AntibIoTic: the Fog-enhanced distributed security system to protect the (legacy) Internet of Things

De Donno, Michele; Fafoutis, Xenofon; Dragoni, Nicola

Published in:
Journal of Computer Security

Link to article, DOI:
10.3233/JCS-210027

Publication date:
2022

Document Version
Peer reviewed version

AntibIoTic: the Fog-enhanced distributed security system to protect the (legacy) Internet of Things

Michele De Donno [a], Xenofon Fafoutis [a] and Nicola Dragoni [a,*]
[a] *DTU Compute, Technical University of Denmark, Denmark*
E-mails: mido@dtu.dk, xefa@dtu.dk, ndra@dtu.dk

Abstract. The Internet of Things (IoT) is evolving our society; however, the growing adoption of IoT devices in many scenarios brings security and privacy implications. Current security solutions are either unsuitable for every IoT scenario or provide only partial security. This paper presents `AntibIoTic` 2.0, a distributed security system that relies on Fog computing to secure IoT devices, including legacy ones. The system is composed of a backbone, made of core Fog nodes and Cloud server, a Fog node acting at the edge as the gateway of the IoT network, and a lightweight agent running on each IoT device. The proposed system offers fine-grained, host-level security coupled with network-level protection, while its distributed nature makes it scalable, versatile, lightweight, and easy to deploy, also for legacy IoT deployments. `AntibIoTic` 2.0 can also publish anonymized and aggregated data and statistics on the deployments it secures, to increase awareness and push cooperations in the area of IoT security. This manuscript recaps and largely expands previous works on `AntibIoTic`, providing an enhanced design of the system, an extended proof-of-concept that proves its feasibility and shows its operation, and an experimental evaluation that reports the low computational overhead it causes.

Keywords: Security System, Internet of Things, Fog Computing

1. Introduction

The Internet of Things (IoT) can be defined as a network of computing devices, such as vehicles, home appliances, sensors, and industrial robots, able to exchange data over the Internet without human interaction.

Statista estimated the number of connected IoT devices, also referred to as *things*, to be around 35 billion by 2021, and more than doubled by 2025 [Sta16]. Also, according to Cisco [Cis18], between 2017 and 2022, the overall IP traffic will grow at a Compound Annual Growth Rate (CAGR) of 26%, culminating in 2022 with more than 80% of traffic expected to be driven by non-PC devices, 51% of which being generated by Machine-to-Machine (M2M) connections. This increasing interconnection of a large number of "smart" devices in different sectors, such as smart cities, smart buildings, Industry 4.0, healthcare, smart vehicles, and so on, is rapidly and inevitably evolving our society, both for consumer and industrial actors, leading to new business models and enhancing user experience.

[*]Corresponding author. E-mail: ndra@dtu.dk.

However, the recent IoT (r)evolution comes with several security and privacy implications. Many are the security threats raised by the Internet of Things and affecting the authentication, confidentiality, integrity, availability, privacy, and non-repudiation of the IoT services [KBL18, MCZ+19, NBHC+19]. Indeed, countless have been cases of IoT devices reported to be poorly secured and thus easy prey of cyberattacks [DGM17, GDS16, FTAK+20, GDDD18]. Amongst the others, Distributed Denial of Service (DDoS) attacks are surely one of the most challenging threats to face [DDDGS17]. As a matter of fact, security experts still remember 2016 as the year of Mirai: the IoT malware that changed the world perception of IoT security. The attacker compromised about 600.000 IoT devices using a set of around 60 default username/password pairs. Later, in October 2016, this Mirai botnet was used to attack, amongst others, the Domain Name System (DNS) provider Dyn reaching a peak of traffic of 1.2 Tbps [DDDGS18].

Today, the situation is still critical. The plethora of insecure IoT devices quickly being deployed worldwide are often used as the source to generate large DDoS attacks [New20b, New20a], leading to the first quarter of 2020 characterized by a rise in the number of DDoS attacks of more than 542% when compared to the end of 2019, as documented by Nexusguard [Nex20]. It is thus clear that security is still a crucial aspect to consider at every step of the IoT lifecycle, ranging from the design and manufacturing of secure IoT devices to the conscious use of new and enhanced IoT services.

Recently, Fog computing has also attracted more and more attention, especially when related to the IoT [KS20]. Fog computing is a relatively new paradigm born from the need to overcome the challenges that the IoT (r)evolution has posed to Cloud computing, bridging the gap between Cloud and IoT [BMZA12]. Acting as a sort of middleware between Cloud and IoT, Fog computing can improve current solutions in several respects, including scalability, interoperability, Quality of Service (QoS), network bandwidth, latency, location awareness, and so on [Gro17]. Among others, IoT (in)security is one of the main challenges that Fog computing promises to help solving.

Problem statement. Security solutions for IoT deployments often rely on specific hardware components, such as Hardware Security Modules (HSM) and Trusted Execution Environments (TEE), to ensure host-level security [Jin19], and methods such as Intrusion Detection Systems (IDS) [EAH18] to monitor for network-level threats. When combined, these approaches represent a solid security defence for future IoT deployments, especially when integrated from the beginning of the IoT lifecycle, and can help reach a high-security level both at the network and the device level. However, in some IoT settings, devices are not equipped with ad-hoc security hardware, but security needs to be granted. For instance, some Industrial IoT (IIoT) implementations, also referred to as brownfield IIoT, include legacy industrial machinery and an infrastructure not initially designed with connectivity in mind, but later transformed to support the interconnection with new solutions and components [AHdHS19, Con16]. In these IoT deployments, devices are often resource-constrained and offer very limited hardware support, but they are critical assets that need to be deployed for long periods (even decades). For such devices, a minimum security level has to be ensured, without disrupting existing business processes with security controls and events [Con16].

As another example, consider consumer IoT deployments, here referred to as networks of connected IoT devices having relationships to associated services and used by the consumer, typically in the home or as electronic wearables [24]. In such environments, it is common to have legacy devices, such as IP cameras, baby monitors, and smartwatches, with limited hardware support and no possibility for updates, that transmit, process, and store large quantities of sensitive information which needs to be secured [FTAK+20].

In those IoT scenarios where legacy devices play a crucial role, adopting network security solutions is often the most effective technique to grant a minimum security level while integrating legacy endpoints in modern IoT systems [Con19]. However, while this approach is often quick and cheap to implement and can provide a consistent level of security across heterogeneous devices, it focuses on the network level and does not ensure device-level security with a fine-level control on the endpoints [Con16].

The security solution presented in this paper, namely `AntibIoTic`, aims at addressing the shortage of comprehensive security solutions able to grant, along with good network-level security, fine-grained device-level protection, also suitable for legacy IoT deployments.

In this manuscript, the terms `AntibIoTic`, `AntibIoTic` 1.0, and `AntibIoTic` 2.0 are used according to software versioning common practices. Specifically, the term `AntibIoTic` refers to the security system in general, without a specific reference to a single version. The terms `AntibIoTic` 1.0 and `AntibIoTic` 2.0 refer respectively to the first and second version of the proposed security solution. To give an example, in an analogy to a popular operating system like Windows, the term `AntibIoTic` is equivalent to the term Windows, while `AntibIoTic` 1.0 and `AntibIoTic` 2.0 are comparable to the terms Windows 8 and Windows 10.

Contribution of the paper. This paper presents `AntibIoTic` 2.0, a distributed security system that relies on Fog computing to protect IoT endpoints, including legacy ones. The system is composed of a backbone, made of core Fog nodes and Cloud servers, a Fog node at the edge acting as the network gateway of the IoT network, and an agent running on each IoT device. The `AntibIoTic` backbone coordinates and supports the security operations performed at the edge by the `AntibIoTic` gateway and the `AntibIoTic` agent to secure each IoT deployment, while publishing useful data and statics accessible from the community. `AntibIoTic` 2.0, due to its distributed nature, can combine fine-grained device-level security with network-level security, while keeping the overhead on each IoT endpoint extremely low. The solution has also been designed to be versatile, scalable, and easy to deploy (also in legacy IoT settings), thanks to the integration with Fog computing. An overview of `AntibIoTic` is depicted in Figure 1.

The work presented in this manuscript summarizes and expands previous papers on `AntibIoTic` [DDDGM16, DDD19, DDFD19], largely enhancing and improving the solution with respect to design, implementation, and evaluation. This paper offers the following main improvements with respect to previous works on `AntibIoTic`.

- *Full integration of `AntibIoTic` with the Fog computing N-tier architecture.* This paper provides a new enhanced design of the entire `AntibIoTic` architecture that includes a backbone composed of core Fog nodes and Cloud servers. It describes each component along with its internal modules and its functions.
- *Extended Proof-of-Concept.* This paper offers an extended proof-of-concept of `AntibIoTic` that involves the use of two additional IoT devices (three in total) and the implementations of new features, such as: the creation and transmission of reports about the security status of the IoT devices, the continuous transmission of keep-alive messages and security status updates from the IoT endpoints, and the automatic upload of the agent on each IoT host.
- *Video demo.* Along with the proof-of-concept, an example of operation for `AntibIoTic` in a real IoT setting is presented that has been recorded and published as a video demo [DD20a].

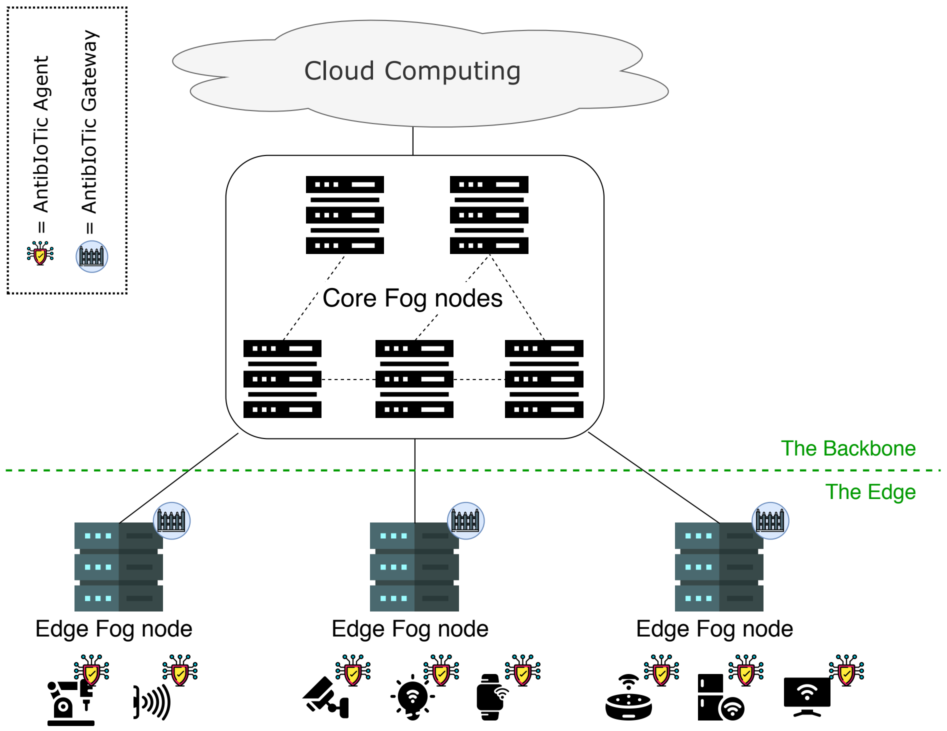


Fig. 1. `AntibIoTic` 2.0: the distributed Fog-enhanced security system that secures the Internet of Things. It is composed of an agent running on each IoT device and a Fog node acting as the gateway for the IoT network; the `AntibIoTic` gateway is connected to the backbone of `AntibIoTic` which provides coordination and support.

- *Evaluation and analysis.* This paper provides a detailed evaluation and analysis of the resources used by `AntibIoTic` when running on IoT devices, showing that the solution is lightweight and suitable also for legacy IoT endpoints.
- *List of services.* A high-level list of services offered by `AntibIoTic` is presented, grouped by component.
- *Enhanced illustrations.* New enhanced illustrations of `AntibIoTic` are provided to better describe the solution and clarify the structure and interactions of each component.

Outline of the paper. The rest of this paper is organized as follows. Section 2 provides the background knowledge necessary to understand this work. Section 3 gives a first high-level description of `AntibIoTic`, while Section 4 and 5 provide more details respectively on the design and deployment of the solution. Section 6 presents a proof-of-concept implementation of `AntibIoTic` 2.0 and Section 7 contains an evaluation of the resources `AntibIoTic` needs to run on IoT devices. Finally, Section 8 reviews related work and Section 9 wraps up the paper. Appendix A contains a demo of `AntibIoTic` 2.0.

2. Background

This section provides some concepts necessary for a full understanding of this work. First, it defines the main scope of `AntibIoTic` within the broad IoT landscape. Then, it aligns the reader with the authors' vision on Fog computing. Finally, it provides an overview of the previous version of `AntibIoTic`, with related motivation for its evolution.

2.1. Internet of Things: definition and scope

To date, the definition of Internet of Things is still discussed by the scientific community and, over the years, many have been the attempts to provide a formalization for it [A+09, AIM10, Sun20, Kha20, AAA+20]. In this work, the Internet of Things is defined as *a network of computing devices, such as vehicles, home appliances, sensors, and industrial robots, able to exchange data over the Internet without human interaction.*

Even after agreeing on a definition for the IoT, determining whether a specific device is considered a *thing* is still a challenging task. Over the years, computational, communication, and storage capabilities have become more accessible and often included in embedded devices. It is thus difficult today to use limited available resources as the key parameter to draw a clear line between *things* and other Internet-connected devices. For instance, on the one side, sensing devices with very limited resources (such as temperature, light, humidity, or proximity sensors) are today used in numerous IoT applications (such as Smart Home, Smart City, Smart Agriculture, and Smart Health, just to mention a few) [SG19]. On the other side, more powerful devices like Single Board Computers (e.g., Raspberry Pi) and Mini PCs (e.g., Intel NUC) are also being increasingly used in IoT deployments [JACSC16, WBD+17]. Thus, this raises the question: "where should one draw the line for `AntibIoTic`?"

This paper focuses on a class of devices that are referred to as *legacy* IoT, and that delimits the main scope of `AntibIoTic`. The scope considers *legacy* an IoT device that is equipped with outdated software and/or hardware but which is still in use due to its critical role in a specific scenario (e.g., healthcare, industrial settings) or its wide use (e.g., popular consumer devices). Also, the term refers to those devices not originally designed with connectivity features but later adapted to be connected to the network, such as brownfield IIoT systems [AHdHS19, Con16]. Legacy devices are typically difficult or impossible to update, either because no longer supported by the manufacturer or because not built with updates in mind from the beginning, and they typically lack the newest ad-hoc security components, such as TEE or HSM [Jin19], which limits their support to modern security solutions.

Please note that this does not want to be a strict selection of what devices can or cannot be protected by `AntibIoTic`, rather a baseline to help the reader in understanding the full potential and rationale behind the proposed solution. Challenging the scope of `AntibIoTic` is encouraged and expected.

2.2. Fog computing

Fog computing is an emerging computing paradigm originally thought to help in addressing the challenges that the IoT transformation posed to Cloud computing. It first appeared in the literature in 2012 [BMZA12] and it has soon become a popular research topic. To date, the exact definition of Fog computing is still debated in the literature and it is often overlapped with similar paradigms, such as Mobile Edge computing, Mobile Cloud computing, Edge computing, and so on. Nevertheless, a thorough analysis of Fog computing and its related paradigms is out of the scope of this paper. Interested readers can find such analysis in [DDTD19].

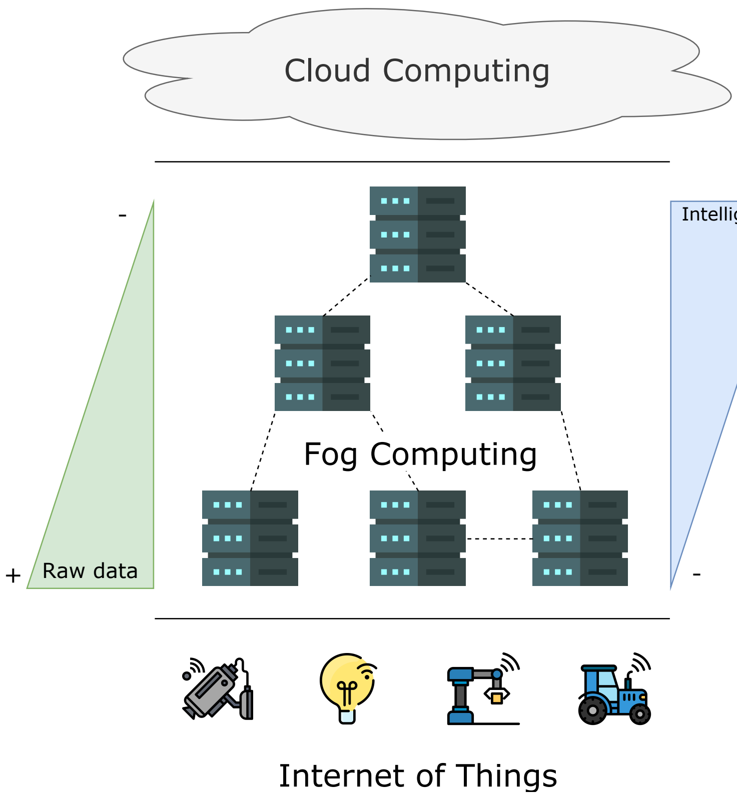


Fig. 2. Fog computing N-tier architecture (inspired from [Ass18]).

This section presents the definition and architectural model of Fog computing that is assumed in the rest of this work.

2.2.1. Definition

As defined in the IEEE standard 1934-2018, Fog computing is "a horizontal, system-level architecture that distributes computing, storage, control and networking functions closer to the users along a cloud-to-thing continuum" [Ass18]. Fog computing should thus be considered as an extension (not a replacement) of Cloud computing that bridges the gap between Cloud and IoT by providing services such as computing, storage, and networking, along the cloud-to-thing continuum. In this scenario, a Fog node is similar to a server in Cloud computing and can be defined as "the physical and logical network element that implements Fog computing services" [Ass18].

2.2.2. Architectural model

Fog computing is based on a hierarchical architecture referred to as N-tier architecture [Ass18]. It is an expansion of the 3-layer architecture [DDTD19] composed of Cloud computing, Fog computing, and Internet of Things, where the Fog layer is further structured in several tiers of Fog nodes, as depicted in Fig. 2. The more Fog nodes are far from the IoT layer, the more "intelligence" and computational capabilities they gain. Fog nodes at each tier can also be linked together to provide additional services, such as fault tolerance, resilience, and load balancing. A bottom-up description of each layer of the N-tier Fog architecture is provided next [DDFD19].

IoT layer. The bottom layer is composed of IoT devices, typically sensors and actuators. Endpoints at this layer are often widely distributed geographically and mainly aim to collect data or actuate commands. Data collected at this level are usually sent to the upper layer for processing and storage.

Fog layer. The intermediate layer is composed of one or more tiers of Fog nodes and it is the core of the Fog computing architecture. Fog nodes can be placed anywhere between the Cloud and the IoT and are usually able to compute, store, and transmit data. The Fog layer is directly connected to both IoT devices and Cloud servers, to which it interacts by providing and obtaining services. This layer can be further structured in tiers based on the proximity of Fog nodes to IoT devices and Cloud servers.

- *IoT-to-Fog tier*: the lowest tier is composed of Fog nodes placed in proximity to IoT devices and mainly focused on acquiring, normalizing, and collecting data from the lower layer.
- *Fog-to-Fog tiers*: on top of the previous tier, there could be one or more tiers of Fog nodes aimed at filtering, compressing, and transforming the data flowing to the Cloud.
- *Fog-to-Cloud tier*: the highest tier is composed of Fog nodes directly connected to the Cloud infrastructure and mainly in charge of helping Cloud servers in aggregating data and building knowledge from it.

Cloud layer. The top layer comprises multiple servers with high resources in charge of performing the most challenging tasks, as in the traditional Cloud architecture [LTM+11]. However, compared to the Cloud layer of the 2-layer Cloud-IoT architecture, in the N-tier architecture, some of the load can be moved to the Fog, increasing the overall efficiency and improving the user experience.

Looking at the N-tier architecture for Fog computing depicted in Fig. 2, it should be clear how it naturally intersects with the `AntibIoTic` infrastructure, detailed in Fig. 1. This intersection will be further explored in the rest of the work.

2.3. The `AntibIoTic` evolution

This manuscript merges and extends previous research conducted on `AntibIoTic` [DDDGM16, DDD19, DDFD19]. Over the last few years, the solution has been improved and the design has slightly changed to achieve an effective distributed security system for the IoT. This section summarizes the history behind `AntibIoTic`, underlining challenges and choices that motivated its evolution.

2.3.1. `AntibIoTic` 1.0

In 2016, the Mirai malware attracted the world attention to the need for solutions against Distributed Denial of Service (DDoS) attacks sourced by IoT devices. As a result, after studying and understanding Mirai [DDDGS18], the first version of `AntibIoTic` [DDDGM16] was designed, referred to as `AntibIoTic` 1.0, as a palliative solution against IoT-driven DDoS attacks.

Infrastructure and modus operandi of `AntibIoTic` 1.0 were strongly inspired by Mirai and based on the assumption that the intrinsic insecurity of IoT devices could be used as the solution to the IoT security problem. In fact, `AntibIoTic` 1.0 was a "white worm" operating similarly to the homonym medication used to treat bacterial infections of the human body: it infected vulnerable IoT devices and added them to a "white botnet" of safe endpoints, removing them from the clutches of attackers. It spread with the same unrestrained and highly effective method as Mirai and then it secured the controlled IoT devices instead of using them for malicious purposes. The infrastructure used to support the “white botnet” was also designed to include additional features aimed at increasing the awareness of the IoT security problem and encouraging all actors involved in the IoT devices lifecycle, such as device manufacturers, final users, and security experts, to work together toward a more secure IoT. More details on `AntibIoTic` 1.0 can be found in [DDDGM16], including a description of its infrastructure and a detailed list of its functionalities.

2.3.2. The need for a new design

The idea behind `AntibIoTic` 1.0 was sound and seemed promising, however, it hid some issues that made it unfeasible to be deployed in a legal and controlled manner.The legal concerns mainly arose from the `AntibIoTic` intent of gaining access and partial control of IoT devices without the explicit consent of their owner, even if only for security purposes. Article 3, 4, and 5 of the EU directive on attacks against information systems [EP13] clearly state that accessing and tampering with the functioning and/or the data of any information system, without right, is punishable as a criminal offence. The very first action that `AntibIoTic` 1.0 performed in order to secure vulnerable IoT devices was to intentionally gain access, without right, to the endpoint, which would infringe article 3 – “illegal access to information systems” – of the EU directive. Subsequently, `AntibIoTic` 1.0 was designed to secure the hosting IoT device by acting on its functioning and data (e.g., changing login credentials or updating the firmware), which would violate article 4 – “illegal system interference” – and 5 – “illegal data interference” – of the EU directive. `AntibIoTic` 1.0 was designed as a white worm and, as such, it was not possible to fully control what devices it targeted in order to require prior consent from their owner. Thus, even if a *white* one, `AntibIoTic` 1.0 could still be legally considered a computer worm, and it is thus illegal to deploy it.

The legal (and ethical) issues behind `AntibIoTic` 1.0, along with the huge potential of the idea had, inspired the re-engineering of the system by including Fog computing in the design [DDD19]. The result is `AntibIoTic` 2.0, a distributed security system that relies on Fog computing to overcome the legal limitations of `AntibIoTic` 1.0 and to unleash its full potential with a new Fog-enhanced design.

3. System overview

`AntibIoTic` 1.0 represented a promising solution against IoT malware, however, it had some legal and ethical issues that required its re-design. The result is `AntibIoTic` 2.0, an enhanced version of `AntibIoTic` that relies on Fog computing to secure IoT deployments.

In this section, a high-level description of `AntibIoTic` 2.0 is provided. First, an overview of the rationale behind `AntibIoTic` 2.0 is given and its main features are summarized. Then, the security assumptions that drove the design of the solution are detailed.

3.1. `AntibIoTic` 2.0 at glance

`AntibIoTic` 2.0 is a Fog-enhanced distributed security system for IoT devices. It can be seen as composed of two main parts: the edge and the backbone.

At the edge, there is one `AntibIoTic` gateway for each IoT deployment that needs to be secured. The `AntibIoTic` gateway is a piece of software running on an edge Fog node configured to be the network gateway of the IoT setting and ideally allowing only secure IoT devices to access the

Internet. The `AntibIoTic` gateway uploads on each IoT device a software agent, the `AntibIoTic` agent, that monitors and improves the local security level of the host and continuously interacts with the Fog node. Based on the interactions with the agents and the rest of the `AntibIoTic` infrastructure, *i.e.* the backbone, each `AntibIoTic` gateway makes decisions on how to improve the security posture of the IoT network and whether a single IoT host is allowed or not to access the Internet, also depending on the security level configured for the specific IoT deployment.

The `AntibIoTic` gateways controlling each IoT deployment are then connected to and coordinated by the network of core Fog nodes. The edge Fog nodes interact with the rest of the infrastructure by sending local information to the upper layers and receiving upgrades and updates. The data collected at the upper layers are then aggregated and processed in Cloud computing servers, and the results are used to both improve the solution and generate metrics and statistics made accessible to stakeholders. The Cloud computing servers and the core Fog nodes, constitute the backbone of `AntibIoTic` 2.0. An overview of `AntibIoTic` 2.0 is presented in Fig. 1.

3.2. Main features

In this section, a high-level overview of the main features of `AntibIoTic` 2.0 is provided.

Quick installation at the edge. `AntibIoTic` is designed to be easily used to secure every IoT environment. When a new IoT network needs to be secure, it is sufficient to install a Fog node, with the `AntibIoTic` gateway on, as the only gateway of the network and properly configure it. The system will then start working transparently to the users and without the need to configure each IoT device in the network manually. `AntibIoTic` is an especially good fit for large IoT deployments with many IoT devices of the same type.

Host security. The agent running on each IoT device grants `AntibIoTic` the possibility to act on every endpoint ensuring device-level security. For instance, it can close suspicious ports, kill malicious processes, or remove infected files. The actions performed and the level of protection granted depends on the type of hosting device.

Network security. The edge Fog node acting as the gateway of the IoT network allows `AntibIoTic` to monitor the network traffic and implement solutions (e.g., Intrusion Detection Systems, Intrusion Prevention Systems, and firewalls) aimed at protecting each IoT deployment against network-level threats. This ensures a consistent minimum level of security across IoT devices, regardless of their host security level.

Data driven. Data collected by `AntibIoTic` components throughout the infrastructure (e.g., common threats, type of infected devices, and possible countermeasures) are polished, anonymized, aggregated, and correlated in order to derive knowledge from them. On the one side, anonymized live data and statistics can be displayed back to the community via a Web server and local dashboards, with the aim of providing a grasp on the current security level of the IoT. On the other side, data are used internally to constantly improve `AntibIoTic` with the help of automated and manual analysis executed at the backbone.

Versatility and scalability. The modular infrastructure of `AntibIoTic` strongly based on Fog computing makes it an extremely versatile and scalable solution. On the one side, it is extremely easy to add new features by acting at the backbone and then propagating the upgrade to the edge, without the need to disrupt the operations of the IoT networks. On the other side, it is simple to quickly scale up by adding Fog nodes both at the backbone and at the edge of the `AntibIoTic` architecture to support increases in the number of IoT devices.

Lightweight and legacy-friendly. AntibIoTic is designed to secure nearly any IoT deployment, including legacy ones: the AntibIoTic agent is lightweight (see Section 7 for details) and compatible to potentially any IoT device; the AntibIoTic gateway is easy to install and transparent to use, suitable for any IoT setting.

Note that the features provided above are a high-level summary of the main functionalities designed for AntibIoTic 2.0 Extensions and improvements are foreseen and encouraged. For the list of services offered by AntibIoTic 2.0 refer to Table 1 in Section 4.1.

3.3. Security assumptions

In this section, the security assumptions made while designing AntibIoTic 2.0 are illustrated. Addressing each of these could be a research topic of its own, thus, it is considered future work for this paper. Nevertheless, each of the security assumptions is discussed, and some solutions that could be integrated with AntibIoTic to release the assumptions are mentioned. In some cases, these solutions cannot be included in AntibIoTic as they are and some work is needed to adapt them; however, their existence proves the realism and feasibility of the hypothesis.

Fog nodes are trusted. Fog nodes, both at the edge and in the backbone, are key components of the AntibIoTic infrastructure. These are responsible for regulating the Internet access of IoT devices and processing aggregated data, issuing updates, upgrades, and alerts. If a Fog node is compromised, it could potentially affect the security of the whole system. *In this paper, each Fog node is assumed to be a trusted entity, adequately secured and not controlled by malicious actors.*

The security of Fog nodes is still a debated topic in the scientific community and solutions are constantly being proposed to address this concern. For instance, SYSGO is developing PikeOS, a separation kernel based on Multi Independent Levels of Security (MILS) [SYS20], which could be used on each Fog node to make sure it is not by-passable and tamper-proof. Aslam et al. propose FoNAC, an automated Fog node audit and certification scheme [AMNR20] that ensures trusted, updated, and vulnerability free Fog nodes.

Secure communication. AntibIoTic 2.0 is a distributed system that strongly relies on components interactions, both at the edge and in the backbone, to ensure the security of IoT devices. If confidentiality, integrity, and authenticity of transmitted data cannot be ensured, the operation of the whole infrastructure is destabilized. *In this paper, the IoT-Fog, Fog-Fog, and Fog-Cloud communications are assumed to be properly secured, granting the confidentiality, integrity, and authenticity of data in transit.*

Securing the communication between network devices is a well-established practice today. Many traditional techniques can be adopted to grant the security of transmitted data by acting at different layers of the ISO/OSI model, such as using the 802.11i wireless security standard WPA2, IPsec, TLS, and HTTPS, to mention the most common ones. Besides, there are novel solutions specifically focused on securing the communication of IoT devices [SMPS19], also using Fog computing [FMC19].

The agent is trusted. At the edge of the AntibIoTic architecture, the security of IoT endpoints is granted via the collaboration between the AntibIoTic agent and the AntibIoTic gateway. Suppose the agent running on an IoT device is compromised; in that case, the security level of that device cannot be granted, and the information collected from that endpoint could be distorted, affecting the permission given to that device and potentially also the aggregated statistics for the whole system. *In this paper, the AntibIoTic agent running on each IoT device is assumed to be*

Table 1
Services offered by `AntibIoTic` 2.0, grouped by component (this table does not include the services required for the internal functioning of `AntibIoTic`)

Service	*Description*	*Module*
	`AntibIoTic` Agent	
SANITIZE	Clean the IoT device from host-level threats.	Sanitizer
SECURE	Secure the perimeter of the IoT device to avoid intrusions.	Sanitizer
LOGGING	Generate reports on the security posture of the IoT device.	Reporter
	`AntibIoTic` Gateway	
ACCESS	Regulate the access to the Internet of the IoT devices.	Handler
UPLOAD	Automatically upload the agent on each IoT device.	Loader
UPDATE	Update and upgrades the agent running on each IoT device.	Loader
OVERVIEW	Show local security data and statistics about the IoT deployment.	Local Panel
NOTIFY	Show security alerts received from the backbone.	Local Panel
CONFIG	Allow to locally tune and configure the system.	Local Panel
NET SEC	Protect against network-level threats.	Watchdog
	`AntibIoTic` Backbone	
PUBLISH	Publish aggregated trends, data, and statistics to provide an overview of the security status of the IoT.	Web Server
RELEASE	Release updates and upgrades to improve the whole system.	Interpreter
ALERT	Identify new potential threats and issue security alerts.	Interpreter

trusted, ensuring the integrity and authenticity of the information it transmits about the security posture of the hosting device.

The execution of trusted code in a potentially hostile environment, such as a vulnerable IoT device, is a problem still discussed in the scientific community, with some solutions being proposed to implement remote attestation techniques also for legacy IoT devices lacking ad-hoc security hardware components [AAD+16]. In legacy IoT scenarios, hybrid and software techniques are most likely to be deployed, for instance, using solutions like HAtt [ABD+20] or SIMPLE [ACT20].

4. Design

After presenting the general idea behind `AntibIoTic` 2.0, this section introduces details on its design. First, it presents the architecture of `AntibIoTic` 2.0 with a description of each component. Then, it provides an example of how `AntibIoTic` 2.0 works in a real-world setting, to help the reader with understanding the solution. The list of `AntibIoTic` services, together with the module of each component implementing them, is reported in Table 1.

4.1. Architecture

The architecture of `AntibIoTic` 2.0 is depicted in Fig. 3. It is composed of two main parts, which are referred to as *edge* and *backbone*. The *edge* of the `AntibIoTic` infrastructure can be compared

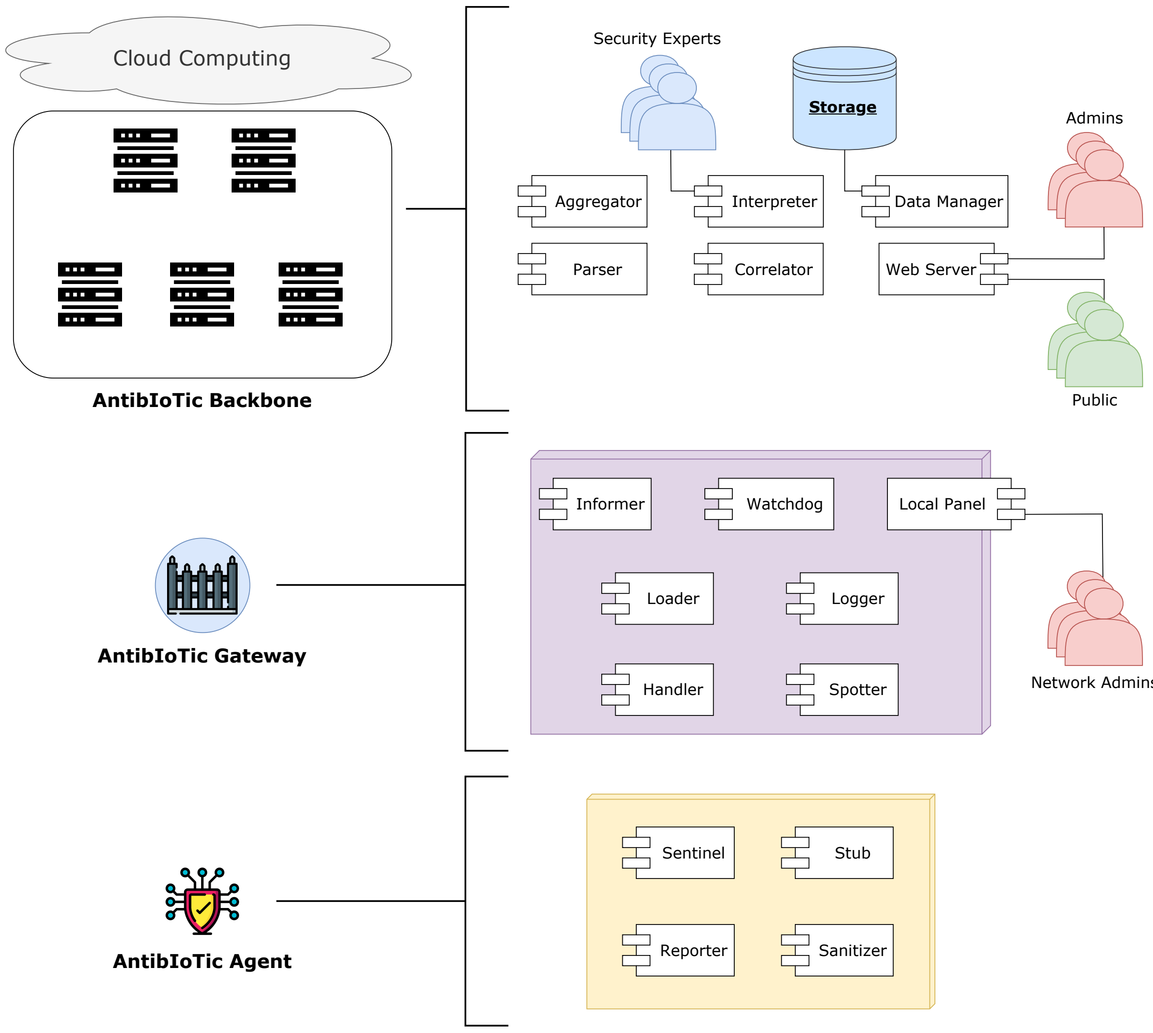


Fig. 3. AntibIoTic 2.0 system architecture: dissection of the AntibIoTic infrastructure with details on modules forming each main component.

to the IoT and IoT-to-Fog layers, i.e. the lowest layers, of the Fog N-tier reference architecture (see Section 2.2) and it includes two main components acting on each IoT deployment: AntibIoTic gateway, AntibIoTic agent. The *backbone* of AntibIoTic refers to the Fog-to-Fog, Fog-to-Cloud, and Cloud layers of the Fog reference architecture (see Section 2.2), and it is composed of a network of core Fog nodes interacting with Cloud servers to support the operations at the *edge*.

4.1.1. Edge

AntibIoTic agent and AntibIoTic gateway are the two components acting at the edge of the AntibIoTic infrastructure to secure each IoT deployment.

The AntibIoTic agent is the software running on each IoT device. It assesses the security posture of the host device and performs actions aimed at increasing the security level of the endpoint. To this aim, it continuously interacts with, sends security reports to, and receives communications from the AntibIoTic gateway. The AntibIoTic agent is composed of the following main modules.

- *Stub.* This module is the spine of the agent. It is the first module to be executed, and it establishes the initial connection with the gateway. Then, it starts the other modules and listens for communications from the gateway, such as commands, updates, or upgrades.
- *Sentinel.* This module is in charge of continuously sending keep-alive messages to the gateway, proving that the agent is successfully running on the host device.
- *Sanitizer.* This module is the core of the agent. It assesses the security posture of the hosting IoT device and performs the host-level operations needed to secure it. The number and type of actions performed depend on the nature of the host device and on the `AntibIoTic` configuration for the specific IoT deployment. For instance, this module can kill processes listening on specific ports and bind to those ports to avoid further intrusions or scan executables to detected known malicious patterns and remove corresponding programs.
- *Reporter.* This module is responsible for creating, updating, transmitting, and deleting host-level security reports. A report is a high-level log of the operations performed by the agent on the host device. A new report is transmitted to the gateway when the security posture of the host device changes or when requested from the gateway.

The `AntibIoTic` gateway is the software running on an edge Fog node acting as the gateway of the IoT network. It interacts with the `AntibIoTic` agent to improve the security posture of the IoT endpoints and decides whether IoT devices are allowed to access the Internet depending on the information continuously exchanged with the `AntibIoTic` agent and on the configuration set for the specific deployment. It provides network-level protection via traditional network security solutions (e.g., IDS, IPS, and firewall), and it interacts with the backbone of the `AntibIoTic` architecture to transmit local information and receive upgrades, updates, and alerts. It also offers a human interface to access system configuration and statistics. The `AntibIoTic` gateway is composed of the following main modules:

- *Handler.* This module is the main interface towards the IoT network. It receives all connection requests coming from the agents and processes them with the help of the other modules. Then, it decides whether each endpoint is allowed to access the Internet based on the security posture of the IoT device and the configuration set at the gateway level.
- *Loader.* This module has the responsibility to load the agent on each IoT device. It also updates and upgrades the agents when needed.
- *Spotter.* This module receives and tracks keep-alive messages from the agents. When it does not receive keep-alive messages from an endpoint, it raises an alert that can block the device access to the Internet.
- *Logger.* This module is responsible for collecting and aggregating the security reports coming from each IoT endpoint. Each report is checked for integrity (e.g., performing a basic check on hash values or using more advanced techniques [GGT15]), anonymized, and stored locally for further analysis and transmission to the backbone.
- *Informer.* This module is responsible for interacting with the backbone of the infrastructure, *i.e.,* core Fog network and Cloud servers. It transmits local information collected from the IoT deployment (e.g., logs) and receives updates, upgrades, and alerts.
- *Local panel.* This module offers an interface to human actors. It shows local data and statistics, allows system tuning and configuration, and displays notifications sent by the backbone.
- *Watchdog.* This module offers protection against network-level threats by implementing traditional network security measures, such as IDS, IPS, and firewall.

4.1.2. Backbone

The backbone of AntibIoTic is composed of core Fog nodes and Cloud servers organized in a hierarchical structure and interacting with each other to support and improve the security operations performed at the IoT deployment. Similarly to a distributed Security Information and Event Management (SIEM) system, the AntibIoTic backbone collects local log data from the AntibIoTic gateways, parses the logs (*i.e.*, security reports) to extract relevant information, and correlates them to identify trends, metrics, and statistics. The knowledge extracted is interpreted to issue upgrades, updates, or alerts propagated till the edge to each AntibIoTic gateway. The aggregated and anonymized information extracted from the logs can also be made publicly accessible to the community via the built-in website.

The AntibIoTic backbone is composed of the following modules. Please note that these are logical components that can be physically distributed throughout the backbone, ranging from the core Fog network to the Cloud, depending on nodes and resources available for the specific AntibIoTic deployment.

- *Aggregator.* This module receives the security reports from each AntibIoTic gateway and aggregates them, providing consolidated data ready to be processed.
- *Parser.* This module contains the parsing engine that interprets the aggregated security reports and extracts relevant information, such as identified threats, adopted countermeasures, type of device infected, network port used by malware, and so on.
- *Correlator.* This module is responsible for correlating the information extracted from the security report in order to create metrics, generate statistics, and identify patterns that, on the one side, help in having a better understanding of the global status of the IoT security, on the other side, can be used to improve the system. This module generally relies on Artificial Intelligence (AI) and Machine Learning (ML) to produce real-time results.
- *Interpreter.* This module interprets the knowledge extrapolated from the data and turns it into updates, upgrades, or alerts to be transmitted to the AntibIoTic infrastructure in order to improve the system. This step is supervised by security experts who review the results before deployment.
- *Data manager.* This module ensures concurrent and secure access to data. Each component interacts with this module to access or store data.
- *Web server.* This module is a traditional Web server composed of a public Website and a private admin panel. The Website publishes aggregated trends, data, and statistics, providing an overall picture of the live security status of the IoT landscape. The admin panel is used by administrators to tune, configure, and maintain the AntibIoTic infrastructure.

4.2. A real-world example: AntibIoTic 2.0 vs Mirai

In this section, a practical example of how AntibIoTic 2.0 would work in a real-world scenario is shown. The aim is to give a first taste of how the solution practically works, before providing further details. For more details on how AntibIoTic 2.0 acts in a real setting, refer to Section 6.

Let us assume that the AntibIoTic backbone is properly set up and that, at the edge, the Fog node hosting the AntibIoTic gateway is configured to be the only access to the Internet for the entire IoT deployment. Let us suppose that the highest security level is required, i.e. the *strict* operation mode is set (operation modes for AntibIoTic 2.0 will be discussed in Section 5.2), and that one of the legacy IoT devices in the network, namely a Netgear DGN1000 router, is

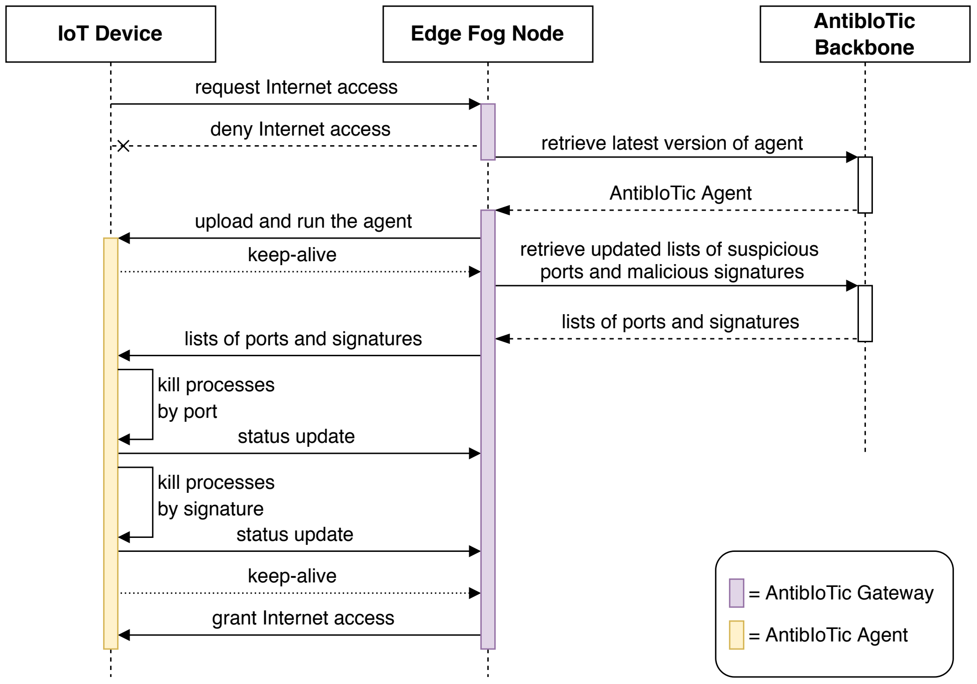


Fig. 4. Example of interaction between the main components of `AntibIoTic` 2.0.

infected with the Mirai malware. Let us see how `AntibIoTic` 2.0 acts in this situation. A graphical representation of the interaction between the `AntibIoTic` components is depicted in Fig. 4.

At first, the infected IoT device requests Internet access to the Fog node. The `AntibIoTic` gateway, in execution on the Fog node, detects that the legacy device does not have the `AntibIoTic` agent in execution, thus, it does not allow the IoT host to access the Internet. Instead, the `AntibIoTic` gateway retrieves the latest version of the agent from the `AntibIoTic` backbone and uploads the agent on the remote IoT device, running it.

Once the `AntibIoTic` agent is executing on the infected IoT device, it starts sending keep-alive messages to the gateway and begins to scan the hosting device for security threats. First, the agent kills processes listening on the suspicious ports defined by the `AntibIoTic` backbone (e.g., TCP/22 SSH, TCP/23 Telnet, TCP/80 HTTP) and binds itself on those ports to prevent further intrusions. Then, the agent scans all running programs in the system memory and terminates the processes having a signature matched in the list of malicious patterns provided by the backbone. In our scenario, where the Mirai malware is assumed to run on a Netgear DGN1000 router, both actions performed by the `AntibIoTic` agent while executing on the router would be effective to secure the hosting device. In fact, Mirai usually binds to Telnet, SSH, and HTTP ports after infecting a device and it has a signature recognizable with a memory scan [DDDGS18]. As a result, the `AntibIoTic` agent terminates the Mirai malware and sends an update to the `AntibIoTic` gateway, communicating that the hosting device is now secure.

The Fog node receives the status update from the `AntibIoTic` agent and allows the legacy IoT device to access the Internet. From now on, the Netgear router is secure and has full Internet access. Nevertheless, the `AntibIoTic` agent keeps running on the device periodically looking for security threats and notifying the AntibIoTic gateway in case of a change in the security posture of the hosting device.

5. Deployment and operations

After discussing the design choices behind AntibIoTic 2.0, this section provides details on the deployment of the solution. First, it describes some models for the deployment of AntibIoTic 2.0. Then, it introduces operation modes that can be configured at the edge of the AntibIoTic infrastructure to best fit different IoT scenarios. Finally, it highlights why AntibIoTic 2.0 is considered an enhanced version of AntibIoTic 1.0.

5.1. Deployment models

This section describes the deployment models for AntibIoTic 2.0 supported by some examples.

5.1.1. Private AntibIoTic

In a private deployment of AntibIoTic, the user is responsible for the whole AntibIoTic infrastructure, from the backbone to the edge. For instance, consider a large company relying on several Industrial IoT deployments to support its business. In such a scenario, AntibIoTic 2.0 can be deployed as an in-house security solution to grant a consistent level of security across all the IoT deployments of the company.

In this case, the organization needs to install all components required for the functioning of the AntibIoTic infrastructure. The nodes required to implement the backbone depend on needs and available resources, and can be installed in different locations or at the central organization headquarter. At the edge, a Fog node is required for each IoT deployment. Thus, a minimal AntibIoTic installation requires at least one powerful node acting as the backbone of the infrastructure, and one Fog node installed at each IoT deployment and configured to be the only access to the Internet for all IoT devices in the scope. Once every node is installed, and the initial configuration of the system is performed, AntibIoTic 2.0 will start working seamlessly, securing each IoT setting without requiring any human interaction with the IoT endpoints. IT administrators can access all information collected by the agents and modify the configuration of the system both at the backbone and the edge level.

The private deployment of AntibIoTic is the most demanding way to benefit from the AntibIoTic solution. It requires qualified personnel to install and monitor the infrastructure and the economic resources needed to maintain the nodes. However, this ensures the best control over the AntibIoTic chain, granting full customization and privacy. The private deployment of AntibIoTic is recommended for big corporations, such as government organizations and multinational companies.

5.1.2. AntibIoTic as a service

AntibIoTic 2.0 can be offered as a security solution from external entities to secure both consumer and industrial IoT deployments. For instance, consider a consumer who wants to secure its private IoT network (e.g., home network, surveillance network, etc.) without having the right competencies and resources to deploy a security solution itself.

In this scenario, the customer can contact an external entity, such as an Internet Service Provider (ISP) or a specialized security company, who offers AntibIoTic as a service and ask them to add its private IoT deployment in their infrastructure. The AntibIoTic provider installs and configures the AntibIoTic gateway at the customer premises and includes the new IoT deployment in the architecture. Once the AntibIoTic gateway is fully installed, configured, and acts as the only gateway of the network, the customer can use its IoT devices as usual, without changing their configuration. The external entity manages both the backbone and the edge of the system and sells AntibIoTic as a security service, potentially including additional maintenance or assistance packages.

AntibIoTic as a service is the easiest and most inexpensive way for a company or a consumer to secure their network of IoT devices. It does not require competencies from the customer, and the maintenance of the system is outsourced to the AntibIoTic provider. Nevertheless, this deployment gives no control to the customer on the AntibIoTic chain, and specific agreements need to be stipulated with the provider to ensure the desired level of privacy and customization.

5.1.3. Hybrid AntibIoTic

AntibIoTic can also be deployed as a hybrid solution where the customer is responsible for the edge of the infrastructure and relies on an external entity for the backbone. For instance, consider a medium-sized company with a brownfield IoT deployment where a consistent security level needs to be granted without disrupting business processes.

In this case, the corporation can contact an external entity who provides AntibIoTic as a service and request the integration of a self-managed local deployment of AntibIoTic in the provider infrastructure. The external entity offers support for installing the AntibIoTic gateway at the customer premises, but it is the company responsibility to configure and manage the system at the edge. Once the installation, configuration, and integration of the edge Fog node with the AntibIoTic backbone is completed, the AntibIoTic gateway will act as the only point of access to the Internet and the system will secure the new IoT deployment according to the custom local settings, without the need to manually configure each IoT device.

The hybrid deployment of AntibIoTic is a relatively inexpensive security solution that gives the customer the possibility to control its local configuration according to the needs without having to maintain the backbone of the system. It still requires qualified personnel to configure and maintain the local AntibIoTic deployment as well as the interaction with the provider infrastructure. However, this ensures partial control over customer data privacy and high customization (e.g., having the possibility to grant a lower security level to reduce the impact of the solution on business processes). The private deployment of AntibIoTic is ideal for medium-sized companies who need a tailored security solution for their IoT deployment but are not willing or capable of handling the whole AntibIoTic infrastructure.

5.2. Operation modes at the edge

At the edge of the AntibIoTic infrastructure, the Fog node acts as the network gateway for the IoT deployment and it is in charge of deciding whether an IoT device is allowed to access the Internet, depending on its security posture. However, AntibIoTic 2.0 is designed to work in different IoT scenarios characterized by various security and connectivity requirements. For instance, in an IoT network of safety-critical systems, the availability of each host is of utmost importance, while security is only desired. Thus, any security solution adopted in this context should not impact the functionality of the IoT application by disrupting the connectivity of IoT devices while trying to secure the system. On the opposite, an IoT deployment belonging to a

Table 2

Operation modes at the edge of the AntibIoTic 2.0 infrastructure; for each operation mode, the table reports the security level ensured, the impact on the IoT applications AntibIoTic might have, and what requirements the IoT devices need to meet to be allowed to access the Internet, in terms of AntibIoTic agent execution and security flaws identified on the system

Operation Mode	Security	Impact	Devices Requirements	
			AntibIoTic Agent	Security Flaws
Strict	High	High	Running	None
Moderate	Medium	Medium	Running	Minor
Lenient	Low	Low	Running	Any

military environment might require the highest security level possible, even to the detriment of connectivity. For this reason, when AntibIoTic is deployed in a specific IoT setting, an initial configuration is performed. First, the installer defines the parameters the AntibIoTic gateway uses for classifying the security flaws identified by the AntibIoTic agent on each IoT device, for instance, considering severe security flaws all those vulnerabilities leading to remote code execution or having a vulnerability score higher than a specific threshold. The operation mode is then set based on the balance between security and impact on the IoT application that is desired within the particular deployment.

In the following, a set of operation modes for AntibIoTic 2.0 are presented, and summarized in Table 2 [DDFD19]: *Strict*, *Moderate*, *Lenient*. Please note that the information presented below represents a reasonable set of configurations that might be suitable for most IoT deployments. However, this set might not be exhaustive, thus, modifications or additions are highly encouraged.

Strict. The strict operation mode is the most secure one. When this mode is set, only IoT devices that are correctly running the AntibIoTic agent and can be fully secured by AntibIoTic are allowed to access the Internet. The AntibIoTic gateway does not give Internet access to any IoT endpoint not running the AntibIoTic agent, *i.e.* devices from which the AntibIoTic gateway does not regularly receive keep-alive messages, or presenting any type of open security flaw. Only when AntibIoTic can ensure the highest security level for the hosting device, this will be allowed to the Internet. For instance, if the AntibIoTic agent running on a specific IoT device detects an insecure Telnet server in execution, it will report this to the AntibIoTic gateway who will not allow this endpoint to access the Internet.

This operation mode is designed for scenarios where security is the utmost concern, even if this can have a high impact on the operation of the IoT applications. For those scenarios where the connectivity of IoT endpoints is a key concern, this mode of operation is not recommended.

Moderate. The moderate operation mode is the most balanced one. When this mode is set, IoT devices correctly running the AntibIoTic agent and not presenting severe security flaws are allowed to the Internet. The AntibIoTic gateway does not allow to the Internet those IoT endpoints not running the AntibIoTic agent or presenting severe security threats that can profoundly impact the security posture of the IoT deployment. IoT devices with minor security flaws are allowed to access the Internet while trying to be secure, as long as they still have the AntibIoTic agent in execution closely controlling their security status. For instance, if the AntibIoTic agent reports to the gateway that the hosting IoT device runs a web application with the X-Content-Type-

Options Header missing, this endpoint will still be allowed to the Internet if this vulnerability can be classified as a minor security flaw.

This operation mode is designed for scenarios where security and connectivity are desired but not critical concerns. In this mode of operation, AntibIoTic works to grant a good security level for IoT devices without having a high impact on the operation of the IoT applications.

Lenient. The lenient operation mode is the one with the lowest impact on the regular operation of IoT applications, to the detriment of security. When this mode is set, all IoT devices running the AntibIoTic agent are allowed to access the Internet, regardless of their security posture. AntibIoTic still works to ensure the best security level possible for each IoT device, however, the AntibIoTic gateway does not prevent insecure devices to access the Internet. Only IoT devices not running the AntibIoTic agent are not allowed to the Internet. For instance, if the AntibIoTic gateway does not receive keep-alive messages from a specific IoT endpoint, this device will not be allowed to the Internet.

This operation mode is ideal for those scenarios where the connectivity and availability of IoT devices are of utmost importance, while security is only desired.

5.3. Why is AntibIoTic 2.0 better than its predecessor?

Now that the details of AntibIoTic 2.0 have been unfolded, it is the time to underline what makes it an improvement when compared to its predecessor.

AntibIoTic 2.0 is an enhanced version of AntibIoTic 1.0 which relies on Fog computing to secure IoT devices (instead of acting as a “white worm” that creates a botnet of safe systems, as in AntibIoTic 1.0). The integration of Fog computing in the design provides multiple benefits.

First of all, the new design solves the main problem of AntibIoTic 1.0, namely the legal issue. In AntibIoTic 2.0, there is a fine-grained control of the IoT devices that are protected, allowing to obtain consensus from device owners before uploading and executing code on the hosting endpoint. Secondly, introducing Fog computing in the system architecture creates a hierarchical structure that is more modular, scalable, and intuitive if compared to the AntibIoTic 1.0 , leading to additional benefits. On the one side, the backbone of the AntibIoTic 2.0 infrastructure can be used to analyse collected data and seamlessly improve the system while in use, transforming the original idea of a simple “white worm” into a more sophisticated and comprehensive solution, similarly to what industry often refers to as a Security Information and Event Management System (SIEM) [BMZ14]. On the other side, the new architecture easily allows the addition of new features and the integration with the existing solution without manually changing the configuration on each IoT device. For instance, network security techniques have been added to the AntibIoTic gateway to complement the device-level security provided by the agent running on each endpoint. As another example, a new malware detection technique could be adopted and easily deployed in each existing AntibIoTic setting just by broadcasting a new software update to all agents. Finally, relying on a distributed paradigm that involves powerful Fog and Cloud nodes to support relatively constrained IoT devices, allows the easy adoption of more complex techniques, for instance based on Machine Learning and Artificial Intelligence, which are generally more challenging to run directly on IoT nodes.

All this is achieved while maintaining the mission of AntibIoTic to offer host- and network-level security and to publish data and statistics aimed at increasing the awareness about the IoT security problem, pushing the collaboration of all stakeholders to reach a more secure Internet of Things.

Table 3

Equipment used to implement the proof-of-concept of AntibIoTic 2.0
*The amount of RAM for the Netgear router was not explicitly declared by the device manufactured, thus, it was directly retrieved from the device running the command '`cat /proc/meminfo`'.

	Intel's Fog Reference Design	Udoo X86 II Advanced Plus	Raspberry Pi 3 (Model B+)	Netgear DGN1000
Role	Fog node	IoT device	IoT device	IoT device
CPU	Intel XEON E3-1275 v5 3.60 Ghz	Intel Celeron N3160 2.24 Ghz	Cortex-A53 SoC 64-bit 1.4GHz	4KEc V6.12
Arch.	x64	x86	ARMv8	MIPS
RAM	2 GB DDR4	4 GB DDR3L	1 GB LPDDR2	16 MB*
OS	Ubuntu 18.04.5 LTS	Ubuntu 20.04 LTS	Raspbian 9	-
Kernel	Linux 4.15.0-112-generic	Linux 5.4.0-37-generic	Linux 4.19.66-v7+	Linux 2.6.20

6. Proof-of-Concept implementation

In this section, a proof-of-concept implementation of AntibIoTic 2.0 is presented to prove the feasibility of the solution and to provide more details on the way it operates at the edge. This proof-of-concept is focused on the edge of the AntibIoTic architecture, thus, it involves the communications and interactions at the IoT-to-Fog layer and does not include the Fog-to-Fog and the Fog-to-Cloud layers. This choice was made because the backbone of the AntibIoTic infrastructure is considered a relatively standard distributed system similar to others implemented today (such as [DKFD20, CRM19, AD14, LYZL18], to mention a few), thus, its feasibility does not need to be further proven.

The rest of this section is organized as follows. First, it describes the equipment and the layout used to implement the proof-of-concept. Then it provides a summary of the features implemented for the proof-of-concept. More details are provided in Appendix A, where an example of operations showing the main features of AntibIoTic 2.0 deployed in an IoT setting is presented.

The source code of AntibIoTic is available on GitHub [DD20b].

6.1. Equipment

The equipment used for the proof-of-concept is described below and reported in Table 3.

The Fog node used to run the AntibIoTic gateway was Intel's Fog Reference Design. It is a fully integrated system equipped with an Intel XEON CPU E3-1275 v5 3.60 GHz, 32 GB RAM DDR4, 250 GB SATA SSD, and running Ubuntu 18.04.5 LTS with kernel Linux 4.15.0-112-generic. It is also furnished with Wi-Fi, Ethernet, and Bluetooth adapters. Additional Ethernet-to-USB adapters were used to address the need for additional Ethernet ports (due to our layout, described in Section 6.2).

The IoT devices used for the experiment were based on three different processor architectures: MIPS, ARM, x86. As representative for MIPS devices, the Netgear DGN1000 wireless router was chosen, equipped with the MIPS CPU 4KEc V6.12 (big-endian), approximately 16 MB of RAM[1], and running Linux 2.6.20; this device was chosen because it is a legacy IoT device vulnerable to the Mirai malware and based on a MIPS processor, thus, it perfectly represents the IoT devices targeted by `AntibIoTic` 2.0. To test the solution on an ARM device, it was used the Raspberry Pi 3 (Model B+) equipped with a Broadcom BCM2837B0, Cortex-A53 (ARMv8) SoC 64-bit 1.4GHz, 1 GB of SDRAM LPDDR2, 16 GB of MicroSD Card, and running Raspbian GNU/Linux 9 with kernel Linux 4.19.66-v7+. Finally, the x86 architecture was tested using the Udoo X86 II Advanced Plus equipped with a CPU Intel Celeron N3160 2.24 Ghz, 4 GB of RAM DDR3L, 32GB eMMC, and running Ubuntu 20.04 LTS with kernel Linux 5.4.0-37-generic.

6.2. Layout

The proof-of-concept layout is depicted in Fig. 5 and refers to the edge of the `AntibIoTic` infrastructure. The IoT deployment was composed of three IoT devices based on different architectures, as described in Section 6.1: Netgear DGN1000 (MIPS), Raspberry Pi 3 (ARM), UDOO x86 (x86). The IoT devices, residing in LAN 1, were connected to Intel's Fog node via Ethernet; this was due to limitations in the Wi-Fi module of some of the devices adopted. The Fog node was the only gateway between LAN 1 and the university network with Internet access; the use of a second Local Area Network (LAN) was due to limitations given by the network configuration of the university where the experiment was conducted (namely, Technical University of Denmark). Although the `AntibIoTic` gateway was not the direct gateway to the Internet, the Fog node was still the only point to access the Internet, thus, this layout can be considered equivalent to the one depicted in Fig. 1.

6.3. Summary

The proof-of-concept goal was to prove the feasibility of the solution at the edge of the network. It has been implemented using a Fog node, Intel's Fog Reference Design, and 3 IoT devices, the Netgear DGN1000 Wireless Router, the Raspberry Pi 3 (Model B+), and the Udoo x86 Advanced Plus. In this setting, the following features have successfully been implemented.

Quick installation. The proof-of-concept includes several helper scripts that can be used to configure and install `AntibIoTic` in the IoT deployment, granting a quick and relatively simple installation. Also, the `AntibIoTic` gateway is able to compile and upload the `AntibIoTic` agent automatically on each IoT device in the network, allowing the system to work without human intervention on each IoT endpoint.

Concurrent interactions. The two main components at the edge of the `AntibIoTic` infrastructure have successfully been implemented and can interact as expected in a concurrent way. In fact, the `AntibIoTic` gateway is able to receive concurrent keep-alive messages and status updates from all `AntibIoTic` agents in the network. At the same time, it can request periodic summary reports and send additional commands, for instance to update the list of malicious signatures and suspicious ports of the agents or to reboot the IoT devices. Although the interactions have been tested only with three devices in the network, the concurrent implementation of the gateway allows it to be scaled up to a wide number of endpoints.

[1]The amount of RAM for this device was not explicitly declared by the device manufacturer, thus, it was directly retrieved from the device running the command '`cat /proc/meminfo`'.

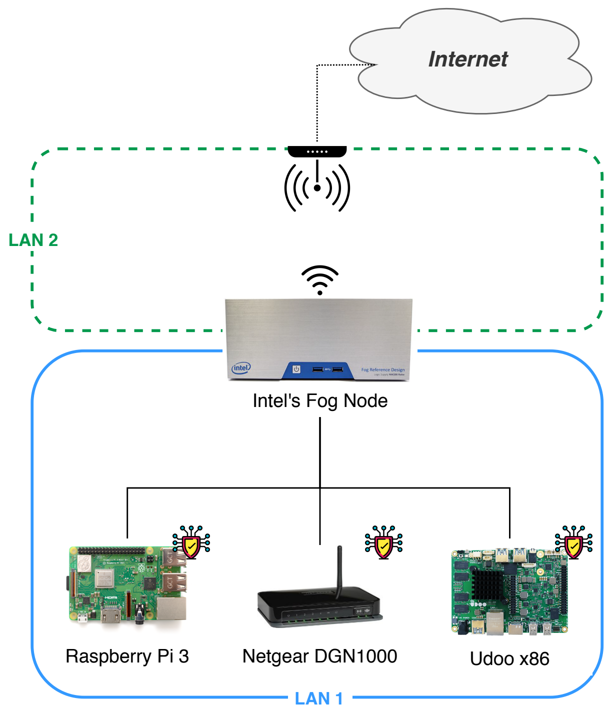


Fig. 5. Proof-of-concept layout

Network security. The `AntibIoTic` gateway can be executed on the Intel Fog node and it successfully acts as the only access point to the Internet. It implements a simple firewall based on '`iptables`' rules to filter the ingoing and outgoing traffic and ensure a minimum network security level. The `AntibIoTic` gateway is also able to discriminate whether a device should be allowed to the Internet or not, depending on the data received from the `AntibIoTic` agent running on the IoT host. Additional network security solutions can easily be implemented.

Host security with low resources. The `AntibIoTic` agent can run on IoT devices based on three different architectures: MIPS, ARM, and x86. It can ensure a minimum level of host security by periodically scanning the hosting device for malicious code and removing it from the device. The agent removes processes that match the malicious signatures in its list and kills potential backdoor listening on suspicious ports, binding itself on the same port to avoid further intrusions. This is achieved by keeping a low usage of resources (see Section 7 for details) and ensuring only a single instance of the agent running on the device.

Operation mode: lenient. The *leninet* operation mode has been implemented. The `AntibIoTic` agent can detect and kill malicious code running on IoT devices, however, the only requirement they have to access the Internet is that the `AntibIoTic` gateway can detect the latest `AntibIoTic` agent running on the IoT endpoint.

7. Evaluation

In this section, the resources usage for `AntibIoTic` 2.0 is evaluated. Specifically, the focus is on testing the `AntibIoTic` agent and analyzing the impact it has on the resources of the hosting IoT devices, in terms of CPU, memory, network, and storage. The detection rate of the `AntibIoTic` agent is not tested because introducing a novel malware detection technique for IoT devices is out of the scope of this paper. A simple signature-based detection approach is used to show how the solution works, but the system architecture proposed is modular enough to allow easy integration of the desired malware detection method [NNNN20].

The focus of the experiments is on the `AntibIoTic` agent. On the one side, that is because it runs on IoT devices where available resources are critical assets, especially for legacy ones. On the other side, because the `AntibIoTic` gateway and the other components of the `AntibIoTic` backbone are expected to be executed on devices with considerable available resources, thus, the overhead the security solution causes is insignificant. In fact, while monitoring the resources usage of the `AntibIoTic` gateway running on the Intel Fog node, it could be seen that the impact was minimal (e.g., CPU average usage = 0%). Thus, it can be assumed that the same applies to the other nodes of the `AntibIoTic` backbone, making their evaluation negligible.

The rest of this section is organized as follows. First, the methodology used to evaluate the `AntibIoTic` agent is described. Then, the data collected during the experiments are provided, organizing them in tables and representing them in graphs. Finally, the data are analysed and discussed.

7.1. Methodology

The equipment and layout used to evaluate `AntibIoTic` was the same used in the proof-of-concept and described in Section 6.1 and Section 6.2: the Intel Fog Node configured to be the gateway of the network and connected to three IoT devices, namely a Netgear DGN1000 router, a Raspberry Pi 3, and an Udoo x86 board. Four scripts were used to perform the evaluation. All the scripts are available on GitHub in the '`tools`' folder [DD20b]

The script '`simulation.sh`' was used to execute emulated malicious code on the hosting device periodically. The script sleeps for a random time interval ranging from 1 second to '`INTERVAL`' seconds, where '`INTERVAL`' is a value defined by the user ('`INTERVAL=10`' in our case), and then executes two programs emulating a malware and a backdoor.

The script '`profiling.sh`' was used to measure the CPU, memory, and network usage of the `AntibIoTic` agent in execution. '`ps`' was used for the CPU, '`pmap`' for the memory, and '`nethogs`' for the network. Every '`INTERVAL`' seconds, where '`INTERVAL`' is a value defined by the user ('`INTERVAL=1`' in our case), the script reads the current CPU usage (in percentage), virtual memory size (in KB), and network data transmitted and received (in KB) for the `AntibIoTic` agent, and stores them in a file for later processing.

The script '`evaluation.sh`' was used to automate and coordinate the execution of the two previous scripts and to repeat the simulation multiple times (10 in our case). The script '`parse-data.py`' was finally used to parse and process the collected data.

In order to collect a significant sample of data, multiple rounds of evaluation were performed on each device, executing the '`evaluation.sh`' script ten times on both the Raspberry Pi 3 and the Udoo x86 board. Unfortunately, due to restrictions of the device itself, it was not possible to run the same script on the Netgear router, thus, some data were manually collected from it.

Table 4
Parameters used while evaluating the resource usage of the AntibIoTic agent; sentinel interval: time between consecutive keep-alive messages; sanitizer interval: time between consecutive local scans for malicious programs; duration: execution time of the script; ‘simulation.sh’ interval: maximum time between consecutive executions of malicious code; ‘profiling.sh’ interval: sampling rate

	AntibIoTic Agent		‘simulation.sh’		‘profiling.sh’	
	Sentinel interval	Sanitizer interval	Interval	Duration	Interval	Duration
Time (s)	15	10	10	112	1	120

7.2. Experimental data

The relevant parameters involved in the evaluation are summarized in Table 4. The AntibIoTic agent was configured with a sentinel interval of 15 seconds and a sanitizer interval of 10 seconds, *i.e.*, the sentinel module of the agent was sending keep-alive messages to the AntibIoTic gateway every 15 seconds and the sanitizer module was performing a full scan of the hosting IoT device every 10 seconds, looking for malicious programs. The ‘simulation.sh’ script was executed on the Udoo x86 and Raspberry Pi with an interval of 10 seconds and a duration of 112 seconds, thus, the simulation run for 112 seconds in total while resembled malicious code was executed at random intervals ranging from 1 to 10 seconds. Finally, the ‘profiling.sh’ script runs on the same two devices for 120 seconds (duration) with a sampling rate of 1 sample per second (interval).

The data collected for each device are summarized in Tables 5, and 6, and represented in Figures 6, 7, 8, and 9. Table 5 reports, for each round of evaluation, the data on CPU and network usage of the AntibIoTic agent while running on Raspberry Pi 3 and Udoo x86 for 120 seconds. Specifically, it reports the minimum, maximum, and average CPU usage during the execution, along with the total Kilobytes (KB) transmitted and received by the agent.

Figure 6 represents more in detail the trend of CPU usage for the AntibIoTic agent running on the Udoo x86 board, second by second. The trend line was obtained as the average, per second, of the CPU usage values collected in the ten rounds of evaluation. The red bars represent the maximum and minimum values obtained, at every second, in the ten rounds. Figure 7 plots the same data for the Raspberry Pi 3.

Figure 8 depicts, for every round of evaluation, the network usage of the AntibIoTic agent running on the Udoo x86 board. The graph represents the total amount of data sent and received by the agent for each round, along with the average for the ten executions. Figure 9 shows the same data for Raspberry Pi 3.

Finally, table 6 lists, for each device, the binary size of the AntibIoTic agent and the total program size in memory while executing.

Although it was not possible to measure the CPU usage and the total network data exchanged by the Netgear router, the data for this device can be expected to be aligned with the ones collected from the other devices.

7.3. Data analysis and discussion

Let us now analyze and discuss the data obtained from the evaluation.

Table 5

For each evaluation round, data on CPU and network usage of the `AntibIoTic` agent running for 120 seconds: minimum, maximum, and average CPU usage; total Kilobytes transmitted and received

	Raspberry Pi 3 (Model B+)					**Udoo x86 Advanced Plus**				
Round	CPU Usage (%)			Network Data (KB)		CPU Usage (%)			Network Data (KB)	
	Min	Max	Avg	Sent	Received	Min	Max	Avg	Sent	Received
1	1.00	15.00	3.31	20.10	2.45	3.00	18.00	3.90	21.33	2.64
2	1.00	9.30	3.36	21.14	2.51	2.00	17.50	3.76	28.64	3.09
3	1.00	14.00	3.44	19.31	2.58	2.00	17.50	3.72	24.53	2.90
4	2.90	28.00	3.79	18.01	2.32	2.00	17.50	3.70	21.46	2.77
5	1.00	14.50	3.60	18.20	2.45	3.00	18.00	3.93	23.42	2.77
6	1.00	16.50	3.63	21.20	2.58	3.00	18.00	3.83	19.25	2.51
7	1.00	16.50	3.78	23.36	2.77	3.00	17.50	3.74	23.42	2.77
8	1.00	15.50	3.68	18.20	2.38	2.00	17.50	3.84	23.61	2.96
9	1.00	17.00	3.80	25.32	2.77	3.00	17.50	3.77	21.46	2.77
10	1.00	16.50	3.78	22.18	2.58	3.00	18.00	3.82	19.31	2.58

Table 6

For each device, program memory size while executing and binary size of the `AntibIoTic` agent

Device	Architecture	Memory size (KB)	Binary size (KB)
Netgear DGN1000 router	MIPS	356 (≈ 2.17%)	63.164
Udoo x86 II Advanced Plus	x86	84592 (≈ 2.02%)	75.576
Raspberry Pi 3 (Model B+)	ARM	19400 (≈ 1.85%)	91.992

First of all, it is important to fully understand the parameters of the `AntibIoTic` agent, reported in Table 4, and their implication on resource usage and the security of the IoT devices. The other parameters reported in the same table are relevant to interpret the resulting data, but are only used for the evaluation and have been reported to grant transparency and reproducibility to the experiments.

The sentinel interval is the time between consecutive keep-alive messages sent from the sentinel module of the agent to the spotter module of the `AntibIoTic` gateway. On the one side, a shorter interval grants a more fine control of the `AntibIoTic` gateway on the agents since it is faster to detect an agent that is not working properly or goes offline. On the other side, the lower the interval, the higher is the impact on CPU and network usage of the agent, since it needs to generate more keep-alive messages causing a higher CPU usage and more transmitted data.

The sanitizer interval is the time between consecutive scans from the sanitize module of the agent, looking for malicious code running on the hosting device. Similarly to the sentinel interval, the shorter the interval, the higher the impact on the CPU usage of the `AntibIoTic` agent, since it has to scan the whole system more often. However, a shorter interval grants a higher security level since the timeframe in which malicious code can run undetected on the IoT device is lower.

The sentinel interval was set to 15 seconds and the sanitizer interval to 10 seconds because those values were considered a good trade-off between security and resource usage. However, depending

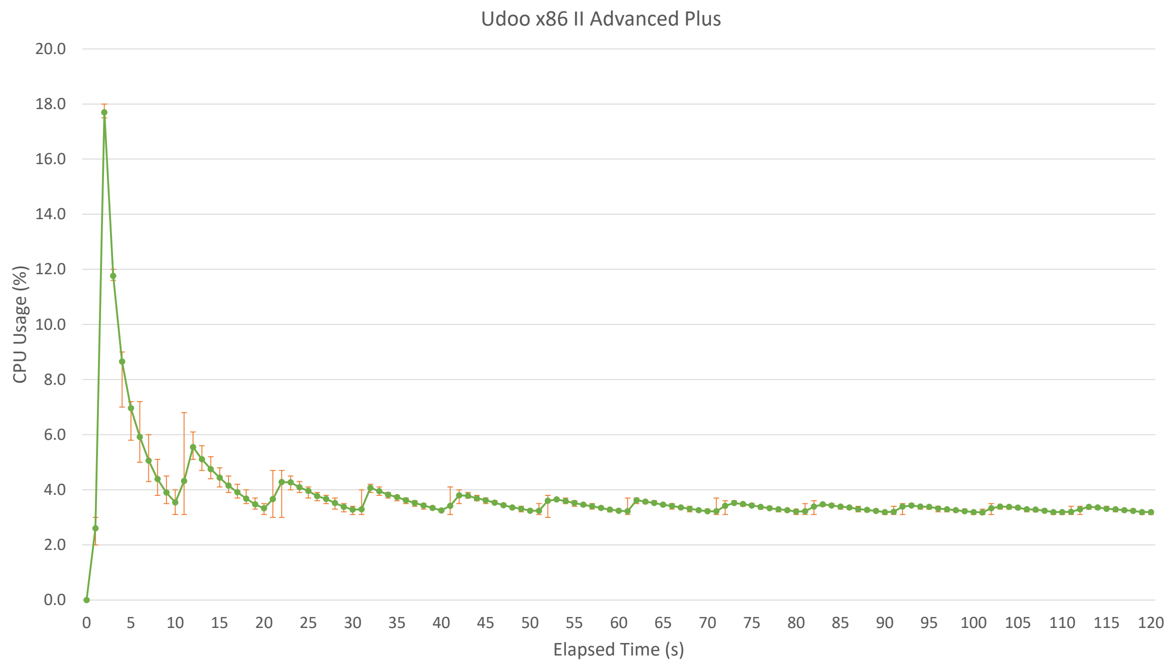


Fig. 6. CPU usage of the `AntibIoTic` agent running on Udoo x86 II Advanced Plus for 120 seconds. Green line: average, every second, of the values collected in the 10 rounds of evaluation. Red bars: maximum and minimum values obtained, each second, in the 10 rounds.

on the specific IoT deployment requirements, these parameters can be adjusted to reduce resource usage or increase the security level.

Looking at the data related to the CPU usage of the `AntibIoTic` agent, reported in Table 5, it can be deduced that the overhead caused by the agent on the IoT devices is minimal. The average CPU usage among the ten rounds of evaluation is 3.80% for the Udoo x86 board, with peaks of 18%, and 3.62% for the Raspberry Pi, with one peak of 28%. Observing Figures 6 and 7 is also easy to identify a clear trend. There is a peak of CPU usage in the first seconds of execution due to the initialization of all the agent modules starting in the background, establishing the connections with the `AntibIoTic` gateway, and the execution of a first full scan of the system. Then, local peaks can be seen approximately every 10 seconds due to the periodic local scan performed by the sanitizer module. As anticipated, tuning the sanitizer interval allows to adjust the CPU usage according to the requirements of the specific IoT deployment. Also, the CPU usage can be further reduced by adding some delay within each scan, for instance pausing the execution (e.g., with a '`sleep()`') when moving from one process to the other. In this way, the scans take longer to finish but require less CPU.

Analysing the network data transmitted and received by the `AntibIoTic` agent, reported in Table 5 and depicted in Figures 8 and 9, some other considerations can be drawn. On the one side, the average amount of data sent in the ten rounds of evaluation is 22.64 KB ($\approx$ 1509 bps) for the Udoo x86 board, with a peak of 28.64 KB (1909 bps), and 20.70 KB ($\approx$ 1380 bps) for the Raspberry Pi, with a peak of 25.32 KB ($\approx$ 1688 bps). This traffic is mainly generated by the keep-alive messages sent every 15 seconds and the status updates sent to the `AntibIoTic` gateway every

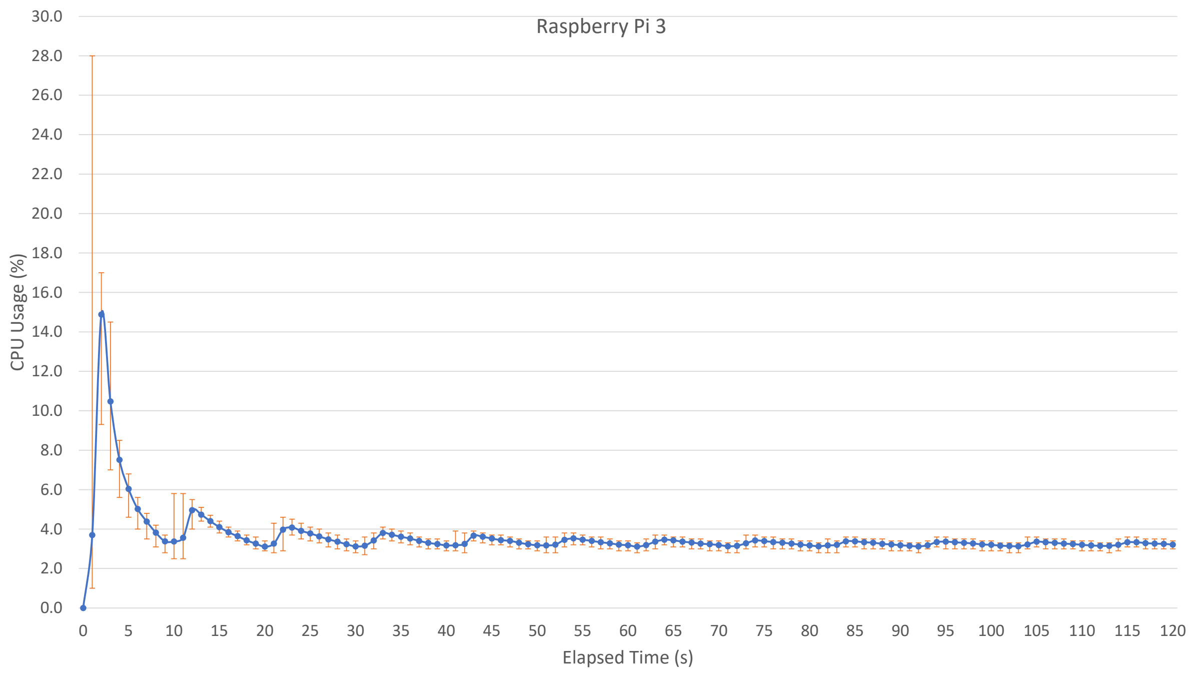


Fig. 7. CPU usage of the `AntibIoTic` agent running on Raspberry Pi 3 (Model B+) for 120 seconds. Blue line: average, every second, of the values collected in the 10 rounds of evaluation. Red bars: maximum and minimum values obtained, each second, in the 10 rounds.

time a new action is performed, such as killing a malicious process. Thus, these values depend on the sentinel interval set for the `AntibIoTic` agent and on the random number of emulated malicious processes generated by the '`simulation.sh`' script at each round of evaluation. On the other side, the average amount of data received in the ten rounds of evaluation is 2.78 KB (≈ 185 bps) for the Udoo x86 board, with a peak of 3.09 KB (≈ 206 bps), and 2.54 KB (≈ 169 bps) for the Raspberry Pi, with a peak of 2.77 KB (≈ 185 bps). The received traffic is almost constant since the communications from the `AntibIoTic` gateway to the agent are very limited in the described scenario.

Looking at the data on the total memory used by the `AntibIoTic` agent while executing, it can be seen that it varies with the available RAM of the device. This might be, for instance, due to the pre-loading of shared libraries for efficiency reasons. Nevertheless, it can be concluded that the total memory used by the `AntibIoTic` agent while executing is around 2 % of the available RAM, even when this is very limited (e.g., 16 MB in the Netgear router).

Finally, the binary size for each architecture, reported in Table 6, provides an approximation of the minimum storage requirement to load the `AntibIoTic` agent on an IoT device. Although this value might change depending on several factors (e.g., the compiler used, optimization techniques adopted, standard C library used, etc.), it can be inferred that the `AntibIoTic` agent requires about 64 KB of free storage on MIPS devices, 76 KB on x86 endpoints, and 92 KB on ARM hosts.

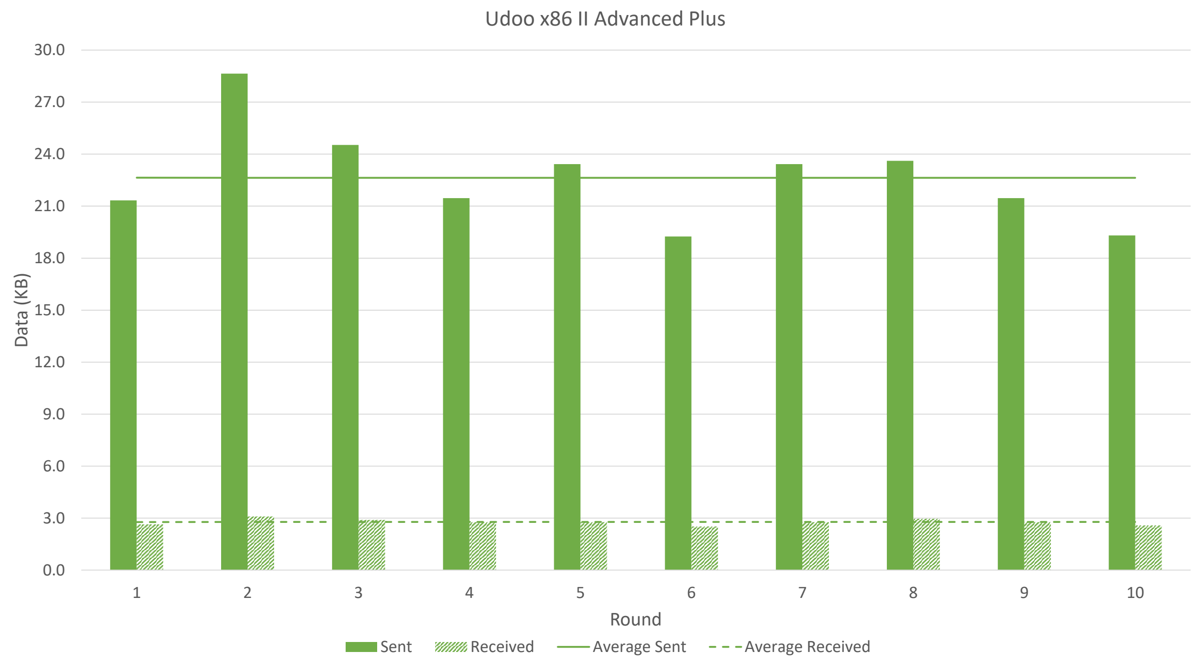


Fig. 8. For each evaluation round, network usage of the `AntibIoTic` agent running on Udoo x86 II Advanced Plus for 120 seconds. Green bars: total Kilobytes sent; light green bars: total Kilobytes received; green line: average data sent; dotted green line: average data received.

8. Related work

To the best of our knowledge, `AntibIoTic` is the first distributed security system of its kind that uses Fog computing to provide security, both at the network- and device-level, to existing IoT deployments, including legacy ones. Nevertheless, some research works can be related to `AntibIoTic`. This section briefly reviews these works.

Roman et al. [RROL18] propose a "virtual immune system" that relies on edge technologies to protect IoT devices. Even though this solution seems closely related to `AntibIoTic` as it reiterates the metaphor of the human body and is based on a distributed approach that involves edge technologies, it has a significantly different architectural model compared to `AntibIoTic`'s one. Also, it is still in its very early stages, and a working implementation proving its feasibility is not available yet.

Soukup et al. [SHŠ+19] propose a security framework to address the security challenges in heterogeneous IoT networks. The framework is composed of a software IoT gateway supported by Cloud/Fog devices. Although the architecture of this solution resembles the `AntibIoTic` structure, Soukup's solution mainly acts at the network level, analyzing the traffic IoT devices generate, without providing device-level protection.

Razouk et al. [RSS17] propose an IoT security middleware that acts as a smart gateway between IoT devices, Fog, and Cloud nodes, and it is composed of several modules, among which a security one, aimed at supporting operations of constrained IoT devices and increasing their security level. However, the research is still in its infant stage without a working implementation yet, and it is

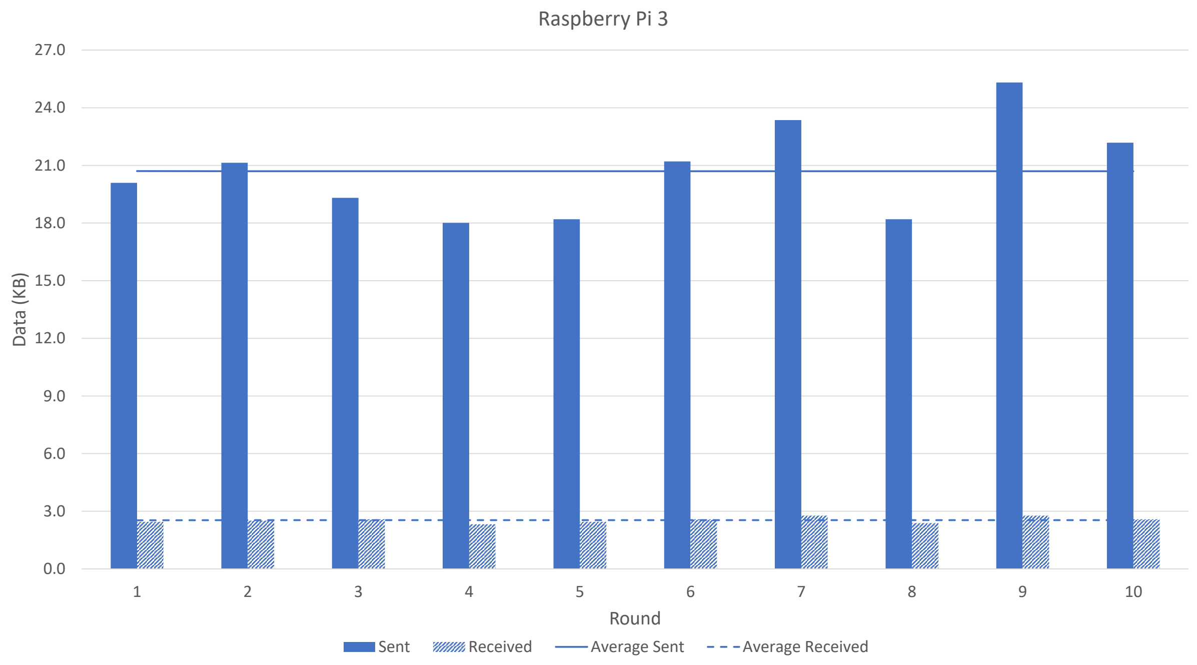


Fig. 9. For each evaluation round, network usage of the `AntibIoTic` agent running on Raspberry Pi 3 (Model B+) for 120 seconds. Blue bars: total Kilobytes sent; light blue bars: total Kilobytes received; blue line: average data sent; dotted blue line: average data received.

presented at a high level without enough details on how the middleware actually works to protect the IoT endpoints.

Kim et al. [KLD19] propose an Edge computing-based IoT security solution, the Secure Swarm Toolkit (SST), that provides authentication and authorization services for the IoT, increasing the resilience to DoS attacks. Sengupta et al. [SRB20] propose a secure Fog-based Industrial IoT architecture that includes some security features. By offloading some of the tasks to Fog nodes, this architecture reduces overhead on IoT devices and latency in decision making, while eliminating trust in the Cloud. Although both solutions have the same aim as `AntibIoTic`, *i.e.,* increasing the security level of the IoT, they have a different scope compared to `AntibIoTic` since they need to be included in the designing phase of IoT networks rather than being used as a solution to secure existing IoT deployments.

Zhou et al. [ZGD19] propose a distributed DDoS mitigation scheme for industrial IoT systems. The solution is based on traffic analysis and Virtualized Network Functions (VFNs) assigned to multiple distributed locations close to the IoT devices, while coordinated by Cloud central servers. Alharbi et al. [ARM+18] propose a Fog computing-based security system, FOCUS, to protect IoT devices against cyber attacks. The solution adopts a Virtual Private Network (VPN) to secure the communication channels and a challenge-response authentication to protect the IoT system against DDoS attacks. Although, similarly to `AntibIoTic` 1.0, both solutions aim at mitigating DDoS attacks, they are intrinsically different from `AntibIoTic`. On the one side, FOCUS and Zhou's solution rely on Fog computing to protect IoT devices against external DDoS attacks, mainly acting at the network level. On the other side, `AntibIoTic` aims at securing each IoT endpoint and, even if not possible, allowing only the secure ones to access the Internet, thus, drastically reducing the possibility of perpetrating large-scale DDoS attacks generated from IoT botnets.

Finally, there are a few works [SYZ20, AAAS20, NS20, dSWM+20, HVAP+16] proposing solutions that, due to their computational capacity and proximity to the IoT endpoints, rely on Fog nodes to detect anomalies, intrusions, or attacks. However, on the one hand, these solutions mainly perform detection at the network level, while `AntibIoTic` also works at the device level. On the other hand, these are passive solutions, *i.e.*, they detect security issues and raise alerts but usually do not actively work to fix the security flaw, which `AntibIoTic` does.

Although the works presented in this section are different when compared to `AntibIoTic`, many of them present traits and techniques that might be compatible with `AntibIoTic`. New approaches are emerging to secure the IoT (such as [A.B21] and [AFK21] to mention only a few examples). Therefore, we call for collaborations to join the efforts of the scientific community to increase the global security level of the IoT.

9. Conclusion

This paper has presented `AntibIoTic` 2.0, to the best of our knowledge, the first Fog-enhanced distributed security system aimed at protecting IoT deployments, while being scalable, versatile, and easy to deploy, even in legacy IoT settings. The system is composed of a core network, the `AntibIoTic` backbone, and two components acting at the edge, the `AntibIoTic` gateway and the `AntibIoTic` agent. At the edge, the agent, running on each IoT device, and the gateway, controlling the access to the Internet, cooperate to offer fine-grained, host-level security coupled with network-level security, while having a minimal impact on the resources of the IoT devices, thanks to the distributed approach. The backbone publishes anonymized data and statistics about the secured IoT deployments, and coordinates and supports the operation at the edge.

First, the solution was described from a high-level perspective to let the reader become familiar with the system. Then, more details on the design and deployment of the solution were provided. Subsequently, an extended proof-of-concept of the solution was shown, and its resource usage was evaluated, showing the low computational impact it has on the involved devices. The source code of `AntibIoTic` is openly available online [DD20b]. Appendix A shows a concrete example of operation of `AntibIoTic` 2.0 when deployed in an IoT setting, including a video-demo [DD20a] that illustrates its main features.

The presented system results from enhancements and improvements made over a number of MSc and BSc theses, a PhD thesis, and previous research [DDDGM16, DDD19, DDFD19] conducted at the Technical University of Denmark.

Please note that, at the edge, deploying `AntibIoTic` 2.0 in a heterogeneous consumer IoT network might be more challenging than in an Industrial IoT one. In general, `AntibIoTic` 2.0 is easier to install in networks with a large number of the same type of devices rather than with a small number of different hosts. The initial configuration of the `AntibIoTic` gateway becomes more complicated with the increasing variety of devices present in the network. Thus, IIoT networks, usually characterized by a large number of homogeneous devices (e.g., robots, sensors, etc.), are more suitable for deploying `AntibIoTic` compared to private IoT networks, often composed of a small number of heterogeneous devices (e.g., IP cameras, smart-homes, smartwatches, tablets, smartphones, smart fridges, etc.). Also, the malware detection technique implemented in the proof-of-concept is a basic pattern-matching approach used to prove the feasibility of the approach, but it can be easily replaced with more effective techniques [NNNN20].

9.1. Future work

`AntibIoTic` is an ambitious and complex solution and, as such, it presents some open challenges that serve as pointers for future work.

On the one side, the security assumptions behind `AntibIoTic` 2.0 (discussed in Section 3.3) should be relaxed. There are already solutions in development that can be analysed and potentially integrated into `AntibIoTic` to relax those assumptions. For instance, the use of a certification scheme to ensure that Fog nodes are trusted [AMNR20], the implementation of existing network security standard to grant secure communications, and the adoption of remote attestation techniques like [ABD+20, ACT20] to have a trusted `AntibIoTic` agent.

On the other side, `AntibIoTic` 2.0 should be further implemented and tested. First, the backbone of the `AntibIoTic` infrastructure should be implemented to have a working testbed of the entire architecture. Then, the solution should be tested in real settings with a large number of (legacy) IoT devices and Fog nodes, evaluating effectiveness and performances. Finally, the full set of features of `AntibIoTic` 2.0 should be implemented, for instance including different operation modes and reports processing capabilities on the `AntibIoTic` gateway, and state-of-the-art techniques should be adopted to replace PoC implementations, such as the malware detection based on a list-matching approach and the host identification done via IP address.

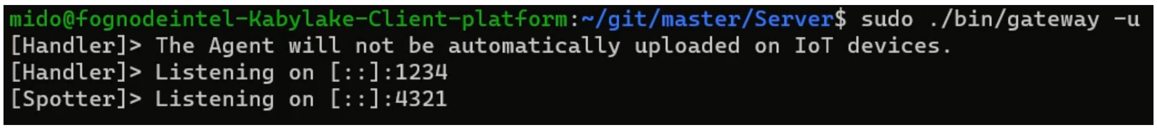
Appendix A. Example of operation

This Appendix demonstrates the feasibility of `AntibIoTic` 2.0 and shows how it concretely operates at the edge of the infrastructure. To this aim, a demo was prepared to show some of the core features of `AntibIoTic` 2.0 when acting in an IoT deployment as the one showed in Fig. 5. A demo has been video recorded and published online [DD20a]. This example of operation, recorded and available online is composed of four main parts detailed below: *startup*, *malware execution*, *agent execution*, *agent termination.*

For more details about the configuration required for each device and the way each functionality is implemented, refer to the GitHub repository of `AntibIoTic` [DD20b].

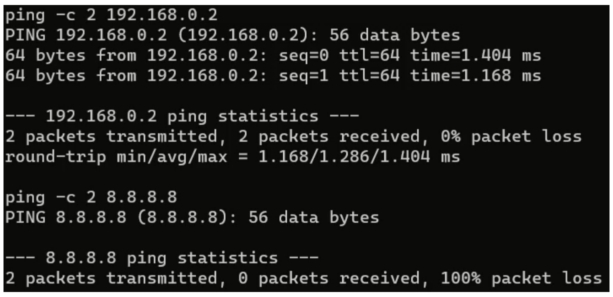
Startup. Initially, `AntibIoTic` was running neither on the IoT devices nor on the Fog node. First, the configuration of each device was checked. Then, as shown in Fig. 10, the `AntibIoTic` gateway was executed on the Fog node with the '`-u`' flag to avoid the automatic uploading of the `AntibIoTic` agent on each IoT device. This choice was made to provide a step-by-step proof-of-concept of how the system works

At the startup, the `AntibIoTic` gateway automatically blocks all IP addresses in the LAN from accessing the Internet and starts listening for incoming connections from the `AntibIoTic` agents on two ports: one (TCP/4231) for keep-alive messages and one (TCP/1234) for interactive communications that include status updates and commands. The Internet access is regulated by using a combination of '`iptables`' rules and an '`ipset`'. Specifically, only the IP addresses added to a specific IP-set are allowed to access the Internet, thus, removing all IPs from the set results in preventing all devices in the LAN from accessing the Internet.



```
mido@fognodeintel-Kabylake-Client-platform:~/git/master/Server$ sudo ./bin/gateway -u
[Handler]> The Agent will not be automatically uploaded on IoT devices.
[Handler]> Listening on [::]:1234
[Spotter]> Listening on [::]:4321
```

Fig. 10. Startup of the `AntibIoTic` gateway on the Fog node using the '`-u`' flag to avoid the automatic uploading of the `AntibIoTic` Agent on each IoT device.



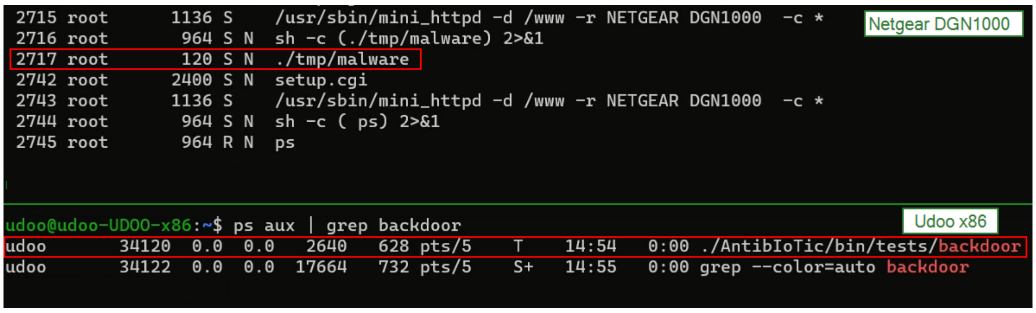
```
ping -c 2 192.168.0.2
PING 192.168.0.2 (192.168.0.2): 56 data bytes
64 bytes from 192.168.0.2: seq=0 ttl=64 time=1.404 ms
64 bytes from 192.168.0.2: seq=1 ttl=64 time=1.168 ms

--- 192.168.0.2 ping statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max = 1.168/1.286/1.404 ms

ping -c 2 8.8.8.8
PING 8.8.8.8 (8.8.8.8): 56 data bytes

--- 8.8.8.8 ping statistics ---
2 packets transmitted, 0 packets received, 100% packet loss
```

Fig. 11. Connectivity check from the Netgear router using the '`ping`' network utility. As expected, the local device with IP `192.168.0.2` is reachable, while the external host `8.8.8.8` is not.



```
 2715 root      1136 S    /usr/sbin/mini_httpd -d /www -r NETGEAR DGN1000  -c *
 2716 root       964 S N  sh -c (./tmp/malware) 2>&1
 2717 root       120 S N  ./tmp/malware
 2742 root      2400 S N  setup.cgi
 2743 root      1136 S    /usr/sbin/mini_httpd -d /www -r NETGEAR DGN1000  -c *
 2744 root       964 S N  sh -c ( ps) 2>&1
 2745 root       964 R N  ps

udoo@udoo-UDOO-x86:~$ ps aux | grep backdoor
udoo       34120  0.0  0.0   2640   628 pts/5    T    14:54   0:00 ./AntibIoTic/bin/tests/backdoor
udoo       34122  0.0  0.0  17664   732 pts/5    S+   14:55   0:00 grep --color=auto backdoor
```

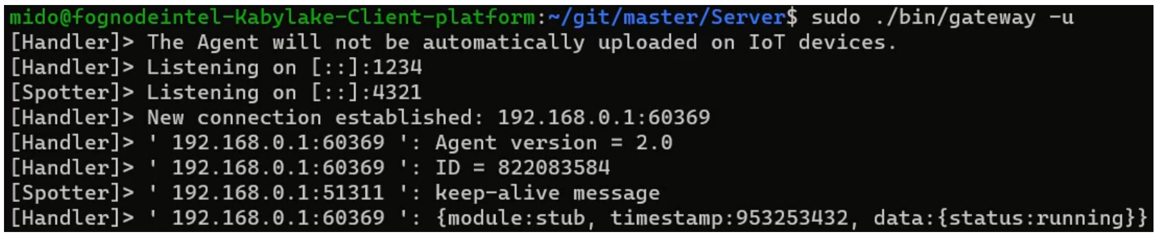
Fig. 12. At the top, a program simulating a malware in execution on the Netgear router. At the bottom, a program simulating a backdoor in execution on the Udoo x86 board.

Once the `AntibIoTic` gateway was in execution on the Fog node, the connectivity of each IoT device was verified using the '`ping`' network utility towards an IP address within the LAN and an external one, as shown in Fig. 11. As expected, all IoT devices could reach out to IP addresses within the LAN, but no communication was possible with external hosts.

Malware Execution. In order to show how the `AntibIoTic` agent behaves when it finds harmful code on the hosting IoT device, two test programs simulating the behaviour of a malware and a backdoor were executed on different IoT devices. Specifically, a program with the signature of a malware was run on the Netgear router and a program acting similarly to a backdoor listening on TCP port 1111 was executed on the Udoo x86 board, as shown in Fig. 12. The Raspberry Pi 3 was instead left with no malicious code to show how the `AntibIoTic` agent behaves in this scenario.

Agent Execution. Once the `AntibIoTic` gateway was executing on the Fog node and two IoT devices were "infected", the `AntibIoTic` agent was executed on all IoT hosts.

At the startup, the `AntibIoTic` agent running on each device establishes two connections with



```
mido@fognodeintel-Kabylake-Client-platform:~/git/master/Server$ sudo ./bin/gateway -u
[Handler]> The Agent will not be automatically uploaded on IoT devices.
[Handler]> Listening on [::]:1234
[Spotter]> Listening on [::]:4321
[Handler]> New connection established: 192.168.0.1:60369
[Handler]> ' 192.168.0.1:60369 ': Agent version = 2.0
[Handler]> ' 192.168.0.1:60369 ': ID = 822083584
[Spotter]> ' 192.168.0.1:51311 ': keep-alive message
[Handler]> ' 192.168.0.1:60369 ': {module:stub, timestamp:953253432, data:{status:running}}
```

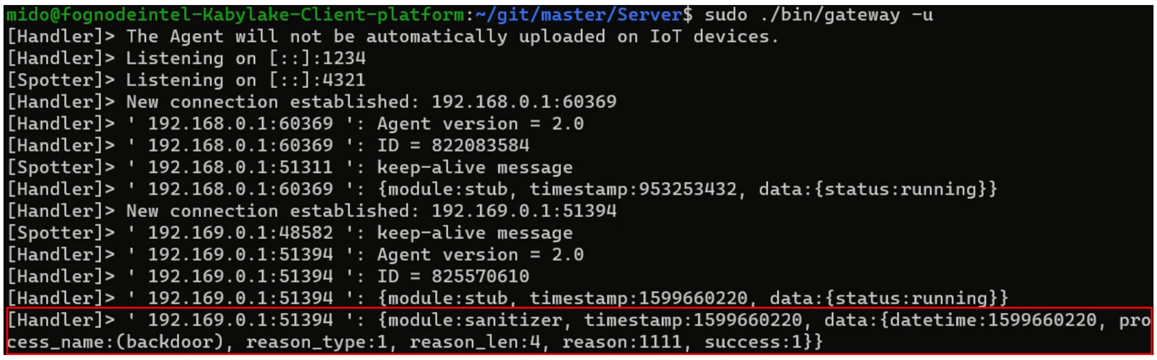
Fig. 13. The `AntibIoTic` gateway accepts two connections from the `AntibIoTic` agent running on the Netgear router (192.168.0.1): one on port TCP 4321 handled by the *spotter* module and one on port TCP 1234 managed by the *handler* module. The spotter module receives the first keep-alive message and the handler receives the startup information from the agent: agent version, device ID, and a first status update.

the Fog node: one with the *spotter* module of the gateway on port TCP 4321 and one with the *handler* module of the gateway on port TCP 1234. The connection with the spotter is used to send keep-alive messages; the connection with the handler is used to send relevant information to the gateway and to receive commands. As shown in Fig. 13 for the Netgear router, the first information transmitted from the agent to the handler are: the version of the agent in execution on the IoT device, device ID, and a first status updated confirming that the agent is correctly running on the hosting device.

In the meantime, the `AntibIoTic` agent starts a background scan of the hosting IoT device looking for malicious code; this process is repeated periodically with a time interval that can be set by the `AntibIoTic` gateway. The agent maintains a list of suspicious ports (that can be updated remotely by the gateway) and performs a scan of all open ports on the system. Suppose one of the suspicious ports is open; in that case, the agent kills the process listening on that port, binds itself to the same port to avoid further intrusions, adds an entry in the log file (also referred to as report), and sends a status update to the gateway. This scenario was tested on the Udoo x86 board running a backdoor and resulted in the termination of the malicious process, as proved by the status update received by the gateway and shown in Fig. 14. Similarly, the agent maintains a list of malware signatures (that can be updated remotely by the gateway) and performs a signature-based scan of all processes in execution. If a process matches one of the signatures in the list, the agent kills the process, adds an entry in the log file, and sends a status update to the gateway. This happened on the Netgear router running a malware and resulted in the termination of the malicious process and the generation of the status update shown in Fig. 15. The `AntibIoTic` agent was also executed on the Raspberry Pi where no malicious code was detected, thus, no status update was received by the gateway.

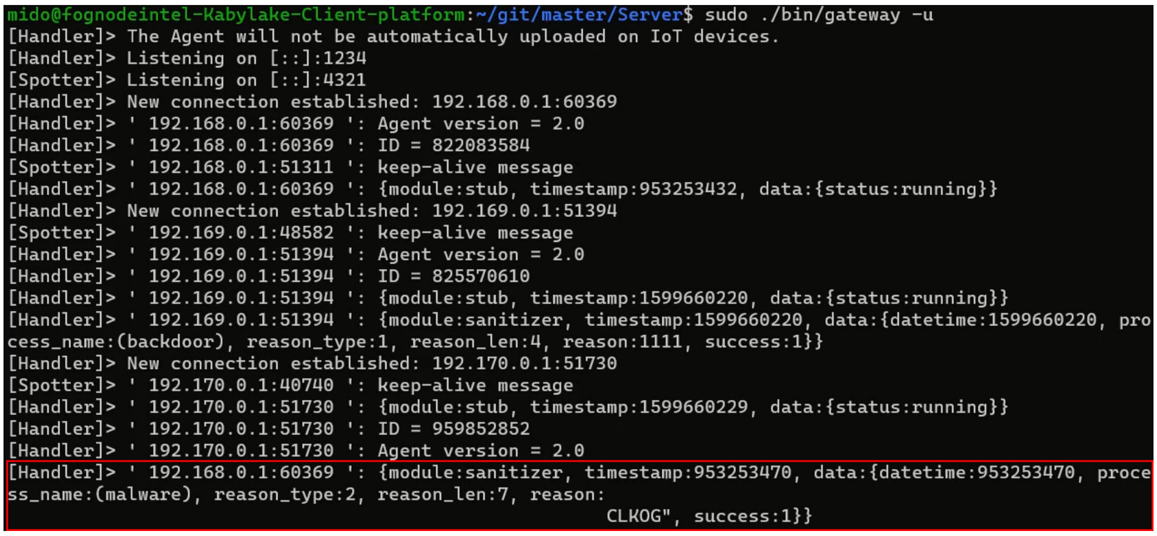
At this point, all IoT devices were correctly running the `AntibIoTic` agent, as proven by the keep-alive messages received by the gateway, and they were cleaned from malicious code, as shown by the status updates. Thus, all three devices were allowed to access the Internet. This was tested using the ‘`ping`’ network utility, as shown in Fig. 16 for the Udoo x86 board. Also, during its execution, the `AntibIoTic` agent kept its resources usage very limited, having an average CPU usage of around 4%, as largely discussed in Section 7.

Agent Termination. To conclude the proof-of-concept, it is shown what happens if the `AntibIoTic` agent running on an IoT device is terminated. This can occur due to different reasons. For example, the agent can be intentionally or unintentionally stopped by a user, it can be killed by the operating system to free resources or by another process running on the device, or it can



```
mido@fognodeintel-Kabylake-Client-platform:~/git/master/Server$ sudo ./bin/gateway -u
[Handler]> The Agent will not be automatically uploaded on IoT devices.
[Handler]> Listening on [::]:1234
[Spotter]> Listening on [::]:4321
[Handler]> New connection established: 192.168.0.1:60369
[Handler]> ' 192.168.0.1:60369 ': Agent version = 2.0
[Handler]> ' 192.168.0.1:60369 ': ID = 822083584
[Spotter]> ' 192.168.0.1:51311 ': keep-alive message
[Handler]> ' 192.168.0.1:60369 ': {module:stub, timestamp:953253432, data:{status:running}}
[Handler]> New connection established: 192.169.0.1:51394
[Spotter]> ' 192.169.0.1:48582 ': keep-alive message
[Handler]> ' 192.169.0.1:51394 ': Agent version = 2.0
[Handler]> ' 192.169.0.1:51394 ': ID = 825570610
[Handler]> ' 192.169.0.1:51394 ': {module:stub, timestamp:1599660220, data:{status:running}}
[Handler]> ' 192.169.0.1:51394 ': {module:sanitizer, timestamp:1599660220, data:{datetime:1599660220, pro
cess_name:(backdoor), reason_type:1, reason_len:4, reason:1111, success:1}}
```

Fig. 14. The AntibIoTic gateway receives a status update from the AntibIoTic agent running on the Udoo x86 board (192.169.0.1) announcing that a process named 'backdoor' was killed because listening on the suspicious port '1111', as reported in the 'reason' field.



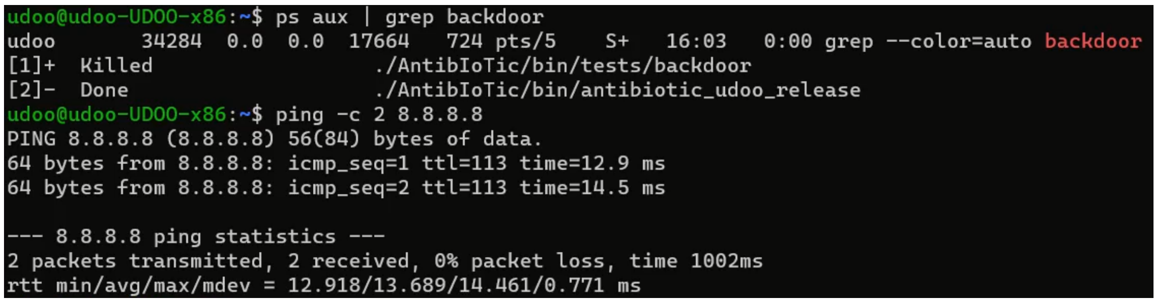
```
mido@fognodeintel-Kabylake-Client-platform:~/git/master/Server$ sudo ./bin/gateway -u
[Handler]> The Agent will not be automatically uploaded on IoT devices.
[Handler]> Listening on [::]:1234
[Spotter]> Listening on [::]:4321
[Handler]> New connection established: 192.168.0.1:60369
[Handler]> ' 192.168.0.1:60369 ': Agent version = 2.0
[Handler]> ' 192.168.0.1:60369 ': ID = 822083584
[Spotter]> ' 192.168.0.1:51311 ': keep-alive message
[Handler]> ' 192.168.0.1:60369 ': {module:stub, timestamp:953253432, data:{status:running}}
[Handler]> New connection established: 192.169.0.1:51394
[Spotter]> ' 192.169.0.1:48582 ': keep-alive message
[Handler]> ' 192.169.0.1:51394 ': Agent version = 2.0
[Handler]> ' 192.169.0.1:51394 ': ID = 825570610
[Handler]> ' 192.169.0.1:51394 ': {module:stub, timestamp:1599660220, data:{status:running}}
[Handler]> ' 192.169.0.1:51394 ': {module:sanitizer, timestamp:1599660220, data:{datetime:1599660220, pro
cess_name:(backdoor), reason_type:1, reason_len:4, reason:1111, success:1}}
[Handler]> New connection established: 192.170.0.1:51730
[Spotter]> ' 192.170.0.1:40740 ': keep-alive message
[Handler]> ' 192.170.0.1:51730 ': {module:stub, timestamp:1599660229, data:{status:running}}
[Handler]> ' 192.170.0.1:51730 ': ID = 959852852
[Handler]> ' 192.170.0.1:51730 ': Agent version = 2.0
[Handler]> ' 192.168.0.1:60369 ': {module:sanitizer, timestamp:953253470, data:{datetime:953253470, proce
ss_name:(malware), reason_type:2, reason_len:7, reason:
                                                   CLKOG", success:1}}
```

Fig. 15. The AntibIoTic gateway receives a status update from the AntibIoTic agent running on the Netgear router (192.168.0.1) announcing that a process named 'malware' was killed because it matched one of the signatures in the list; the exact signature is reported in the 'reason' field but it is also composed of non-printable characters.

be closed due to a device reboot. In the demo, the AntibIoTic agent was manually killed on all three IoT devices.

When the AntibIoTic agent is terminated, the AntibIoTic gateway detects that the connection the agent has with its spotter module is closed and immediately revokes the Internet access to the IoT endpoint. Figure 17 shows this scenario for the Netgear router. When a new connection is established between the agent and the gateway, the Fog node starts receiving the keep-alive messages again and will allow the IoT endpoint to access the Internet.

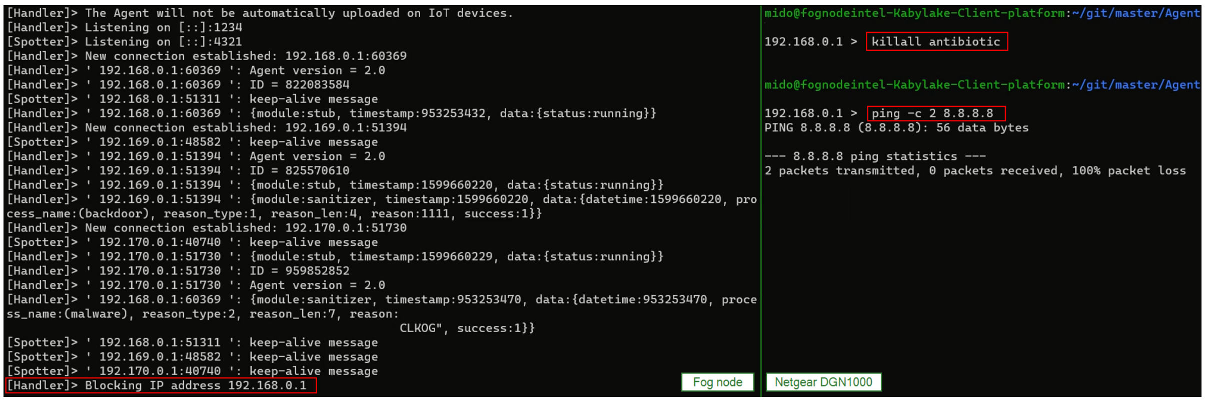
Additional features: The source code of the proof-of-concept of AntibIoTic 2.0 [DD20b] includes all the features needed to perform the demo shown in the previous section. In addition, it includes features not presented in the example above not to make it too long and chaotic, but worth mentioning. At the time of writing this manuscript, the additional features implemented



```
udoo@udoo-UDOO-x86:~$ ps aux | grep backdoor
udoo       34284  0.0  0.0  17664   724 pts/5    S+   16:03   0:00 grep --color=auto backdoor
[1]+  Killed                  ./AntibIoTic/bin/tests/backdoor
[2]-  Done                    ./AntibIoTic/bin/antibiotic_udoo_release
udoo@udoo-UDOO-x86:~$ ping -c 2 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=113 time=12.9 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=113 time=14.5 ms

--- 8.8.8.8 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1002ms
rtt min/avg/max/mdev = 12.918/13.689/14.461/0.771 ms
```

Fig. 16. Once the `AntibIoTic` agent removed the backdoor from the Udoo x86 board, the `AntibIoTic` gateway allowed the device to access the Internet as proven by the successful execution of a ‘`ping`’ towards an external host (8.8.8.8).



```
[Handler]> The Agent will not be automatically uploaded on IoT devices.
[Handler]> Listening on [::]:1234
[Spotter]> Listening on [::]:4321
[Handler]> New connection established: 192.168.0.1:60369
[Handler]> ' 192.168.0.1:60369 ': Agent version = 2.0
[Handler]> ' 192.168.0.1:60369 ': ID = 822083584
[Spotter]> ' 192.168.0.1:51311 ': keep-alive message
[Handler]> ' 192.168.0.1:60369 ': {module:stub, timestamp:953253432, data:{status:running}}
[Handler]> New connection established: 192.169.0.1:51394
[Spotter]> ' 192.169.0.1:48582 ': keep-alive message
[Handler]> ' 192.169.0.1:51394 ': Agent version = 2.0
[Handler]> ' 192.169.0.1:51394 ': ID = 825570610
[Handler]> ' 192.169.0.1:51394 ': {module:stub, timestamp:1599660220, data:{status:running}}
[Handler]> ' 192.169.0.1:51394 ': {module:sanitizer, timestamp:1599660220, data:{datetime:1599660220, pro
cess_name:(backdoor), reason_type:1, reason_len:4, reason:1111, success:1}}
[Handler]> New connection established: 192.170.0.1:51730
[Spotter]> ' 192.170.0.1:40740 ': keep-alive message
[Handler]> ' 192.170.0.1:51730 ': {module:stub, timestamp:1599660229, data:{status:running}}
[Handler]> ' 192.170.0.1:51730 ': ID = 959852852
[Handler]> ' 192.170.0.1:51730 ': Agent version = 2.0
[Handler]> ' 192.168.0.1:60369 ': {module:sanitizer, timestamp:953253470, data:{datetime:953253470, proce
ss_name:(malware), reason_type:2, reason_len:7, reason:
                                                       CLKOG", success:1}}
[Spotter]> ' 192.168.0.1:51311 ': keep-alive message
[Spotter]> ' 192.169.0.1:48582 ': keep-alive message
[Spotter]> ' 192.170.0.1:40740 ': keep-alive message
[Handler]> Blocking IP address 192.168.0.1
```

```
mido@fognodeintel-Kabylake-Client-platform:~/git/master/Agent

192.168.0.1 > killall antibiotic


mido@fognodeintel-Kabylake-Client-platform:~/git/master/Agent

192.168.0.1 > ping -c 2 8.8.8.8
PING 8.8.8.8 (8.8.8.8): 56 data bytes

--- 8.8.8.8 ping statistics ---
2 packets transmitted, 0 packets received, 100% packet loss
```

Fig. 17. The `AntibIoTic` agent running on the Netgear router was manually killed (on the right). As a result, the `AntibIoTic` gateway (on the left) refrains the IoT device (192.168.0.1) to access the Internet, as proven by the unsuccessful execution of the ‘`ping`’ command towards an external host (8.8.8.8).

in the proof-of-concept available on GitHub [DD20b] and not discussed in the demo include the following.

Periodic report. The `AntibIoTic` gateway periodically asks all `AntibIoTic` agents to send a report of recent activities performed on the hosting IoT device. The report includes status updates similar to the one the agent sends to the gateway when a malicious process is terminated.

Automatic uploads of the agent. The `AntibIoTic` gateway automatically compiles and uploads the `AntibIoTic` agent on each IoT device. This feature was inhibited during the demo by using the ‘`-u`’ flag when running the gateway.

Commands execution. After the first connection with the handler module of the gateway is established, the `AntibIoTic` agent can receive and execute a range of commands if requested by the gateway. For instance, the gateway can update both the list of malware signatures and the list of suspicious ports used by the agent, modify the time interval the agent waits to scan for malicious code, reboot the IoT device, or terminate the execution of the agent. The number and type of commands implemented by the agent on each IoT device might differ and heavily depend on the type of hosting endpoint.

Ensure single instance. The `AntibIoTic` agent implements an internal control feature that ensures only one concurrent instance of the agent is running on each device at the same time. If another running instance of the `AntibIoTic` agent is detected, that instance is terminated before proceeding.

Helper scripts. The GitHub repository of `AntibIoTic` 2.0 [DD20b] contains several helper scripts that can be used to automate the configuration and setup required to deploy `AntibIoTic` at the edge, both server- and client-side. This makes it quick and simple to deploy `AntibIoTic` in many IoT settings.

References

[A+09] K. Ashton et al., That 'Internet of Things' thing, *RFID journal* **22**(7) (2009), 97–114.

[AAA+20] K. Alieyan, A. Almomani, R. Abdullah, B. Almutairi and M. Alauthman, Botnet and Internet of Things (IoTs): A Definition, Taxonomy, Challenges, and Future Directions, in: *Security, Privacy, and Forensics Issues in Big Data*, IGI Global, 2020, pp. 304–316.

[AAAS20] J. Arshad, M.A. Azad, M.M. Abdeltaif and K. Salah, An intrusion detection framework for energy constrained IoT devices, *Mechanical Systems and Signal Processing* **136** (2020). doi:10.1016/j.ymssp.2019.106436.

[AAC+20] S. Aldhaheri, D. Alghazzawi, L. Cheng, A. Barnawi and B.A. Alzahrani, Artificial Immune Systems Approaches to Secure the Internet of Things: A Systematic Review of the Literature and Recommendations for Future Research, *Journal of Network and Computer Applications* **157** (2020), 102537. doi:10.1016/j.jnca.2020.102537.

[AAD+16] T. Abera, N. Asokan, L. Davi, F. Koushanfar, A. Paverd, A.-R. Sadeghi and G. Tsudik, Things, Trouble, Trust: on Building Trust in IoT Systems, in: *Proceedings of the 53rd Annual Design Automation Conference*, 2016, pp. 1–6. doi:10.1145/2897937.2905020.

[A.B21] A.B. Feroz Khan, G. Anandharaj, The Embedded Framework for Securing the Internet of Things, *Journal of Engineering Research* **9** (2021). doi:doi.org/10.36909/jer.v9i2.9823.

[ABD+20] M.N. Aman, M.H. Basheer, S. Dash, J.W. Wong, J. Xu, H.W. Lim and B. Sikdar, HAtt: Hybrid Remote Attestation for the Internet of Things with High Availability, *IEEE Internet of Things Journal* (2020). doi:10.1109/JIOT.2020.2983655.

[ACT20] M. Ammar, B. Crispo and G. Tsudik, SIMPLE: A Remote Attestation Approach for Resource-constrained IoT devices, in: *Proceedings of the 11th International Conference on Cyber-Physical Systems (ICCPS)*, IEEE, 2020, pp. 247–258. doi:10.1109/ICCPS48487.2020.00036.

[AD14] I. Anastasov and D. Davcev, SIEM implementation for global and distributed environments, in: *Proceedings of the World Congress on Computer Applications and Information Systems (WCCAIS)*, IEEE, 2014, pp. 1–6. doi:10.1109/WCCAIS.2014.6916651.

[AFK21] G.A. A.B. Feroz Khan, A Multi-layer Security approach for DDoS detection in Internet of Things, *International Journal of Intelligent Unmanned Systems* **9** (2021), 178–191. doi:doi.org/10.1108/IJIUS-06-2019-0029.

[AGMAA+20] M.A. Al-Garadi, A. Mohamed, A. Al-Ali, X. Du, I. Ali and M. Guizani, A Survey Of Machine And Deep Learning Methods For Internet Of Things (IoT) Security, *IEEE Communications Surveys & Tutorials* (2020). doi:10.1109/COMST.2020.2988293.

[AHdHS19] M. Al-Hawawreh, F. den Hartog and E. Sitnikova, Targeted Ransomware: A New Cyber Threat to Edge System of Brownfield Industrial Internet of Things, *IEEE Internet of Things Journal* **6**(4) (2019), 7137–7151. https://ieeexplore.ieee.org/abstract/document/8703829.

[AIM10] L. Atzori, A. Iera and G. Morabito, The Internet of Things: A Survey, *Computer networks* **54**(15) (2010), 2787–2805.

[AMNR20] M. Aslam, B. Mohsin, A. Nasir and S. Raza, FoNAC-An automated Fog Node Audit and Certification scheme, *Computers & Security* **93** (2020). doi:10.1016/j.cose.2020.101759.

[ARM+18] S. Alharbi, P. Rodriguez, R. Maharaja, P. Iyer, N. Bose and Z. Ye, FOCUS: A Fog Computing-based Security System for the Internet of Things, in: *Proceedings of the 15th Consumer Communications & Networking Conference (CCNC)*, IEEE, 2018, pp. 1–5. ISSN 2331-9860. doi:10.1109/CCNC.2018.8319238.

[Ass18] I.S. Association, 1934-2018-IEEE Standard for Adoption of OpenFog Reference Architecture for Fog Computing (2018). https://ieeexplore.ieee.org/document/8423800.

[BEK14] C. Bormann, M. Ersue and A. Keranen, Terminology for Constrained-Node Networks, *Internet Engineering Task Force (IETF), Request for Comments: 7229* (2014). https://tools.ietf.org/html/rfc7228.

[BMZ14] S. Bhatt, P.K. Manadhata and L. Zomlot, The operational role of security information and event management systems, *IEEE security & Privacy* **12**(5) (2014), 35–41. doi:10.1109/MSP.2014.103.

[BMZA12] F. Bonomi, R. Milito, J. Zhu and S. Addepalli, Fog computing and its role in the internet of things, in: *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*, 2012, pp. 13–16. doi:10.1145/2342509.2342513.

[Cis18] Cisco, Cisco Visual Networking Index: Forecast and Trends, 2017-2022, Technical Report, 2018. https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white-paper-c11-741490.pdf.

[Con16] I.I. Consortium, Industrial Internet of Things Volume G4: Security Framework, Technical Report, 2016. https://www.iiconsortium.org/IISF.htm.

[Con19] I.I. Consortium, The Industrial Internet of Things Volume G1: Reference Architecture, Technical Report, 2019. https://www.iiconsortium.org/IIRA.htm.

[CRM19] F. Concone, G.L. Re and M. Morana, A Fog-Based Application for Human Activity Recognition Using Personal Smart Devices, *ACM Transactions on Internet Technology (TOIT)* **19**(2) (2019), 1–20. doi:10.1145/3266142.

[24] E.T.C.C.S. (CYBER), Cyber Security for Consumer Internet of Things: Baseline Requirements, Technical Report, 2020. shorturl.at/fvGK4.

[DD17] M. De Donno, The AntibIoTic Against DDoS Attacks, 2017, M.Sc. Thesis, Technical University of Denmark (DTU).

[DD20a] M. De Donno, AntibIoTic 2 0 - Demo [Video], 2020. https://youtu.be/xiIKLREo3vY.

[DD20b] M. De Donno, AntibIoTic [source code], 2020. https://github.com/michele-dedonno/AntibIoTic.

[DDD19] M. De Donno and N. Dragoni, Combining AntibIoTic with Fog Computing: Antibiotic 2.0, in: *Proceeding of the 3rd International Conference on Fog and Edge Computing (ICFEC)*, IEEE, 2019, pp. 1–6.

[DDDGM16] M. De Donno, N. Dragoni, A. Giaretta and M. Mazzara, AntibIoTic: Protecting IoT Devices Against DDoS Attacks, in: *International Conference in Software Engineering for Defence Applications*, Springer, 2016, pp. 59–72.

[DDDGS17] M. De Donno, N. Dragoni, A. Giaretta and A. Spognardi, Analysis of DDoS-capable IoT malwares, in: *Proceedings of the Federated Conference on Computer Science and Information Systems (FedCSIS)*, IEEE, 2017, pp. 807–816.

[DDDGS18] M. De Donno, N. Dragoni, A. Giaretta and A. Spognardi, DDoS-capable IoT malwares: Comparative analysis and Mirai investigation, *Security and Communication Networks* **2018** (2018). doi:10.1155/2018/7178164.

[DDFD19] M. De Donno, J.M.D. Felipe and N. Dragoni, ANTIBIOTIC 2.0: a fog-based anti-malware for internet of things, in: *Proceedings of the European Symposium on Security and Privacy Workshops (EuroS&PW)*, IEEE, 2019, pp. 11–20.

[DDTD19] M. De Donno, K. Tange and N. Dragoni, Foundations and Evolution of Modern Computing Paradigms: Cloud, IoT, Edge, and Fog, *IEEE Access* **7** (2019), 150936–150948.

[DF18] J.M. Donaire Felipe, Using Fog Computing to Secure the IoT, 2018, M.Sc. Thesis, Technical University of Denmark (DTU).

[DGM17] N. Dragoni, A. Giaretta and M. Mazzara, The Internet of Hackable Things, in: *Proceedings of the 5th International Conference in Software Engineering for Defence Applications*, P. Ciancarini, S. Litvinov, A. Messina, A. Sillitti and G. Succi, eds, Springer, 2017, pp. 129–140. ISBN 978-3-319-70578-1.

[DKFD20] M. Dabbaghjamanesh, A. Kavousi-Fard and Z. Dong, A Novel Distributed Cloud-Fog Based Framework for Energy Management of Networked Microgrids, *IEEE Transactions on Power Systems* (2020). doi:10.1109/TPWRS.2019.2957704.

[dSWM+20] C.A. de Souza, C.B. Westphall, R.B. Machado, J.B.M. Sobral and G. dos Santos Vieira, Hybrid Approach to Intrusion Detection in Fog-based IoT Environments, *Computer Networks* **180** (2020). doi:10.1016/j.comnet.2020.107417.

[EAFVR19] R. El-Awadi, A. Fernández-Vilas and R.P.D. Redondo, Fog Computing Solution for Distributed Anomaly Detection in Smart Grids, in: *Proceedings of the International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, IEEE, 2019, pp. 348–353. doi:10.1109/WiMOB.2019.8923222.

[EAH18] M.F. Elrawy, A.I. Awad and H.F. Hamed, Intrusion Detection Systems for IoT-based Smart Anvironments: a Survey, *Journal of Cloud Computing* **7**(1) (2018), 21. doi:10.1186/s13677-018-0123-6.

[EP13] C. EU Parliament, Directive of the European Parliament and of the Council of 12 August 2013 on Attacks Against Information Systems and Replacing Council Framework Decision 2005/222/JHA, Vol. 218, 2013, [Accessed on July 15th, 2020]. https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32013L0040&from=EN.

[FMC19] L. Ferretti, M. Marchetti and M. Colajanni, Fog-based Secure Communications for Low-power IoT Devices, *ACM Transactions on Internet Technology (TOIT)* **19**(2) (2019), 1–21. doi:10.1145/3284554.

[FME+18] X. Fafoutis, L. Marchegiani, A. Elsts, J. Pope, R. Piechocki and I. Craddock, Extending the battery lifetime of wearable sensors with embedded machine learning, in: *Proceedings of the 4th World Forum on Internet of Things (WF-IoT)*, 2018, pp. 269–274. doi:10.1109/WF-IoT.2018.8355116.

[FTAK+20] M. Favaretto, T. Tran Anh, J. Kavaja, M. De Donno and N. Dragoni, When the Price Is Your Privacy: A Security Analysis of Two Cheap IoT Devices, in: *Proceedings of 6th International Conference in Software Engineering for Defence Applications*, P. Ciancarini, M. Mazzara, A. Messina, A. Sillitti and G. Succi, eds, Springer International Publishing, 2020, pp. 55–75.

[GDDD18] A. Giaretta, M. De Donno and N. Dragoni, Adding salt to pepper: A structured security assessment over a humanoid robot, in: *Proceedings of the 13th International Conference on Availability, Reliability and Security*, 2018, pp. 1–8.

[GDS16] R. Goyal, N. Dragoni and A. Spognardi, Mind the Tracker You Wear: A Security Analysis of Wearable Health Trackers, in: *Proceedings of the 31st Annual ACM Symposium on Applied Computing*, SAC '16, ACM, 2016, pp. 131–136. ISBN 978-1-4503-3739-7. doi:10.1145/2851613.2851685.

[GGT15] M.S. Giri, B. Gaur and D. Tomar, A survey on data integrity techniques in Cloud Computing, *International Journal of Computer Applications* **122**(2) (2015).

[Gro17] O.C.A.W. Group, OpenFog Reference Architecture for Fog computing, Technical Report, 2017. https://iiconsortium.org/pdf/OpenFog_Reference_Architecture_2_09_17.pdf.

[HVAP+16] F. Hosseinpour, P. Vahdani Amoli, J. Plosila, T. Hämäläinen and H. Tenhunen, An intrusion detection system for fog computing and IoT based logistic systems using a smart data approach, *International Journal of Digital Content Technology and its Applications* **10** (2016). https://jyx.jyu.fi/handle/123456789/54088.

[ITU12] T.S.S.O. ITU, Overview of the Internet of Things, *Recommendation ITU-T Y* **2060** (2012), 22. https://www.itu.int/rec/T-REC-Y.2060-201206-I.

[JACSC16] S.J. Johnston, M. Apetroaie-Cristea, M. Scott and S.J. Cox, Applicability of Commodity, Low Cost, Single Board Computers for Internet of Things Devices, in: *Proceedings of the 3rd World Forum on Internet of Things (WF-IoT)*, IEEE, 2016, pp. 141–146. doi:10.1109/WF-IoT.2016.7845414.

[Jin19] Y. Jin, Towards Hardware-Assisted Security for IoT Systems, in: *Proceeding of the Computer Society Annual Symposium on VLSI (ISVLSI)*, IEEE, 2019, pp. 632–637.

[KBL18] D.E. Kouicem, A. Bouabdallah and H. Lakhlef, Internet of things security: A top-down survey, *Computer Networks* **141** (2018), 199–221.

[Kha20] L. Khalid, Internet of Things (IoT), in: *Software Architecture for Business*, Springer, 2020, pp. 107–127. doi:10.1007/978-3-030-13632-1_7.

[KLD19] H. Kim, E.A. Lee and S. Dustdar, Creating a Resilient IoT With Edge Computing, *Computer* **52**(8) (2019), 43–53. doi:10.1109/MC.2018.2888768.

[KS20] K. Kaur and M. Sachdeva, Fog computing in IoT: An overview of new opportunities, in: *Proceedings of ICETIT 2019*, Springer, 2020, pp. 59–68. doi:10.1007/978-3-030-30577-2_5.

[LTM+11] F. Liu, J. Tong, J. Mao, R. Bohn, J. Messina, L. Badger and D. Leaf, NIST Cloud Computing Reference Architecture, Technical Report, 2011, 2011.

[LYZL18] Y. Lai, F. Yang, L. Zhang and Z. Lin, Distributed Public Vehicle System Based on Fog Nodes and Vehicular Sensing, *IEEE Access* **6** (2018), 22011–22024. doi:10.1109/ACCESS.2018.2824319.

[MCZ+19] F. Meneghello, M. Calore, D. Zucchetto, M. Polese and A. Zanella, IoT: Internet of Threats? A survey of practical security vulnerabilities in real IoT devices, *IEEE Internet of Things Journal* **6**(5) (2019), 8182–8201.

[NBHC+19] N. Neshenko, E. Bou-Harb, J. Crichigno, G. Kaddoum and 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.

[New20a] T.H. News, Dark Nexus: A New Emerging IoT Botnet Malware Spotted in the Wild, 2020, [Accessed on July 1st, 2020]. https://thehackernews.com/2020/04/darknexus-iot-ddos-botnet.html.

[New20b] T.H. News, Mukashi: A New Mirai IoT Botnet Variant Targeting Zyxel NAS Devices, 2020, [Accessed on July 1st, 2020]. https://thehackernews.com/2020/03/zyxel-mukashi-mirai-iot-botnet.html.

[Nex20] Nexusguard, DDoS Threat Report 2020 Q1, Technical Report, 2020. https://blog.nexusguard.com/threat-report/ddos-threat-report-2020-q1.

[NNNN20] Q.-D. Ngo, H.-T. Nguyen, L.-C. Nguyen and D.-H. Nguyen, A survey of IoT malware and detection methods based on static features, *ICT Express* (2020). doi:10.1016/j.icte.2020.04.005.

[NS20] B.A. NG and S. Selvakumar, Anomaly detection framework for Internet of things traffic using vector convolutional deep learning approach in fog environment, *Future Generation Computer Systems* **113** (2020), 255–265. doi:10.1016/j.future.2020.07.020.

[PB18] B. Paharia and K. Bhushan, Fog Computing as a Defensive Approach Against Distributed Denial of Service (DDoS): A Proposed Architecture, in: *Proceedings of the 9th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, IEEE, 2018, pp. 1–7. doi:10.1109/ICCCNT.2018.8494060.

[PSRC17] A. Perrig, P. Szalachowski, R.M. Reischuk and L. Chuat, *SCION: a Secure Internet Architecture*, Springer, 2017. https://www.scion-architecture.net/pdf/SCION-book.pdf.

[RROL18] R. Roman, R. Rios, J.A. Onieva and J. Lopez, Immune System for the Internet of Things Using Edge Technologies, *IEEE Internet of Things Journal* **6**(3) (2018), 4774–4781. doi:10.1109/JIOT.2018.2867613.

[RSS17] W. Razouk, D. Sgandurra and K. Sakurai, A New Security Middleware Architecture Based on Fog Computing and Cloud To Support IoT Constrained Devices, in: *Proceedings of the 1st International Conference on Internet of Things and Machine Learning*, 2017, pp. 1–8. doi:10.1145/3109761.3158413.

[SG19] D. Sehrawat and N.S. Gill, Smart Sensors: Analysis of Different Types of IoT Sensors, in: *Proceedings of the 3rd International Conference on Trends in Electronics and Informatics (ICOEI)*, IEEE, 2019, pp. 523–528. doi:10.1109/ICOEI.2019.8862778.

[SHŠ+19] D. Soukup, O. Hujňák, S. Štefunko, R. Krejčí and E. Grešák, Security Framework for IoT and Fog Computing Networks, in: *Proceedings of the 3rd International conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud)(I-SMAC)*, IEEE, 2019, pp. 87–92. doi:10.1109/I-SMAC47947.2019.9032592.

[SL18] M. Svarrer-Lanthén, Bot Module for AntibIoTic, 2018, B.Sc. Thesis, Technical University of Denmark (DTU).

[SMPS19] G. Selander, J. Mattsson, F. Palombini and L. Seitz, Object Security for Constrained RESTful Environments (OSCORE), *Work in Progress* (2019). https://www.hjp.at/doc/rfc/rfc8613.html.

[SRB20] J. Sengupta, S. Ruj and S.D. Bit, A Secure Fog Based Architecture for Industrial Internet of Things and Industry 4.0, *IEEE Transactions on Industrial Informatics* (2020). doi:10.1109/TII.2020.2998105.

[Sta16] Statista, Internet of Things (IoT) connected devices installed base worldwide from 2015 to 2025(in billions), 2016, [Accessed on June 29th, 2020]. https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide/.

[Sun20] A. Sunyaev, The Internet of Things, in: *Internet Computing*, Springer, 2020, pp. 301–337. doi:10.1007/978-3-030-34957-8_10.

[SYS20] SYSGO, PikeOS Certified Hypervisor, 2020, [Accessed on August 14th, 2020]. https://www.sysgo.com/products/pikeos-hyperviso.

[SYZ20] A. Samy, H. Yu and H. Zhang, Fog-Based Attack Detection Framework for Internet of Things Using Deep Learning, *IEEE Access* **8** (2020), 74571–74585. doi:10.1109/ACCESS.2020.2988854.

[WBD+17] P. Woznowski, A. Burrows, T. Diethe, X. Fafoutis, J. Hall, S. Hannuna, M. Camplani, N. Twomey, M. Kozlowski, B. Tan, N. Zhu, A. Elsts, A. Vafeas, A. Paiement, L. Tao, M. Mirmehdi, T. Burghardt, D. Damen, P. Flach, R. Piechocki, I. Craddock and G. Oikonomou, SPHERE: A Sensor Platform for Healthcare in a Residential Environment, in: *Designing, Developing, and Facilitating Smart Cities*, Springer, 2017, pp. 315–333. doi:10.1007/978-3-319-44924-1_14.

[ZGD19] L. Zhou, H. Guo and G. Deng, A Fog Computing Based Approach to DDoS Mitigation in IIoT Systems, *Computers & Security* **85** (2019), 51–62. doi:10.1016/j.cose.2019.04.017.