119: "Improper Restriction of Operations within the Bounds of a Memory Buffer", CWE-121: "Stack-Based Buffer Overflow", CWE-122: "Heap-based Buffer Overflow", CWE-125: "Out-of-bounds Read", CWE-131: "Incorrect Calculation of Buffer Size" [87].

  Several variations exist of buffer overflow attacks: Stack-based buffer overflow refers to the act of overwriting the memory of a program's stack, which is responsible for storing local variables and function calls. This manipulation allows for the execution of arbitrary code or the modification of the program's control flow. A heap-based buffer overflow refers to overwriting the memory allocated in a program's heap, which is responsible for dynamically allocating memory. This can result in the execution of arbitrary code or the modification of the program's behaviour [80].

  The integer overflow attack refers to a security breach that arises when a mathematical computation generates a result that exceeds the chosen data type's capacity, leading to an inaccurate representation of the value. It is commonly observed when the outcome of an operation exceeds the upper or lower limit that may be represented by the given data type, resulting in the value "wrapping around" and returning to a lower or higher value [80].

  Return-oriented programming (ROP) is a technique that involves utilizing pre-existing code snippets, referred to as gadgets, within a program to run arbitrary code without the need to inject new code. Jump-oriented programming (JOP) refers to utilizing pre-existing code snippets within a program to execute arbitrary code by navigating to various memory regions, all without introducing new code injection [80].

  The global offset table (GOT) buffer overflow is a specific buffer overflow attack that focuses on exploiting the GOT data structure located within a program's memory. Notably, software programs commonly utilize the global offset table (GOT) to hold the memory

locations of dynamically linked functions and variables. By manipulating the GOT software, an attacker can reroute the program's execution flow towards their malicious code [80].

The unicode overflow attack is a type of hack that takes advantage of a vulnerability resulting from inadequate processing of unicode-encoded data. Like other forms of buffer overflow, this attack is executed by transmitting malicious unicode input that surpasses the allocated buffer size. As a result, memory corruption occurs, potentially leading to the execution of malicious code. The Return-To-Libc attack is devised and executed to circumvent security measures, such as a non-executable stack. Instead of directly inserting and running malicious code into the stack, the attacker replaces the return address with the address of a targeted function that already exists in a standard C library, such as printf or scanf, among others [80].

- *Vulnerable Firmware*: The exploitation of vulnerabilities in firmware, which refers to the software integrated into physical devices, can enable malicious actors to modify the behaviour of the device or gain unauthorized control over it.
- *Man-in-the-Middle (MitM) Attacks*: Man-in-the-middle (MitM) attacks manifest when an unauthorized individual gains access to the communication channel connecting two end systems. This is achieved by introducing a malicious node between the legitimate nodes or exploiting vulnerabilities in the communication protocols inside an IoT network. The idea of man-in-the-middle (MitM) allows hackers to manipulate the flow of data, modify the configuration of the network architecture, fabricate counterfeit identities, and generate deceptive and inaccurate information to undermine the security of an IoT system. The many forms of man-in-the-middle (MitM) attacks include eavesdropping, Sybil attacks, Wormhole attacks, Identity replication attacks, Node replication attacks, and others [88]. MitM attacks may be categorized into two basic types: passive attacks and active attacks [89].
  - *Passive Man-in-the-Middle (MitM) Attack*: A passive man-in-the-middle (MitM) attack is characterized by the absence of active alteration of communication by the attacker. In contrast, the assailant clandestinely intercepts the conversation to obtain unauthorized access to confidential data. Despite the absence of communication manipulation by the attacker, they are still capable of acquiring sensitive information, such as usernames, passwords, and other secret data. This is mostly due to inadequate encryption of the majority of traffic. Eavesdropping can be classified as a passive man-in-the-middle (MitM) attack [89].
  - *Active man-in-the-middle (MitM) attack*: An active man-in-the-middle (MitM) attack refers to the deliberate interception and alteration of communication between two entities by an attacker. Once an unauthorized individual has access to the communication channel, they can control the communication process by intercepting, altering, or introducing new messages. The most prevalent forms of active Man-in-the-Middle (MitM) attacks are impersonation, spoofing attacks, and data manipulation [89].

    It is essential to acknowledge that MitM represents a multilayered threat, including all layers of the IoT architecture. The veracity of this claim may be demonstrated using a spoofing attack. DNS Spoofing impacts the application layer, IP spoofing affects the network layer, and ARP Spoofing operates at the link layer. Moreover, those who do malicious acts can employ rogue access point attacks to intercept a network at the physical layer. In addition to the sorts mentioned above of man-in-the-middle attacks, several tactics can be employed to execute these attacks [89].
  - *Sniffing*: As mentioned in the Section 7.1.2.
  - *Packet Injection*: Packet injection is a method that enables malicious actors to control the flow of network traffic by introducing malicious packets into the network. Within the domain of the IoT, malicious actors may employ this methodology to disrupt the communication linkages between IoT devices and cloud-based services or to introduce commands that potentially alter those devices' operational characteristics. Scapy and MitM are robust tools for implementing this strategy.
  - *Session Hijacking*: Session hijacking is a technique employed by malicious actors to acquire a user's session information illicitly, granting them the ability to assume the user's identity and gain unauthorized access to their data. Within the domain of the IoT, someone with malicious intent can employ session hijacking as a means to enter the user's IoT devices illicitly and then exert remote control over them. Bettercap is widely recognized as a highly effective tool for implementing this strategy [89].
  - *SSL Stripping*: SSL stripping is a method that involves converting an encrypted SSL connection into an unencrypted one. In the IoT context, malicious actors can employ SSL stripping techniques to surreptitiously intercept and modify the data transmitted between IoT devices and cloud-based services. SSLStrip is widely recognized as a tool that may be utilized to execute this particular strategy [89].
- *Zero-Day Exploits*: As mentioned in the Section 7.1.2
- *Authentication Attacks*: Authentication attacks in the context of the IoT are activities aimed at circumventing or altering the authentication systems employed by IoT devices, networks, or services. Authentication refers to the procedure through which the identification of individuals, devices, or entities is confirmed before authorizing their access to resources or services. In IoT settings, malicious actors exploit authentication mechanisms to attain illegal entry, pilfer confidential data, cause disruptions to operational processes, or execute destructive activities [90].
  - *Password Guessing*: Attackers employ trial-and-error techniques to make educated guesses to ascertain usernames and passwords. Passwords that are weak or easily guessable are especially vulnerable to this attack [90].
  - *Brute Force Attacks*: Brute force password attacks employ a systematic approach to systematically test all conceivable combinations to ascertain a password. This approach demonstrates efficacy when used with passwords that possess a limited number of characters and exhibit a relatively low complexity. Even with the most advanced contemporary systems, the feasibility of utilizing a password consisting of eight or more characters can be compromised. If a password only consists of alphabetic characters, encompassing both uppercase and lowercase letters, it is estimated that around 8,031,810,176 attempts would be required to decipher it successfully. This assumes that the potential attacker knows the length and complexity criteria of the password. Additional considerations are numerical values, the distinction between uppercase and lowercase letters, and special characters within the context of the specific language being localized [90].

A brute force attack can finally discover the password given the appropriate settings. The computational resources and execution duration frequently render brute force testing inconsequential upon its completion. The duration of attacks is contingent upon the duration required to produce all conceivable permutations of passwords. Subsequently, the consideration of the target system's reaction time is considered. Brute force password attacks are often regarded as the least efficient approach to password cracking. Therefore, threat actors employ them as a final option [90].

- *Credential Stuffing*: Credential stuffing is an automated hacking technique that uses stolen credentials. The credentials consist of collections of usernames, email addresses, and passwords. The method commonly utilizes automation to initiate login requests targeted at an application and to record successful login attempts for subsequent exploitation [90].
  Credential stuffing attacks do not engage in the process of systematically attempting to crack or speculate passwords. The threat actor employs specialized tools to automate authentication by leveraging previously obtained credentials. This methodology may include doing a large number of iterations to ascertain the probable instances when a user may have employed the same login credentials on a different website or service. Credential stuffing attacks exploit the practice of password reuse, leveraging the fact that many users use identical credential combinations across different online platforms [90].
- *Phishing & Vishing*: Phishing and vishing, which refers to voice calls, are frequently utilized as means of acquiring information for further attacks, as well as to introduce malicious software onto an endpoint. The malware, as mentioned above, can illicitly extract credentials. It can also be utilized as a faked password reset attack component. A prevalent strategy employed by phishing emails is the inclusion of a hyperlink that prompts the recipient to reset their account password. The attackers may assert that this occurrence is attributable to the probable breach of a previously utilized password. The email may display the emblem and resemblance of a commercial entity, such as a financial institution, vendor, or service provider [91].
  Nevertheless, the hyperlink inside the electronic mail redirects the target individual to a deceptive interface designed to reset passwords. Subsequently, the attacker procures the authentic password to get unauthorized access to the target individual's valid account. In the context of an employee, it is noteworthy that a deceptive email regarding password reset may manifest as originating from the corporate help desk. These attacks serve as a persistent reminder of the significance of end-user training. Users must exercise constant vigilance to verify the legitimacy of solicited email addresses or phone numbers.
- *Man-in-the-Middle (MitM) Attacks:* As mentioned in Section 7.1.3.
- *Session Hijacking:* As mentioned in Section 7.1.3.
- *Token Replay*: Attackers use vulnerabilities to acquire authentication tokens, afterwards employing them to assume the identity of the authorized user and get unlawful access [92].
- *Password Spraying*: Password spraying is an attack that relies on credentials and aims to gain unauthorized access to several accounts by employing a limited set of commonly used passwords. In terms of conceptualization, this phenomenon might be considered the opposite of a brute-force password attack. The brute force method involves systematically and repetitively generating many password combinations to obtain unauthorized access to a specific user account [90].
  In the context of a password spray attack, the attacker systematically tests a singular, often employed password (e.g., "12345678" or "Passw0rd") across several accounts before moving on to test an alternative password [90].
  The threat actor systematically attempts to authenticate each user account in their roster using a uniform password, resetting the roster and testing the next password. This strategy effectively reduces the likelihood of the threat actor being detected and experiencing lockouts on a single account due to the time intervals between successive attempts. The probability of a threat actor successfully accessing a resource increases when an individual user does not possess proper password hygiene or on a specific account [90].
- *Biometric Spoofing*: In the context of IoT systems employing biometric authentication methods such as fingerprint or face recognition, malicious actors have the potential to exploit the system by utilizing counterfeit biometric data to circumvent the authentication process.
  Biometric spoofing pertains to fraudulent activities in biometric authentication wherein counterfeit samples, such as fabricated fingerprints, face scans, or iris scans, are presented as a means of executing a "presentation attack". Although biometric identification technologies are more robust than password-based systems, it is important to note that they are not impervious to vulnerabilities. Modern threats are undergoing a process of adaptation to circumvent the safeguards mentioned above. Therefore, several stringent authentication prerequisites outlined in compliance and regulatory frameworks need liveness detection mechanisms to ascertain the authenticity of the provided credentials [93]. Some of the different Biometric Spoofing attacks are:
  * *Facial Recognition Spoofing Attacks*: Attackers may trick face recognition systems using various methods. Print Attack employs a printed image of the target's face to deceive the facial recognition software. This is one of the most basic techniques, and it works well against less complex systems (the majority of which must be used where they would safeguard sensitive data). Replay Attack involves hackers filming the target's face on camera and replaying it in front of the camera. This strategy frequently works better than a print attack because it includes motion, which certain facial recognition systems might need [93].
    During a 3D mask attack, the attacker fashions a lifelike 3D mask of the target's face and puts it on. While this strategy might be more complicated to detect, it can also be challenging to use effectively without the right tools and abilities. In a deep fake attack, a video of the target's face is made using machine learning or AI software. Thanks to deepfake technology, some face recognition algorithms find it challenging to distinguish between actual and synthetic facial movements and expressions [93].
  * *Fingerprint Recognition Spoofing Attacks*: Fingerprint verification systems, although often robust, may still be susceptible to spoofing without adequate countermeasures. In fake fingerprints, hackers make synthetic fingerprints that resemble the target

user's fingerprint pattern, frequently derived directly from a fingerprint, using materials like gelatin. The attacker's finger or a fake finger can then be put over the fake fingerprint to fool the fingerprint scanner [93].

In latent fingerprints, a perpetrator uses adhesive tape or other means to extract a target user's latent fingerprint from a surface and transfer it to a substance that can deceive the fingerprint scanner. 3D-printed fingerprint describes a complex attack that entails using digital technology to create a 3D model of the target user's fingerprint, which is subsequently printed using materials that have qualities similar to those of human skin. This technique can produce accurate copies that can trick some fingerprint readers [93].

* *Iris Recognition Spoofing Attacks*: While Iris recognition is typically considered a highly secure biometric modality, it can still be subject to spoofing attacks if the proper safeguards are not taken. Digital Iris images are one type of iris recognition spoofing attack that involves projecting a digital image or video of the target user's Iris onto a device screen, like a smartphone or tablet, and then presenting it to the iris scanner. This technique can trick some biometric scanners by altering gadgets' lighting and sharpness settings [93].

  Physical eyes, albeit an uncommon and extreme option, can also deceive the iris identification system by employing a preserved cadaver eye with the target user's iris pattern. These can be tougher to detect if the contacts are correctly made. This deterrence causes someone to steal a deceased subject's eye and utilize it quickly to be effective [93].

- *Phishing Attacks*: This attack is widespread and frequently used to steal sensitive user information. It happens when a hacker poses as a reliable entity and tricks people into downloading an attachment or entering personal information on a phoney website, which leads to the installation of malware or the revealing of sensitive data. Specialized phishers target the absence of specialized active security measures by systems and the lack of knowledge or attentiveness of users with compromised attacks that combine social engineering and sophisticated tactics known as compromised attacks for critical infrastructures. Zero-day malware, link manipulation, filter evasion, obscuring brand logos, website forgery, covert redirect, etc., are some of the tactics used. These are initially targeted at vendor/remote websites, followed by the hacking of IIoT systems and, generally speaking, the taking over of operational systems that are linked to it [94].

  The IIoT is often accessed or entered at a front-end level by malicious users. He stays there while doing reconnaissance and network mapping until the best opportunity to launch a broad attack is identified. He then pivots between different systems to apply the necessary exploits to hack ICS systems [94].
- *Sql Injection Attacks*: As mentioned in Section 7.1.3.
- *Reverse Engineering Attacks*: Reverse engineering attacks against IoT (Internet of Things) devices entail disassembling, examining, and comprehending the inner workings of these devices to find vulnerabilities, exploit flaws, and perhaps obtain unauthorized access or control over them. Although reverse engineering might be a difficult task, it is a vital tool for anybody involved in IoT security. Security experts may discover possible attack mechanisms, create new security protocols, and enhance system security by studying the underlying processes of a device or system. Because IoT devices sometimes have low computing capabilities and may not be equipped with sufficient security protections, these attacks can represent substantial security threats [95].

  Although usually based on static analysis, reverse engineering may occasionally be carried out through dynamic analysis while utilizing debugger tools. Reverse engineering attacks on IoT devices can serve various purposes, including extracting sensitive data (passwords, encryption keys, personal information), changing a device's firmware to allow unlawful functionality, creating bespoke firmware for maliciousness and taking control of a device to create a botnet or perform remote control operations [96]. Some of the common steps in reverse engineering attacks are [97]:
  - *Device Acquisition & Inspection of the Device*: Acquire the desired IoT device. This might entail buying the gadget, getting one used, or discovering one in a controlled setting. Crack up the gadget to see what's inside. Obtaining access to the electronics and memory can entail removing the shell, soldering, or doing other physical operations.
  - *Extraction of Firmware and Data*: Remove the firmware (software that regulates the operation of the device) from the device's memory. To do this, it may be necessary to replicate the firmware from storage components like flash memory using specialist tools and methods.
  - *Analyzing the Firmware*: Examine the extracted firmware to learn more about its components, code, and structure. This could entail looking at the code and data structures with tools like disassemblers, decompilers, and debuggers.
  - *Vulnerability Identification*: Look for vulnerabilities in the firmware, such as buffer overflows, unsafe communication protocols, inadequate encryption, and hardcoded passwords.
  - *Exploitation*: Use the vulnerabilities to your advantage to take over the system or get access without authorization. This can entail creating malicious code, inserting it into the firmware, or taking advantage of software flaws.
  - *Modification or Repackaging*: Change the firmware to accomplish certain objectives, including backdoors, changing functionality, or modifying the gadget for malicious purposes.
- *Sniffing Attacks*: As mentioned in the Section 7.1.2.
- *Logic Bombs*: A logic bomb is a sequence of instructions in a program that may attack an operating system, a program, or a network by delivering a malicious payload. It doesn't start unless certain criteria are satisfied. These restrictions might be as straightforward as a certain day or hour. An even more complicated illustration is when a company terminates an employee and records the termination in its database. Logic bombs frequently include a computer worm or virus. Even though some individuals confuse the two names, they don't refer to the same kinds of malware [98,99].

  A logic bomb technically isn't malware, but the code inside is harmful. A logic bomb attack is often a sort of cyber sabotage conducted by an inside attacker, who is typically malicious, in contrast to certain malware-like viruses or worms that break into a safe system on

their own. An example of this may be a dissatisfied current or former employee who has access to sensitive data or administrative control over systems, such as a programmer or information technology administrator [98,99].

Logic bombs might be made by workers who fear being dismissed as retaliation against their employers. They may disarm the explosives daily if their firms still employ them. Once free, they can fire attacks whenever they choose to do the most damage. Logic bombs also provide certain external hazards from malicious programs. WORM_SOHANAD.FM is one example. Logic bombs can be hazardous due to their secrecy [98,99].

In addition to the fact that they are dormant and ready to explode like a volcano, their payloads pose a mystery hazard. These strikes have the potential to surprise their targets completely. Furthermore, it might be difficult to identify the threat actor who attacked since evidence can be erased when a logic bomb is being finished. Additionally, attackers might utilize the additional time to hide their identities [98,99].

- *Exploitation of a Misconfiguration*: Several systems and components must be configured for IoT applications to function successfully. Consequently, each of these components has to be correctly configured for security. They are easily exploitable by a malicious actor if they are not correctly set. Operating systems, servers, frameworks, database management systems, and any other applications must be appropriately set up for a secure IoT environment [71].
- *Cross Site Scripting*: As mentioned as in Section 7.1.3

7.1.4. Encryption attacks

IoT security threats may include attacks on encryption schemes. Side-channel attacks target the implementation of cryptographic methods rather than the algorithms themselves. By examining physical measurements made during calculation and the internal state of the physical device during processing, attackers can deduce the encryption key [100].

Cryptanalysis attacks look for vulnerabilities in the cryptographic algorithm to derive encryption keys. Cryptanalysis attacks can be ciphertext-only, chosen-plaintext, adaptive-chosen-plaintext, chosen-ciphertext, or adaptive-chosen-ciphertext, depending on the information the attacker has access to Ref. [100].

A malicious actor can intercept communications between two users and use those communications to decrypt data using keys shared by both users, making encryption methods vulnerable to MitM attacks. Users continue to believe they are simply speaking to one another, as they have done in previous MitM attacks [100].

7.1.5. Data attacks

The term “data attack” in the context of the IoT refers to a variety of malicious actions intended to jeopardize the availability, confidentiality, integrity, and general security of data inside IoT systems. The IoT is a network of interconnected smart devices that gather, share, and analyze data, making them vulnerable to various data-related attacks. Some common data attacks in IoT include.

- *Data Interception and Eavesdropping*: Attackers intercept and eavesdrop on the communication between IoT devices and data servers. This can lead to unauthorized access to sensitive information, such as personal data, passwords, and confidential business data.
- *Data Tampering*: The data exchanged between IoT devices and servers are subject to alteration by malicious actors. This may entail the alteration of sensor readings, directives, or any other kind of data transmission. An instance of tampering might be manipulating temperature sensor readings within an industrial environment, resulting in erroneous decision-making processes based on unreliable data.
- *Replay Attacks*: As mentioned in Section 7.1.2
- *Man-in-the-Middle (MitM) Attacks*:As mentioned in Section 7.1.3
- *Denial of Service (DoS) and Distributed Denial of Service (DDoS) Attacks*: As mentioned in Section 7.1.2
- *Data Falsification*: A data falsification attack within the context of the IoT pertains to a malicious act in which an attacker deliberately modifies or manipulates data inside an IoT system to deceive, disrupt, or compromises its functioning, integrity, or security. Attackers introduce fabricated information into the system by capitalizing on weaknesses or actively changing device inputs. This phenomenon can potentially result in erroneous decision-making processes founded upon facts of questionable reliability [101].
- *Device Impersonation*: Device spoofing refers to a method of attack when a malicious device alters the IP address, MAC address, or other identifying details of a genuine device, therefore assuming the identity of a valid device [102,103].
- *Firmware and Software Exploitation*: As mentioned in the Section 7.1.3
- *Insecure APIs and Interfaces*: The presence of vulnerabilities in APIs (Application Programming Interfaces) and interface services or frameworks utilized for communication with IoT devices can result in unauthorized access to or modification of data.

7.1.6. Side channel attacks

Side channel attacks are a class of attacks that exploit “side channel information”, which refers to information that may be obtained from encryption devices other than the plaintext or the ciphertext produced by the encryption process. Encryption devices provide quantifiable time information, statistics on power usage, and several other data.

Side channel attacks leverage several types of information to get the cryptographic keys employed by the targeted device. The premise of this statement is rooted in the observation that logic operations possess physical attributes contingent upon the input data they receive. Instances of side channel information include temporal attacks, power analysis attacks, interference analysis attacks, electromagnetic attacks, and environmental attacks [2,104].

7.2. IoT attack based on IoT threat type

An IoT threat may be defined as a potential danger, vulnerability, or risk that has the potential to undermine the security, privacy, functioning, or dependability of IoT devices, systems, and networks. The risks mentioned earlier can originate from many origins and possess the capacity to inflict substantial damage upon persons, companies, and even vital infrastructure [48]. IoT attacks can be categorized into two threat types: privacy and security threats. However, before that, we must identify the difference between privacy and security. Security and privacy are closely related concepts, but they refer to different information and data protection aspects. The primary objective of security is to safeguard data, systems, and networks from unauthorized access, attacks, and threats.

The concept involves implementing strategies to proactively mitigate, identify, and address diverse cyber attacks, unauthorized data access incidents, and other malicious behaviours. The primary objective of security is to uphold the principles of confidentiality, integrity, and availability concerning information. The process includes implementing various technologies, policies, and procedures to protect digital assets against unauthorized access, malicious activities, hacking and malware, and other possible threats. Security measures encompass a range of strategies and technologies to safeguard information and systems against unauthorized access or malicious activities. These measures commonly include the implementation of firewalls, encryption protocols, multi-factor authentication mechanisms, intrusion detection systems, and the frequent application of software updates [105,106].

The security field frequently necessitates the adoption of a proactive stance to safeguard against possible dangers. Privacy, conversely, pertains to the regulation of entry to personal information and the guarantee of people's entitlement to maintain the confidentiality of their data. The concept pertains to safeguarding an individual's personal information, thwarting any unlawful acquisition, utilization, or dissemination of this information [105,106].

The idea of privacy centers around empowering individuals to make well-informed decisions regarding collecting and utilizing their data, particularly emphasizing the actions of organizations and enterprises. Privacy legislation, such as the General Data Protection Regulation (GDPR) implemented by the European Union and the California Consumer Privacy Act (CCPA), provides criteria for the appropriate management, storage, and dissemination of personal data. The concept of privacy frequently revolves around the rights and consent of individuals concerning their data [105,106].

- *Privacy Threat*: The concept of privacy within the context of the IoT can be categorized into three distinct areas:
  (1) The recognition and understanding of potential privacy risks associated with smart devices and services that are connected to the data subject.
  (2) The ability for individuals to have control over the gathering and handling of their personal information by the smart devices in their vicinity.
  (3) The awareness and control over these entities' subsequent utilization and sharing of personal information with any external entity that falls outside the individual's sphere of control.

  In a smart home scenario, the area around an individual's residence or immediate surroundings may be called their sphere, depending on the specific circumstances. The impression and expectations of privacy tend to differ among individuals, resulting in a lack of clarity around the idea of personal information. Hence, it is imperative to thoroughly evaluate the sensitivity of the data and the corresponding user needs while developing novel systems and services [48]. Some examples of Privacy threats are:

– *Identification*: Identification refers to associating a (permanent) identifier, such as an individual's address, name, or alias, with the individual and relevant information about them. The potential risk is in the process of associating an individual's identification with a certain level of privacy, breaching contextual boundaries. This action also serves to trigger and enable other dangers. For example, the practice of profiling and monitoring individuals or aggregating diverse data sources. The current prominence of identification risk mostly lies in the information processing stage inside backend services, where a substantial volume of data is gathered in a centralized location outside the control of the individual [48].

  One of the primary obstacles encountered in the realm of identification is the development of IoT systems that prioritize localized processing over centralized approaches and horizontal interactions over vertical ones. This design philosophy aims to limit the extent to which identifying data is accessible outside an individual user's domain [48].

– *Localization and Tracking*: Localization and tracking refer to identifying and recording an individual's precise whereabouts over time and across different physical locations. Identifying tracking needs is essential to establish a consistent association between continual localization and a specific individual. Currently, the ability to monitor and trace individuals is facilitated by several methods, including analyzing internet traffic, utilizing global positioning system (GPS) technology, and tracking mobile phone whereabouts [48].

  The majority of concrete privacy breaches associated with this danger have been established, such as GPS tracking, the leaking of private information, and the pervasive sense of being pursued. Localizing and monitoring typically do not result in privacy infringements in the context of near physical proximity. For instance, those near the subject can immediately view their whereabouts. The process of localization and tracking presents a potential concern, particularly at the information processing stage, since it involves the creation of location traces stored in backend systems beyond the control of the individual [48].

  The primary obstacles encountered in the field of localization and tracking pertain to the recognition of tracking amongst the accumulation of passive information, management of shared location data inside interior settings, and the implementation of privacy-preserving protocols for connection with IoT systems [48].

– *Profiling*: Profiling refers to gathering and organizing information files about individuals to infer their interests through linkage with other profiles and data. Profiling methods are primarily employed in e-commerce to facilitate personalization, including recommender systems, newsletters, and adverts. Additionally, these approaches are utilized for internal optimization by using consumer demographics and preferences. Instances in which profiling has resulted in infringements against privacy include pricing cases discrimination, the dissemination of unwanted adverts, the manipulation of individuals using social engineering tactics, and the occurrence of erroneous automated judgements, such as Facebook's automatic identification of individuals deemed sexual offenders [48].
Gathering and trading personal information about someone is widely regarded as a breach of privacy. The instances mentioned above demonstrate that the risk of profiling primarily arises at the dissemination stage, affecting both third parties and the subject through the potential for inaccurate or discriminatory judgements. The methodologies mentioned above have the potential to be implemented in IoT scenarios but with necessary modifications to accommodate the prevalent paradigm that presupposes a centralized database. Considering the multitude of dispersed data sources inherent to the IoT ecosystem is essential [48].
The recalibration of measurements and redesign of algorithms necessitates significant effort, as exemplified by recent research in differential privacy for dispersed data sources. Collecting data is a fundamental aspect of the IoT and plays a significant role in its implementation. Therefore, reconciling the competing interests of firms seeking to engage in profiling and data analysis with the privacy needs of individuals is widely acknowledged as a significant difficulty [48].
– *Privacy Violating Interaction and Presentation*: In this context, sensitive personal information is transmitted over a publicly accessible platform and then disclosed to those with malicious intent. Various IoT applications, including those in industrial, infrastructure, medical, and healthcare systems, require significant connections between users. In these systems, it is plausible that consumers are presented with relevant information by using smart objects in their environment, for example, by employing various lighting approaches and utilizing television or desktop displays to display video content [48].
On the other hand, people exert control over systems using an alternate, intuitive approach by leveraging smart objects present in their surroundings, including sensing and communication capabilities. However, it is important to note that many communication and organizational processes are inherently public. This thus raises privacy concerns when confidential information is exchanged between the user and the system [48].
– *Lifecycle Transitions*: Privacy is compromised when smart objects disclose confidential information by manipulating administrative domains throughout their existence. This pertains to observing undermining pictures and videos commonly encountered on cameras and other modern gadgets. The primary cause of privacy controversies arising from the life cycle of IoT devices is information collection, which is contingent upon the information level of the IoT reference model [48].
The current life cycle design for many customer care products is a one-time purchase with limited further advancements in performance. Smart objects can potentially enhance the life cycle by facilitating activities such as exchanging, lending, gifting, and disposing in an invaluable manner. Hence, it is imperative to acknowledge the necessity for flexible outcomes that effectively address specific challenges. Certain life cycle transitions, such as sharing a smart product, necessitate the secure attachment of confidential information for a transitory phase. The rightful owner can access and utilize the undisclosed information consistently [48].
– *Inventory Attack*: The term “inventory attack” refers to the unauthorized collection of information about personal electronic device specifications and features. The interconnection of IoT devices is widely recognized as a significant and dynamic characteristic of the IoT. Smart objects can be accessed and queried over the Internet through the utilization of several Internet protocols. Authorized organizations can access information from various sources, including certified system users and owners [48].
On the other hand, unauthorized organizations can exploit this access to gather comprehensive data pertaining to specific locations, such as office buildings, public institutions, or industrial areas. Despite the ability of smart objects to efficiently distinguish between permitted and unauthorized organizations, it is possible to classify further and identify these companies based on their transmission rate and other unique qualities. With the expected advancement of wireless sensor networks (WSNs) technology, fingerprinting methods can be shown in a passive manner by acting as a covert listener near the vicinity of a victim's residence [48].
– *Linkage*: Linkage refers to a kind of threat wherein previously independent system devices are interconnected, resulting in the disclosure of information from disparate data sources that were before undisclosed to the previously isolated entities. The consumers lack awareness of the comprehensive assessment and potential data loss when integrating diverse datasets and authorizations. One such instance of privacy infringement through connection may be observed in the fast proliferation of unidentified data sources [48].
The expansion of the IoT will be further intensified by the emergence of linkage hazards, mainly due to two key factors. Initially, the implementation of a parallel interconnection would facilitate the integration of systems across several companies, resulting in the creation of a diversified system capable of offering novel services that were previously unavailable inside individual systems. Additionally, establishing a thriving network of such entities would necessitate a flexible exchange of information and collaboration across diverse persons [48].

- *Security Threat*: The primary distinction between a security thing and a security attack is their respective characteristics [48]. A security thing refers to an entity that fulfils all of the IAS-Octave security criteria. In contrast, a security attack denotes an intentional act that poses a risk to at least one of the IAS-Octave security criteria as defined in Table 12. Some examples of security threat attacks are DoS, tag cloning, tag swapping, node replication selective forwarding, etc.

7.3. Attack execution

The concepts of centralized and decentralized IoT attacks pertain to distinct methodologies concerning the arrangement, governance, and implementation of cyber attacks directed against IoT devices and systems.

Table 12
Security requirements for IAS using Octave methodology.

Security requirements	Definition	Abbreviations
Confidentiality	Only authorised objects or users can obtain access to the data	CONF
Integrity	Data completeness and accuracy is preserved	INTG
Non-repudiation	IoT system can validate the occurrence of any event	NREP
Availability	Ensuring accessibility of an IoT system and its services	AVAL
Privacy	Presence of privacy rules or policies	PRIV
Auditability	Monitoring of the IoT object activity	AUDI
Accountability	End users can take charge of their actions	ACNT
Trustworthiness	Reliability on IoT object identity	TRST

- *Centralized Attacks*: Centralized attacks refer to a type of cyber attack wherein a singular point of control or a central coordinating body is responsible for orchestrating the attack on several IoT devices. In this methodology, the attacker commonly employs a command and control (C&C) server to orchestrate the hacked devices in executing malicious activities. Centralized attacks exhibit a higher degree of organization and efficiency in terms of disseminating orders and collecting data from compromised devices. Nevertheless, the identification and takedown of the command and control (C&C) server might potentially facilitate the detection and mitigation of these threats [17]. A centralized botnet attack is an example of a centralized execution attack, as shown in Fig. 5.
- *Decentralized/Peer to Peer Attacks*: Decentralized attacks entail a dispersed and autonomous strategy, wherein each hacked IoT device operates independently to accomplish the goals of the attacker. In the given context, the devices have the capability to establish direct communication with one another or autonomously make decisions relying on their own programming. The detection and mitigation of decentralized attacks pose more challenges due to the absence of a centralized point of control that may be specifically targeted. Fig. 6 shows the execution architecture of decentralized/peer to peer attacks.

A comparison of C&C and P2P architecture is illustrated in Table 13.

7.4. Software surfaces

- *Operating System Attacks*: Operating system (OS) attacks within the realm of the IoT pertain to instances of security breaches and vulnerabilities that explicitly aim at compromising the operating systems that are operational on IoT devices. IoT devices frequently

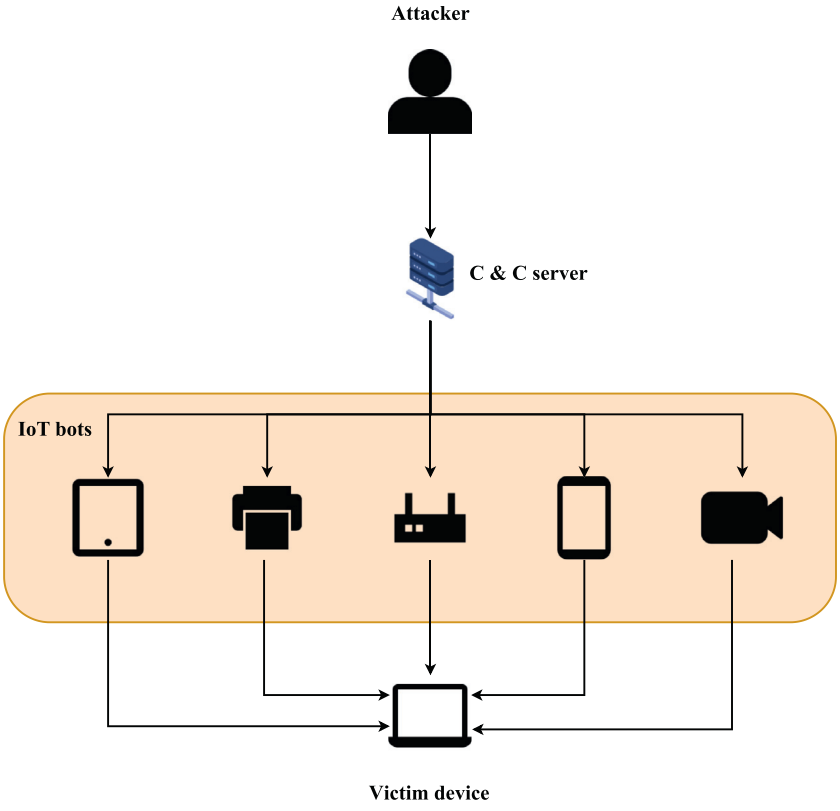

Fig. 5. C&C botnet architecture.

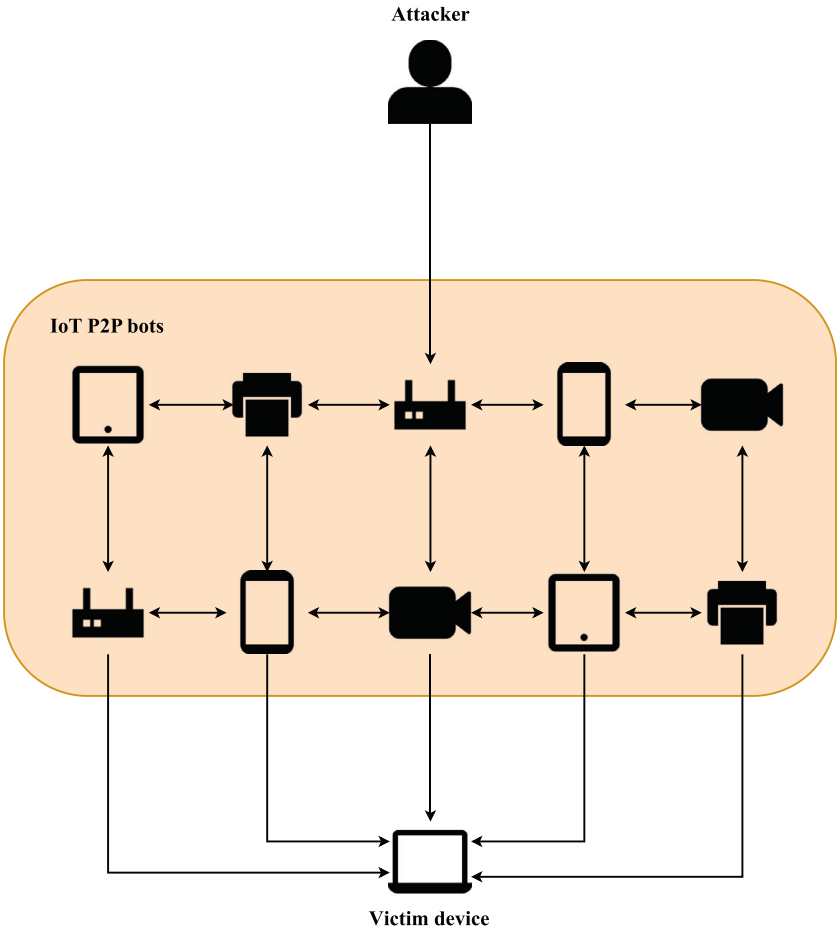

Fig. 6. P2P botnet architecture.

Table 13
Comparison on C&C and P2P architecture.

	C&C architecture	P2P architecture
Mode of distribution	Centralized	Decentralized
Potential actions	Easy to take down once discovered	Hard to take down
Bots information	Must contain the list of all bots in the network	Contains the list of neighbouring peers only
Anomaly detection	Easy to discover malicious traffic due to the unusual traffic size	Hard to distinguish malicious and legitimate traffic

exhibit resource limitations, encompassing restricted computational capabilities, memory capacity, and security attributes. This renders them susceptible to a multitude of attacks, encompassing those that capitalize on vulnerabilities inside their respective operating systems. IoT devices have compact physical dimensions, are characterized by a substantial quantity, possess limited storage capacity, and consume minimal energy. There are two distinct categories of devices: high-end and low-end. High-performance devices, such as smartphones and Raspberry Pi, have enhanced computational capabilities and power efficiency, enabling them to use conventional operating systems such as Linux effectively [107].

The prevalence of Linux as the primary operating system in the IoT gateway device industry has been noted, with over 70% of IoT devices utilizing this platform. Nevertheless, low-end devices have significant limitations regarding available resources, rendering them incapable of effectively operating standard operating systems. Hence, it is imperative to establish a de facto standard operating system (OS) specifically designed for resource-constrained IoT devices. This OS should be capable of accommodating diverse applications and operational demands inside a heterogeneous network (HetNet) [107].
Creating, implementing, and maintaining large-scale IoT software requires a suitable operating system (OS). IoT devices often rely on battery power as their primary energy source and possess a memory capacity of around 100 kilobytes (KB). The devices are equipped with 8-bit microcontrollers, significantly less than the processing power seen in contemporary PCs and laptops running Windows,

Unix, or Mac operating systems. The constraints associated with IoT devices necessitate the development of a system that is efficient, lightweight, portable, and versatile while also having minimal RAM and ROM requirements [107].

Microsoft, Linux, Arrayent, ARM, IFTTT, and several other contenders are actively developing and designing IoT operating systems. Prominent suppliers are increasingly prioritizing developing and implementing real-time operating systems (RTOS) and fully functional operating systems (FFOS) powered by compact OSs. The embedded operating system (OS) business has exhibited a degree of stability in recent years, with the coexistence of real-time operating systems (RTOS) and full-function operating systems (FFOS) in parallel marketplaces [107].

Presently, the IoT operating system (OS) industry is undergoing a rapid paradigm shift due to the maturation of Linux and the necessity to reevaluate the creation of value and generation of money. Two distinct categories of operating systems (OSs) for IoT devices exist: open-source and closed-source. The open-source operating systems encompass a range of options, such as Contiki, RIOT, FreeRTOS, TinyOS, OpenWSNnuttX, eCos, mbedOS, L4 microkernel family, uClinux, Android, Brillo etc. The closed-source operating systems (OSs) encompass a range of options such as ThreadX, QNX, VxWorks, Wind River Rocket, PikeOS, embOS, Nucleus RTOS, Sciopta, Hawei LiteOS, etc.

The lack of a dominating operating system (OS) in the IoT field might be attributed to the developing nature of this specialism, despite the concept itself having existed for some time. There is a scarcity of professionals in the field of the IoT, with a significant number of researchers, developers, and enterprises still in the early stages of acquiring knowledge and expertise in this domain [107]. Several prevalent operating system attacks exist inside the IoT domain. Some of them are:

- *Remote Code Execution (RCE)*: As mentioned in section in Section 7.1.3.
- *Denial of Service (DoS) and Distributed Denial of Service (DDoS)*:Section 7.1.2.
- *Memory Corruption Attacks*: Memory corruption attacks can be categorized into spatial and temporal memory corruption attacks. Spatial corruption occurs when a program accesses a memory buffer outside its designated boundaries. On the other hand, temporal corruption occurs when an application accesses memory either before it has been initialized or after it has been released [108].

  One instance of spatial corruption may be observed in the well-known buffer overflow vulnerability when the absence of a boundary check results in the corruption of contiguous memory. Instances of temporal corruption encompass uninitialized memory and use-after-free vulnerabilities. The program accesses a memory value in the above-case scenario without undergoing appropriate setup or verification processes. If the attacker can establish the memory value before its retrieval, the program executes its calculations using an invalid value [108].

  The consequences of this action vary depending on the purpose for which the value is employed, perhaps resulting in additional memory corruption or the acquisition of control over the application. The latter scenario pertains to situations where the program retains several references to a memory buffer created and subsequently releases it without invalidating all references. Use-after-free vulnerabilities are prevalent in C++ applications that adhere to a modular design, such as contemporary web browsers, document viewers, and office applications [108]. Some of the memory corruption attacks are:

  * *Buffer Overflow*: A buffer overflow is a type of software vulnerability that arises when a program attempts to write data beyond the allocated bounds of a buffer, resulting in the overwriting of neighbouring memory addresses. Errors of this nature sometimes arise throughout the coding process, often due to limited acquaintance with the programming language or a lack of attentiveness to specific aspects. Programming languages like C/C++ lack automated memory management features, such as dynamic bounds checking and automatic buffer resizing [109].

    String manipulation methods such as strcpy and gets, because of their lack of sufficient string length checking, have the potential to be susceptible to buffer overflow vulnerabilities. Other library methods, such as memcpy, can be susceptible when provided with improper arguments and zero-sized mallocs [109].
  * *Integer Overflow*: The issue of integer overflow, which is often referred to as wraparound, arises when the result of an arithmetic operation exceeds the memory space available for storing the numeric value or beyond the range limitations of the integer value provided. In most programming languages, integers are typically assigned a finite number of bits for storage [110].

    As an illustration, consider a 16-bit integer variable capable of storing either an unsigned integer within the range of 0∼65,535 or a signed integer within the scope of −32,768 to 32,767. During an arithmetic operation, if the outcome exceeds the designated storage capacity, such as in the case of $65{,}535 + 1$, the compiler has the potential to one possible course of action is to disregard the error entirely or terminate the application. Most compilers tend to ignore overflow occurrences, resulting in the storage of unexpected output or errors. The consequences of this action may manifest in a range of attacks, including but not limited to buffer overflow, which is widely recognized as the most prevalent form of attack. Such attacks can result in the execution of malicious programs or the unauthorized elevation of privileges [110].

    According to the 2020 CWE top 25 most dangerous software weaknesses, integer overflow is ranked as the 11th most hazardous software flaw and is identified by the CWE-190 identifier. This vulnerability is classified as critical due to its straightforward identification and susceptibility to exploitation. Exploitation can result in the entire subjugation of a system, unauthorized acquisition of data, extraction of data from a system, or disruption of proper program functionality. This issue is commonly encountered during the implementation phase of the software development lifecycle [110].

    The common vulnerability exposure (CVE) database has a substantial number of vulnerabilities, namely 1113, linked to integer overflow attacks. Among the numerous vulnerabilities identified in the common vulnerabilities and exposures (CVE) database, buffer overflow attacks emerge as a prevalent and favoured technique employed by hackers and adversaries [110].

    Buffer overflows result from inadequate sanitization or validation of user input by a developer before allocating memory space for it within a buffer. The occurrence of integer overflow results in the activation of a buffer overflow vulnerability, enabling the

attacker to acquire a shell and then raise their privileges upon successfully exploiting such vulnerability. The integer overflow vulnerability effectively turns off the validation checks, leading to a buffer overflow during execution [110].

* *Format String Attacks*: As mentioned in Section 7.1.3.
* *Stack Overflow*: A stack refers to a continuous segment of memory that functions utilize to store data for processing purposes. Multiple operations are specified on stacks. Stack overflows, specifically classified as CWE-121, pertain to a subset of the buffer overflow vulnerability (refer to K69961311). These vulnerabilities have the potential to impact applications developed in various programming languages [111].

  The term "stack overflow" describes instances where software endeavours to transfer data from a particular memory location to a fixed-length buffer allocated on the stack. However, the allocated buffer is insufficient in size to accommodate the entirety of the data. There are two crucial instructions within the context of stack manipulation, namely "PUSH" and "POP". The former is responsible for placing data into the stack, while the latter facilitates the removal of data from the stack [109,111].

  The system operates on a last-in, first-out (LIFO) principle and grows toward lower memory addresses, as in Intel, Motorola, SPARC, and MIPS-based systems. In every stack, there exists a register known as the stack pointer (SP), which is capable of indicating the position of the topmost or the adjacent free memory region, depending on the specific implementation of the stack. The stack comprises frames placed onto the stack when a function is invoked and subsequently popped off when the execution is completed. Theoretically, a local variable can be accessed by providing offsets relative to the stack pointer (SP) [109,111].

  Nevertheless, the corresponding offsets undergo alteration when the number of PUSH and POP operations increases. Therefore, the task of monitoring the number of words necessitates the execution of many commands or meticulous oversight. Therefore, to provide expedited access to variables, frame pointers are employed. The frame pointer is a register that serves the purpose of indicating a static position inside a frame. The displacement of the variables from the frame pointer remains constant regardless of the occurrence of push and pop operations [109,111].
* *Heap Overflow*: Heap overflows, also known as CWE-122, are a specific type of vulnerability known as buffer overflow. This vulnerability has the potential to impact applications developed in various programming languages. "Heap overflow" refers to instances where the software attempts to transfer data from a specific memory location to a fixed-length buffer allocated on the heap. However, the allocated buffer is insufficient in size to accommodate the entirety of the data being moved [112].

  Heap overflow is analogous to stack overflow when an adversary can access or modify data outside the allocated memory buffer. One significant distinction is that a heap data structure does not retain a stored pointer that may be overwritten, rendering it more challenging to exploit through malicious attacks. Overwriting a piece of stored information in a heap or overwriting it with vulnerable data is relatively straightforward, but it cannot be categorized as a general class of attacks. A doubly linked list facilitates the heap's management of free memory chunks. Every assigned heap chunk contains metadata to manage the heap [109,112].

– *Privilege Escalation Attacks*: A privilege escalation attack refers to a type of cyberattack specifically devised to obtain unauthorized access to privileged levels of a system. Attackers leverage human habits, design weaknesses, or oversights in operating systems or online apps to carry out their attacks. This phenomenon strongly correlates with lateral movement strategies employed by cyber attackers, wherein they navigate across a network intending to access valuable assets of significant importance. The outcome entails the presence of a user, either internal or external, who possesses system rights without proper authorization. The potential harm caused by malicious individuals might vary in severity depending on the scope of the breach. The present scenario might involve an unsanctioned electronic communication or a malicious software attack targeting enormous amounts of information. If left unnoticed, attacks have the potential to give rise to advanced persistent threats (APTs) targeting operating systems [113–115].

  Typically, adversaries initiate privilege escalation by employing a social engineering strategy that hinges on manipulating human behaviour. One of the most fundamental cyber threats is phishing, which refers to disseminating electronic messages that include malicious links. When an attacker successfully infiltrates one individual's account, the entirety of the network becomes vulnerable. Attackers actively seek vulnerabilities inside an organization's defensive measures, which enable them to get initial access or obtain fundamental rights by means of credential theft. As elaborated upon further, the exploitation of these vulnerabilities facilitates the attainment of heightened privileges. Therefore, an effective plan must incorporate prevention, detection, and prompt intervention tactics [113].

  Privilege escalation techniques can be done either locally or remotely. The initiation of local privilege escalation often occurs within the premises, frequently involving an individual who is affiliated with the organization. The initiation of remote escalation is possible from virtually any location. Both approaches can be successful for an attacker who is determined [113].

  Attacks may be classified into two main categories:

  Horizontal privilege escalation, also known as account takeover, refers to the unauthorized acquisition of elevated privileges by an attacker who gets access to a conventional user account with lower-level rights. The potential attacker can acquire an employee's login credentials, therefore obtaining unauthorized entry to email accounts, files, and any web-based applications or interconnected networks associated with those credentials. Once the attacker has established an initial position, they can traverse the network horizontally, progressively extending their level of authorized access to other accounts with comparable privileges [113,115].

  Vertical privilege escalation, also known as privilege elevation, starts in a similar manner when an attacker utilizes a foothold to initiate a vertical escalation, therefore acquiring unauthorized access to accounts endowed with elevated privileges. As an illustration, potential targets may include accounts endowed with elevated rights, such as those possessed by IT helpdesk personnel or system administrators. A privileged account possesses the capability to infiltrate and compromise other accounts. The continuous

evolution of cyberattack techniques has led to the emergence of many methods for unauthorized access and system penetration. However, it is worth noting that phishing remains prominent in this regard [113,115].

The individuals responsible for crafting these fraudulent communications, whether they adopt a wide and indiscriminate approach or meticulously focus on specific targets, aim to deceive users into divulging their login information, install malicious software, or compromise network security by granting unauthorized access. Both Windows servers and Linux operating systems are susceptible to security breaches. Windows privilege escalation frequently utilizes token manipulation, bypassing user account control or hijacking dynamic link libraries (DLLs). Various methods for escalating privileges on Linux systems, such as enumeration, kernel exploitation, and using sudo capabilities to acquire root access, are often seen. The acquisition of stolen credentials grants significant privileges, incentivizing attackers to actively seek novel methods for escalating Linux privileges [113].

– *Kernel Exploits*: A kernel exploit pertains to a kind of security vulnerability and attack that specifically focuses on compromising the kernel component of an operating system. The kernel serves as the central component of an operating system and is responsible for managing system resources, hardware control, and provision of vital services to other software components. The act of exploiting the kernel has the potential to result in illegal access, privilege escalation, and the compromise of the system's overall security [108].

  Kernel exploits often entail exploiting faults, vulnerabilities, or deficiencies present in the code of the kernel. The vulnerabilities encompass a range of sorts, such as buffer overflows, use-after-free, race situations, and others. Upon successfully exploiting a kernel vulnerability, the attacker can acquire elevated privileges and execute arbitrary code within the kernel space. This capability enables someone to circumvent security safeguards, deploy malicious software, get entry to confidential information, and potentially assume full control of the entire system [108].

  Kernel exploits are well acknowledged for their significant potency and inherent peril since they provide unauthorized access to the kernel, which is the most pivotal component of an operating system. Operating system developers and security researchers consistently collaborate to detect and rectify these flaws to mitigate such attacks. It is highly recommended that users maintain their systems by regularly updating them with the most recent security patches and upgrades. This practice helps to mitigate the potential risks associated with kernel exploits and other security vulnerabilities [108].

  One instance of a kernel exploit is represented by the common vulnerabilities and exposures identifier CVE-2017-1000251. The Linux Kernel's native Bluetooth stack, known as BlueZ, exhibits a stack overflow vulnerability inside the L2CAP configuration response processing. This vulnerability is present in Linux kernel versions 2.6.32 through 4.13.1, and it can potentially lead to the execution of remote code within the kernel space [116].

  In CVE-2017-1000410, the Linux kernel, starting from version 3.3-rc1 and onwards, is susceptible to a vulnerability that resides in the handling of incoming L2CAP instructions, specifically ConfigRequest and ConfigResponse messages. This information leak may be attributed to the presence of uninitialized stack variables, which can be accessed by an attacker in their uninitialized state. Through the manipulation of code flows preceding the handling of configuration messages, an assailant can have a certain level of influence on the data that will be stored in uninitialized stack variables. This technique enables the circumvention of kernel address space layout randomization (KASLR) and stack canary protection mechanisms, as the exposure of both pointers and stack canaries can occur using this method. The potential exploitation of the remote code execution (RCE) vulnerability in L2CAP configuration parsing (CVE-2017-1000251) can be enhanced by using the aforementioned vulnerability. This combination has the potential to enable attackers to exploit the RCE against kernels that have implemented the aforementioned mitigations [108,117]. The following are the details about this vulnerability: The variable in question is declared without initialization in the l2cap_-parse_conf_rsp and l2cap_parse_conf_req functions. The variable “efs” is an instance of the structure “l2cap_conf_efs”. Furthermore, during the processing of input configuration parameters in each of these procedures, it is possible for the switch case responsible for handling EFS elements to bypass the memcpy call, which is responsible for writing to the efs variable [116,117].

  In the given code snippet, there is a conditional statement that checks if the length of the input data, denoted as Olen, is equal to the size of the efs variable. If this condition is satisfied, the efs variable is copied from the input data using the memcpy function. It is important to note that the value of open is determined by the attacker and may be controlled by them. Despite the conditional check, both of these functions ultimately include the efs variable in the outgoing configuration request that is being constructed. The function l2cap_add_conf_opt is invoked with the parameters &ptr, L2CAP_CONF_EFS, sizeof(efs), and (unsigned long) &efs. By transmitting a configuration request or response that includes an L2CAP_CONF_EFS element, but with an element length that does not match the size of the efs variable, it is possible to prevent the memcpy operation from copying data to the uninitialized efs variable. Consequently, the uninitialized variable would be exposed to the attacker, resulting in the disclosure of 16 bytes of data [116,117].

– *IoT Botnets*: An IoT botnet refers to a network of IoT devices, primarily routers, which have been compromised by malware, particularly designed for IoT botnet purposes, and are now under the command of malevolent individuals. IoT botnets have gained notoriety for its utilization in orchestrating distributed denial-of-service (DDoS) attacks on specific targets, with the intention of causing disruption to their operations and services. The pivotal function of routers within networks additionally presents more prospects for malicious entities to use IoT botnets in carrying out more detrimental operations. The accessibility of IoT botnets to cybercriminals is evident from their advertisement on underground forums [118].

  The efficacy of a botnet is mostly derived from the quantity of devices comprising it. The utilization of botnet malware families to infiltrate devices underscores the inherent nature of the danger. These malware families are specifically engineered to acquire a substantial number of devices while safeguarding against competing botnet software. In general, botnets are managed by a singular command and control (C&C) server, which establishes connections with all the compromised devices, commonly called

"bots". However, the utilization of peer-to-peer (P2P) networking inside certain botnets obviates the necessity for a command and control (C&C) server, hence augmenting the challenge of dismantling them [118].
  - *Zero-Day Exploits*: As mentioned in Section 7.1.2.
  - *Man-in-the-Middle (MitM) Attacks*: As mentioned in Section 7.1.3.
- *Firmware Attacks*: A firmware attack refers to infiltrating malicious code into a device by exploiting a backdoor in the processor's software. Backdoors refer to certain pathways inside the code that enable select persons to circumvent security measures and gain unauthorized access to the system. The backdoor typically evades detection as a result of its intricate nature, although its exploitation by hackers can lead to significant repercussions [29].
  One prevalent illustration of a firmware attack is the occurrence of an unsanctioned upgrade on a computer or mobile device, leading to the introduction of malicious software or other forms of cybercriminal behaviour. This phenomenon occurs due to the inclusion of undisclosed features or functions in several updates, which might serve malicious purposes such as surreptitiously intercepting data and disabling essential functionality. These updates maintain a facade of innocence, disguising their true intentions [29].
- *Application Software Attacks*: As mentioned in Section 7.1.3.

7.5. IoT protocols

Before delving into prevalent IoT protocols, it is important to establish a comprehensive definition of the term "protocol". Protocols encompass a collection of regulations governing the transmission of data among electronic devices, according to predetermined agreements pertaining to information organization and the manner in which data is exchanged between parties involved. Similarly, IoT protocols encompass a set of established norms that facilitate the interchange and conveyance of data between the Internet and peripheral devices. The categorization of IoT protocols may be delineated into three distinct groups [119].

- *IoT Communication Protocols (A1)*: IoT communication protocols encompass a collection of rules and standards that dictate the manner in which devices and systems inside the IoT network interact and exchange information. These protocols provide the smooth interchange, control, and synchronization of data among diverse devices, sensors, and applications. Communication protocols in the IoT play a crucial role in governing the formatting, transmission, and reception of data. These protocols are designed to account for several considerations, including energy efficiency, data size, latency, and dependability [120].
- *IoT Connectivity Protocols (A2)*: Connectivity protocols encompass a specialized category of communication protocols specially designed to facilitate the connection between IoT devices and networks. The primary objective of these protocols is to facilitate the establishment and sustenance of connections between devices and the internet or other nearby networks. Connectivity protocols are paramount in facilitating the reliable and effective transmission and reception of data across devices [120].
- *IoT Network Protocols (A3)*: Network protocols refer to a comprehensive range of protocols that govern the regulations and conventions employed for data transmission and establishing routes inside networks, encompassing IoT networks. The aforementioned protocols provide the parameters for data transmission among network devices, encompassing aspects such as addressing, error control, and other network administration responsibilities [120].

IoT networks provide connectivity across diverse objects, facilitating seamless communication, data sharing, and collaboration. When developing an IoT application, the initial decision pertains to selecting a network. There are several categories of IoT networks, each with distinct attributes, functionalities, and use cases. IoT networks may be categorized into four distinct kinds as follows as shown in Fig. 7 and Table 14 [122–126].

- *LAN/PAN (B2)*: Personal area networks (PANs) and local area networks (LANs) are types of networks that have restricted geographical coverage. While the cost-effectiveness of data transit across PAN and LAN networks is often perceived, its reliability is not always guaranteed. WiFi and Bluetooth are wireless technologies often employed in IoT connection solutions, facilitating personal and local area network communication. The integration of many access points into a larger network enables the utilization of WiFi for both local and distributed applications. If the gadget does not consistently receive data, a solitary battery powered by Bluetooth low energy (BLE) can endure for up to five years. The Bluetooth low energy (BLE) wireless network protocol is characterized by its superior energy efficiency [127].
- *Low Power Wide Area Networks (LPWAN) (B3)*: IoT devices that utilize low power wide area networks (LPWANs) often transfer little data packets, infrequently and often over considerable distances. The development of this particular wireless network was initiated as a direct reaction to the initial challenges encountered in the field of cellular communication. LPWAN is promoted as possessing a superior coverage area compared to WiFi and Bluetooth, while also exhibiting lower power consumption than cellular networks [127].
  The LoRaWAN protocol, which utilizes the LoRa (Long-Range) communication network, is a well-recognized and extensively employed IoT network protocol within this domain. LoRaWAN offers advantages for IoT devices, such as decreased power consumption, resulting in extended battery life and cost-effective chipsets. Under optimal conditions, a solitary base station or gateway functioning inside a long-range network can provide connectivity to an extensive region, spanning many kilometres, particularly in densely populated urban areas [127].
- *MeshpProtocols (B4)*: The characterization of mesh networks is most effectively achieved by analyzing their connection configuration, which pertains to how the various components communicate with one other. Within the context of mesh networks, the

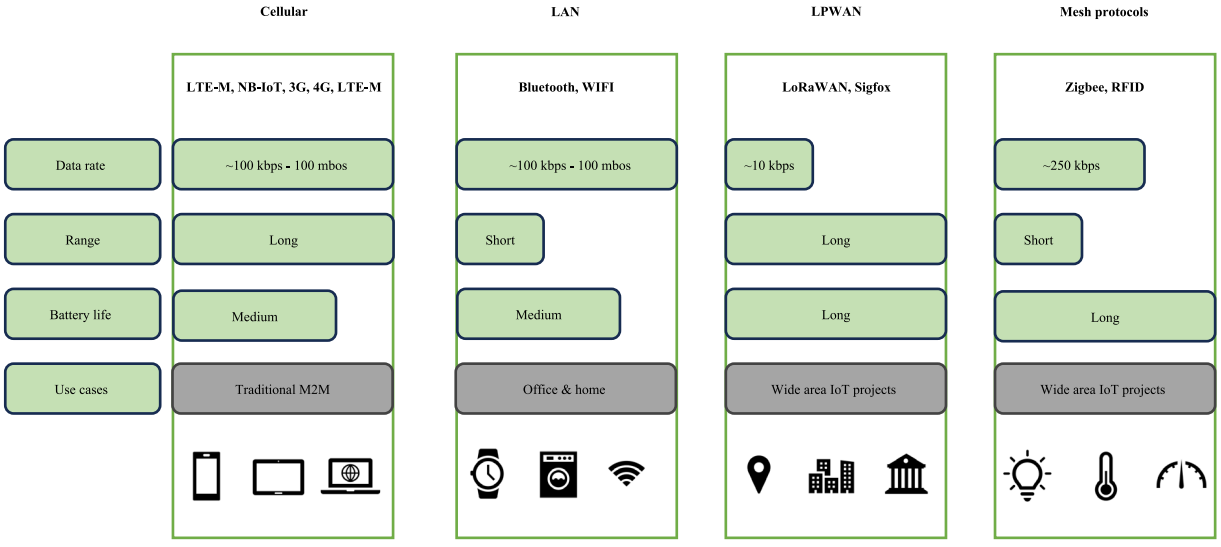

Fig. 7. Comparison of IoT Protocols based on IoT Networks.

Table 14
Relationship between IoT networks, protocols, and OSI layers (new similar diagram).

IoT Network	IoT Protocols	OSI Layer(s)
Cellular	MQTT, CoAP, HTTP, MQTT-SN	Application (7)
	TCP/IP, UDP/IP	Transport (4)
	LTE-M, NB-IoT	Physical (1)
LAN/PAN	MQTT, CoAP, HTTP, WebSocket	Application (7)
	TCP/IP, UDP/IP	Transport (4)
	Ethernet	Data link (2)
Low power wide area networks (LPWAN)	LoRaWAN, Sigfox	Data link (2)
	CoAP, MQTT-SN	Application (7)
	LoRaWAN MAC	Data link (2)
Mesh network	Zigbee, Thread	Network (3)
	6LoWPAN, Bluetooth mesh	Network (3)
	Z-Wave	Physical (1)
	MQTT, CoAP, HTTP	Application (7)
	IEEE 802.15.4 (underlying layer)	Data link (2)

collaborative efforts of sensor nodes are directed towards the mutual exchange of data among themselves, with the ultimate objective of transmitting this data to the gateway. Zigbee may be identified as an example of an IoT wireless network technology. Mesh networks have a constrained spatial reach, necessitating the deployment of additional sensors within a building or the utilization of repeaters to get the desired coverage for a given application. Moreover, the manner in which these networks engage with each other might result in an excessive utilization of power, particularly when aiming for rapid communication, as exemplified by an application designed for intelligent lighting. Mesh networks are commonly utilized due to their high level of resilience, ability to efficiently identify optimal data transmission routes that are both fast and dependable, and ease of installation.
The Fig. 8 shows the relationship between IoT protocols and OSI layer.

7.6. Attacks based on device property

Devices may be classified into two categories based on their properties: low-end devices and high-end devices. These attacks affect the IoT system. The implementation of the IoT has the potential to lead to critical system failures or instances of anomalous behaviour, which may be attributed to the inherent characteristics of the devices involved [128,129].

- *Attack on Low-End Device Class*: The attack on the low-end device class refers to a malicious activity wherein low-power devices target and compromise the IoT system. In this regard, the cost of this class is minimized since it just relies on establishing a connection between the system and external entities over a radio link. Both the potential and network configuration exhibit comparable behaviour. The accessibility of IoT device sensor nodes is limited to a select few. As an illustration, the wristwatch can remotely manage household equipment such as smart TVs and refrigerators [128,129]. Sweyntooth refers to a number of Bluetooth low energy (BLE) technology flaws that a group of researchers from the SUTD asset research division found. Currently, it has a set of 18 vulnerabilities. The vulnerabilities exist in several BLE software development kits (SDKs) of prominent systems on chips (SoCs),

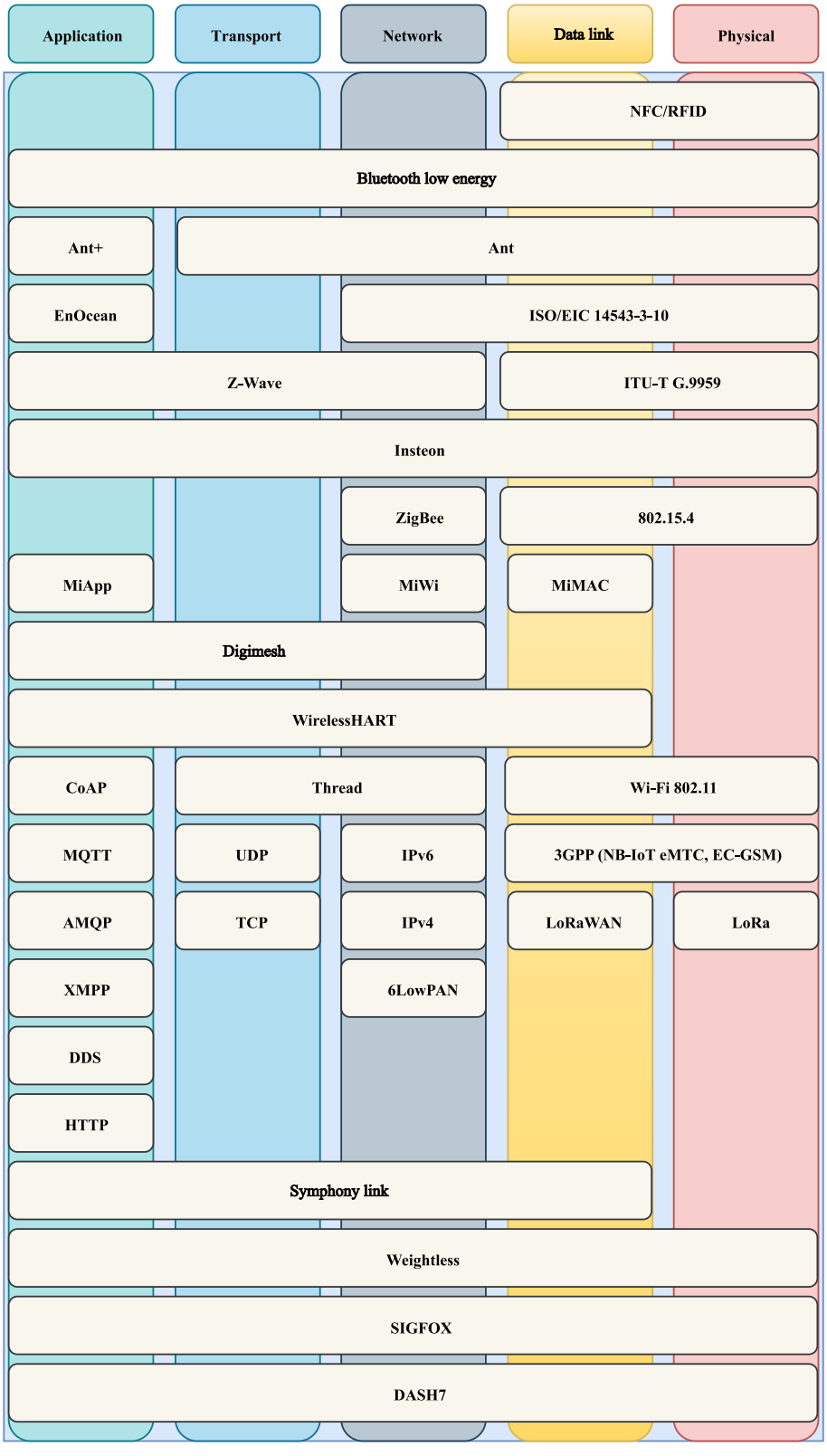

Fig. 8. Relationship between IoT protocols and OSI layers [121].

which expose flaws in the BLE implementation. Depending on the specific apps being used, this vulnerability enables an adversary within the radio range to intentionally cause crashes, deadlock, buffer overflows, security bypasses, and other related issues. It impacted many IoT devices with a vulnerable BLE stack from many manufacturers [130].

- *Attack on High-End Device Class*: Unlike low-end devices, high-end device class attacks utilize fully equipped devices to initiate attacks against IoT systems. This class establishes connectivity between IoT devices, enabling them to be accessed remotely by a high-performance laptop with superior CPU capabilities, regardless of location or time [128,129]. CNN reported in 2017 that "The FDA confirmed that St. Jude Medical's implantable cardiac devices have vulnerabilities that could allow a hacker to access a device. Once in, they could deplete the battery or administer incorrect pacing or shocks, the FDA said" [131].

7.7. Attacks based on adversary location

An adversary can launch attacks on the IoT system from any location. Insider or outsider attackers refer to different types of attacks that are distinguished by the attacker's location [128,129].

- *Internal Attacks*: An internal attack refers to a security breach initiated by a component situated within the perimeter of the IoT security framework, commonly known as an "insider". To initiate the attack, the perpetrator endeavours to execute their malevolent malware targeting IoT devices. Insider attacks may be classified into four distinct types: compromised actors, inadvertent actors, emotional attackers, and technology perception actors [128,129]. A day after departing, in November 2021, a former employee at the South Georgia Medical Center in Valdosta, Georgia, transferred confidential information from the facility's computers to his USB drive for no apparent reason. This is an example of a malicious insider threat, where the insider intended to damage the company for personal gain or because they were irate or unhappy [132].
- *External Attacks*: External attacks refer to instances where adversaries positioned outside the IoT network (public) can remotely access IoT devices. The individuals lack knowledge of the architecture of the IoT that they are attempting to get access to. Typically, an iterative trial and error methodology is employed to facilitate effective connectivity inside the appropriate IoT native network [128,129]. Cybercriminals attacked Mailchimp and its partners in 2022, resulting in many attacks. Malicious actors were able to successfully execute a phishing attack in January 2023, fooling at least one Mailchimp employee into disclosing their credentials [132].

7.8. Attacks based on information damage level

IoT devices are sensors responsible for monitoring the fluctuations in various parameters. The vulnerability of information to manipulation by malicious actors is a significant concern. The information damage level is classified into six distinct categories:

- *Disruption Attacks*: Disruption attacks primarily focus on compromising the availability of a system. Resource depletion is a potential consequence that may occur as a result of an interruption being initiated. During the operation of the Internet of Things (IoT), it is possible for disruptions to occur, resulting in the IoT device transitioning into a state of shutdown [128].
- *Eavesdropping*: As mentioned in Section 7.1.2.
- *Modification Attack*: Altering or modifying information within IoT devices poses a significant danger to the integrity of security requirements for IoT systems, hindering their ability to perform effectively according to their intended design. The motivation behind the perpetrators engaging in such discourteous behaviour is to manipulate the communication protocol to deceive [133].
- *Fabrication Attack*: The fabrication attack poses a challenge to the authentication of IoT systems since attackers can introduce counterfeit data into the standard architecture of the IoT. The deliberate inundation of the IoT network by the perpetrator results in the degradation of the informational integrity of the IoT device [54].
- *Message Replay Attack*: As mentioned in Section 7.1.2.
- *Man-in-the-Middle Attack*: As mentioned in Section 7.1.3.

8. Comparison between IoT attack datasets

The CICIoT 2023 dataset is a comprehensive and up-to-date dataset and benchmark that focuses on the occurrence of large-scale attacks inside the IoT ecosystem. The dataset is a comprehensive and substantial IoT attack data collection. Its purpose is to facilitate the advancement of security analytics applications inside real-world IoT operations. To achieve this objective, 33 attacks are carried out within an IoT architecture comprising 105 interconnected devices. The aforementioned attacks may be categorized into seven distinct classifications, specifically DDoS (Distributed Denial of Service), DoS (Denial of Service), Recon (Reconnaissance), Web-based, Brute Force, Spoofing, and Mirai [57].

Ultimately, malicious IoT devices perpetrate all attacks to compromise additional IoT devices. The dataset known as CICIoT2023 is accessible in two distinct file formats, pcap and csv. Pcap files comprise the primary data created and gathered inside the CIC IoT network across various situations. The aforementioned files encompass the whole of sent packets and possess the potential to be utilized for the extraction and manipulation of additional attributes. In addition, csv files offer a more straightforward approach to importing and using the data [57]. Table 15 compares the publicly available IoT attack datasets.

Table 15
Comparison of IoT attack datasets.

Dataset	Domain	Availability	Kind	Features	Label	No of devices	No of attacks	Size (in GB)
N-BaIoT (2018) [134]	Home	Yes	Real	Yes	Yes	9	10	2 GB
BoT-IoT (2019) [135]	Home	Yes	Simulated	Yes	Yes	5	6	69.3 GB
Kitsune (2019) [136]	Home	Yes	Simulated	Yes	Yes	9	9	19 GB
IoTNIDS (2019) [137]	Home	Yes	Real	No	Yes	2	12	0.55 GB
MedBIoT (2020) [138]	Home	Yes	Real/simulated	Yes	Yes	7	3	10 GB
IoT-23 (2020) [139]	Home	Yes	Real	No	Yes	3	20	21 GB
IoTIDS (2020) [140]	Home	Yes	Real	Yes	Yes	2	2	0.55 GB
MQTT (2020) [141]	Home/Office	Yes	Simulated	Yes	Yes	8	10	0.99 GB
MQTT-IoT-IDS (2020) [141]	Home	Yes	Simulated	Yes	Yes	1	5	1.35 GB
TON IoT (2020)	Home/Industry	Yes	Simulated	Yes	Yes	10	9	65 GB
Edge-IIoTSet (2022) [142]	Industry	Yes	Real	Yes	Yes	10	14	1.48 GB
CIC IoT Attack 2023	Home/Industry/ Office	Yes	Real	Yes	Yes	105	33	548 GB

The CICIoT2023 dataset expands upon current knowledge in IoT security by incorporating a comprehensive network structure that includes a diverse range of IoT devices. It also introduces various attacks that have not been previously included in a single IoT security dataset. Additionally, the dataset evaluates the performance of commonly used machine learning (ML) techniques in different classification scenarios [57].

9. IoT attacks detection methods

Identifying and mitigating these attacks are of utmost importance in safeguarding the security of IoT ecosystems. Numerous approaches for detecting attacks have been devised to tackle these aforementioned issues. The subsequent techniques pertain to the detection of attacks in the IoT domain are:

9.1. Anomaly detection

An anomaly is a specific data point that deviates from the expected behaviour within a modelled system. Anomalies refer to infrequent occurrences or observations that exhibit notable deviations from established norms or patterns within a particular data point, contextual setting, or temporal segment (such as a season or quarter) or over the whole dataset. Anomalies, in essence, arise due to external factors, such as sensor malfunction or external attack. A detection algorithm's primary objective is to identify an anomaly's location and categorize or deduce its underlying cause [143].

Anomaly identification, sometimes called outlier or event detection, is the process of doing data analysis to identify atypical situations inside a given system. The anomaly detection algorithms check incoming traffic at several levels, including the IoT network level up to the data centre. The importance of anomaly detection rests in its ability to identify and analyze irregularities within IoT data. These anomalies, albeit infrequent, can provide valuable insights and actionable information across various industries such as healthcare, manufacturing, finance, transportation, and energy. Anomaly detection in the IoT is utilized within the betting and gambling industry to identify instances of insider trading through the analysis of trends in trade activity [143].

In contrast, industrial machinery employs a detection algorithm to guarantee the safety of production processes. Presently, most anomaly detection techniques used in the IoT domain need substantial human involvement and the customization of solutions at the local level. Understanding an anomaly is considered straightforward in principle, and given sufficient time, a domain expert can identify anomalous data [143].

Nevertheless, developing an automated model in an IoT context presents many challenges. Defining and categorizing all forms of anomalous data accurately can be difficult, mainly when there is little or no access to labelled training data. The concept of normal behaviour undergoes continuous transformation and development across all disciplines. An illustrative instance involves a modification in the number of individuals residing in a family, leading to a corresponding alteration in the electricity demand. In addition, it is common for data to be subject to noise. In situations when the ratio of signal to noise is low, the noise might closely resemble genuine abnormalities. The level of complexity escalates in proportion to the expansion of interconnected systems and the wide range of input data types [143].

The categorization of IoT anomaly detection methods is based on their approach to the problem, their application, the technique type, and the algorithm's latency. Fig. 9 offers a comprehensive summary of IoT anomaly detection techniques.

- *By Method*: There are three distinct methodologies for addressing the problem, including geometric, statistical, and machine learning techniques. Geometrical approaches are founded upon the underlying concept that the segregation of anticipated and anomalous data occurs when employing distance- and density-based strategies to describe a particular dataset. In a given collection of data points, isolation or density-based approaches operate on the premise that anomalies manifest in locations with low data density. The approaches employ either a static or dynamic threshold "t" on the estimated distance "d" to classify anomalies is given as [143]:

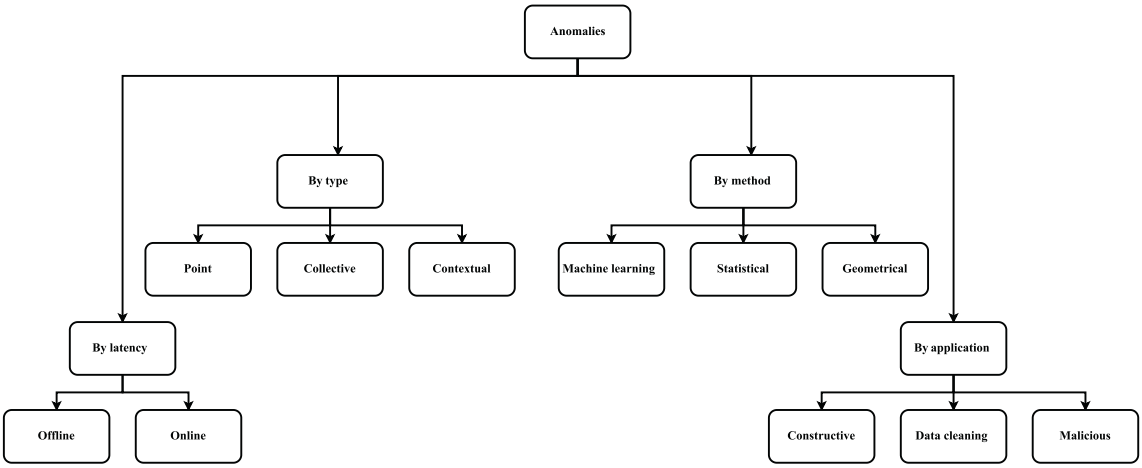

Fig. 9. IoT anomaly detection.

$$d = \left\{ \begin{matrix} < t, \text{Normal } (\text{under threshold}) \\ > t, \\ \text{Anomaly } (\text{above threshold}) \end{matrix} \right\} \tag{1}$$

Statistical methods, such as the minimum volume approach, aim to employ mathematical models and distributions to represent normal data. The purpose of the minimum volume technique is to construct an n-dimensional simplex that encompasses the provided data cloud, also known as the ground truth. The goal is to reduce the occupied volume of the simplex while maximizing the inclusion of the ground truth data points. An anomaly is any data point deviating from the expected pattern or distribution [143].

Another forecasting method that may be utilized is known as exponential smoothing. The proposed methodology uses historical data points and a smoothing parameter to predict future data points. Anomalous data, derived from statistical methods, refers to observations exhibiting deviations from the established model. A significant amount of academic research supports traditional geometric and statistical techniques and depends on a comprehensive comprehension of the true nature of the subject matter. These methodologies fail to acknowledge several real-world scenarios in which data models exhibit significant temporal dependencies. Hence, utilizing data-driven machine learning and deep learning methodologies becomes imperative to facilitate adaptable adjustments [143].

The frequency of publications on machine learning and deep learning models has grown notably in recent years, becoming the third subcategory. The model selection is contingent upon the characteristics of the provided data. For instance, sequential data inputs like audio, video, and time series are favoured by models like long short-term memory (LSTM) and transformers.

In contrast, the convolutional neural network (CNN) and autoencoder (AE) prefer non-sequential data formats, particularly those about visual inputs. The algorithms aim to differentiate between typical and atypical behaviour by defining a decision boundary, such as using the support vector machine (SVM) classifier or predicting future values in streaming data using LSTM networks. The categorization of these approaches is contingent upon the presence or absence of training labels, resulting in the classification of supervised, semi-supervised, self-supervised, or unsupervised methods [143].

- *By Application*: An application employs three distinct pathways for classifying anomalies: constructive, destructive, and data cleaning. Constructive applications are characterized by their productivity and capacity to bring positive outcomes, thereby contributing value to society. An example of such an application can be found in fall prevention for the elderly, as explored in the study by Ref. [144]. This research investigates the utilization of image descriptors to monitor the daily behaviour of elderly individuals, intending to prevent falls.

  The study further examines and compares the performance of three different classifiers, namely multilayer perceptron (MLP), K-nearest neighbours (KNN), and support vector machine (SVM). Additional examples of research in the field include a study conducted by researchers [145] that explores the use of reinforcement learning techniques in the context of unmanned aerial vehicle (UAV) applications, specifically focusing on its application in the domain of smart farming.

  Furthermore, another noteworthy study conducted by researchers [146] investigates the implementation of a federated learning strategy for anomaly detection in smart home applications. Destructive applications are designed to disrupt normal operations to get potentially illicit financial benefits, inflict harm upon the network and the application data that traverses the IoT network or impede crucial business procedures. These applications have a detrimental influence on society.

  For instance, the scholarly article authored by Ref. [147] examines several cyberattacks targeting the IoT and provides an overview of the most recent advancements in IoT security. These applications necessitate a thorough investigation of potential remedies, such as RAPPER (Ransomware Prevention via Performance Counters) and NBaIoT (Network-Based Detection of IoT Botnet Attacks Using Deep Autoencoders), employing an anomaly detection approach. This research addresses preventive measures, proactive activities implemented before an unauthorized occurrence, and the post-incident identification and response strategies employed.

In conclusion, data cleaning or data cleansing applications, such as DeepAnT (Deep Learning Approach for Unsupervised Anomaly Detection in Time Series) [148], employ deep convolutional neural networks to eliminate undesired data spikes and sensor noise present in the input signal.

- *By Anomaly Type*: One commonly observed type is the circumstance-specific type, which is the point, contextual, and collective. The deviation of a single data point from the anticipated pattern characterizes a point anomaly. One instance that exemplifies this concept is the identification and prevention of credit card fraud, as discussed in the scholarly article [149].
  A contextual anomaly refers to a single incident that may be deemed unusual within a certain context. This implies that comparing numerous views on a given data item does not consistently lead to abnormal or unexpected outcomes. A contextual anomaly is identified when the combined analysis of contextual and behavioural factors is undertaken. In traffic violations, the discrepancies seen are contingent upon the geographical location data [150].
  In contrast to a single point or contextual anomaly, the collective anomaly pertains to examining the complete dataset. One instance of a collective anomaly may be observed in the application of electrocardiograms (ECGs) for the purpose of monitoring and identifying abnormalities or malfunctions in the human cardiac system [151].
- *By Latency*: The determination of whether a detection algorithm is conducted in real-time during the data collecting stage or deferred to a subsequent storage step is contingent upon the algorithm's latency and scalability. An online algorithm can sequentially analyze data as individual data points or as a sliding window without accessing the entire input dataset. Traditional methods for analyzing geometric and statistical data in online settings encompass distance-based, density- and deviation-based, and angle-based methodologies. Two examples of online techniques include the IoT-Keeper developed by Ref. [152], which utilizes the fuzzy C-means algorithm, and the ensemble approach proposed by Ref. [153].
  In contrast, offline algorithms possess the ability to utilize comprehensive data. These algorithms are characterized by their high computational complexity and advanced nature, enabling them to address the problem within a feasible timeframe efficiently. Nevertheless, it is crucial to acknowledge that in recent times, algorithms such as the one proposed by the authors in the publication [154] utilize long short-term memory (LSTM) and Gaussian naive bayes techniques. Additionally, these algorithms, along with the aforementioned models, undertake the training phase of the model offline and afterwards deploy the model online.
  Various sources of anomalies include intrusion detection systems, fraud detection mechanisms, and data leakage detection systems [155].
- *Intrusion Detection*: Identifying illegal or malicious activity inside an IoT ecosystem is called Intrusion detection. This includes identifying illegal access to IoT devices, atypical data transmission patterns, or other activities that may jeopardize the integrity of IoT systems. To safeguard the security of IoT networks and devices, intrusion detection systems (IDS) employ various approaches, including anomaly detection and signature-based detection, to detect and identify possible threats and vulnerabilities.
- *Fraud Detection*: Fraud detection in the IoT context refers to identifying and mitigating fraudulent actions occurring inside IoT networks. This may include identifying deceptive sensor data, illegal intrusion into IoT devices, or fraudulent activities inside IoT apps. Fraud detection systems in the IoT context frequently depend on machine learning algorithms and real-time data analysis. These systems aim to identify atypical patterns or behaviors that may signify fraudulent activities, ensuring IoT systems' dependability and credibility.
- *Data Leakage*: Data leakage in the IoT context refers to the inadvertent or deliberate disclosure of confidential information or data originating from IoT devices or networks. These events might be attributed to insufficient implementation of security protocols, illegal entry into networks, or weaknesses in IoT systems.
  The phenomenon of data leakage presents substantial concerns in terms of privacy and security, as it has the potential to result in the unauthorized disclosure of sensitive data. Preventing data leakage in the IoT necessitates implementing several measures, such as encryption, access control, secure communication protocols, and routine security assessments. These measures are essential for ensuring the confidentiality and integrity of IoT data.
  Several significant issues and challenges arise when it comes to anomaly detection.
- *Missing Data Points*: The presence of missing data points can be attributed to the data loss caused by external factors, making it challenging to identify and quantify these missing values.
- *Data Corruption*: The corruption of data can be attributed to either external influences or device malfunction, resulting in the challenge of distinguishing between anomalous and corrupted data.
- *Encrypted Data*: Detecting anomalies in encrypted data is a significant challenge.
- *Sensor Fusion*: Sensor fusion presents challenges in collecting and aggregating data from diverse sensors.
- *Real-time Detection*: Real-time anomaly detection entails the analysis of high-speed streaming data, necessitating prompt and expedient responses.
- *Noisy Data*: Noise in electronic transmission necessitates the removal of such noise from edge computing devices in the IoT before sending the data to the cloud.
- *Traffic Surge*: A traffic surge might lead to the potential overload of the anomaly detection process due to the significant volume of data involved.
- *Multivariate Data*: When analyzing multivariate data, it is important to take into account the occurrence of frequent changes in the data.

9.2. Signature-based detection

Signature-based detection necessitates the proactive formulation of predetermined rules or signatures by security professionals to identify known attack patterns. This approach is particularly effective in detecting known attacks with signatures already in the database. Conversely, the absence of signatures within the database renders it incapable of identifying any unknown attacks. The stochastic gradient descent (SGD) algorithm has superior computational efficiency compared to alternative methods. However, it is limited in its ability to accurately detect and classify novel attack types that have not been previously encountered or included in its training data [156].

Signature-based techniques provide a notable advantage over anomaly-based methods due to their inherent simplicity and ability to function in real-time online environments. The method commonly referred to as signature-based detection is alternatively recognized as misuse detection. The misuse detection strategy involves the analysis of network events to identify known threats, often employing string-searching algorithms [156].

Misuse detection, also known as signature detection, is a technique that relies on a predetermined set of criteria. These rules, such as specific byte sequences in network traffic or known sequences of dangerous instructions used by malware, are put into a system and compared against observed events. When an event that raises suspicion is noticed, it triggers an alarm. This particular intrusion detection system (IDS) demonstrates effectiveness in identifying and mitigating known attacks. However, it cannot detect previously undiscovered zero-day attacks that lack established signatures. Signature-based detection is a favoured approach in the field of cyber security solutions due to its simplicity of implementation and effectiveness in recognizing known attacks. This method, as highlighted in a scholarly article on network intrusion detection for IoT security based on learning techniques, exhibits a high detection rate while maintaining a low false alarm rate [62].

9.3. Behavioural analysis

Dynamic code analysis is troubleshooting potentially malicious software inside a physical or virtual environment. The program's source code is executed using various test inputs to detect security vulnerabilities that may arise from its code while interacting with other programs or systems. Dynamic analysis is a methodology employed for the examination of IoT attack behaviours [157].

Using behaviour analysis for detecting IoT attacks presents several advantages compared to static analysis. Dynamic analysis can detect known and zero-day threats due to comparable behaviours shared by many attacks. Dynamic analysis is conducted by utilizing sandbox tools such as Cuckoo Sandbox or CWSandbox, which enable the observation of malware behaviours during runtime [157].

The dynamic analysis behavioural report provides a comprehensive examination of the behaviours exhibited by a malware executable file following its execution on the host system of the victim. These behaviours include downloading files contaminated with malware from the Internet, executing harmful operations within the operating system, deleting and generating files, exfiltrating sensitive data through remote access, denying users access to resources, or intentionally slowing down the network. Memory analysis can also be employed to research the behaviour of IoT attacks. Memory analysis is conducted by acquiring memory dumps of the compromised system's physical memory during execution [157].

The extraction of behavioural information pertaining to IoT attacks may be achieved by utilizing sophisticated tools like the volatility framework, which enables the analysis of memory dumps. Furthermore, it is infeasible for memory to conceal the previously indicated malevolent actions. For a program, whether benign or malicious, to be run, it must be loaded into memory.

Consequently, the technique of memory analysis is considered the most appropriate and favoured approach for detecting advanced IoT attack behaviours. The utilization of hybrid analysis presents itself as an alternative approach for investigating IoT attack behaviour. This approach combines the features of other analytical approaches outlined before. It is noteworthy that integrating several methodologies for analyzing IoT attacks might yield more insights into malware, hence enhancing detection capabilities [157].

The term “IoT attack analysis testbed” encompasses a range of settings utilized to observe and evaluate the behaviours shown by IoT attacks. IoT attacks should be studied inside a secure and isolated environment to mitigate potential harm during the analysis process [157]. The Table 16 compares various testbeds for IoT attacks.

- *Bare-Metal*: One method of examining the behaviours of an IoT attack is to attack a pristine system operating on physical hardware. This type of analysis environment for IoT attacks is sometimes called a “bare-metal” setup. Investigating IoT attacks on a bare-metal system may primarily be conducted in either user or kernel mode. Nevertheless, this technique is only convenient for analyzing user-mode IoT attacks, as in the case of kernel-mode IoT attacks such as a rootkit; the malware can interrupt the analysis and severely infect the system or impair the device's hardware.

Table 16
Comparison of testbeds for IoT attacks [157].

Testbed	Easily deployable	Fast system restore	Hardware damage	Requires high cost
Bare-metal	Yes	No	Yes	Yes
Sandboxing	Yes	Yes	No	No
Virtual machine	Yes	Yes	No	No
MA (hardware-based)	No	No	No	Yes
MA (software-based)	Yes	Yes	Yes	No
System emulation	Yes	Not required	No	No

Due to the potential for causing harm or leaving a system in an unsafe state, it is imperative to restore a clean state after performing each user-mode IoT attack. This entails reversing all activities performed by the IoT attacks. To mitigate a kernel-mode IoT attack, it is necessary to recover the affected system by completing the infected system's formatting after each attack analysis. This process involves the installation of a fresh and untainted operating system.

- *Sandbox Environment*: The sandbox environment provides a platform for researchers and security analysts to execute and analyze IoT attack programs while ensuring that the native operating system, its apps, and the underlying hardware remain unaffected by these activities. Sandboxes safeguard procedures and network infrastructures from stealthy attacks or vulnerabilities from unknown IoT attacks.
- *Virtual Machines*: Utilizing virtual machines (VMs) to install and execute IoT threat analysis tools can safeguard the hardware, operating system, and other host system applications. A virtual machine enables the creation of a pristine system snapshot, which may be utilized to restore an infected system to its original, uninfected condition following the execution and analysis of each malware executable.

  Hence, it is imperative to capture and preserve the snapshot of the pristine system consistently. In addition, it is possible to grab a snapshot of the system. At the same time, the IoT attack is being executed, allowing the system to be restored to its prior condition by utilizing the acquired clean snapshot.
- *System emulation*: A further option for an IoT attack testbed is using a system emulator, which facilitates the complete isolation of IoT attack execution from the underlying hardware. An emulator is a software application that enables a primary computer system to function as a secondary computer system, known as the guest computer. This capability allows the host computer to perform an IoT attack that simulates operation within the guest machine. It is strongly recommended to refrain from opening additional programs while an emulator runs due to the shared hardware resources between the emulator and the host computer.

  Furthermore, it is essential to note that IoT attacks are not carried out on tangible hardware but rather on an internal guest software component generated by the emulator. The guest component is responsible for monitoring the actions taken by the IoT attack.
- *Volatile memory acquisition*: The process of memory acquisition (MA) entails the collection of a memory dump, also known as a memory picture, during the operation of an IoT attack or a benign application. This dump is then saved onto the host system or an external storage medium, facilitating further analysis. An IoT attack can be carried out using the above testbed methods, which require the installation of a memory acquisition tool in the analytical environment. The acquisition of memory dumps using hardware can be accomplished by utilizing specialized hardware tools such as “Tribble” (through a PCI express card) or “Firewire”. In contrast, acquisition tools that are based on software offer a high level of usability and user-friendliness. Additionally, they are readily accessible at a relatively cheap cost and may be easily implemented. However, specific advanced malicious software can identify software-based memory acquisition methods. Furthermore, many advanced forms of malware employ anti-debugging tactics that hinder the acquisition tool's ability to capture photos of volatile memory. Consequently, a blank memory picture is acquired when any anti-debugging and anti-memory dumping mechanisms are active in the system's memory.
- *Commercial and Industrial Testbed*: Security experts and systems security analysts predominantly favour the utilization of commercial testbeds. The IBM X-Force Command Centre is recognized as the pioneering commercial malware testbed. The implementation involves the utilization of a mobile command Cyber Tactical Operations Center, which offers a range of functions. The Ixia BreakingPoint and Cyber Exata are further testbeds that are commercially accessible. Ixia can replicate over 28,000 instances of live malware attacks and transmit them to the devices under examination, whilst Cyber Exata offers a range of cyber range services.
- *Internet of thing(IoT) Testbed*: IoT testbeds have become imperative due to their ability to identify vulnerabilities in IoT applications, devices, and communication. Additionally, they provide analysis and assessment reports that may be utilized in creating and advancing novel ways of detecting IoT attacks.

9.4. Honeypots

The honeypot is a cybersecurity technology designed to replicate a genuine and valuable network, attracting potential attackers. It operates within an isolated and segregated network environment. The system mentioned above may be perceived as a simulated entity designed to resemble an authentic system to entice potential attackers to engage with it. This strategy enables the monitoring of the ensuing interaction between the attackers and the compromised device [158].

Over time, the concept of honeypots has gained significant attention among researchers in information security due to its relevance in attack identification and developing deception toolkits. A honeypot is a computer resource designed to be probed, attacked, or hacked, serving as a decoy to gather information about potential threats. The primary purpose of a honeypot is to trick potential attackers, diverting their attention away from the intended target or collecting valuable information about attack trends [158].

Moreover, the value of a honeypot may be assessed when it is subjected to malicious attacks. The honeypot may be described as a tool designed to entice potential attackers into attempting to breach a system by offering a simulated environment resembling a genuine system. Consequently, if the perpetrators successfully breach the system, there is a strong probability that they will access this particular segment of the network to acquire valuable data. In essence, the primary objective of a honeypot is to effectively monitor and record incoming data to utilize this information to mitigate and preemptively deter subsequent analogous cyber-attacks [158]. Fig. 10 shows the overview of honeypots.

Honeypots may be classified into two distinct categories: production honeypots and research honeypots.

A production honeypot is frequently employed as a means for organizations to enhance the security of their internal IT infrastructure. Implementing this particular honeypot mitigates the risks encountered by a specific firm by deploying its information technology infrastructure to detect and identify malicious cyber attacks. Due to the limited range of services this honeypot offers, its installation is

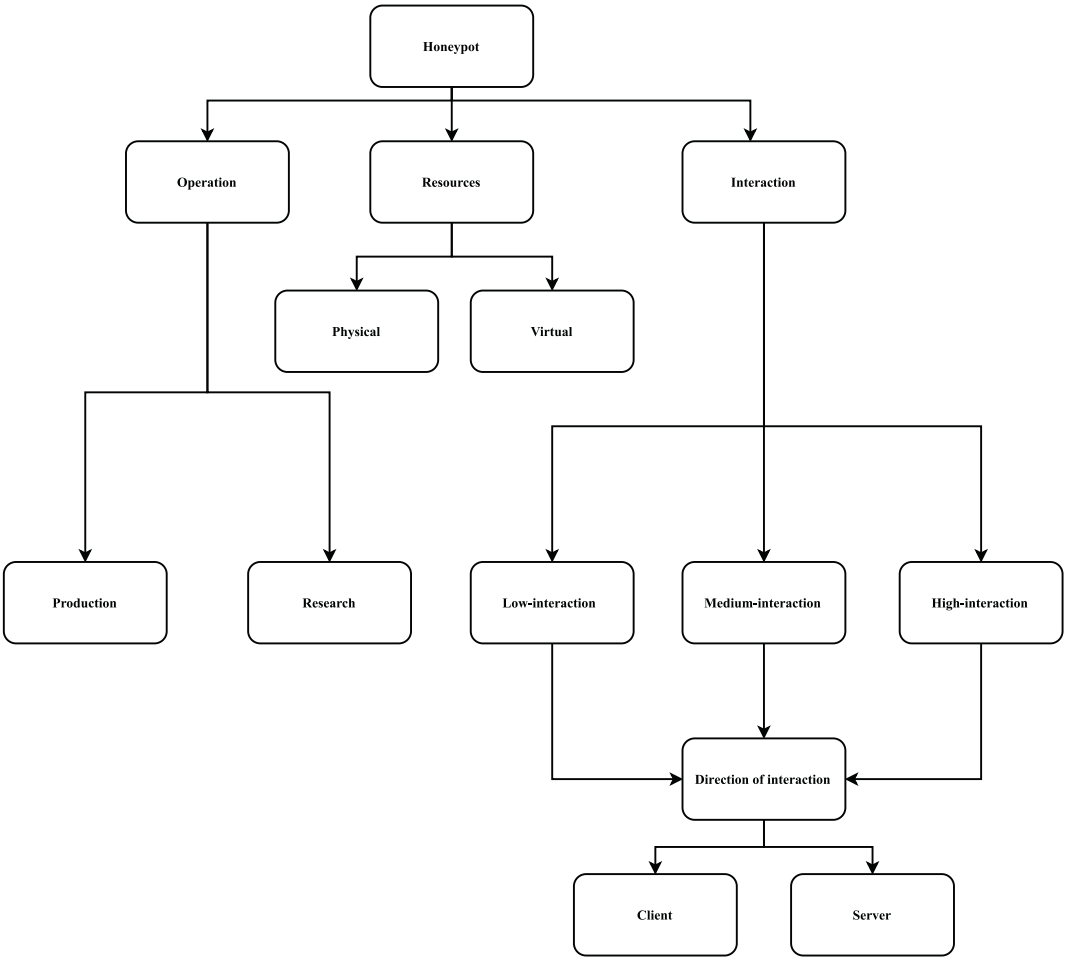

Fig. 10. Overview of honeypots.

frequently characterized by simplicity. One might argue that a production honeypot entails a trade-off between operational simplicity and the volume of information gathered in a research honeypot. The level of sophistication exhibited by this honeypot is considerable since it has been specifically engineered to gather a broad range of information from potential attackers. Consequently, its deployment presents increased challenges. The data collected will assist network forensic experts in gaining a more comprehensive comprehension of the attacker's behaviour patterns [158].

An additional categorization of honeypots may be made according to the extent of interaction they allow, which includes three types:

(1) Low-interaction honeypot (LIH).
(2) Medium-interaction honeypot (MIH).
(3) High-interaction honeypot (HIH).

The low-interaction honeypot (LIH) is equipped with a limited range of services, including SSH, Telnet, and FTP. It is designed to restrict the attacker's access to the operating system, as the honeypot does not own an installed operating system. Hence, the attacker's engagement is restricted to a login endeavour as attempting to guess the password. The low information handler (LIH) generates a limited response mostly utilized for statistical analysis. This particular honeypot may be classified as a production honeypot due to its straightforward installation process and high susceptibility to compromise. The medium-interaction honeypot (MIH) lacks an operating system but offers a more advanced degree of simulated services designed to entice potential attackers [158].

Consequently, this honeypot generates a substantial volume of responses, catalyzing the initiation of subsequent attack stages. The likelihood of being compromised is proportional to the extent of engagement capabilities. In contrast, the high-interaction honeypot (HIH) can be characterized as a multifaceted and advanced honeypot. The implementation and maintenance of this system pose significant challenges due to the provision of an unrestricted operating environment, which enables attackers access to a wide range of services. In other words, HIH imitates the service's functionality within the operating system and executes it itself. This allows for the comprehensive collection and analysis of an attacker's behaviour. This particular honeypot is commonly utilized for research purposes [158].

The honeypot can fulfil the role of either a server or a client honeypot. A server honeypot remains passive until attackers contact it, whereas client honeypots actively seek prospective entities that may begin an interaction.

Honeypots can be categorized as either physical or virtual honeypots. A physical honeypot refers to a tangible device deployed within a network. In contrast, a virtual honeypot is a synthetic entity that resides on a host system and receives network traffic sent to it. A singular server can accommodate several virtual honeypots. The honeypot in question is often classified as a high-interaction honeypot [158].

9.5. Security information and event management

Security information and event management (SIEM) is a security system that aids businesses in the identification and mitigation of possible security threats and vulnerabilities, hence preempting any potential disruptions to business operations. Security information and event management (SIEM) systems play a crucial role in assisting corporate security teams in identifying deviations in user behaviour. Additionally, these systems leverage artificial intelligence (AI) to streamline and automate certain labour-intensive tasks related to identifying potential threats and subsequent incident response [159].

The initial iterations of SIEM platforms were primarily designed as log management solutions, integrating security information management (SIM) and security event management (SEM) functionalities. These platforms facilitated the continuous monitoring and analysis of security-related events in real-time while also facilitating the tracking and logging of security data to ensure compliance and support auditing processes [159].

Over time, the field of security information and event management (SIEM) software has seen advancements to include the integration of user and entity behaviour analytics (UEBA), among other sophisticated security analytics, artificial intelligence (AI), and machine learning techniques. These enhancements enable the identification of abnormal behaviours and the detection of advanced threats by analysing various indicators [159].

In contemporary security operation centers (SOCs), utilizing security information and event management (SIEM) has become a fundamental security monitoring and compliance management component. At a fundamental level, all security information and event management (SIEM) systems engage in data collection, consolidation, and sorting processes to detect potential risks and ensure compliance with data regulations. Although there may be variations in their capabilities, most solutions provide the same essential functionalities [159].

SIEM systems are responsible for ingesting event data from many sources throughout an organization's comprehensive IT architecture, encompassing both on-premises and cloud environments. Real-time analysis involves collecting, correlating, and examining event log data derived from many sources, including users, endpoints, applications, data sources, cloud workloads, and networks. Data obtained from security hardware and software, such as firewalls or antivirus software, is also included in this process.

Certain security information and event management (SIEM) solutions can link with external threat intelligence feeds from third-party sources. This integration enables the correlation of internal security data with pre-existing threat signatures and profiles. The utilization of real-time threat feeds facilitates the ability of teams to prevent or identify novel forms of attack signatures effectively.

Event correlation plays a crucial role within the framework of a security information and event management (SIEM) system. By employing sophisticated analytics techniques, event correlation enables the identification and comprehension of complex data patterns, offering valuable insights for promptly detecting and mitigating possible risks to an organization's security [159].

SIEM systems substantially impact enhancing the MTTD (Mean Time To Detect) and MTTR (Mean Time To Respond) for IT security teams. This is achieved by relieving the burden of manual workflows in conducting thorough security event analyses.

Security information and event management (SIEM) is an operational technology that integrates and centralizes the analysis of security-related data into a unified dashboard. This allows security teams to monitor and assess activities efficiently, prioritize alerts, detect potential threats, and take appropriate actions to mitigate risks or address security incidents. In addition to their primary functions, SIEM dashboards commonly incorporate real-time data visualizations, which facilitate identifying anomalies or patterns indicative of potentially malicious behaviour by security analysts. Administrators can promptly receive alerts and implement necessary measures to minimize potential security threats using configurable and predefined correlation criteria. This proactive approach enables them to address these dangers before they escalate into more substantial security concerns [159].

SIEM systems have gained significant popularity among enterprises that must adhere to regulatory compliance standards. Due to its automated data collection and analysis capabilities, using SIEM is an invaluable asset in acquiring and validating compliance data throughout an organization's infrastructure. Security information and event management (SIEM) systems can produce compliance reports in real-time for various standards such as Payment Card Industry Data Security Standard (PCI-DSS), General Data Protection Regulation (GDPR), Health Insurance Portability and Accountability Act (HIPAA), Sarbanes-Oxley Act (SOX), and other relevant compliance frameworks. This functionality alleviates the challenges associated with security management and facilitates the timely identification of any violations, enabling prompt remedial actions. Numerous SIEM solutions have pre-configured add-ons to create automatic reports tailored to fulfill compliance obligations [159].

Irrespective of the scale of a business, it is imperative to use proactive measures to monitor and address IT security issues. SIEM systems offer several advantages to organizations and have emerged as a crucial element in optimizing security procedures.

SIEM systems facilitate the consolidation of compliance audits and reporting processes over a complete organizational architecture. Implementing advanced automation enables the efficient gathering and analysis of system logs and security events, decreasing internal resource usage while adhering to rigorous compliance reporting requirements. Contemporary security information and event management (SIEM) solutions have evolved to incorporate advanced security orchestration, automation, and response (SOAR) systems. This integration offers notable benefits to IT teams regarding time and resource efficiency while handling organizational security measures.

These systems leverage deep machine learning techniques to autonomously acquire knowledge from network activity. As a result, they can efficiently manage intricate threat detection and incident response procedures, surpassing the time efficiency of physical teams [159].

SIEM can be a crucial catalyst for enhancing interdepartmental efficiency because it can improve visibility inside IT settings. Utilizing a single dashboard facilitates the consolidation of system data, warnings, and notifications, improving team communication and collaboration effectiveness in addressing security risks and events. Given the dynamic nature of the cybersecurity environment, enterprises must have access to dependable solutions capable of identifying and addressing familiar and unfamiliar security risks. By leveraging integrated threat intelligence feeds and artificial intelligence (AI) technologies, security information and event management (SIEM) systems have the potential to enhance the efficiency of security teams in their response to all types of attacks.

Security information and event management (SIEM) systems are highly suitable for performing computer forensic investigations following a security event. Security information and event management (SIEM) solutions enable enterprises to effectively gather and analyze log data from their digital infrastructure under a centralized platform. This empowers individuals to replicate previous occurrences or examine novel ones to scrutinize dubious behaviour and establish more efficient security protocols. The process of compliance audits and reporting is both a vital and demanding endeavour for several firms. SIEM systems significantly decrease the resource allocation necessary for process management by providing real-time audits and on-demand regulatory compliance reporting as and when required.

Due to the increasing prevalence of remote work arrangements, the utilization of software-as-a-service (SaaS) applications, and the implementation of bring-your-own-device (BYOD) rules, companies must possess sufficient visibility to address network vulnerabilities beyond conventional network boundaries effectively. Security Information and Event Management (SIEM) systems can monitor and record all network activity, encompassing people, devices, and applications. This comprehensive approach enhances visibility throughout the whole infrastructure, enabling the identification of potential risks and threats, irrespective of the location from which digital assets and services are accessible [159].

9.6. *Machine learning*

Machine learning techniques have demonstrated significant potential in identifying and detecting malicious software attacks [160]. The subsequent machine learning techniques are categorized according to their commonalities in learning approaches. Fig. 11 offers a comprehensive summary of IoT attack detection methods.

- *Instance-based Learning*: The instance-based learning approach, also known as lazy learning or memory-based learning, is a supervised learning classifier that uses similarity metrics to categorize. The families of instance-based learning approaches include K-nearest neighbour (KNN) [161], self-organising map (SOM) [162,163], and support vector machine (SVM) [164–166].
- *Decision Tree*: The decision tree (DT) technique is a version of the supervised learning approach. The initial step in decision tree classification is the representation of the complete dataset, which includes benign and malicious behavioural aspects, as the root node. In the subsequent phase, the classifier employs feature selection approaches, such as information gain [167], to identify the most pertinent characteristics from the dataset.
  The third stage is partitioning the whole dataset into subsets that consist of the most pertinent attributes and corresponding values. During the fourth stage, the DT classifier produces internal nodes, also known as decision tree nodes, encompassing the most optimal characteristics. In the last stage, denoted as the fifth step, the classifier proceeds to iteratively generate decision trees by utilizing the subset of the most pertinent data, as established in step 3. The iterative process persists until no more nodes can be categorized, and the last node is referred to as a leaf node. This leaf node represents the anticipated outcome of a specific occurrence, which may be benign or malicious.
  J48, known as the C4.5 decision tree, logistic model tree (LMT), iterative dichotomiser 3 (ID3), and classification and regression tree (CART) are among the several types of decision tree classifiers. The LMT [168] is a supervised learning classifier constructed using a hybrid logistic regression approach and a decision tree classifier.
- *Regression Method*: The regression technique involves constructing a detection model using input features from a training dataset. This model is then used to predict the class of a new instance. The output of a regression classifier is contingent upon the knowledge acquired by the classifier during the training phase. Logistic regression (LR), also known as binary logistic regression, is a widely utilized family of regression classifiers in malware detection. It has the distinction of being the most often employed regression classifier [169].
  Multinomial logistic regression (MLR) constitutes an additional category of regression classifiers. The MLR classifier is commonly employed in multi-class classification scenarios, wherein many characteristics within the training set influence the classifier's prediction [169,170]. The multiple linear regression (MLR) model encompasses a range of potential predictions for the target variable, making it a suitable approach for forecasting several forms of a certain malware executable.
- *Probabilistic Method*: The probabilistic methodology is a method that utilizes conditional probabilities to predict the label or class of unknown samples. This algorithm provides a solution by estimating the probability distribution. Various techniques fall under the heading of Naive Bayes, including Naive Bayes (NB), Gaussian Naive Bayes (GNB), Bernoulli Naive Bayes (BNB), Multinomial Naive Bayes (MNB), Bayesian belief network (BBN), Bayesian-logistic regression (BLR), and hidden Markov model (HMM) [171–178].
- *Clustering Method*: The clustering approach is an unsupervised learning method that derives knowledge from an unstructured dataset. K-means, K-medoids, hierarchical clustering (HC), expectation-maximization (EM), mean-shift clustering (MSC), and density-based spatial clustering of applications with noise (DBSCAN) are widely recognized clustering approaches. The K-means algorithm is a partitioning approach that generates clusters that are more compact and efficient/robust [175]. K-means clustering is a method that seeks to divide a

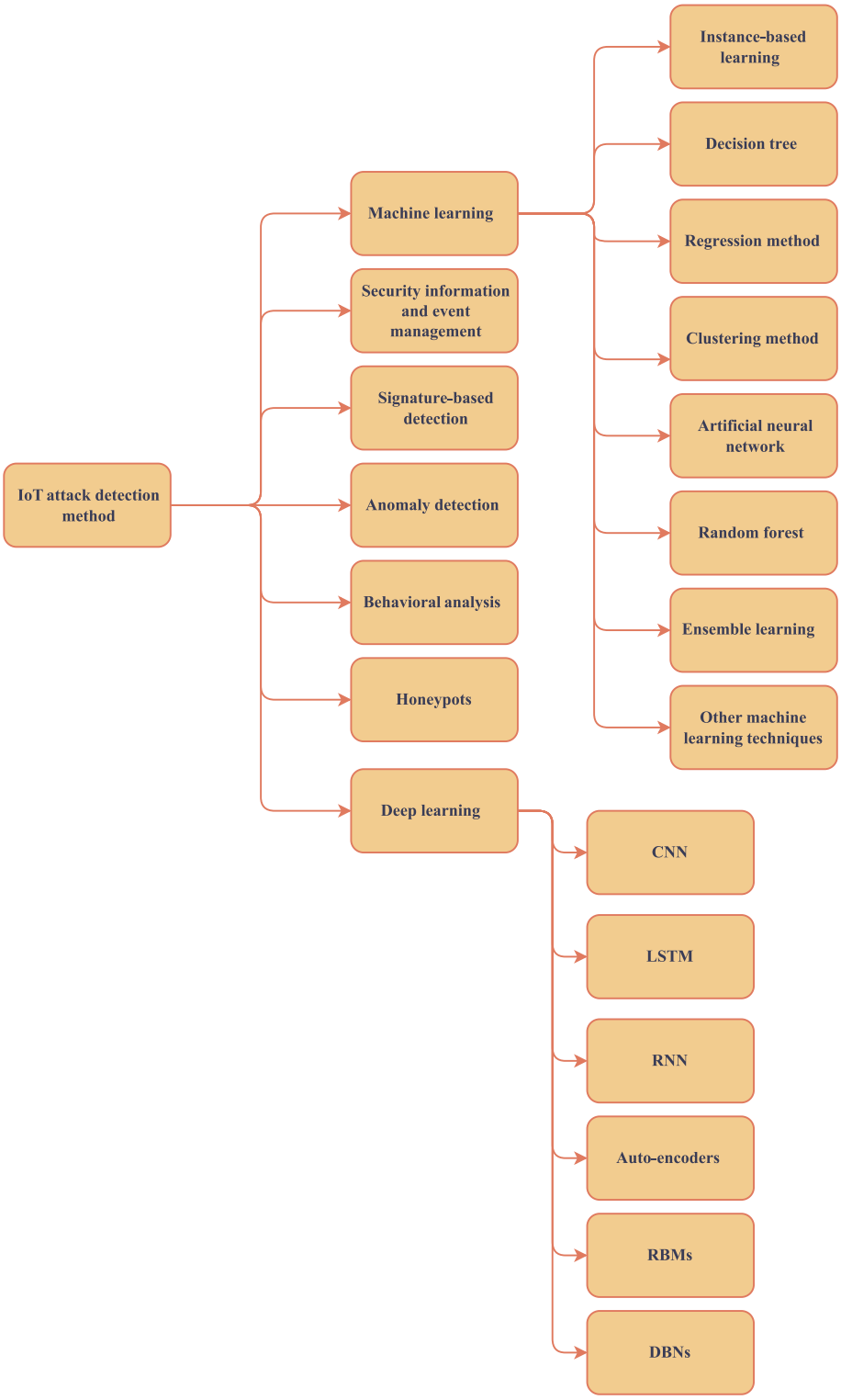

Fig. 11. Overview of IoT attack detection methods.

set of observations, denoted as n, into k distinct clusters, denoted as C (where C = c_1, c_2, c_3, ⋯, c_k, with c_k representing the final cluster). This partitioning is based on similarity measures, and each observation is assigned to the cluster with the closest mean, which acts as a representative prototype for that cluster. The similarity between observations may be determined by utilizing distance metrics [175,179].

The K-medoids algorithm is a modified version of the partitioning clustering approach. It is designed to generate more compact clusters when applied to a collection of unlabeled data [180]. Hierarchical clustering, also known as hierarchical cluster analysis (HCA), is an unsupervised learning method that organizes a set of supplied observations or objects into a hierarchical structure of clusters. In the context of expectation-maximization (EM), measuring distances between observations is conducted by utilizing probability distributions [181]. The authors of [182] have applied the MSC and DBSCAN algorithms for grouping malware.

- *Artificial Neural Network (ANN)*: The Artificial Neural Network (ANN) technique is a computational model inspired by the structure and function of biological neural networks. The Artificial Neural Network (ANN), commonly referred to as a neural network, is a machine learning technique that aims to uncover the inherent associations among observations within a provided dataset by emulating the human brain's cognitive processes.
  The topology of an artificial neural network (ANN) model typically consists of interconnected nodes or neurons, including input, hidden, and output layers, which interact with each other. In the context of malware binary classification, the resulting output can be categorized as either malware or benign, as indicated by Ref. [183]. The detection of malware has been facilitated by the utilization of several artificial neural network (ANN) approaches, such as the multilayer perceptron (MLP) [184], back-propagation (BP) [185], and stochastic gradient descent (SGD) [186].
- *Ensemble Learning*: The ensemble learning approach, also known as the ensemble technique, falls under machine learning, encompassing both supervised and unsupervised learning. It involves amalgamating diverse machine learning algorithms and several feature sets, also called boosting, to use their respective strengths and potentials. Boosting is not a singular, distinct learning algorithm; instead, it is a general method that aims to enhance the learning process of many weak algorithms by combining them.
  AdaBoost, often referred to as discrete AdaBoost, is a prominent boosting technique utilized in the domain of classification problem-solving. The weakness of a particular learning algorithm is assessed by calculating the mistake rate at each iteration. AdaBoost identifies misclassified observations and assigns higher weights for the subsequent algorithm to prioritize their importance, enhancing the classification outcome. The procedure persists until the desired outcomes are attained, and no further classifier may be appended.
  Bagging is an ensemble method that utilizes a mixture of many algorithms to enhance the overall predictive performance. The approach partitions the training set into numerous subsets, each serving as a training sample. A classifier is then constructed for each training sample, resulting in many classifiers that learn from distinct training samples derived from a common initial training set. The anticipated results of all classifiers are aggregated by calculating either an average or a majority vote [187]. In Ref. [188], a bagged-J48 algorithm was introduced.
- *Random Forest*: The random forest method is a supervised learning algorithm capable of learning from a provided training dataset by constructing an ensemble of decision trees. The learning process of this method is dependent on bagging processes. Hence, the random forest algorithm constructs a collection of several decision trees to create an ensemble, combining the predictions of these trees to enhance the overall performance [189].
  The authors of [190] utilized the API call sequences obtained during ransomware execution to develop detection techniques based on random forest and AdaBoost algorithms. Stacking, also known as a stacked generalization, is a widely used ensemble approach in machine learning [187].
  Gradient boosting machines (GBM) is an ensemble approach commonly employed for classification and regression tasks. The approach involves the integration of models derived from several learning algorithms to create a novel iterative model. The method was first introduced as AdaBoost, which subsequently led to the development of several ensemble approaches like GBM, model-based boosting (MBoost), and subsequently CatBoost, XGBoost, and LightGBM [191]. In a previous study, the authors introduced a malware detection model based on lightGBM [191].
- *Other Machine Learning Techniques*: Various other machine-learning techniques are available for behaviour-based malware detection. These include the inductive rule learner (e.g., RIPPER) [192], reinforcement learning [193], similarity-based and threshold-matching [194], fuzzy clustering approach [195], and evolutionary techniques [196].

9.7. Deep learning

Deep learning (DL) refers to a category of neural network algorithms characterized by utilizing numerous hidden layers in their architecture. Hidden layers in neural networks allow the model to acquire knowledge about benign and malicious properties by learning abstractions. Deep learning models can effectively handle large datasets that include high dimensions. These models can automatically extract and select high-level abstract characteristics without the need for human experience. Research has shown that deep learning models perform comparatively better than typical machine learning models [197].

Additionally, it is worth noting that data labelling is unnecessary for training deep learning models. These models can learn from labelled and unlabeled data, yielding accurate findings while minimizing false-positive outcomes [198]. Hence, in comparison to conventional supervised machine learning methods that heavily rely on human labour and domain experts for feature extraction and selection [199] and are susceptible to a high false positive rate (FPR) along with their simplistic assumptions for generating pertinent features [200], deep learning models have gained prominence among researchers for their ability to address the limitations of traditional machine learning approaches.

Various deep learning techniques such as convolutional neural networks (CNNs) [201], recurrent neural networks (RNNs) [202,203], long short-term memory networks (LSTMs) [204], stacked auto-encoders (SAEs) [205], deep feedforward neural networks (DFNNs), also referred to as feedforward networks (FNs) [206], restricted Boltzmann Machines (RBMs) [207],

and deep belief networks (DBNs) [208] are employed for behaviour-based malware detection. The aforementioned procedures were also utilized in previous studies [209–211].

10. Challenges and research opportunities

The topic of security in the IoT has been a subject of significant worry, with the primary factors contributing to this concern being the absence of established security standards and hardware-related vulnerabilities. This part, however, centers its attention on the obstacles and areas for further investigation in the realm of IoT attacks.

The examination of current methodologies for detecting attacks in the IoT reveals that scholars and industry professionals are actively engaged in addressing the security vulnerabilities within this domain. Nevertheless, some obstacles in the realm of IoT threat detection strategies need consideration in developing innovative approaches in further research endeavors.

(1) **Availability of Dataset**: The dataset holds utmost importance as the fundamental component of methodologies employed for detecting IoT attacks. According to Ref. [212], in the context of machine learning and deep learning approaches, the rate at which a model learns and improves is directly proportional to the size of the data it is fed. Nevertheless, a significant dearth of reliable data on IoT attacks is evident. IoTPOT and IoT-23 are now considered the leading benchmark IoT datasets in IoT malware research. Nevertheless, given that most of the research utilizes an identical dataset, any flaws or imperfections in the dataset can significantly influence the overall model.
Additionally, it is important to consider that each dataset may have a unique emphasis, potentially limiting its usefulness in the suggested model. As an illustration, the IoTPoT is implemented on a limited set of Internet protocol (IP) addresses, especially engaging with Telnet queries [213]. Hence, generating a dataset that accurately reflects real-world threats is imperative.
Future Scope: Create a dataset by implementing IoT attack samples on a collection of authentic IoT devices with varying architectures and operating systems.

(2) **Presence of Diverse Attack Vectors**: Investigating various attack vectors is an essential and ever-evolving field of research within the realm of IoT (Internet of Things) attacks. IoT ecosystems' intricate and linked structure makes them very vulnerable to a diverse array of attack vectors. Understanding and classifying various attack vectors is vital in developing efficacious security solutions.
Future Scope: The investigation of improved anomaly detection techniques can contribute to the identification of atypical behaviour in IoT devices, hence facilitating early detection of potential threats and the anticipation of diverse attack modalities.

(3) **Lack of Real-world Experiments**: In the context of dynamic analysis approaches, it is important to acknowledge that certain IoT attacks can potentially cause significant harm to the devices employed inside the testbed. Despite the increased difficulty and complexity of analyzing real devices, the optimal detection strategy must assess its performance in a real-world setting.
Future Scope: It is advisable to incorporate actual devices to analyze and evaluate the proposed model design, as this would provide a more comprehensive assessment of its effectiveness. In addition, it is possible to build strategies for detecting anti-virtual machine measures.

(4) **Scalability of Detection Methods**: With the rapid proliferation of IoT devices and networks, traditional security mechanisms may struggle to keep pace with the increasing scale and complexity of the ecosystem. Researchers are challenged to develop detection methods that can efficiently monitor and protect large-scale IoT deployments. Scalability is crucial to ensure that security solutions can handle the growing number of devices and the massive volumes of data generated by IoT system.
Future Scope: Develop module-based detection approaches that can be extended later when new malware of different architectures or operating systems are evolved. Opportunities in this area involve designing lightweight, distributed, and adaptive detection algorithms that can be seamlessly integrated into IoT infrastructures. Additionally, exploring the potential of edge and fog computing for decentralized security measures is vital for achieving scalability. By addressing these scalability challenges, researchers can contribute to the development of IoT security solutions that can effectively safeguard the expanding IoT landscape [214,215].

(5) **Experimental Environment**: Working on real devices is expensive and time-consuming as the hardware and software aspects must be studied before the implementation. In this situation, most detection mechanisms prefer working in simulation environments that are cost-effective and flexible. However, the robust IoT attack detection methods in the simulation environment might not be ideal for real devices. Furthermore, its effectiveness in a virtual environment cannot confirm that the model is efficient.
Future Scope: Cost-effective approaches, including testbed-as-a-service [216] and re-usage of old IoT devices, can be considered while working with real testbeds.

(6) **Lack of Honeypots**: Honeypot is a security means that makes a virtual trap to attract attackers to capture their behaviour patterns, attack vectors, and security issues [217]. However, there are fewer existing honeypots for capturing IoT malware behaviour. As a result, there is a lack of behavioural patterns or rules that can be used in detection tools or mechanisms.
Future Scope: Implement honeypots that handle massive data and support more protocols and increased IoT devices to capture the behaviour of IoT attacks.

(7) **Emphasis of Adversarial Attacks**: Adversarial attacks, which introduce deliberate perturbations to the data, provide a distinct method for inducing inaccuracies in the model's output [218]. However, there is a scarcity of detection systems that consider adversarial attacks.

Future Scope: While implementing the detection model, experimenting with adversarial attacks can improve the detection model.

(8) **Absence of Cross-Platform Detection Methods**: In most of the existing works, ARM-based IoT malware samples are used even though there exist different IoT devices of Intel, ARC, SPARC, MIPS architectures [219]. Focusing on a particular type of architecture won't solve the security issues in IoT devices.
Future Scope: Develop IoT attack detection methods that consider cross-architecture malware samples rather than concentrating on one type.

(9) **Selection of Features**: Feature selection refers to identifying and prioritizing specific attributes or characteristics of IoT devices and networks relevant to attack detection and prevention. Given IoT threats' diverse and ever-evolving nature, identifying effective features is crucial for developing accurate and efficient security models. Research in this domain offers the potential to enhance the precision of anomaly detection, reduce false positives, and enable proactive threat mitigation.
Future Scope: Future work could focus on the automated extraction and selection of pertinent features, leveraging machine learning and data-driven approaches to continually adapt to the changing threat landscape. By refining feature selection techniques, researchers can contribute to more robust and adaptive security solutions for IoT ecosystems. Consider memory-related features [220] and log files for attack detection in addition to the widely used network features and opcode sequences.

(10) **Determination of Evaluation Metrics**: Determining evaluation metrics is pivotal to IoT (Internet of Things) attacks. When assessing the effectiveness of security measures and strategies against IoT threats, it is crucial to have well-defined evaluation metrics in place. These metrics help quantify the performance and efficiency of security solutions, enabling organizations to understand their vulnerabilities better. In IoT attacks, evaluation metrics can encompass various factors, including detection accuracy, false-positive rates, response time, and the ability to handle diverse attack scenarios.
Accurate and comprehensive evaluation metrics are essential for identifying the strengths and weaknesses of security measures, ultimately aiding in the refinement of defence strategies. The dynamic nature of IoT attacks necessitates a continuous evolution of evaluation metrics to keep pace with emerging threats and ensure robust protection of IoT ecosystems [215].
Future Scope: Introduce novel evaluation metrics and focus on multiple metrics rather than concentrating on one.

(11) **Absence of Software Tools**: The absence of software tools in the context of IoT (Internet of Things) attacks represents a critical gap in the cybersecurity landscape. IoT devices, often constrained by resource limitations, require specialized security tools for threat detection and mitigation. However, the diversity and sheer number of IoT devices make developing and deploying universal software tools challenging. This deficiency can leave IoT ecosystems vulnerable to various attacks, including malware infections, data breaches, and device manipulation.
The absence of specialized software tools poses challenges in properly monitoring, managing, and safeguarding IoT networks. To tackle this matter effectively, it is imperative to devise security measures specifically designed for lightweight Internet of Things (IoT) systems, considering the distinctive attributes of IoT devices. Future research and investment in the field of IoT cybersecurity should place greater emphasis on the development of technologies aimed at enhancing the safeguarding of IoT ecosystems [215].
Future Scope: Create a software-based implementation to be integrated into real devices, reducing the security issues in IoT devices.

(12) **Presence of Obfuscation**: Obfuscation in the context of IoT (Internet of Things) attacks is a concerning security aspect. Obfuscation is a technique malicious actors use to hide their malicious code, making it difficult to detect and analyze. In IoT attacks, obfuscation is often employed in developing malware and exploits. This poses a significant challenge for security professionals and researchers who aim to identify and mitigate these threats. The use of obfuscation in IoT attacks can complicate the detection process, as it requires specialized tools and techniques to unravel the obscured code.
Furthermore, it underscores the evolving nature of IoT attacks, where threat actors continuously adapt and refine their methods to avoid detection. To address this, the security community must invest in advanced threat detection and analysis methods capable of deciphering obfuscated code, thereby enhancing the security of IoT devices and networks [221].
Future Scope: Consider the measure of randomness techniques in the model so that code obfuscation and data compression can also be detected.

11. Conclusion and future work

The IoT, considered the third industrial revolution, has positively impacted the economy through the digitalization of numerous sectors like education, military, healthcare, tourism, and others. The increasing proliferation of IoT devices and inadequate security measures have given rise to many malicious attacks, such as hacking, espionage, intrusion methods, and malware attacks. The primary causes of attacks may be attributed to inadequate credentials, insufficient adherence to security standards, and a lack of user awareness on security measures [17].

Researchers and security professionals have focused on various detection and prevention strategies to address this issue. However, the lack of knowledge and the increasing prevalence of IoT threats pose significant challenges to developing effective security measures for IoT devices.

To facilitate the attention of academics and industrialists towards this particular area, we provide our study, which encompasses four primary contributions. A comprehensive investigation was undertaken to examine various dimensions of attacks on the IoT. These attacks were classified within a taxonomy based on several factors, including attack domains, threat types, execution methods, software surfaces, IoT protocols, device properties, adversary locations, and information damage levels.

Furthermore, a comprehensive examination was conducted on several attacks, resulting in a mapping correlating the categories with the list of common vulnerabilities and exposure databases (CVEs).

Next, we conducted an examination of several strategies for detecting attacks in the IoT context, encompassing approaches such as machine learning and deep learning-based detection techniques. One may see that machine learning strategies have been categorized according to shared characteristics in learning methodologies. In contrast, deep learning-based detection approaches were developed to enable the model to acquire information pertaining to both benign and harmful features through the process of learning abstractions.

In addition, we have highlighted twelve challenges that have yet to be solved due to the dynamic nature of IoT threats. Furthermore, we have indicated potential areas for future study that might effectively mitigate these challenges.

The incorporation of the IoT into the daily lives of individuals has yielded several benefits, including enhanced data analytics capabilities, increased automation, and the utilization of intelligent gadgets. Concerns over security and vulnerability complement the convenience. The diverse composition of the IoT amplifies the potential vulnerabilities that attackers may exploit, hence expanding the overall attack surface. Additionally, integrating IoT devices into existing networks presents novel security risks that were not previously present.

A potential breach of the system's security can have severe consequences for its users. When designing the system, it is important to consider total security in order to ensure the mitigation of critical vulnerabilities. In order to mitigate the occurrence of attacks, it is imperative to effectively adopt appropriate rules and processes. This study has provided an analysis of attacks targeting the IoT within the context of the rapidly developing IoT infrastructure. Additionally, it has discussed several approaches for detecting these attacks. Nonetheless, IoT devices' many characteristics and limitations render any proposed solution unsuitable and outdated. Furthermore, it is anticipated that an increasing number of countermeasures and vulnerabilities will be revealed soon due to the dynamic nature of technology [222].

Acknowledgment

This project was supported in part by collaborative research funding from the National Research Council of Canada's Artificial Intelligence for Logistics Program.

Declaration of competing interest

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

References

[1] W. Ma, L. Ma, K. Li, J. Guo, Few-shot IoT attack detection based on SSDSAE and adaptive loss weighted meta residual network, Information Fusion 98 (2023) 101853.
[2] J. Sengupta, S. Ruj, S. Das Bit, A comprehensive survey on attacks, security issues and blockchain solutions for IoT and IIoT, Journal of Network and Computer Applications 149 (C) (2020) 102481.
[3] F. Dahlqvist, M. Patel, A. Rajko, J. Shulman, Growing opportunities in the Internet of Things, McKinsey & Company, July 2019 [06-04-2023], https://www.mckinsey.com/industries/private-equity-and-principal-investors/our-insights/growing-opportunities-in-the-internet-of-things.
[4] A. Simmons, Internet of Things (IoT) architecture: Layers explained November 2022 [06-04-2023], https://dgtlinfra.com/internet-of-things-iot-architecture/.
[5] M. Abomhara, G.M. Køien, Cyber security and the Internet of Things: Vulnerabilities, threats, intruders and attacks, Journal of Cyber Security and Mobility 4 (1) (2015) 65–88.
[6] K. Gülen, IoT protocols 101: The essential guide to choosing the right option, January 2023 [06-04-2023], https://dataconomy.com/2023/01/03/iot-protocols-comparison/.
[7] IoT Lens: Edge layer, March 2023 [06-04-2023], https://docs.aws.amazon.com/wellarchitected/latest/iot-lens/edge-layer.html.
[8] R. Schmid, The importance of the edge layer in an IoT ecosystem, Jun 2018 [06-04-2023], https://www.techtarget.com/iotagenda/blog/IoT-Agenda/The-importance-of-the-edge-layer-in-an-IoT-ecosystem.
[9] S. Shock, The 5 layers of IoT architecture that give it super powers, January 2023 [06-04-2023], https://novotech.com/learn/m2m-blog/blog/2023/01/03/the-5-layers-of-iot-architecture-that-give-it-super-power/.
[10] R. Agar, IoT architecture guide: Major and additional layers of IoT system, November 2022 [06-04-2023], https://www.helpwire.app/blog/iot-architecture/.
[11] A. Amir, What are IoT attacks?, 2023 [06-04-2023], https://www.educative.io/answers/what-are-iot-attacks.
[12] Common IoT attacks that compromise security, May 2022 [06-04-2023], https://socradar.io/common-iot-attacks-that-compromise-security/.
[13] IoT. Attacks, 6 security risks to be aware of, July 2023 [06-04-2023], https://www.byos.io/blog/iot-attacks.
[14] A. Arampatzis, Top 10 vulnerabilities that make IoT devices insecure, October 2022 [06-04-2023], https://venafi.com/blog/top-10-vulnerabilities-make-iot-devices-insecure/.
[15] E. Bertino, N. Islam, Botnets and Internet of Things security, Computer 50 (2) (2017) 76–79.
[16] [06-04-2023], https://wiki.owasp.org/index.php/OWASP_Internet_of_Things _Project#tab=IoT_Top_10, Internet of Things (IoT) top 10 2018.
[17] P. Victor, A.H. Lashkari, R. Lu, T. Sasi, P. Xiong, S. Iqbal, IoT malware: An attribute-based taxonomy, detection mechanisms and challenges, Peer-to-Peer Networking and Applications 16 (3) (2023) 1380–1431.
[18] L. Sun, Q. Du, A review of physical layer security techniques for Internet of Things: Challenges and solutions, Entropy 20 (10) (2018) 730.
[19] M.M. Ogonji, G. Okeyo, J.M. Wafula, A survey on privacy and security of Internet of Things, Computer Science Review 38 (2020) 100312.
[20] B.B. Zarpelão, R.S. Miani, C.T. Kawakani, S.C. de Alvarenga, A survey of intrusion detection in Internet of Things, Journal of Network and Computer Applications 84 (2017) 25–37.
[21] S. Hajiheidari, K. Wakil, M. Badri, N.J. Navimipour, Intrusion detection systems in the Internet of Things: A comprehensive investigation, Computer Networks 160 (2019) 165–191.
[22] D.E. Kouicem, A. Bouabdallah, H. Lakhlef, Internet of things security: A top-down survey, Computer Networks 141 (2018) 199–221.
[23] R.R. Krishna, A. Priyadarshini, A.V. Jha, B. Appasani, A. Srinivasulu, N. Bizon, State-of-the-art review on IoT threats and attacks: Taxonomy, challenges and solutions, Sustainability 13 (16) (2021) 9463.

[24] S.M. Tahsien, H. Karimipour, P. Spachos, Machine learning based solutions for security of Internet of Things (IoT): A survey, Journal of Network and Computer Applications 161 (2020) 102630.
[25] M.A. Khan, K. Salah, IoT security: Review, blockchain solutions, and open challenges, Future Generation Computer Systems 82 (2018) 395–411.
[26] H.A. Abdul-Ghani, D. Konstantas, M. Mahyoub, A comprehensive IoT attacks survey based on a building-blocked reference model, International Journal of Advanced Computer Science and Applications 9 (3) (2018) 355–373.
[27] H.J.F. Bel, S. Sabeen, A survey on IoT security: Attacks, challenges and countermeasures, Webology 19 (1) (2022) 3741–3763.
[28] X. Liang, Y. Kim, A survey on security attacks and solutions in the IoT network, in: Proceedings of 2021 IEEE 11th Annual Computing and Communication Workshop and Conference (CCWC), IEEE, Piscataway, 2021, pp. 853–859.
[29] Y. Shah, S. Sengupta, A survey on classification of cyber-attacks on IoT and IIoT devices, in: Proceedings of 2020 11th IEEE Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON), IEEE, Piscataway, 2020, pp. 406–413.
[30] G. Fersi, Fog computing and Internet of Things in one building block: A survey and an overview of interacting technologies, Cluster Computing 24 (4) (2021) 2757–2787.
[31] L. Sun, Q. Du, A review of physical layer security techniques for Internet of Things: Challenges and solutions, Entropy 20 (10) (2018) 730.
[32] N.K. Tran, M.A. Babar, J. Boan, Integrating blockchain and Internet of Things systems: A systematic review on objectives and designs, Journal of Network and Computer Applications 173 (2021) 102844.
[33] R.A. Memon, J.P. Li, J. Ahmed, M.I. Nazeer, M. Ismail, K. Ali, Cloud-based vs. blockchain-based IoT: A comparative survey and way forward, Frontiers of Information Technology & Electronic Engineering 21 (4) (2020) 563–586.
[34] P.J. Taylor, T. Dargahi, A. Dehghantanha, R.M. Parizi, K.-K.R. Choo, A systematic literature review of blockchain cyber security, Digital Communications and Networks 6 (2) (2020) 147–156.
[35] L. Cui, S. Yang, F. Chen, Z. Ming, N. Lu, J. Qin, A survey on application of machine learning for Internet of Things, International Journal of Machine Learning and Cybernetics 9 (8) (2018) 1399–1417.
[36] L. Xiao, Y. Li, G. Han, G. Liu, W. Zhuang, Phy-layer spoofing detection with reinforcement learning in wireless networks, IEEE Transactions on Vehicular Technology 65 (12) (2016) 10037–10047.
[37] National Vulnerability Database (NVD), January 2022 [03-10-2023], https://www.nist.gov/programs-projects/national-vulnerability-database-nvd.
[38] Chinese National Vulnerability Database, September 2020 [21-08-2023], https://dbpedia.org/page/Chinese_National_Vulnerability_Database.
[39] JVN iPedia, October 2023 [21-08-2023], https://jvndb.jvn.jp/en/.
[40] ISC-CERT-CN [21-08-2023], https://www.ics-cert.org.cn/portal/index.html.
[41] United States Computer Emergency Readiness Team (US-CERT), August 2023 [21-08-2023], https://en.wikipedia.org/wiki/United_States_Computer_Emergency_Readiness_Team.
[42] Zero day initiative [06-04-2023], https://www.zerodayinitiative.com/.
[43] BugTraq [06-04-2023], https://en.wikipedia.org/wiki/Bugtraq.
[44] Vulners [06-04-2023], https://vulners.com/.
[45] exploitee.rs [06-04-2023], https://exploitee.rs/.
[46] CVE, August 2023 [06-04-2023], https://cve.mitre.org/.
[47] N. Neshenko, E. Bou-Harb, J. Crichigno, G. Kaddoum, N. Ghani, Demystifying IoT security: An exhaustive survey on IoT vulnerabilities and a first empirical look on internet-scale IoT exploitations, IEEE Communications Surveys & Tutorials 21 (3) (2019) 2702–2733.
[48] E. Shaikh, I. Mohiuddin, A. Manzoor, Internet of Things (IoT): Security and privacy threats, in: Proceedings of the 2019 2nd International Conference on Computer Applications & Information Security (ICCAIS), IEEE, Piscataway, 2019, pp. 1–6.
[49] W.H. Hassan, Current research on Internet of Things (IoT) security: A survey, Computer networks 148 (2019) 283–294.
[50] L. Dineshwori, The right security for IoT: Physical attacks and how to counter them, June 2019 [06-04-2023], https://iot.electronicsforu.com/headlines/the-right-security-for-iot-physical-attacks-and-how-to-counter-them/.
[51] A. Gangolli, Q.H. Mahmoud, A. Azim, A systematic review of fault injection attacks on IoT systems, Electronics 11 (13) (2022) 2023.
[52] A.S. Gillis, Fault injection testing [06-04-2023], https://www.techtarget.com/searchsoftwarequality/definition/fault-injection-testing.
[53] Attacks in the IoT [06-04-2023], https://ebrary.net/180673/computer_science/attacks.
[54] A.B. Usman, J. Gutierrez, Toward trust based protocols in a pervasive and mobile computing environment: A survey, Ad Hoc Networks 81 (2018) 143–159.
[55] R. Krejčí, O. Hujňák, M. Švepeš, Security survey of the IoT wireless protocols, in: Proceedings of the 2017 25th Telecommunication Forum (TELFOR), IEEE, Piscataway, 2017, pp. 1–4.
[56] M.M. Ahemd, M.A. Shah, A. Wahid, IoT security: A layered approach for attacks & defenses, in: Proceedings of the 2017 International Conference on Communication Technologies (ComTech), IEEE, Piscataway, 2017, pp. 104–110.
[57] E.C.P. Neto, S. Dadkhah, R. Ferreira, A. Zohourian, R. Lu, A.A. Ghorbani, CICIoT2023: A real-time dataset and benchmark for large-scale attacks in IoT environment, Sensors 23 (13) (2023) 5941.
[58] A. Mitrokotsa, M.R. Rieback, A.S. Tanenbaum, Classifying RFID attacks and defenses, Information Systems Frontiers 12 (5) (2010) 491–505.
[59] L. Wallgren, S. Raza, T. Voigt, Routing attacks and countermeasures in the RPL-based Internet of Things, International Journal of Distributed Sensor Networks 9 (8) (2013) 794326.
[60] WPA2 Hole196 vulnerability: Exploits and remediation strategies, 2010. http://securedsolutions.com.my/pdf/WhitePapers/WPA2-Hole196-Vulnerability.pdf.
[61] M. Hamza, Common attacks on routing protocols and how to mitigate them, November 2020 [06-04-2023], https://hamzamhirsi.medium.com/common-attacks-on-routing-protocols-and-how-to-mitigate-them-11ec0cad08d7.
[62] N. Chaabouni, M. Mosbah, A. Zemmari, C. Sauvignac, P. Faruki, Network intrusion detection for IoT security based on learning techniques, IEEE Communications Surveys & Tutorials 21 (3) (2019) 2671–2701.
[63] V.P. Singh, S. Jain, J. Singhai, Hello flood attack and its countermeasures in wireless sensor networks, International Journal of Computer Science Issues (IJCSI) 7 (3) (2010) 23.
[64] K. Hameed, S. Garg, M.B. Amin, B. Kang, A. Khan, A context aware information based clone node attack detection scheme in Internet of Things, Journal of Network and Computer Applications 197 (2022) 103271.
[65] L. Buttyán, J.-P. Hubaux, Security and cooperation in wireless networks: Thwarting malicious and selfish behavior in the age of ubiquitous computing, Cambridge University Press, 2007.
[66] G. Nakibly, M. Arov, Routing loop attacks using IPv6 tunnels, in: Proceedings of the 3rd USENIX conference on Offensive technologies, ACM, New York, 2009, pp. 1–7.
[67] S. Pamarthi, R. Narmadha, Literature review on network security in wireless mobile ad-hoc network for IoT applications: Network attacks and detection mechanisms, International Journal of Intelligent Unmanned Systems 10 (4) (2022) 482–506.
[68] A. Dizdar, What is DNS attack and how to prevent them, May 2022 [06-04-2023], https://brightsec.com/blog/dns-attack/.
[69] E. Borges, The most popular types of DNS attacks, November 2022 [06-04-2023], https://securitytrails.com/blog/most-popular-types-dns-attacks.
[70] Fast flux networks, August 2023 [08-08-2023], https://www.fortinet.com/resources/cyberglossary/fast-flux-networks.
[71] S. Rizvi, A. Kurtz, J. Pfeffer, M. Rizvi, Securing the Internet of Things (IoT): A security taxonomy for IoT, in: Proceedings of the 2018 17th IEEE International Conference on Trust, Security and Privacy in Computing and Communications/12th IEEE International Conference on Big Data Science and Engineering (TrustCom/BigDataSE), IEEE, Piscataw
[72] R. Taylor, Four major DNS attack types and how to mitigate them, May 2023 [08-08-2023], https://bluecatnetworks.com/blog/four-major-dns-attack-types-and-how-to-mitigate-them/.

[73] What is a zero-day attack? - definition and explanation, 2023 [08-08-2023], https://www.kaspersky.com/resource-center/definitions/zero-day-exploit.
[74] CVE [06-04-2023], https://cve.mitre.org/index.html.
[75] National institute of standards and technology [06-04-2023], https://www.nist.gov/.
[76] Known Exploited Vulnerabilities Catalog [06-04-2023], https://www.cisa.gov/known-exploited-vulnerabilities-catalog.
[77] CVEdetails.com [06-04-2023], https://www.cvedetails.com/.
[78] Latest zero days [06-04-2023], https://www.zero-day.cz/.
[79] Bugcrowd glossary section, code injection, August 2023 [08-08-2023], https://www.bugcrowd.com/glossary/code-injection/.
[80] H.A. Noman, O.M.F. Abu-Sharkh, Code injection attacks in wireless-based Internet of Things (IoT): A comprehensive review and practical implementations, Sensors 23 (13) (2023) 6067.
[81] Remote code execution vs. remote command execution vs. code injection vs. command injection vs. RCE, April 2022 [08-08-2023], https://hakluke.com/remote-code-execution-vs-remote-command-execution-vs-code-injection-vs-command-injection-vs-rce/.
[82] O. Barinova, HQL injection exploitation in MySQL, July 2019 [08-08-2023], https://www.trustwave.com/en-us/resources/blogs/spiderlabs-blog/hql-injection-exploitation-in-mysql/.
[83] Hibernate query language (HQL) injection, August 2023 [08-08-2023], https://www.acunetix.com/vulnerabilities/web/hibernate-query-language-hql-injection/.
[84] K. Greshake, S. Abdelnabi, S. Mishra, C. Endres, T. Holz, M. Fritz, Not what you've signed up for: Compromising real-world LLM-integrated applications with indirect prompt injection, in: Proceedings of the 16th ACM Workshop on Artificial Intelligence and Security, ACM, New York, 2023, pp. 79–90.
[85] What is LDAP injection, August 2023 [08-08-2023], https://www.techslang.com/definition/what-is-ldap-injection/.
[86] T. Andrzej Nidecki, O. Dinc, Remote code execution (RCE), August 2023 [08-08-2023], https://www.invicti.com/learn/remote-code-execution-rce/.
[87] I. Zeifman, What is a buffer overflow, attack examples and prevention methods, March 2023, [08-08-2023], https://sternumiot.com/iot-blog/buffer-overflow-attack/.
[88] N. Sivasankari, S. Kamalakkannan, Detection and prevention of man-in-the-middle attack in IoT network using regression modeling, Advances in Engineering Software 169 (2022) 103126.
[89] H. Fereidouni, O. Fadeitcheva, M. Zalai, IoT and man-in-the-middle attacks, 2023. https://arxiv.org/pdf/2308.02479v1.pdf.
[90] Password cracking 101: Attacks & defenses explained, March 2022 [08-08-2023], https://www.beyondtrust.com/blog/entry/password-cracking-101-attacks-defenses-explained.
[91] J. Deogirikar, A. Vidhate, Security attacks in IoT: A survey, in: Proceedings of the 2017 International Conference on I-SMAC (IoT in Social, Mobile, Analytic s and Cloud) (I-SMAC), IEEE, Piscataway, 2017, pp. 32–37.
[92] Tokenisation vs. encryption: Which one is better? [08-08-2023], https://www.vaultree.com/blog/tokenisation-vs-encryption-which-one-is-better/, April 2023.
[93] J. Shah, What is liveness detection? Preventing biometric spoofing, May 2023 [08-08-2023], https://www.1kosmos.com/authentication/liveness-detection/.
[94] K. Tsiknas, D. Taketzis, K. Demertzis, C. Skianis, Cyber threats to industrial IoT: A survey on attacks and countermeasures, IoT 2 (1) (2021) 163–186.
[95] The art of reverse engineering: Uncovering IoT network security vulnerabilities, August 2023 [08-08-2023], https://simeononsecurity.ch/articles/the-art-of-reverse-engineering/.
[96] What you should know about reverse engineering IoT devices, December 2021 [08-08-2023], https://www.redalertlabs.com/blog/what-you-should-know-about-reverse-engineering-iot-devices?categoryId =267021.
[97] O. Shwartz, Y. Mathov, M. Bohadana, Y. Elovici, Y. Oren, Reverse engineering IoT devices: Effective techniques and methods, IEEE Internet of Things Journal 5 (6) (2018) 4965–4976.
[98] What is a logic bomb?, August 2023 [08-08-2023], https://www.malwarebytes.com/logic-bomb.
[99] R. Awati, L. Fitzgibbons, Logic bomb, August 2021 [08-08-2023], https://www.techtarget.com/searchsecurity/definition/logic-bomb.
[100] T. O'Gorman, A primer on IoT security risks, February 2017 [06-04-2023], https://securityintelligence.com/a-primer-on-iot-security-risks/.
[101] S. Nizam, Z.-A. Ibrahim, F.A. Rahim, H. Fadzil, H. Abdullah, M. Mustaffa, Forensic analysis on false data injection attack on IoT environment, International Journal of Advanced Computer Science and Applications(IJACSA) 12 (10) (2021).
[102] Z. Ling, J. Luo, Y. Xu, C. Gao, K. Wu, X. Fu, Security vulnerabilities of Internet of Things: A case study of the smart plug system, IEEE Internet of Things Journal 4 (6) (2017) 1899–1909.
[103] A. Einorytė, What are IoT attacks, July 2023 [06-04-2023], https://nordvpn.com/blog/iot-attacks/.
[104] S. Babar, A. Stango, N. Prasad, J. Sen, R. Prasad, Proposed embedded security framework for Internet of Things (IoT), in: Proceedings of the 2011 2nd International Conference on Wireless Communication, Vehicular Technology, Information Theory and Aerospace & Electronic Systems Technology (Wireless VITAE), IEEE, Piscataway, 2011, pp. 1–5.
[105] Privacy vs. security: What's the difference? [06-04-2023], https://us.norton.com/blog/privacy/privacy-vs-security-whats-the-difference, January 2023.
[106] Difference between security and privacy, August 2023 [06-04-2023], https://www.javatpoint.com/security-vs-privacy.
[107] E.A. Shammar, A.T. Zahary, The Internet of Things (IoT): A survey of techniques, operating systems, and trends, Library Hi Tech 38 (1) (2020) 5–66.
[108] C. Liebchen, Advancing memory-corruption attacks and defenses, February 2018. http://tuprints.ulb.tu-darmstadt.de/8090/.
[109] M. Pal, P. Dey, V. Dokania, Memory corruption-basic attacks and counter measures, International Journal of Engineering Science and Computing 6 (2016) 3511.
[110] M. Luqman, Integer overflow attack and prevention, January 2021 [06-04-2023], https://www.securecoding.com/blog/integer-overflow-attack-and-prevention/.
[111] What is a stack overflow attack, November 2021 [06-04-2023], https://my.f5.com/manage/s/article/K40055166.
[112] What is a heap overflow attack, November 2021 [06-04-2023], https://my.f5.com/manage/s/article/K53293427.
[113] B.L. Bergmans, What is privilege escalation? [06-04-2023], https://www.crowdstrike.com/cybersecurity-101/privilege-escalation/, June 2022.
[114] A.A. Zaid, M.H. Alalfi, A. Miri, A model-driven-engineering approach for detecting privilege escalation in IoT Systems, 2022. https://arxiv.org/pdf/2205.11406.pdf.
[115] K. Yu, L. Tan, S. Mumtaz, S. Al-Rubaye, A. Al-Dulaimi, A.K. Bashir, F.A. Khan, Securing critical infrastructures: Deep-learning-based threat detection in IIoT, IEEE Communications Magazine 59 (10) (2021) 76–82.
[116] CVE-2017-1000251 [06-04-2023], https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2017-1000251.
[117] CVE-2017-1000410 [06-04-2023], https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2017-1000410.
[118] What is an IoT botnet, August 2023 [06-04-2023], https://www.trendmicro.com/vinfo/us/security/definition/iot-botnet.
[119] A 2022 Guide to IoT Protocols and Standards, August 2023 [06-04-2023], https://www.particle.io/iot-guides-and-resources/iot-protocols-and-standards/.
[120] S. Elhadi, A. Marzak, N. Sael, S. Merzouk, Comparative study of IoT protocols, 2018. http://api.semanticscholar.org/CorpusID:69813998.
[121] J. Mocnej, A. Pekár, W. Seah, E. Kajáti, I. Zolotová, Internet of Things unified protocol stack, Acta Electrotechnic a et Informatica 19 (2) (2019) 24–32.
[122] J. Teel, Comparison of wireless technologies: Bluetooth, WiFi, BlE, Zigbee, Z-Wave, 6LoWPAN, NFC, WiFi Direct, GSM, LTE, LoRa, NB-IoT, and LTE-M [21-08-2023], https://predictabledesigns.com/wireless_technologies_bluetooth_wifi_zigbee_gsm_lte_lora_nb-iot_lte-m/, July 2022.
[123] N. Agnihotri, What are different types of IoT networks? [21-08-2023], https://www.engineersgarage.com/types-of-iot-networks-cellular-lpwan-lan-mesh/, August 2023.
[124] I. Muts, 4 types of IoT networks: Overview and use cases, August 2023 [21-08-2023], https://euristiq.com/types-of-iot-networks/.
[125] Selecting right low power short-range protocols for your IoT solutions, August 2022 [21-08-2023], https://www.embeddedtechconvention.com/news/49070/.

[126] 6 leading types of IoT wireless tech and their best use cases, October 2018 [21-08-2023], https://behrtech.com/blog/6-leading-types-of-iot-wireless-tech-and-their-best-use-cases/.
[127] Types of IoT networks, March 2023 [06-04-2023], https://www.tutorialspoint.com/types-of-iot-networks.
[128] M.M. Hossain, M. Fotouhi, R. Hasan, Towards an analysis of security issues, challenges, and open problems in the Internet of Things, in: Proceedings of the 2015 IEEE World Congress on Services, IEEE, Piscataway, 2015, pp. 21–28.
[129] M. Nawir, A. Amir, N. Yaakob, O.B. Lynn, Internet of Things (IoT): Taxonomy of security attacks, in: Proceedings of the 2016 3rd International Conference on Electronic Design (ICED), IEEE, Piscataway, 2016, pp. 321–326.
[130] A. Jha, IoT security - Part 21 (famous IoT attacks & vulnerabilities), December 2020 [06-04-2023], https://payatu.com/blog/iot-attacks-and-vulnerabilities/.
[131] T. Dunlap, The 5 worst examples of IoT hacking and vulnerabilities in recorded history, June 2020, [06-04-2023], https://www.iotforall.com/5-worst-iot-hacking-vulnerabilities.
[132] 7 examples of real-life data breaches caused by insider threats, March 2023 [06-04-2023], https://www.ekransystem.com/en/blog/real-life-examples-insider-threat-caused-breaches.
[133] K. Lounis, M. Zulkernine, Attacks and defenses in short-range wireless technologies for IoT, IEEE Access 8 (2020) 88892–88932.
[134] Y. Meidan, M. Bohadana, Y. Mathov, Y. Mirsky, A. Shabtai, D. Breitenbacher, Y. Elovici, N-baiot—Network-based detection of IoT botnet attacks using deep autoencoders, IEEE Pervasive Computing 17 (3) (2018) 12–22.
[135] N. Koroniotis, N. Moustafa, E. Sitnikova, B. Turnbull, Towards the development of realistic botnet dataset in the Internet of Things for network forensic analytics: Bot-IoT dataset, 2018. https://arxiv.org/pdf/1811.00701.pdf.
[136] Y. Mirsky, T. Doitshman, Y. Elovici, A. Shabtai, Kitsune: An ensemble of autoencoders for online network intrusion detection, 2018. https://arxiv.org/pdf/1802.09089.pdf.
[137] H. Kang, D.H. Ahn, G.M. Lee, J.D. Yoo, K.H. Park, H.K. Kim, IoT network intrusion dataset, 2019, https://doi.org/10.21227/q70p-q449.
[138] A. Guerra-Manzanares, J. Medina-Galindo, H. Bahsi, S. Nõmm, MedBIoT: Generation of an IoT botnet dataset in a medium-sized IoT network, in: Proceedings of the 6th International Conference on Information Systems Security and Privacy ICISSP, SciTePress, 2020, pp. 207–218.
[139] S. Garcia, A. Parmisano, M.J. Erquiaga, IoT-23: A labeled dataset with malicious and benign IoT network traffic, January 2020, https://doi.org/10.5281/zenodo.4743746.
[140] R. Ahmad, I. Alsmadi, W. Alhamdani, L. Tawalbeh, A comprehensive deep learning benchmark for IoT IDS, Computers & Security 114 (2022) 102588.
[141] I. Vaccari, G. Chiola, M. Aiello, M. Mongelli, E. Cambiaso, MQTTset, a new dataset for machine learning techniques on MQTT, Sensors 20 (22) (2020) 6578.
[142] M.A. Ferrag, O. Friha, D. Hamouda, L. Maglaras, H. Janicke, Edge-IIoTset: A new comprehensive realistic cyber security dataset of IoT and IIoT applications for centralized and federated learning, IEEE Access 10 (2022) 40281–40306.
[143] A. Chatterjee, B.S. Ahmed, IoT anomaly detection methods and applications: A survey, Internet of Things 19 (2022) 100568.
[144] Y.M. Galvão, V.A. Albuquerque, B.J.T. Fernandes, M.J.S. Valença, Anomaly detection in smart houses: Monitoring elderly daily behavior for fall detecting, in: Proceedings of the 2017 IEEE Latin American Conference on Computational Intelligence (LA-CCI), IEEE, Piscataway, 2017, pp. 1–6.
[145] H. Lu, Y. Li, S. Mu, D. Wang, H. Kim, S. Serikawa, Motor anomaly detection for unmanned aerial vehicles using reinforcement learning, IEEE Internet Things Journal 5 (4) (2018) 2315–2322.
[146] T.D. Nguyen, S. Marchal, M. Miettinen, H. Fereidooni, N. Asokan, A.-R. Sadeghi, DÏot: A federated self-learning anomaly detection system for IoT, in: Proceedings of the 2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS), IEEE, Piscataway, 2019, pp. 756–767.
[147] M. Alsheikh, L. Konieczny, M. Prater, G. Smith, S. Uludag, The state of IoT security: Unequivocal appeal to cybercriminals, onerous to defenders, IEEE Consumer Electronics Magazine 11 (3) (2022) 59–68.
[148] M. Munir, S.A. Siddiqui, A. Dengel, S. Ahmed, DeepAnt: A deep learning approach for unsupervised anomaly detection in time series, IEEE Access 7 (2019) 1991–2005.
[149] P. Srikanth, An efficient approach for clustering and classification for fraud detection using bankruptcy data in IoT environment, International Journal of Information Technology 13 (6) (2021) 2497–2503.
[150] S. Asoba, S. Supekar, T. Tonde, J.A. Siddiqui, Advanced traffic violation control and penalty system using IoT and image processing techniques, in: Proceedings of the 2020 2nd International Conference on Innovative Mechanisms for Industry Applications (ICIMIA), IEEE, Piscataway, 2020, pp. 554–558.
[151] H. Li, P. Boulanger, A survey of heart anomaly detection using ambulatory electrocardiogram (ECG), Sensors 20 (5) (2020) 1461.
[152] I. Hafeez, M. Antikainen, A.Y. Ding, S. Tarkoma, IoT-keeper: Detecting malicious IoT network activity using online traffic analysis at the edge, IEEE Transactions on Network and Service Management 17 (1) (2020) 45–59.
[153] H.H.W.J. Bosman, G. Iacca, A. Tejada, H.J. Wörtche, A. Liotta, Ensembles of incremental learners to detect anomalies in ad hoc sensor networks, Ad Hoc Networks 35 (2015) 14–36.
[154] D. Wu, Z. Jiang, X. Xie, X. Wei, W. Yu, R. Li, LSTM learning with Bayesian and Gaussian processing for anomaly detection in industrial IoT, IEEE Transactions on Industrial Informatics 16 (8) (2020) 5244–5253.
[155] B. Sharma, L. Sharma, C. Lal, Anomaly detection techniques using deep learning in IoT: A survey, in: Proceedings of the 2019 International Conference on Computational Intelligence and Knowledge Economy (ICCIKE), IEEE, Piscataway, 2019, pp. 146–149.
[156] N.U. Sheikh, H. Rahman, S. Vikram, H. AlQahtani, A lightweight signature-based IDS for IoT Environment, 2018. https://arxiv.org/pdf/1811.04582.pdf.
[157] P. Maniriho, A.N. Mahmood, M.J.M. Chowdhury, A study on malicious software behaviour analysis and detection techniques: Taxonomy, current trends and challenges, Future Generation Computer Systems 130 (2022) 1–18.
[158] M.F. Razali, M.N. Razali, F.Z. Mansor, G. Muruti, N. Jami l, IoT honeypot: A review from researcher' s perspective, in: Proceedings of the 2018 IEEE Conference on Application, Information and Network Security (AINS), IEEE, Piscataway, 2018, pp. 93–98.
[159] What is SIEM, October 2023 [06-04-2023], https://www.ibm.com/topics/siem.
[160] E.B. Karbab, M. Debbabi, MalDy: Portable, data-driven malware detection using natural language processing and machine learning techniques on behavioral analysis reports, Digital Investigation 28 (2019) S77–S87.
[161] D. Xue, J. Li, T. Lv, W. Wu, J. Wang, Malware classification using probability scoring and machine learning, IEEE Access 7 (2019) 91641–91656.
[162] T. Kohonen, Self-organized formation of topologically correct feature maps, Biological cybernetics 43 (1) (1982) 59–69.
[163] D.C. Le, N. Zincir-Heywood, M.I. Heywood, Unsupervised monitoring of network and service behaviour using self organizing maps, Journal of Cyber Security and Mobility 8 (1) (2018) 15–52.
[164] S. Huda, J. Abawajy, M. Alazab, M. Abdollalihian, R. Islam, J. Yearwood, Hybrids of support vector machine wrapper and filter based framework for malware detection, Future Generation Computer Systems 55 (2016) 376–390.
[165] N. Asrafi, D.C.-T. Lo, R.M. Parizi, Y. Shi, Y.-W. Chen, Comparing performance of malware classification on automated stacking, in: Proceedings of the 2020 ACM Southeast Conference, ACM, New York, 2020, pp. 307–308.
[166] J. Moubarak, T. Feghali, Comparing machine learning techniques for malware detection, in: Proceedings of the 6th International Conference on Information Systems Security and Privacy, 2020, pp. 844–851.
[167] J. Suaboot, Z. Tari, A. Mahmood, A.Y. Zomaya, W. Li, Sub-curve HMM: A malware detection approach based on partial analysis of api call sequences, Computers & Security 92 (2020) 101773.
[168] A. Dhammi, M. Singh, Behavior analysis of malware using machine learning, in: Proceedings of the 2015 Eighth International Conference on Contemporary Computing (IC3), IEEE, Piscataway, 2015, pp. 481–486.
[169] T. Ghate, C. Pathade, C. Nirhali, K. Patil, N. Korade, Machine learning based malware detection: A boosting methodology, International Journal of Innovative Technology and Exploring Engineering (IJITEE) 9 (4) (2020) 2241–2245.

[170] H. Sayadi, H.M. Makrani, S.M.P. Dinakarrao, T. Mohsenin, A. Sasan, S. Rafatirad, H. Homayoun, 2SMaRT: A two-stage machine learning-based approach for run-time specialized hardware-assisted malware detection, in: Proceedings of the 2019 Design, Automation & Test in Europe Conference & Exhibition (DATE), IEEE, Piscataway, 2019, pp. 728–733.
[171] E.M. Alkhateeb, M. Stamp, A dynamic heuristic method for detecting packed malware using naive Bayes, in: Proceedings of the 2019 International Conference on Electrical and Computing Technologies and Applications (ICECTA), IEEE, Piscataway, 2019, pp. 1–6.
[172] M. Mambo, Foreword, IEICE transactions on fundamentals of electronics, Communications and Computer Sciences E100.A (1) (2017) 1–2.
[173] M. Schultz, E. Eskin, F. Zadok, S. Stolfo, Data mining methods for detection of new malicious executables, in: Proceedings of the 2001 IEEE Symposium on Security and Privacy, S&P 2001, IEEE, Piscataway, 2001, pp. 38–49.
[174] D. Oyen, B. Anderson, K. Sentz, C. Anderson-Cook, Order priors for Bayesian network discovery with an application to malware phylogeny, September 2017, https://doi.org/10.1002/sam.11364.
[175] K. Hughes, Y. Qu, A theoretical model: Using logistic regression for malware signature based detection, in: Proceedings of the The 10th International Conference on Dependable, Autonomic, and Secure Computing, DASC-2012), 2012.
[176] S.L.S. Darshan, M.A.A. Kumara, C. Jaidhar, Windows malware detection based on cuckoo sandbox generated report using machine learning algorithm, in: Proceedings of the 2016 11th International Conference on Industrial and Information Systems (ICIIS), IEEE, Piscataway, 2016, pp. 534–539.
[177] J. Suaboot, Z. Tari, A. Mahmood, A.Y. Zomaya, W. Li, Sub-curve HMM: A malware detection approach based on partial analysis of API call sequences, Computers & Security 92 (2020) 101773.
[178] Y. Zhang, C. Rong, Q. Huang, Y. Wu, Z. Yang, J. Jiang, Based on multi-features and clustering ensemble method for automatic malware categorization, in: Proceedings of the 2017 IEEE Trustcom/BigDataSE/ICESS, IEEE, Piscataway, 2017, pp. 73–82.
[179] Y. Fang, W. Zhang, B. Li, F. Jing, L. Zhang, Semi-supervised malware clustering based on the weight of bytecode and API, IEEE Access 8 (2020) 2313–2326.
[180] S. Hou, L. Chen, E. Tas, I. Demihovskiy, Y. Ye, Cluster-oriented ensemble classifiers for intelligent malware detection, in: Proceedings of the 2015 IEEE 9th International Conference on Semantic Computing (IEEE ICSC 2015), IEEE, Piscataway, 2015, pp. 189–196.
[181] S. Pai, F.D. Troia, C.A. Visaggio, T.H. Austin, M. Stamp, Clustering for malware classification, Journal of Computer Virology and Hacking Techniques 13 (2) (2017) 95–107.
[182] S.M.K. Raza, J. Caballero, Malware traffic classification: Evaluation of algorithms and an automated ground-truth generation pipeline, 2020. https://arxiv.org/pdf/2010.11627.pdf.
[183] C. Pascariu, I.-D. Barbu, Dynamic analysis of malware using artificial neural networks: Applying machine learning to identify malicious behavior based on parent process hirarchy, in: Proceedings of the 2017 9th International Conference on Electronics, Computers and Artificial Intelligence (ECAI), IEEE, Piscataway, 2017, pp. 1–5.
[184] K.O. Babaagba, S.O. Adesanya, A study on the effect of feature selection on malware analysis using machine learning, in: Proceedings of the 2019 8th International Conference on Educational and Information Technology, ACM, New York, 2019, pp. 51–55.
[185] L.E. Gonzalez, R.A. Vazquez, Malware classification using Euclidean distance and artificial neural networks, in: Proceedings of the 2013 12th Mexican International Conference on Artificial Intelligence, IEEE, Piscataway, 2013, pp. 103–108.
[186] C.W. Kim, NtMalDetect: A machine learning approach to malware detection using native API system calls, 2018. https://arxiv.org/pdf/1802.05412.pdf.
[187] J. Bai, J. Wang, Improving malware detection using multi-view ensemble learning, Security and Communication Networks 9 (17) (2016) 4227–4241.
[188] Y.A. Ahmed, B. Koçer, S. Huda, B.A.S. Al-rimy, M.M. Hassan, A system call refinement-based enhanced minimum redundancy maximum relevance method for ransomware early detection, Journal of Network and Computer Applications 167 (2020) 102753.
[189] L. Breiman, Random forests, Machine Learning 45 (1) (2001) 5–32.
[190] F. Ullah, Q. Javaid, A. Salam, M. Ahmad, N. Sarwar, D. Shah, M. Abrar, Modified decision tree technique for ransomware detection at runtime through API calls, Scientific Programming 2020 (2020) 8845833.
[191] M. Al-kasassbeh, M.A. Abbadi, A.M. Al-Bustanji, Lightgbm algorithm for malware detection, in: Intelligent Computing, Springer, Cham, 2020, pp. 391–403.
[192] F. Ahmed, H. Hameed, M.Z. Shafiq, M. Farooq, Using spatio-temporal information in API calls with machine learning algorithms for malware detection, in: Proceedings of the 2nd ACM Workshop on Security and Artificial Intelligence, ACM, New York, 2009, pp. 55–62.
[193] S.M. Bidoki, S. Jalili, A. Tajoddin, PbMMD: A novel policy based multi-process malware detection, Engineering Applications of Artificial Intelligence 60 (2017) 57–70.
[194] M. Ghiasi, A. Sami, Z. Salehi, Dynamic VSA: A Framework for malware detection based on register contents, Engineering Applications of Artificial Intelligence 44 (2015) 111–122.
[195] M. Alaeiyan, A. Dehghantanha, T. Dargahi, M. Conti, S. Parsa, A multilabel fuzzy relevance clustering system for malware attack attribution in the edge layer of cyber-physical networks, ACM Transactions on Cyber-Physical Systems 4 (3) (2020) 1–22.
[196] M.S. Abbasi, H. Al-Sahaf, I. Welch, Particle swarm optimization: A wrapper-based feature selection method for ransomware detection and classification, in: Proceedings of the Applications of Evolutionary Computation, Springer, Cham, 2020, pp. 181–196.
[197] S. Yuan, X. Wu, Deep learning for insider threat detection: Review, challenges and opportunities, Computers & Security 10 4 (2021) 102221.
[198] M.M. Najafabadi, F. Villanustre, T.M. Khoshgoftaar, N. Seliya, R. Wald, E. Muharemagic, Deep learning applications and challenges in big data analytics, Journal of Big Data 2 (1) (2015) 1–21.
[199] X. Wang, S.M. Yiu, A multi-task learning model for malware classification with useful file access pattern from API call sequence, 2016. https://arxiv.org/pdf/1610.05945.pdf.
[200] R. Pascanu, J.W. Stokes, H. Sanossian, M. Marinescu, A. Thomas, Malware classification with recurrent networks, in: Proceedings of the 2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, Piscataway, 2015, pp. 1916–1920.
[201] D. Nahmias, A. Cohen, N. Nissim, Y. Elovici, Deep feature transfer learning for trusted and automated malware signature generation in private cloud environments, Neural Networks 124 (2020) 243–257.
[202] C. Jindal, C. Salls, H. Aghakhani, K. Long, C. Kruegel, G. Vigna, Neurlux: Dynamic malware analysis without feature engineering, in: Proceedings of the 35th Annual Computer Security Applications Conference, ACM, New York, 2019, pp. 444–455.
[203] P. Dixit, S. Silakari, Deep learning algorithms for cybersecurity applications: A technological and status review, Computer Science Review 39 (2021) 100317.
[204] L. Xiaofeng, J. Fangshuo, Z. Xiao, Y. Shengwei, S. Jing, P. Lio, ASSCA: API sequence and statistics features combined architecture for malware detection, Computer Networks 157 (2019) 99–111.
[205] M. Sewak, S.K. Sahay, H. Rathore, An investigation of a deep learning based malware detection system, in: Proceedings of the 13th International Conference on Availability, Reliability and Security, ACM, New York, 2018, pp. 1–5.
[206] B. Kolosnjaji, A. Zarras, G. Webster, C. Eckert, Deep learning for classification of malware system call sequences, in: Proceedings of the AI 2016: Advances in Artificial Intelligence, Springer, Cham, 2016, pp. 137–149.
[207] R. Benchea, D.T. GavriluŢ, Combining restricted Boltzmann machine and one side perceptron for malware detection, in: Proceedings of the International conference on conceptual structures, Springer, Cham, 2014, pp. 93–103.
[208] O.E. David, N.S. Netanyahu, DeepSign: Deep learning for automatic malware signature generation and classification, in: Proceedings of the 2015 International Joint Conference on Neural Networks (IJCNN), IEEE, Piscataway, 2015, pp. 1–8.
[209] A. Pinhero, M.L. Anupama, P. Vinod, C.A. Visaggio, N. Aneesh, S. Abhijith, S. AnanthaKrishnan, Malware detection employed by visualization and deep neural network, Computers & Security 105 (2021) 102247.

[210] S. Tobiyama, Y. Yamaguchi, H. Shimada, T. Ikuse, T. Yagi, Malware detection with deep neural network using process behavior, in: Proceedings of the 2016 IEEE 40th Annual Computer Software and Applications Conference (COMPSAC), IEEE, Piscataway, 2016, pp. 577–582.
[211] R. Vinayakumar, M. Alazab, K.P. Soman, P. Poornachandran, S. Venkatraman, Robust intelligent malware detection using deep learning, IEEE Access 7 (2019) 46717–46738.
[212] S. Haiba, T. Mazri, Build a malware detection software for IoT network using machine learning, in: Proceedings of the 4th International Conference on Networking, Information Systems & Security, 2021, pp. 1–8.
[213] S. Torabi, M. Dib, E. Bou-Harb, C. Assi, M. Debbabi, A strings-based similarity analysis approach for characterizing IoT malware and inferring their underlying relationships, IEEE Networking Letters 3 (3) (2021) 161–165.
[214] A. Kumar, T.J. Lim, EDIMA: Early detection of IoT malware network activity using machine learning techniques, in: Proceedings of the 2019 IEEE 5th World Forum on Internet of Things, WF-IoT), IEEE, Piscataway, 2019, pp. 289–294.
[215] S. Nižetić, P. Šolić, D.L.I. Gonzalez-De, L. Patrono, Internet of Things (IoT): Opportunities, issues and challenges towards a smart and sustainable future, Journal of cleaner production 274 (2020) 122877.
[216] M. Hossain, S. Noor, Y. Karim, R. Hasan, IoTbed, A generic architecture for testbed as a service for Internet of Things-based systems, in: Proceedings of the 2017 IEEE 2nd International Congress on Internet of Things (ICIOT), IEEE, Piscataway, 2017, pp. 42–49.
[217] M.A. Hakim, H. Aksu, A.S. Uluagac, K. Akkaya, U-Pot, A honeypot framework for UPnP-based IoT devices, in: Proceedings of the 2018 IEEE 37th International Performance Computing and Communications Conference (IPCCC), IEEE, Piscataway, 2018, pp. 1–8.
[218] D. Vasan, M. Alazab, S. Venkatraman, J. Akram, Z. Qin, MTHAEL: Cross-architecture IoT malware detection based on neural network advanced ensemble learning, IEEE Transactions on Computers 69 (11) (2020) 1654–1667.
[219] H. HaddadPajouh, A. Dehghantanha, R. Khayami, K.-K.R. Choo, A deep recurrent neural network based approach for Internet of Things malware threat hunting, Future Generation Computer Systems 85 (2018) 88–96.
[220] J. Jeon, J.H. Park, Y.-S. Jeong, Dynamic analysis for IoT malware detection with convolution neural network model, IEEE Access 8 (2020) 96899–96911.
[221] J. Jeon, B. Jeong, S. Baek, Y.-S. Jeong, Hybrid malware detection based on Bi-LSTM and SPP-Net for smart IoT, IEEE Transactions on Industrial Informatics 18 (7) (2021) 4830–4837.
[222] G. Rajendran, R.S.R. Nivash, P.P. Parthy, S. Balamurugan, Modern security threats in the Internet of Things (IoT): Attacks and countermeasures, in: Proceedings of the 2019 International Carnahan Conference on Security Technology (ICCST), IEEE, Piscataway, 2019, pp. 1–6.

Tinshu Sasi is a Master's student at the University of New Brunswick's (UNB) Faculty of Computer Science in Canada. In 2014, he earned a bachelor's degree in computer science from Manav Rachna International University in India. He also has over 8 years of experience working as a Senior Software Engineer developing enterprise-level software solutions. Cybersecurity, the Internet of Things, machine learning, and deep learning are among his key research interests.

Arash Habibi Lashkari is a Canada Research Chair (CRC) in Cybersecurity. He is a Senior member of IEEE and an Associate Professor at York University. Before this, he was an Associate Professor at the Faculty of Computer Science, University of New Brunswick (UNB), and the Research Coordinator of the Canadian Institute for Cybersecurity (CIC). His research focuses on cyber threat modelling and detection, malware analysis, big data security, internet traffic analysis, and cybersecurity dataset generation. Dr. Lashkari has over 25 years of teaching experience spanning several international universities and was responsible for designing the first cybersecurity Capture the Flag (CTF) competition for post-secondary students in Canada. He has been the recipient of 15 awards at international computer security competitions, including three gold awards, and was recognized as one of Canada's Top 150 Researchers for 2017. In 2020, Dr. Lashkari was recognized with the University of New Brunswick's prestigious Teaching Innovation Award for his personally-created teaching methodology, the Think-Que-Cussion Method. He is the author of ten published books and more than 110 academic articles on a variety of cybersecurity-related topics and the co-author of the national award-winning article series “Understanding Canadian Cybersecurity Laws”, which was recently recognized with a Gold Medal at the 2020 Canadian Online Publishing Awards, remotely held in 2021.

Rongxing Lu is a University Research Scholar and professor at the University of New Brunswick's (UNB) Faculty of Computer Science (FCS). From April 2013 until August 2016, he was an assistant professor at Nanyang Technological University's (NTU) School of Electrical and Electronic Engineering in Singapore. From May 2012 until April 2013, Rongxing Lu was a Postdoctoral Fellow at the University of Waterloo. He received the most prestigious “Governor General's Gold Medal” when he received his PhD degree from the Department of Electrical & Computer Engineering, University of Waterloo, Canada, in 2012; and in 2013, he won the 8th IEEE Communications Society (ComSoc) Asia Pacific (AP) Outstanding Young Researcher Award. Dr. Lu is a member of the IEEE. His research interests include applied cryptography, privacy-enhancing technologies, and the security and privacy of IoT-Big Data. Dr. Lu now holds many positions, including the Mastercard IoT Research Chair, the Chair of IEEE ComSoc CIS-TC (Communications and Information Security Technical Committee), and the founding Co-chair of IEEE TEMS Blockchain and Distributed Ledgers Technologies Technical Committee (BDLT-TC).

Pulei Xiong is a Research Officer in the Cybersecurity team at the National Research Council Canada. His current research interests focus on the study of robust machine learning, encompassing offensive, defensive, and assessment methods for developing and deploying machine learning systems that are resilient against adversarial attacks throughout the entire machine learning pipeline. Currently, he is the Principal Investigator (PI) leading several research projects in this area, collaborating with experts from academia and industry. In addition to his work in robust machine learning, his research interests include privacy-preserving technologies and their application, and security compliance for emerging technologies. Prior to joining NRC, Pulei gained extensive experience in the cybersecurity industry. He is widely recognized as a cybersecurity consultant, renowned for his leadership in developing the Protection Profile for Mobile Devices, which stands as the first industrial security standard of its kind for mobile computing. Pulei has held the certification of GIAC Mobile Device Security Analyst (GMOB) since 2016.

Shahrear Iqbal is a Research Officer at the National Research Council (NRC) Canada. He received his PhD in Cybersecurity from Queen's University, Kingston, ON, Canada 2017. Before joining NRC, he was a researcher at Security Compass, Toronto. He is a specialist in software security in the areas of mobile, IoT, and connected devices. Previously, he was a faculty member in the Department of Computer Science and Engineering (CSE) at Bangladesh University of Engineering and Technology (BUET). He is currently doing research on the security of connected devices with a focus on context-aware authentication and automated defence mechanism.