electronics

MDPI

Article

Towards Transparent and Secure IoT: Improving the Security and Privacy through a User-Centric Rules-Based System

João Lola [1], Carlos Serrão [1,*] and João Casal [2]

[1] Information Sciences, Technologies and Architecture Research Center (ISTAR), Instituto Universitário de Lisboa (ISCTE-IUL), 1600-189 Lisboa, Portugal; joao_pedro_lola@iscte-iul.pt
[2] SCNL Truphone, S.A., 1700-158 Lisboa, Portugal; joao.casal@truphone.com
* Correspondence: carlos.serrao@iscte-iul.pt

Abstract: In recent years, we have seen a growing wave in the integration of IoT (Internet of Things) technologies into society. This has created new opportunities, but at the same time given rise to several critical issues, creating new challenges that need to be addressed. One of the main challenges is the security and privacy of information that is processed by IoT devices in our daily lives. Users are, most of the time, unaware of IoT devices' personal information collection and transmission activities that affect their security and privacy. In this work, we propose a solution that aims to increase the privacy and security of data in IoT devices, through a system that controls the IoT device's communication on the network. This system is based on two basic and simple principles. First, the IoT device manufacturer declares their device's data collection intentions. Second, the user declares their own preferences of what is permitted to the IoT device. The design of the system includes tools capable of analyzing packets sent by IoT devices and applying network traffic control rules. The objective is to allow the declaration and verification of communication intentions of IoT devices and control the communication of such devices to detect potential security and privacy violations. We have created a test-bed to validate the developed solution, based on virtual machines, and we concluded that our system has little impact on how the overall system performed.

Keywords: security; privacy; IoT networks; intent declaration; communication rights and permissions; traffic analysis

check for updates

Citation: Lola, J.; Serrão, C.; Casal, J. Towards Transparent and Secure IoT: Improving the Security and Privacy through a User-Centric Rules-Based System. *Electronics* **2023**, *12*, 2589. https://doi.org/10.3390/electronics12122589

Academic Editors: Tiago Cruz and Bruno Sousa

Received: 13 April 2023
Revised: 29 May 2023
Accepted: 6 June 2023
Published: 8 June 2023

1. Introduction

Information security and privacy are two of the most critical topics when addressing Internet-related technology. They represent the two most fundamental elements in establishing trust between companies and their users. An organisation that does not possess strong data security can be seen by users as less trustworthy than a competitor. Relaxed data security policies can lead to risks of compromising personally identifiable information, personal health information, medical records, banking information, intellectual property, and other types of valuable or sensitive data. Without a proper security strategy, companies can suffer data breaches that may result in huge reputation damages and financial losses. In a worst-case scenario, this may even terminate the organisation itself. Therefore, information security systems must enforce three essential principles: confidentiality, integrity, and availability.

The increasingly rapid growth of Internet-connected IoT devices is causing privacy and security concerns, as reported in several studies [1]. Most of these concerns are related to relaxed communication security policies to prevent security infringements and data leaks. Other problems reside in the fact that IoT devices are poorly designed from a security standpoint, which makes them easily exploitable by any malicious actor. Looking at the literature, it is possible to find multiple examples of this. For instance, Amazon's Alexa sends privately recorded conversations to random numbers on the owner's list of contacts [2]; Google Nest IP cameras, doorbells, and thermostats can be hacked by third parties [3]; Google Home Hub captures footage from Xiaomi Mijia surveillance cameras from other dwellings [4]; and Samsung Smart TVs had a vulnerability that allowed third parties to view everything that was going on in the room where the TV was located via the embedded camera and microphone [5]. This security panorama is of concern.

Another aspect is also related to the IoT device manufacturers' lack of transparency on the way they collect, process, protect, and transmit data about their users. One example was Google's failure to disclose in its Nest Guard product specifications that the device had a built-in microphone [6]. Another example was Amazon's revelation that, in addition to artificial intelligence algorithms, their employees listened to samples of recordings of Amazon Echo devices to improve the quality of service [7,8].

A study conducted by Mozilla revealed that 45% of the 190,000 participants surveyed considered that privacy was the biggest concern when it came to the use of IoT devices, and 34.5% of them considered that it was up to them to keep connected IoT devices secure and private, versus 34% who believed that it was up to device manufacturers to ensure the privacy and security of their devices [9]. The study reveals that consumers do not trust manufacturers to keep their devices secure and feel it is up to them to safeguard their privacy and security. The results of this study reflect that many consumers consider that the privacy and security of data should be assured both by them and the device manufacturers.

In another study conducted by the Economist Intelligence Unit, a global leader in business intelligence, 92% of the 1629 IoT consumers surveyed said that privacy was their primary concern and wished to be informed when IoT devices were collecting their personal information and having greater control over the data collected and transmitted [10].

This work focuses on a specific consumer IoT device that connects through the consumer's networks using a specific gateway. It addresses some of the consumer IoT security risks by proposing a system that will control the data communication between the IoT device on a network and any external systems. The control is established in two different ways. One of the ways is through the IoT device manufacturer's declaration about what data will be collected and sent by the device. The other concerns the specific data permissions the IoT device owner wants to impose on their network. So, in a scenario where we have an IoT device that wants to send video data to the network (and the manufacturer expresses such intention), the user may also use their specific preferences, imposing that the IoT device can only send such video data in a particular time of the day. The proposed system focuses on ensuring transparency of information transmitted by consumers' IoT devices on networks, giving them control over how they communicate and their behaviour to achieve higher levels of privacy and security.

In this article, we start by providing a short introduction to the problem and explain how this problem is relevant in this context. After, a short section about some related work is presented, where some related approaches are presented and how they approach similar problems. The following section introduces and explains the system developed in this work, explains how the system works, and highlights some results. In the final section of this article, some conclusions are presented about the conducted work.

2. Related Work

This section presents a summary of the relevant literature in this field. To conduct this literature review, we used well-known scientific databases (Google Scholar, IEEExplore, and ACM Library) and searched for terms such as "iot security and privacy" and "iot communication control". We have only considered works published in English since 2016.

The scientific literature contains a large number of articles published on the topic of IoT security and privacy. Although the IoT ecosystem brought immense benefits, it creates several challenges, especially regarding security and privacy. Handling these issues and ensuring security and privacy for IoT products and services must be a fundamental priority. Users need to establish trust in IoT devices and related services. Moreover, IoT devices may be under attack, and their functionality may be changed by attackers, therefore, IoT device's behaviour must be controlled. The authors of this work provide a discussion of IoT security, privacy, safety, and ethics [11]. Furthermore, the authors often discuss IoT's different challenges in terms of security and privacy and propose some possible solutions [12–14].

One of the main security concerns regarding IoT devices is their susceptibility to cyber-attacks. Some of the analysed works proposed using intrusion detection systems based on machine learning and blockchain technologies [15–18]. These approaches often use machine learning algorithms to detect abnormal behaviour in IoT devices and blockchain technology to securely store intrusion detection information [19–21]. This is a similar approach to the one followed by this article; however, in our case, we have introduced the opportunity for the user to control what the consumer IoT device can do on the user's home network.

Another important aspect of IoT security is firmware security [22]. Some authors propose an approach that uses a combination of static and dynamic analysis techniques to identify vulnerabilities in firmware and a fuzz testing approach to verify firmware robustness [23,24].

In addition to technical solutions, user education is an important aspect of IoT security. Some authors propose an IoT security awareness framework for end-users [25]. Their approach educates users on basic IoT security measures, such as password management and network security [26–28].

However, despite all the proposed solutions in the literature, IoT security and privacy remains a major challenge. As the IoT scene continues to expand and evolve, addressing security and privacy challenges will require a multifaceted approach that includes technical solutions, user education, and regulatory measures. As reviewed, recent works proposed various frameworks and technologies to enhance the security and privacy of IoT devices, including intrusion detection, firmware security, blockchain, edge computing, machine learning, and supply chain security. However, as the IoT continues to evolve and expand, much work still needs to be conducted to ensure its security and protect users' privacy.

The work that is presented in this article addresses IoT security by exploring the capability of enforcing normative functionality to IoT devices and detecting violations of those behaviours. Moreover, this article presents contributions to the state of the art in two different key aspects. First, it allows the manufacturers of consumer IoT devices to declare their intentions in terms of data collection from the user and communication to the network. These intentions can be expressed through a specific Web interface and REST API (although more options may be available) converted to a specific rules-based controlled environment that will confine the IoT device's communication behaviour. Second, our system also puts the end-user in control of its privacy because it allows users to establish their preferences regarding data that the IoT device can collect and send through the network. Combining the declaration of the manufacturer's intentions and the end-users' preferences will contribute to a more secure and private IoT environment.

3. System Design

This section presents the system prototype implemented and the components that ensure the transparency of the information transmitted by consumers' IoT devices in the networks, providing them with control over their devices' communications and ensuring that they communicate following what is stated on their manifests.

3.1. Architecture

In the architecture of the system (Figure 1), there are two different main actors: the "**IoT Device Manufacturer**" and the "**IoT Device Owner**". The "**Intent Rules**" represents the device manufacturer's traffic control rules, and the "**User Rules**" represents the consumer IoT device owner's traffic control rules.

"**Intent Rules**" depict the intentions and specifications of the "**IoT Device Manufacturer**", to be defined in the IoT device manifest, regarding the device communication requirements, the information the device sensors collects and what is transmitted to the network. These rules will make the communication transparent to the consumers that will be using the device. Furthermore, they will make it possible to check whether the device's behaviour complies with its declared manifest. This is considered normal behaviour if the device complies with the manifest intentions. If not, the system will inform the "**IoT Device Owner**", and it will be up to them to decide what to do next (e.g., to block a device from communicating sound to the Internet). Finally, the "**IoT Device Owner**" is empowered to act upon their device's behaviour with the "**User Rules**".

The "**User Rules**" represents the IoT device owner's intentions and specifications regarding what they permit the device to send to the Internet. These rules give the owner of the device complete control over it. In this way, the owner can create rules to block the device from sending unwanted information that, in their view, violates their privacy and security.

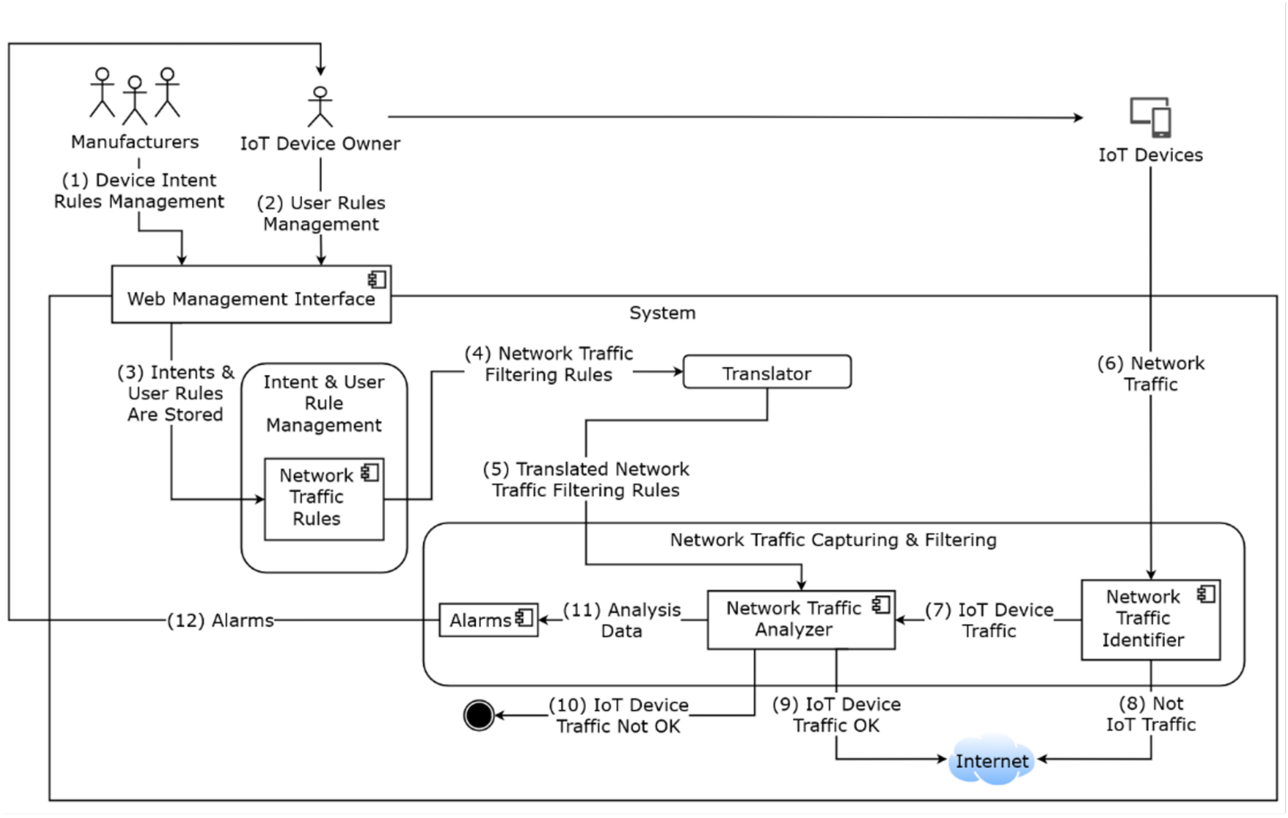

Manufacturers
IoT Device Owner
IoT Devices
(1) Device Intent Rules Management
(2) User Rules Management
Web Management Interface
System
(3) Intents & User Rules Are Stored
Intent & User Rule Management
Network Traffic Rules
(4) Network Traffic Filtering Rules
Translator
(5) Translated Network Traffic Filtering Rules
(6) Network Traffic
Network Traffic Capturing & Filtering
Network Traffic Identifier
(7) IoT Device Traffic
Network Traffic Analyzer
(11) Analysis Data
Alarms
(12) Alarms
(10) IoT Device Traffic Not OK
(9) IoT Device Traffic OK
(8) Not IoT Traffic
Internet

Figure 1. System high-level architecture.

By creating these two types of traffic control rules, we have implemented whitelisting and blacklisting of data communication. The blacklist concept is applied to the "**User Rules**". In this case, we create rules that prevent the packets from reaching their destination to enforce the user's control over their IoT traffic to control their privacy and security. The whitelist concept is applied to the "**Intent Rules**" and aims to describe the rules based on the IoT device declaration of intentions (as described previously). These rules are used by the system to learn when devices are not complying with their manifest by creating rules that trigger specific alarms (e.g., according to the packets' content and destination). We can also generate blacklist rules from the whitelist rules if necessary.

The "**Intent and User Rules Management**" service has two components. The "**Web Management Interface**", provides the appropriate mechanisms for the consumer IoT device owner and manufacturer to enter, modify, and delete user-defined rules and manufacturer declaration of intents, respectively. The "**Network Traffic Rules**" component represents the backend of the interface, where the CRUD (Create, Read, Update, and

Delete) database operations of the user rules and manufacturer declaration of intents are implemented.

The system uses its own rules specification to define the traffic control rules in the “**Intents and User Rule Management**” service. Our traffic control rule specification is compact, generic, and easy to understand because it was created from scratch with network traffic control in mind.

The “**Translator**” is the service that receives the traffic control rules from the “**Intent and User Rules Management**” service and translates them into the rule’s expression technique used by the “**Network Traffic Capturing and Filtering**” service to define traffic control rules. This service is implemented on the same machine where the “**Network Traffic Capturing and Filtering**” service is installed and configured to store the translated rules directly in the rules folder of the “**Network Traffic Capturing and Filtering**” service configuration folders. This translation service could have been implemented directly in the “**Intent and User Rules Management**” service but we decided to implement it in a separate service to ensure its portability and interoperability with other existing traffic policy control systems.

The “**Network Traffic Capturing and Filtering**” service consists of three components: “**Network Traffic Identifier**”, “**Network Traffic Analyser**”, and “**Alarms**”. These three components were implemented using the Snort IDPS installed and configured to perform the IoT device’s network traffic control.

In the case of the implemented system, and also a limitation of it, we assumed that the content of the packets issued by the IoT devices and captured by the “**Network Traffic Capturing and Filtering**” service is transported via HTTP (Hypertext Transfer Protocol) messages and not HTTPS (Hypertext Transfer Protocol Secure) messages, so the content that is parsed is not encrypted.

This option also finds strengths in the research of [29], where some of the devices used in their studies revealed that the communication of device states and credentials was completed via plain HTTP with other devices and the outside world. Their study also showed that some devices received software updates via HTTP and that most of this traffic corresponds to requested images and video thumbnails from manufacturer servers. In another study [30], they noted that all smart home IoT devices, except for smart cameras, access the web over HTTP for some portion of traffic, with health and wearable devices, intelligent assistants, and smart TV corresponding to around 20% of traffic over HTTP.

The “**Network Traffic Identifier**” is the component that identifies whether traffic being sent to the Internet originates from a consumer IoT device. If so, it is sent to the “**Network Traffic Analyser**”; otherwise, the network traffic goes directly to the Internet. For simulation purposes, all the generated consumer IoT traffic captured by this component is created by the Ostinato [31] traffic generator before the “**Network Traffic Capturing and Filtering**” service. Ostinato simulates the consumer IoT device’s behaviour when violating the manufacturer’s manifest communication intents and user-defined rules. The identifier of IoT traffic is contained inside the packet that is captured for analysis and is made up of the TAC (Type Allocation Code), a unique 8-digit code, which allows identification of the manufacturer and model of the device individually within a network for the manufacturer intents, and the IMEI (International Mobile Equipment Identity), a unique 15-digit code which acts as a fingerprint of the device, allowing it to be identified within a network for the user-defined rules.

The “**Network Traffic Analyser**” analyses the IoT traffic according to defined traffic control rules. If the traffic complies with what is specified, it is redirected to the Internet; if not, alarms will be triggered if traffic violates the manufacturer’s intent manifest or blocked from reaching the Internet if the traffic violates the user-defined rules.

The “**Alarms**” component receives information about network traffic that complied with the rules and the traffic that did not. This component mainly alerts the IoT device owner to the system activity, especially to any device with behaviour not declared by the manufacturer. With this information, the user may block traffic from the device that falls outside the intentions manifest. The information presented to the owner consists of the timestamp of the captured packet, the transport protocol, source and destination addresses, and the content it was carrying that triggered the specific policy rule violation.

As previously stated, regarding the "**Network Traffic Capturing and Filtering**" service, we decided to use the Snort IPS Deep Packet Content Filtering and Blocking, a tool designed primarily for network traffic intrusion detection. This tool is configured to act as an IPS, enabling it to perform DPI (Deep Packet Inspection) on packets captured from the network and apply user-defined rules to block packets or allow them to pass through to the network.

The typical usage of the system, according to the architecture presented in Figure 1 could go as follows:

1. The "**IoT Device Manufacturer**" specifies and definies their own intentions that are translated to simple rules ("**Intent Rules**");
2. The "**IoT Device Owner**" specifies and definies their own rules ("**User Rules**");
3. Both the "**Intent Rules**" and the "**User Rules**" are stored on the "**Intent and User Rules Management**" service;
4. Rules are sent to the "**Translator**" service that is responsible for conducting the rules translation to the appropriate network filtering service (in this case, it will be Snort);
5. "**Translator**" service sends the translated rules to the "**Network Traffic Capturing and Filtering**" service through the "**Network Traffic Analyser**" component;
6. "**IoT Devices**" produce network traffic. This network traffic is captured by the "**Network Traffic Capturing and Filtering**" service, through the "**Network Traffic Identifier**" component;
7. If the "**Network Traffic Identifier**" identifies the traffic as having origin in an IoT device, the traffic is redirected to the "**Network Traffic Analyser**";
8. If it is not IoT device traffic, it is redirected to the Internet;
9. If the "**Network Traffic Analyser**" verifies that the traffic is compliant with the rules, then the traffic is routed to the internet;
10. If not, traffic is blocked;
11. "**Network Traffic Analyser**" component produces a set of analytical data that will be used to generate alarms or simply collect information about the system usage;
12. Alarm information is sent by the "Alarms" component of the system to the "**IoT Device Owner**".

3.1.1. Intents and User Rule Management REST API

The Intents and User Rule Management REST API is designed to allow manufacturers to state their communication intentions and enable the IoT device owner to create user rules to control the information the devices communicate. This API provides manufacturers and IoT device owners alike with endpoints to allow the intentions and user-defined rules to be retrieved from the local database to be displayed. The entry methods enable the manufacturer to enter their device communication intent statements and the IoT device owner to insert their defined rules to control the device communication intents into the database. The update methods allow for modifying existing intents and user-defined rules in the database. Finally, the delete method enables the withdrawal of device intents or user-defined rules from the database.

For this REST API, two entities were created to represent the manufacturer's intent and the user-defined rules, designated as intent and rule.

A UUID (Universal Unique Identifier) defines an intent and the device type the TAC represents. Furthermore, part of the intent is the properties, as defined by the content type, such as Audio, Video, or Text, by the communication protocol, which is used to represent the message that contains that content, such as HTTP, RTSP (Real-Time Streaming Protocol), or SIP (Session Initiation Protocol), and by the destination address of the intent, which is represented by a range of IPs to which the content is to be sent.

A rule has the same fields as the intent, except the device identifier field, represented by the device's IMEI.

3.1.2. Translator (Integration Layer with Filtering Tool)

The Translator REST API is designed to allow the translation of intents and IoT device owner rules that it receives from the Intents and User Rule Management REST API into the rules format used by Snort. This REST API provides the methods to create Snort-compatible rules from the manufacturer's intents and user-defined rules created in the Intents and User Rule Management REST API. The update methods allow modification of the existing Snort rules by replacing them with new ones entered by the manufacturer or IoT device owner. The delete method enables the withdrawal of device Snort rules from the rule files located in the Snort configuration folders. To allow Snort to detect and interpret the content of a packet it has captured, a method has been created to transform the content type it receives from the Intents and User Rule Management REST API service from text to hexadecimal when the Snort rule is created.

For this REST API, another entity called a Snort rule has been created to complement existing ones in Intents and User Rule Management REST API. A Snort rule is defined by:

- **Rule Action**—specifies what action the Snort rule performs on the packet it received, such as alert, drop, log, pass, reject, or drop.
- **Rule Protocol**—specifies which protocol should be parsed for suspicious behaviour, such as TCP, UDP, ICMP, or IP.
- **Source IP**—specifies the packet's source IP(s) address(es).
- **Source Port**—specifies which source port(s) of the packet.
- **Flow Direction**—specifies which direction of the packet flow, unidirectional (->) or bidirectional (<>).
- **Destination IP**—specifies the destination address(es) of the packet.
- **Destination Port**—specifies the destination port(s) of the packet.
- **Rule Options**—specify what content the rule should analyse, where it is located within the packet, the packet size, the message shown when the alert is triggered, and the rule ID.

3.1.3. Snort IPS

Snort IPS [32] is the chosen tool to detect manufacturer intention violations and allow IoT device owners to block content communication according to their security and privacy preferences. Snort is an open-source rules-based NIDPS (Network Intrusion Detection and Prevention System) software to detect and prevent malicious attacks or information leakages, with the ability to analyse traffic in real-time, log packets, and generate alerts for users when packet content matches the defined rules.

To enable the option to block packets with content that violates the privacy and security of the owner from reaching their destination, it was necessary to configure Snort using the "`snort.conf`" file, selecting the DAQ (Data Acquisition Library) module "`Afpacket`". This allows us to run Snort in "inline" mode with two network interfaces. Only in "inline" mode can Snort block the packets it captures. In "normal" mode, it only generates alerts for the captured packets. Through the "`buffer_size_mb`" variable, we can define the memory to be assigned to the DAQ, considering the amount of traffic to be analysed, the number of rules activated, and the hardware Snort is running on. It was also necessary to configure Snort to consider the rule files for each device, both for the manufacturer's intentions and for the IoT device owner's rules. The following represents the command that allows Snort to run. It is made up of four main arguments and an optional fifth argument. The alerts mode (1), your configuration file (2), the interfaces where Snort "listens" for traffic to analyse (3), and the operation mode (4):

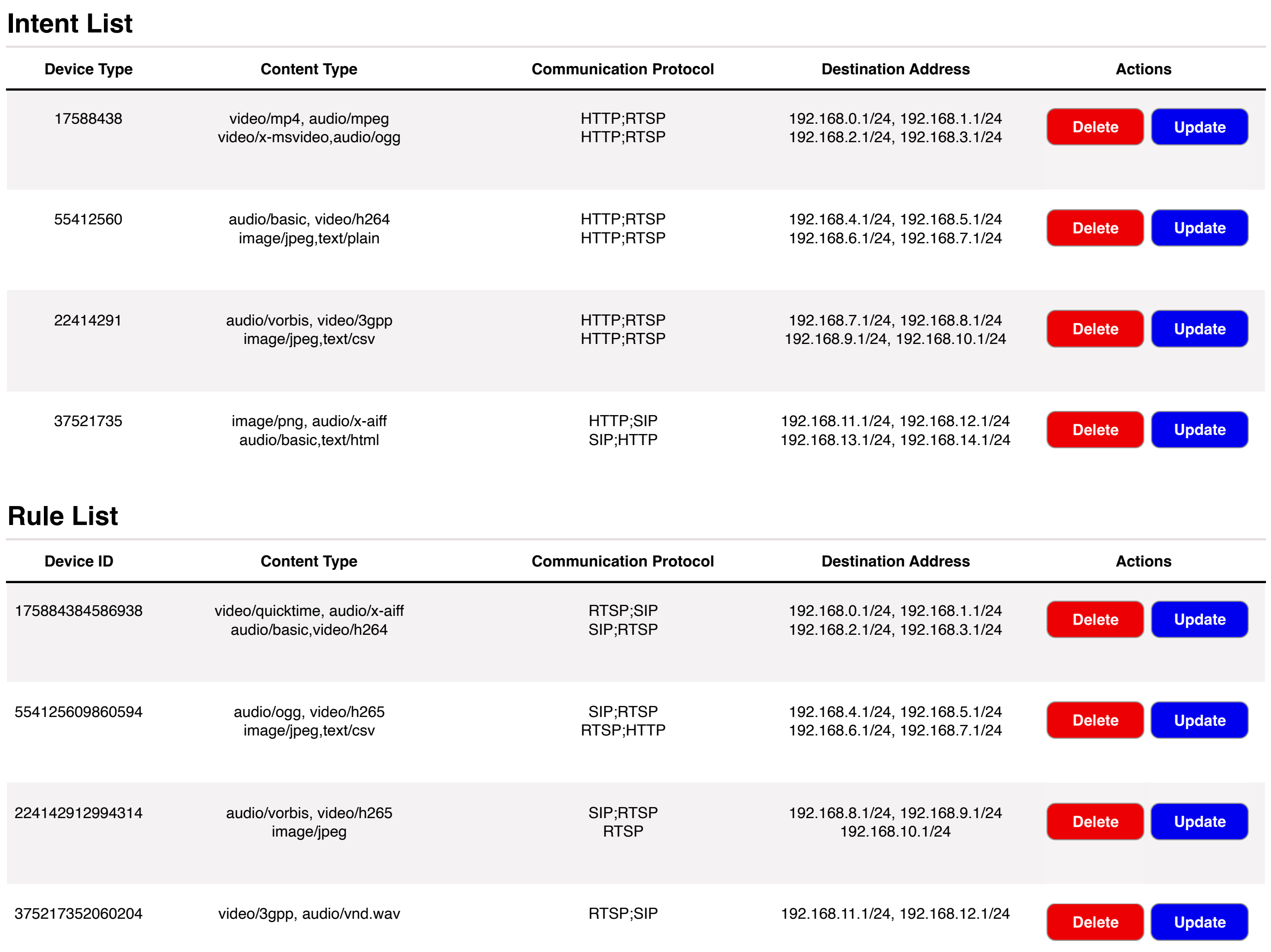
```
sudo snort -A console -c /etc/snort/snort.conf -i ens33:ens34 -Q
```

- `-A console`: this argument configures Snort so that the alerts it generates are sent to the console.
- `-c /etc/snort/snort.conf`: this argument allows you to choose the configuration file of the Snort settings that should be executed.
- `-i ens33: ens34`: this argument chooses the interfaces to listen for traffic.
- `-Q`: this argument allows you to run Snort in inline mode.

3.1.4. Web Management Interface

The Web Management Interface was designed to support the Intent and User Rule Management REST API and, consequently, the Translator backend. This interface allows the manufacturer to read, insert, update, and delete the device intentions rules. Additionally, it enables the IoT device owner to read, insert, update, and delete user-defined rules.

The intentions and user-defined rules are displayed in tables in the intent list and rule list tabs, respectively, as shown in Figure 2. The intents table displays Device Type (TAC), Content-Type, Communication Protocol, Destination Address, and Actions (Delete and Update). The rules table indicates the Device ID (IMEI), Content-Type, Communication Protocol, Destination Address, and Actions (Delete and Update).



Intent List

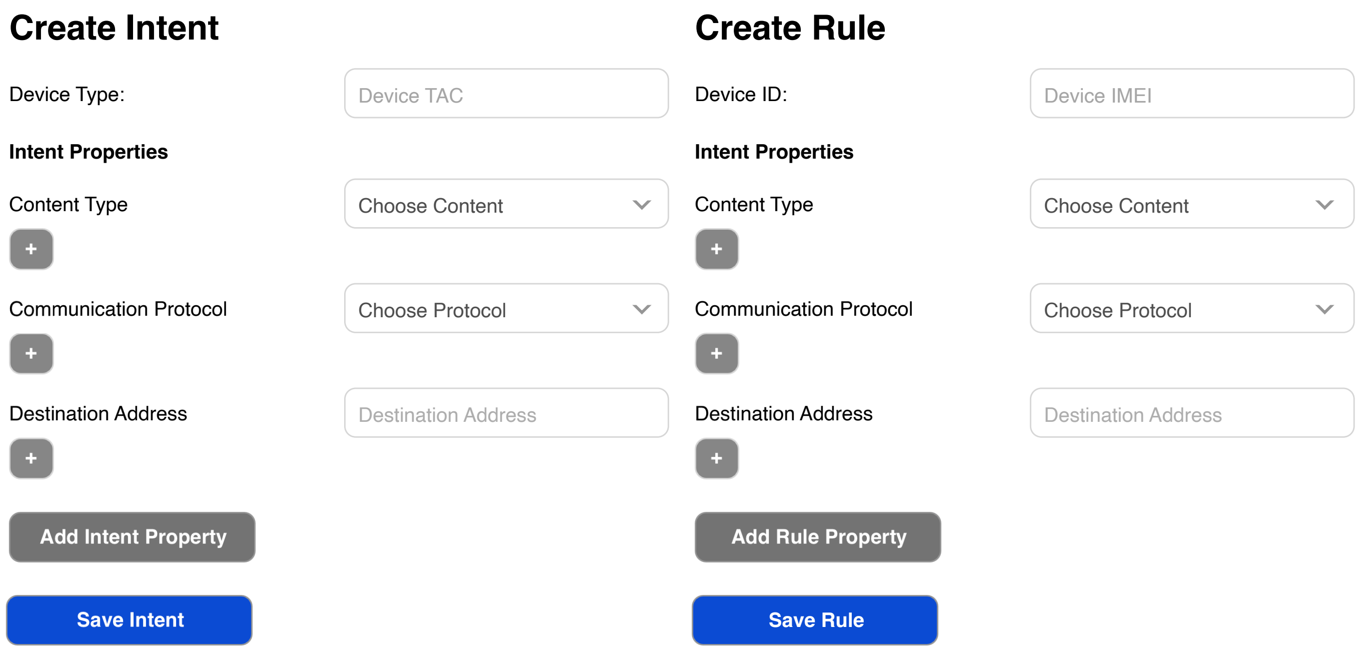
Device Type	Content Type	Communication Protocol	Destination Address	Actions
17588438	video/mp4, audio/mpeg video/x-msvideo,audio/ogg	HTTP;RTSP HTTP;RTSP	192.168.0.1/24, 192.168.1.1/24 192.168.2.1/24, 192.168.3.1/24	Delete Update
55412560	audio/basic, video/h264 image/jpeg,text/plain	HTTP;RTSP HTTP;RTSP	192.168.4.1/24, 192.168.5.1/24 192.168.6.1/24, 192.168.7.1/24	Delete Update
22414291	audio/vorbis, video/3gpp image/jpeg,text/csv	HTTP;RTSP HTTP;RTSP	192.168.7.1/24, 192.168.8.1/24 192.168.9.1/24, 192.168.10.1/24	Delete Update
37521735	image/png, audio/x-aiff audio/basic,text/html	HTTP;SIP SIP;HTTP	192.168.11.1/24, 192.168.12.1/24 192.168.13.1/24, 192.168.14.1/24	Delete Update

Rule List

Device ID	Content Type	Communication Protocol	Destination Address	Actions
175884384586938	video/quicktime, audio/x-aiff audio/basic,video/h264	RTSP;SIP SIP;RTSP	192.168.0.1/24, 192.168.1.1/24 192.168.2.1/24, 192.168.3.1/24	Delete Update
554125609860594	audio/ogg, video/h265 image/jpeg,text/csv	SIP;RTSP RTSP;HTTP	192.168.4.1/24, 192.168.5.1/24 192.168.6.1/24, 192.168.7.1/24	Delete Update
224142912994314	audio/vorbis, video/h265 image/jpeg	SIP;RTSP RTSP	192.168.8.1/24, 192.168.9.1/24 192.168.10.1/24	Delete Update
375217352060204	video/3gpp, audio/vnd.wav	RTSP;SIP	192.168.11.1/24, 192.168.12.1/24	Delete Update

Figure 2. Intent and Rules list table.

To enter a new intent, the manufacturer must navigate to the intent form in the intent tab, where they are greeted with the above-mentioned fields. After they enter the data and save the intent in the database, it is automatically translated by the system into a Snort rule, utterly transparent to the manufacturer. The translated intents will be written in a file corresponding to the device's TAC ID. When the manufacturer deletes a device from the database, the file containing the corresponding Snort rules for that device is also deleted. When the device intents are updated, the intents that existed previously are replaced with those entered by the manufacturer. The same process is followed for the user-defined rules entered by the IoT device owner in the rule tab, but with the slight difference that user-defined rules are written to a file corresponding to the device IMEI ID (Figure 3).


Create Intent

Device Type:	Device TAC
Intent Properties	
Content Type	Choose Content
Communication Protocol	Choose Protocol
Destination Address	Destination Address

Create Rule

Device ID:	Device IMEI
Intent Properties	
Content Type	Choose Content
Communication Protocol	Choose Protocol
Destination Address	Destination Address

Figure 3. Intent and Rules Insert/Update form.

For more advanced users who have experience working with Snort and its rules, we have provided an appropriate interface that allows them to download the text rule files to edit (located in the download tab) and upload them again (located in the upload tab), replacing the existing ones with the ones they have modified to their liking.

3.2. Snort Implementation Workflow

This section presents the Snort workflow for each packet captured by the Network Traffic Filtering and Capturing service. We focus on two different scenarios. The first scenario is when the IoT device violates the communication intents: this happens when the device deviates from its expected behaviour as defined by the manufacturer. This deviation may occur because the manufacturer tries to misbehave or because the device has been compromised. The second scenario occurs when the IoT device violates the user-stipulated rules: this occurs when the device deviates from the behaviour the user sets for it.

The flow that represents the device intent communication violation scenario is presented in Figure 4. Snort starts by capturing a packet sent by an IoT device for inspection (1). Then, the decoder can analyse that packet to identify the transport protocol used and check for conflicts with the intent rules (2). When the decoder finishes the identification and classification process, it sends the packet to the pre-processor (3), which is responsible for preparing the packet to be processed more easily by the detection engine, which flags the traffic flow by looking for a match in the ruleset for traffic that is violating the defined policies (4). If a match is found by TAC (device type) and content type in packet load, an intents violation alert is generated (5), followed by event logging (6). Otherwise, Snort takes another packet and repeats the same process. In this scenario, Snort acts in detection mode only, as it does not perform any action on the packet but simply triggers an alert for a defined policy violation.

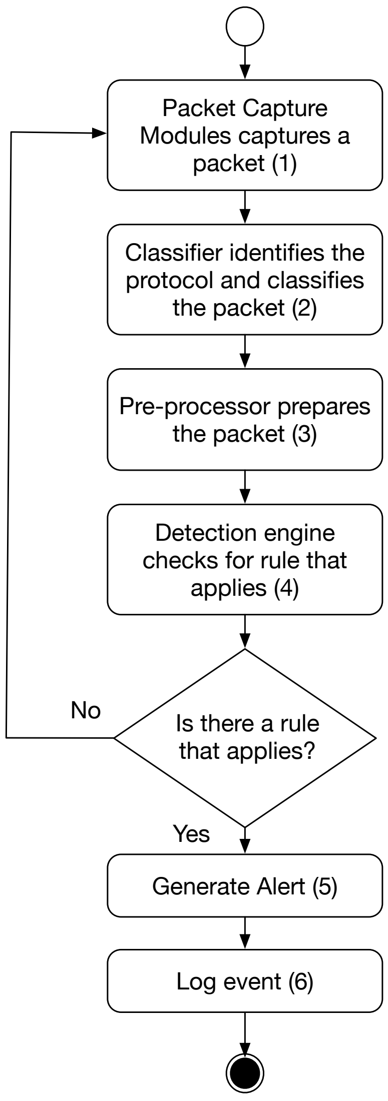

Packet Capture Modules captures a packet (1)
Classifier identifies the protocol and classifies the packet (2)
Pre-processor prepares the packet (3)
Detection engine checks for rule that applies (4)
Is there a rule that applies?
No
Yes
Generate Alert (5)
Log event (6)

Figure 4. Intent scenario—system workflow.

Considering the flow of the user-defined rules violation scenario presented in Figure 5, when it is detected that a device sends a packet that represents a violation of the privacy and security of its owner, the process is the same, but after generating the alert, Snort blocks and discards the packet (6) that contains the device IMEI and the content type the IoT device owner has specified to be blocked, stopping it from reaching its destination, and then logs the event that occurred (7). Again, if no packet matches the user-defined rules, another is captured for analysis. In this scenario, Snort acts in prevention mode because it stops the packet from reaching its destination by performing a drop action on the packet.

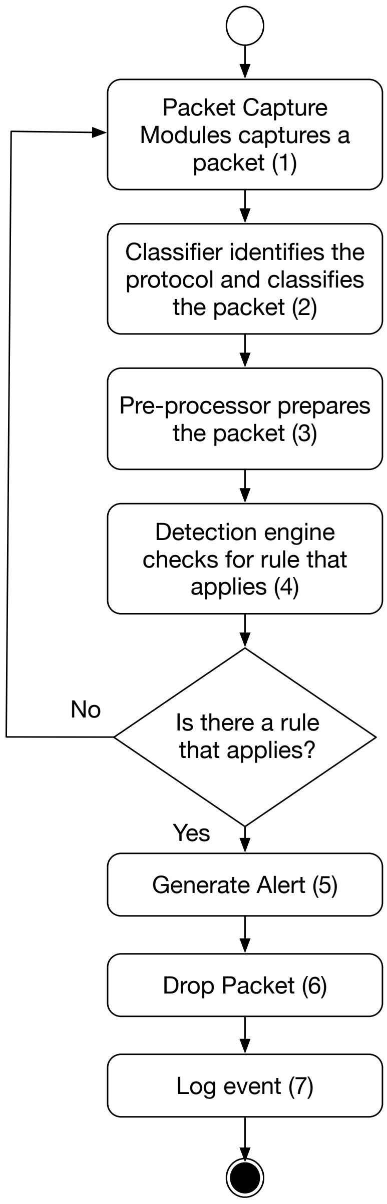

Packet Capture Modules captures a packet (1)
Classifier identifies the protocol and classifies the packet (2)
Pre-processor prepares the packet (3)
Detection engine checks for rule that applies (4)
Is there a rule that applies?
No
Yes
Generate Alert (5)
Drop Packet (6)
Log event (7)

Figure 5. Rule scenario—system workflow.

When the manufacturer enters the intents for the IoT device using the Web Management Interface, the intents are stored and then sent from the Intent and User Management Rest API to the Translator REST API to be translated from intent into alert Snort rules because in the intent case, our aim is only to alert the IoT device owner to the fact that their device is not complying with their communication intents, without blocking the content.

```
alert tcp $HOME_NET any -> [192.168.19.1/24,192.168.20.1/24] any (msg:
"ALERT -DESTINATION ADDRESS NOT OK For Intent Of Device TAC 17020105.
Intent{ContentType: [audio/mpeg, video/mp4] Destination Address:
[192.168.19.1/24, 192.168.20.1/24]";flow:stateless; content:"|31 37
30 32 30 31 30 35|"; offset:4; depth:8; content:"|5b 61 75 64 69 6f
2f 6d 70 65 67 2c 76 69 64 65 6f 2f 6d 70 34 5d|"; distance:14;
within:22;classtype:policy-violation; sid:1137585;)
```

When the IoT device owner enters the user-defined rules using the Web Management Interface, the rules are stored and then sent from the Intents and User Rule Management REST API to the Translator REST API to be translated from a user-defined rule into a drop

Snort rule, because in the user rule case, we want to prevent some of the content sent by the devices reaching the Internet, violating owner privacy and security.

```
drop tcp $HOME_NET any -> [192.168.0.1/24,192.168.1.1/24] any (msg:
"DROP - Rule Device IMEI 175884384586938{ Content-Type:
[video/quicktime, audio/x-aiff] Destination Address: [192.168.0.1/24,
192.168.1.1/24] }"; flow:stateless; content: "|31 37 35 38 38 34 33
38 34 35 38 36 39 33 38|"; offset:5; depth:15; content: "|5b 76 69
64 65 6f 2f 71 75 69 63 6b 74 69 6d 65 2c 61 75 64 69 6f 2f 78 2d
61 69 66 66 5d|"; distance:14; with-in:30;classtype:policy-violation;
sid:8181915;)
```

4. System Validation and Testing

This section describes the test environment implemented to perform the required tests to test the developed solution. The developed solution was tested on a virtualised environment, as described in the next sections of this article.

This system was tested in two different aspects. The first one tested the different functionalities of the developed system throughout two specially defined scenarios: one that analyses the system's capability for detecting violations of the IoT manufacturer intent rules and another that analyses the system's capability for detecting violations of the user-defined preferences. The second aspect that was tested was the system's performance where we analysed the system's behaviour when high volumes of traffic were generated.

4.1. Test System Deployment

The test deployment system presented in Figure 6 was created to allow the testing of the violation of manufacturer-defined intents and violation of the user-defined rules scenarios.

The intents scenario is essential to test if the IoT device is not complying with the behaviour defined by the manufacturer because we need proof that guarantees to us those manufacturers and their devices can be trusted and are not being used to spy on innocent consumers. The user-defined rules scenario is vital to test so that consumers possess a mechanism to protect themselves against an untrustworthy manufacturer using the devices in their home to gather information about them.

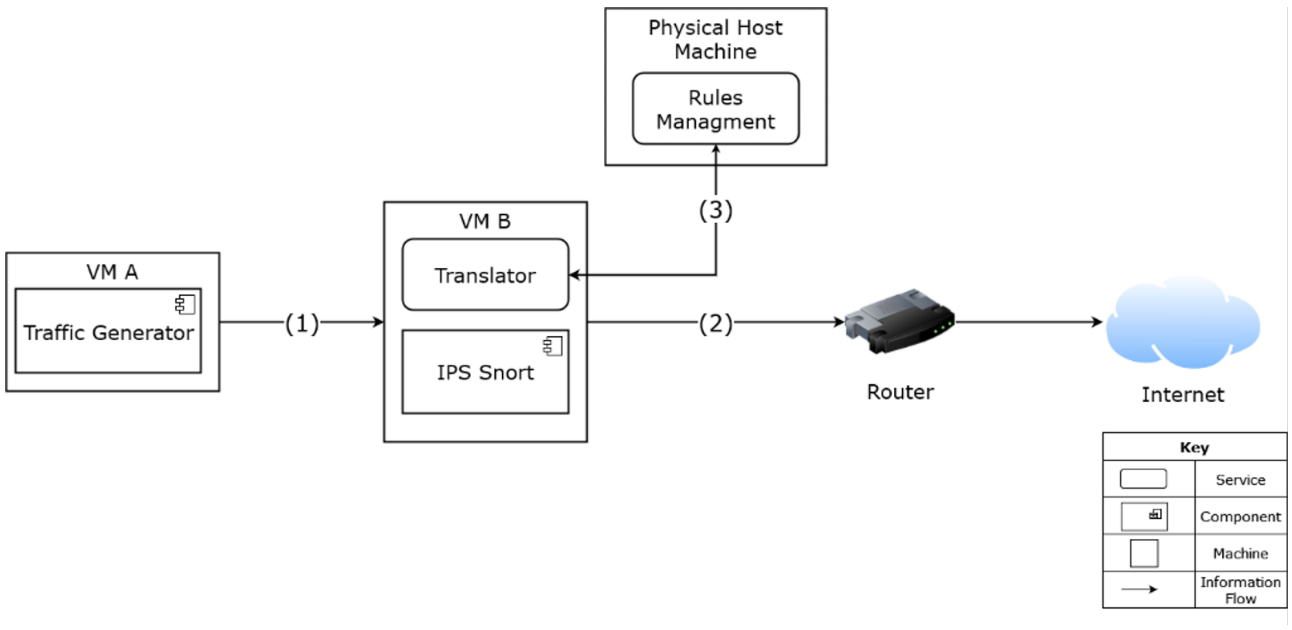


Figure 6. Test Deployment system.

This system comprises two virtual machines, VM A and VM B, and a physical host machine. The approach presented was inspired by a solution offered before [33]. Our system, much like Hafeez's, uses Snort in intrusion "prevention" mode for network traffic control by making all the traffic from the Traffic Generator (VM A) pass through the Translator and Snort IPS (VM B) before reaching the Internet router allowing DPI analysis of the packets and application of the desired traffic control rules to the packet flow.

Virtual machine A (VM A) was created to represent an IoT device to generate network traffic collected and analysed by Snort IPS, located on virtual machine B (VM B) (1). Virtual machine B (VM B) was created to contain the traffic control and analysis software that collects and analyses traffic from the IoT devices connected to this virtual machine. In addition, this machine acts as an internal server to check IoT device network traffic compliance with the traffic control rules and to forward it to the Internet if no violations occurred; if any violation did occur, an alert would be displayed, followed by blocking of the traffic (2). The physical host machine contains the Web Management Interface and the Intent and User Rule Management REST API that allow manufacturers and IoT device users to specify intents and define rules for their devices by creating, reading, modifying, and deleting rules contained within a database. The Interface component of the Rules Management service calls the endpoint of the Translator service (3).

The need for a virtual machine stems from the open-source software selected here, which is designed to run on open-source Linux core-based operating systems. As this is open-source, it is free of use.

An open-source version of the Ostinato software [31] was chosen to generate traffic streams to simulate the various devices present on an IoT network. Ostinato is an open-source tool that allows us to test bandwidth, create, and configure multiple streams, specify how data are sent, set the number of packets or bursts to be sent per second, view real-time statistical data for network monitoring and measurement, read statistical measurements of lost packets per stream, and offer support for the most common standard protocols.

Using this software, we created multiple streams to cover the IoT device owner rules and the manufacturer-specified intent scenarios. For these streams, we defined the protocol data settings, the packet's source and destination IP address, the transport protocol, and the HTTP, RTSP, or SIP text messages. In the intent case, the TAC of the device, the content it transmits, and—in the case of the IoT device owner's rule—the IMEI of the device and the content it sends. Finally, we defined the stream control settings, the type we wish to send—whether packet or burst—the number of packets or bursts, the rate of packets or bursts per second, and what action to take when the stream ends, as explained above.

4.2. Functional Tests

In this section, we introduce the functional scenarios used in the proposal: one that tests intent violations and another that tests the user's control over their devices' communication. We also present and discuss the test results.

These tests are essential to ensure that our implemented system performs as expected. To prove this, we must test the system to see if it can detect violations of the declared manufacturer intents and generate alerts whenever the IoT device transmits packets that carry information that does not match the specifications of the intents manifest of the IoT device manufacturer. To conduct this test, the system creates test packets using the traffic generator to simulate the misbehaviour of the IoT device. These packets must contain information that diverges from the information specified in the manufacturer's intents manifest. This test is successful if alerts are triggered for each packet that violates the manufacturer's intent manifest specifications. Thus, such devices will not be able to compromise the privacy and security of the owner, who can be assured that devices do what is stated in their manifest. To prove that the IoT device owner has control over the privacy and security of the device's communication, we must create test packets using the same traffic generator as mentioned above with content that violates user-defined rules, so when the system detects this content, it blocks it from reaching its destination. In this manner, the system implemented can assure the IoT device owner of control over their device's communication and guarantee their privacy and security.

4.2.1. Test Scenario I—Violations of the Manifest of Intents

In this test scenario, we simulate network traffic that violates manufacturers' intents and analyse the system's reaction. To do so, we use the intents of three different devices, which have been specified by the manufacturer, as set out in Table 1.

Table 1. Test Intents.

Device	Device Type (TAC)	Declared Content-Type	Declared Communication Protocol	Declared Destination Addresses
Samsung Android Tablet	09951213	[video/mp4, audio/mpeg]	[HTTP, RTSP]	[192.168.21.0/24, 192.168.22.0/24]
		[video/x-msvideo, audio/ogg]	[HTTP, RTSP]	[192.168.23.0/24, 192.168.24.0/24]
Apple iOS Smartphone	42245111	[audio/basic, video/h264]	[HTTP, RTSP]	[192.168.25.0/24, 192.168.26.0/24]
		[image/jpeg, text/plain]	[HTTP, RTSP]	[192.168.27.0/24, 192.168.28.0/24]
Huawei Android Smartphone	33292418	[audio/mpeg, video/mp4]	[SIP, RTSP]	[192.168.29.0/24, 192.168.30.0/24]

Analysing the intents defined by the manufacturers in the previous table, we see how they have specified communication intents for three well-known IoT devices. First, the unique eight-digit TAC is required to identify a device by its brand and model on the network that it is connected to. The content type that the device sends to the Internet is specified using MIME (Multipurpose Internet Mail Extensions) types. This content can originate from the device camera, microphone, location sensor, humidity sensor, or other device sensors. Finally, the communication protocol represents the protocol used to transport the TAC and content-type information to one of the IPs defined in the destination address.

The intents content type and destination addresses, as shown in Table 1, must be denied so that it is possible to validate whether the manufacturer-defined intent is fulfilling the device communication intentions. This way, we can evaluate if the content type and destination addresses defined in the device's intents are the same as those the device sends to the Internet. If true, users should not be confronted with them in an execution environment, as they comply with the manufacturer-defined intent. If not, the user should be presented with alert messages in an execution environment, alerting them to non-compliance with the manufacturer-defined intent.

After the intents are created and translated, it is necessary to create network traffic, meaning the packets, content type, and destination address—which must be different from what is specified by the manufacturer so that the alerts can be triggered. Table 2 shows an example of a packet whose content type does not match the content type specified by the manufacturer but whose destination address corresponds to one of the ranges. Table 3 shows a packet with a destination address that differs from the specified one whose contents match the set.

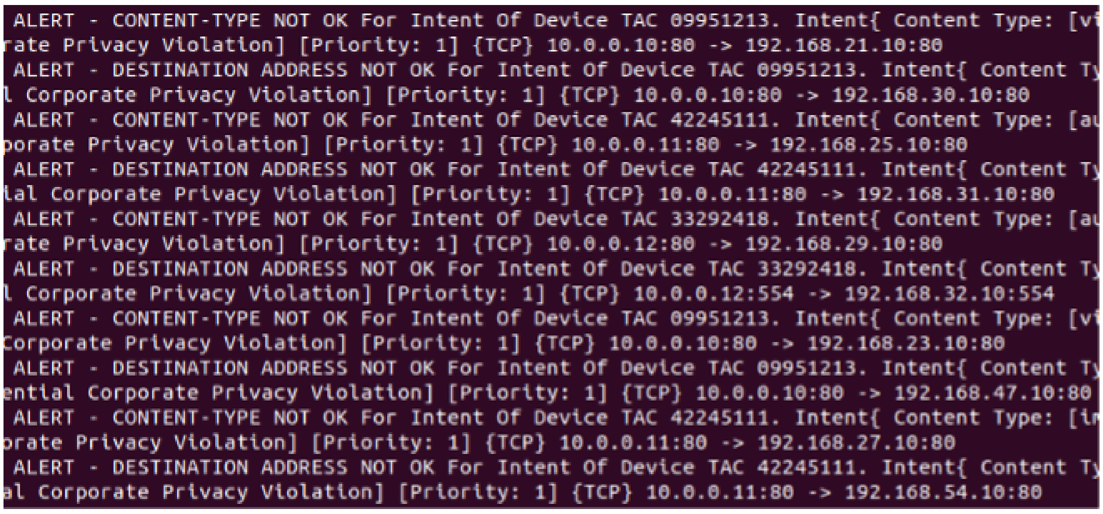
Table 2. Content type not matching.

Protocol Data	
Source IP: 10.0.0.10	Destination IP: 192.168.21.10
Text Protocol	
TAC:09951213 Content-Type: [text/plain, video/h264]	

Table 3. Destination address not matching.

Protocol Data	
Source IP: 10.0.0.10	Destination IP: 192.168.30.10
Text Protocol	
TAC:09951213 Content-Type: [video/mp4, audio/mpeg]	

The following picture (Figure 7) represents the Snort messages displayed when the content and destination address of the previously created packets do not match the content and destination address specified by the manufacturer in the intent.



```
 ALERT - CONTENT-TYPE NOT OK For Intent Of Device TAC 09951213. Intent{ Content Type: [vi
rate Privacy Violation] [Priority: 1] {TCP} 10.0.0.10:80 -> 192.168.21.10:80
 ALERT - DESTINATION ADDRESS NOT OK For Intent Of Device TAC 09951213. Intent{ Content Ty
l Corporate Privacy Violation] [Priority: 1] {TCP} 10.0.0.10:80 -> 192.168.30.10:80
 ALERT - CONTENT-TYPE NOT OK For Intent Of Device TAC 42245111. Intent{ Content Type: [au
porate Privacy Violation] [Priority: 1] {TCP} 10.0.0.11:80 -> 192.168.25.10:80
 ALERT - DESTINATION ADDRESS NOT OK For Intent Of Device TAC 42245111. Intent{ Content Ty
ial Corporate Privacy Violation] [Priority: 1] {TCP} 10.0.0.11:80 -> 192.168.31.10:80
 ALERT - CONTENT-TYPE NOT OK For Intent Of Device TAC 33292418. Intent{ Content Type: [au
rate Privacy Violation] [Priority: 1] {TCP} 10.0.0.12:80 -> 192.168.29.10:80
 ALERT - DESTINATION ADDRESS NOT OK For Intent Of Device TAC 33292418. Intent{ Content Ty
l Corporate Privacy Violation] [Priority: 1] {TCP} 10.0.0.12:554 -> 192.168.32.10:554
 ALERT - CONTENT-TYPE NOT OK For Intent Of Device TAC 09951213. Intent{ Content Type: [vi
Corporate Privacy Violation] [Priority: 1] {TCP} 10.0.0.10:80 -> 192.168.23.10:80
 ALERT - DESTINATION ADDRESS NOT OK For Intent Of Device TAC 09951213. Intent{ Content Ty
ential Corporate Privacy Violation] [Priority: 1] {TCP} 10.0.0.10:80 -> 192.168.47.10:80
 ALERT - CONTENT-TYPE NOT OK For Intent Of Device TAC 42245111. Intent{ Content Type: [ir
orate Privacy Violation] [Priority: 1] {TCP} 10.0.0.11:80 -> 192.168.27.10:80
 ALERT - DESTINATION ADDRESS NOT OK For Intent Of Device TAC 42245111. Intent{ Content Ty
al Corporate Privacy Violation] [Priority: 1] {TCP} 10.0.0.11:80 -> 192.168.54.10:80
```

Figure 7. Intent violation alerts.

Analysing the messages displayed, Snort tells us that, for example, the device with the **TAC 099551213** has committed four intent violations. The first was that the content type the device was sending did not match the content type the manufacturer specified in their intent: the manufacturer specified the device was supposed to send the content `[video/mp4, audio/mpeg]`, but instead, after analysing the packet on Wireshark from source IP `10.0.0.10:80` to destination IP `192.168.21.10:80` and **TAC 099551213**, we saw that the content the device sent was `[text/plain, video/h264]`. The second was that the destination address to which the packet was intended to go did not match the address the manufacturer specified in their intent: the manufacturer specified that the device was supposed to send the content `[video/mp4, audio/mpeg]` to one IP address from the following IP ranges `[192.168.21.1/24, 192.168.22.1/24]`, but instead, after analysing the packet from source IP `10.0.0.10:80` and **TAC 099551213**, we saw that the actual destination IP address was `192.168.30.10`. The two other intent violations are identical to those described above, only with different destination addresses and content types.

In the Snort log folder, we can view the files with the capture timestamp and where the captured packets are recorded, which can be opened in a program such as Wireshark or another relevant program. These logs represent the packets whose content triggered the alarms for intent manifest violations. Thus, we possess detailed reports that users can check for the number of traffic control rule violations and the number of alarms triggered in the process. Furthermore, depending on how the Snort alarm mode is configured, we can also access the log files in the same folder of the Snort events to check the number of alarms and triggered rules.

In this test, we have shown that the detection and alarm functional process succeeded. The system detected that the virtual device content and destination address in the test packets were not compliant with the content type and destination address specified in the device intentions. Therefore, it generated a log of alerts for these events, reporting that the content or content destination was incorrect according to the device intent manifest specifications. If the device was indeed complying with the device intents, no log of alerts would have been generated, meaning no violations of intents occurred that could have jeopardised the privacy and security of the IoT device owner.

In the eventuality of a manufacturer failing to correctly specify their device's real communication intentions, our system will be able to inform the IoT device owner, whether their device is reliable or not, meaning it will tell the device owner if the device is complying with the manufacturer's intents or not.

4.2.2. Test Scenario II—Violations of the User Rules

In this test scenario, we simulate the violation of defined rules of privacy and information security by three devices and how the system is supposed to react when such an event occurs. To do this, we use the following rules specified by the owner, which enable the submission of unapproved content to be blocked, as seen in Table 4.

Table 4. Test User Rules.

Device	Device ID (IMEI)	Content-Type Allowed by Device Owner	Communication Protocol Allowed by Device Owner	Destination Address Allowed by Device Owner
Samsung Android Tablet	099512133857038	[video/quicktime, audio/x-aiff]	[RTSP, SIP]	[192.168.21.0/24, 192.168.22.0/24]
		[audio/basic, video/h264]	[SIP, RTSP]	[192.168.23.0/24, 192.168.24.0/24]
Apple iOS Smartphone	422451118095078	[audio/ogg, video/h265]	[SIP, RTSP]	[192.168.25.0/24, 192.168.26.0/24]
		[image/jpeg, text/csv]	[RTSP, SIP]	[192.168.27.0/24, 192.168.28.0/24]
Huawei Android Smartphone	332924181938544	[image/jpeg]	[RTSP]	[192.168.29.0/24, 192.168.30.0/24]

In the previous table (Table 4), we show user-defined rules to prevent certain types of content from three IoT devices from leaking to the Internet, possibly violating users' privacy and security. The user rules seen here are defined by their IMEI, which acts as the device identifier, and the content type that the user has chosen to block is also specified using the MIME types.

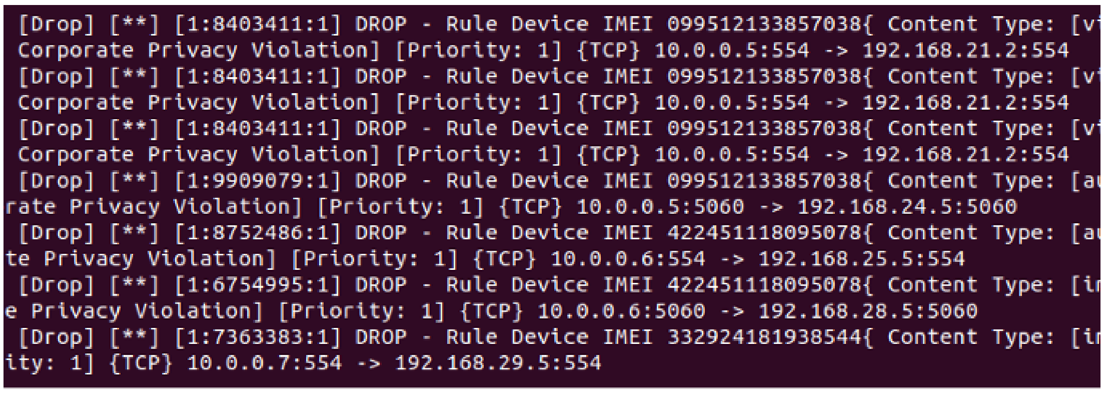
The user-defined rules, shown in Table 4, are written to prevent packets that meet the specified conditions from reaching the Internet. Whenever it is found that a device with a particular IMEI is submitting the specified content to one of the specified addresses, the rule should be activated and the event logged, followed by immediate rejection of the packet.

After creating the rules and their translation, it is necessary to create the packets. The content must correspond to what is specified by the owner so that the packet can be rejected for violating the rules defined by the owner. Table 5 shows an example of a packet whose contents are specified in the rule to be dropped.

Table 5. Content test rule packet.

Protocol Data	
Source IP: 10.0.0.5	Destination IP: 192.168.21.2
Text Protocol	
IMEI: 099512133857038 Content-Type: [text/plain, video/h264]	

The following image (Figure 8) represents the Snort messages that inform us that a packet has been discarded for violating the user-defined rules.



```
 [Drop] [**] [1:8403411:1] DROP - Rule Device IMEI 099512133857038{ Content Type: [v
 Corporate Privacy Violation] [Priority: 1] {TCP} 10.0.0.5:554 -> 192.168.21.2:554
 [Drop] [**] [1:8403411:1] DROP - Rule Device IMEI 099512133857038{ Content Type: [v
 Corporate Privacy Violation] [Priority: 1] {TCP} 10.0.0.5:554 -> 192.168.21.2:554
 [Drop] [**] [1:8403411:1] DROP - Rule Device IMEI 099512133857038{ Content Type: [v
 Corporate Privacy Violation] [Priority: 1] {TCP} 10.0.0.5:554 -> 192.168.21.2:554
 [Drop] [**] [1:9909079:1] DROP - Rule Device IMEI 099512133857038{ Content Type: [a
rate Privacy Violation] [Priority: 1] {TCP} 10.0.0.5:5060 -> 192.168.24.5:5060
 [Drop] [**] [1:8752486:1] DROP - Rule Device IMEI 422451118095078{ Content Type: [a
te Privacy Violation] [Priority: 1] {TCP} 10.0.0.6:554 -> 192.168.25.5:554
 [Drop] [**] [1:6754995:1] DROP - Rule Device IMEI 422451118095078{ Content Type: [i
e Privacy Violation] [Priority: 1] {TCP} 10.0.0.6:5060 -> 192.168.28.5:5060
 [Drop] [**] [1:7363383:1] DROP - Rule Device IMEI 332924181938544{ Content Type: [i
ity: 1] {TCP} 10.0.0.7:554 -> 192.168.29.5:554
```

Figure 8. Snort output showing discarded packets.

Analysing the messages displayed, Snort informs us that, for example, the device with the IMEI 422451118095078 has committed two user-defined policy violations. The first was that the content type of the packet sent by the device—viewed in Wireshark for the packet sent from source IP address `10.0.0.5` to destination IP address `192.168.25.5`—matches the content type defined by the user to be blocked `[audio/ogg, video/h265]`. The second was that the content type of the packet sent by the device—viewed in Wireshark for the packet sent from source IP address `10.0.0.5` to destination address IP `192.168.28.5`—matches the content type that the user has defined to be blocked `[image/jpeg, text/csv]`.

In the same location mentioned in the previous subsection, we can also see the packet files containing the captured packets that have violated the user-defined rules, with the information about the content they were carrying and the capture timestamp. In addition, the log files containing the event logs of the rules triggered by the test packets are also located in the same folder.

In this test, we have shown that the detection and blocking process succeeded. The system was able to detect that the virtual IoT device content present in the test packets violated the user-defined rules, resulting in a compromise of the user-defined privacy and security rules. It, therefore, dropped the packets that violated the user-defined rules, stopping them from reaching their destination.

The results of this test prove that the traffic control rules defined by the user enable them to control how their device communicates, ensuring that the user can guarantee the privacy and security of their personal information when the intentions of manufacturers and the actions of their devices are not to be trusted.

4.3. System Performance

This section describes the performance tests performed for 10,000 packets in the selected traffic generator generated for different packet throughputs. For this test, we used 10 devices for which 19 intent and 15 user-defined rules were created. The metrics chosen for analysing the system performance were:

- **Packets Received**—represents the number of packets that were captured for inspection.
- **Packets Analysed**—represents the number of packets parsed from packets received.
- **Dropped Packets**—represents the number of packets not found and therefore not analysed by the analysis component.
- **Packets Whitelisted/Blacklisted**—represents the number of packets whose content and destination address violated the manufacturer's specifications or IoT device owner rules.
- **Analysis Runtime**—represents the time taken to analyse and classify packets received by the traffic capture component.
- **Dump Runtime**—represents the time taken to generate and send packets to the network.
- **Delay**—represents the additional expense required to analyse the generated packets.

These performance tests aim to test how Snort performs in the worst-case scenarios, to measure if high throughput significantly impacts the time taken to analyse the packet's content and detect that the content violates manufacturer intents or user-defined rules.

4.3.1. Intent Scenario Results

In Figure 9, we can view the data collected from the tests performed for 10,000 packets. The longest Snort analysis runtime recorded for this test was 16 min and 45 s for a throughput of 100 packets/s, and the shortest was 1 min and 36 s for a throughput of 9000 packets/s. The average Snort analysis runtime was approximately 4 min and 23 s. The longest packet dump runtime recorded in this test was 16 min and 39 s for a throughput of 100 packets/s, and the shortest was 10 s for a throughput of 9000 packets/s. The average packet dump runtime was approximately 3 min and 17 s. The longest delay recorded in this test was 3 min and 15 s for a throughput of 9000 packets/s; the shortest was 6 s for 100 packets/s. The average delay was approximately 1 min and 37 s.

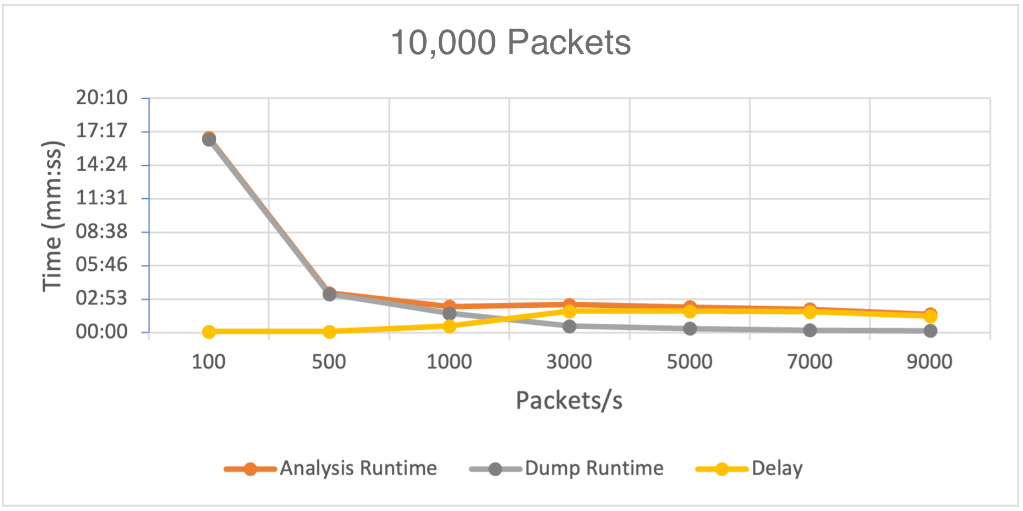


Figure 9. The 10,000 Packets runtime data.

Table 6 shows the remaining data collected for other variables. All the packets captured by Snort during the execution of this test were analysed, meaning no packets were skipped or dropped during that time. Of all the packets captured, a percentage between 2% and 12% corresponded to packets whose content violated the manufacturer's intent on the device manifest. In contrast, the remaining percentage corresponded to miscellaneous traffic captured by Snort.

Table 6. The 10,000 packet variable data.

Packets/s Throughput	Packets Captured	Packets Analysed	Packets Dropped	Packets Blacklisted
100	113,019	100%	0%	88.48%
500	102,240	100%	0%	97.62%
1000	101,391	100%	0%	98.62%
3000	105,846	100%	0%	94.25%
5000	100,941	100%	0%	98.71%
7000	102,817	100%	0%	94.81%
9000	101,165	100%	0%	96.37%

With these results, we proved that the system implemented detected and handled multiple packets at various throughputs without ever showing service breaks during the analysis for high packet loads, maintaining a cadence of approximately 13 s per packet without missing or skipping packets.

According to the results, the analysis of the captured packets took the longest time in 10,000 packets per second tests when the rate of packet throughput was the lowest at 100 packets per second because Snort could analyse only 100 packets at a time. However, with increased throughput, we saw that time decreased progressively to 22 s for the 10,000-packet tests. This gives a packet analysis average after the consistency of approximately 19 s in the 10,000-packet test (Table 7).

Table 7. Intent Scenario Results Summary.

	10,000 Packets/s
Analysis Runtime High/Low	16 min 45 s / 1 min 36 s
Dump Runtime High/Low	16 min 39 s/10 s
Delay Runtime High/Low	1 min 48 s/6 s
Average Analysis Runtime	4min 23 s
Average Dump Runtime	3min 17 s
Average Delay Runtime	1 min 06 s

4.3.2. Rules Scenario Results

Figure 10 shows the data collected from the tests performed for 10,000 packets. The longest Snort analysis runtime recorded in this test was 19 min and 3 s for a throughput of 100 packets per second, and the shortest was 2 min and 50 s for a throughput of 7000 packets per second. This test's average Snort analysis runtime was approximately 5 min and 53 s. The longest packet dump runtime recorded was 16 min and 40 s for a throughput of 100 packets/s, and the shortest was 10 s for a throughput of 9000 packets/s. This test's average packet dump runtime was approximately 3 min and 17 s. The longest delay we recorded was 2 min and 23 s for a throughput of 100 packets/s; the shortest was 1 min and 57 s for a throughput of 500 packets/s. The average delay in this test was approximately 2 min and 36 s.

Table 8 shows the remaining data collected for other variables. All the packets captured by Snort during the execution of this test were analysed, meaning no packets were skipped or dropped during that time. Of all the packets captured, 40% and 97% corresponded to packets whose content violated the user-defined rules, while the remaining percentage corresponded to the miscellaneous traffic captured by Snort.

In the results, we proved that the system implemented, detected, and dropped the packets accordingly, without ever showing service breaks during the analysis for high packet loads, maintaining a cadence of approximately 30 s per packet without missing or skipping packets (Table 9).

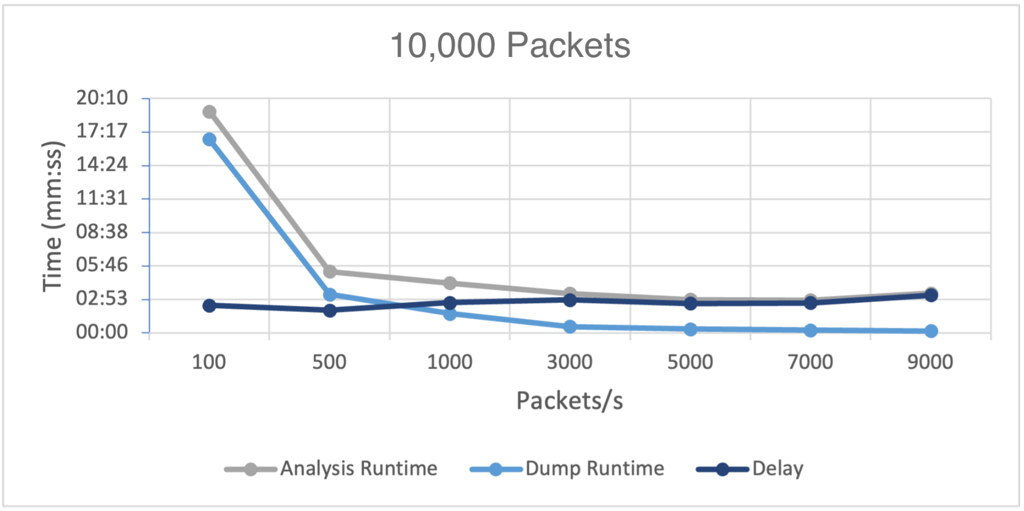


Figure 10. The 10,000 packets of runtime data.

Table 8. The 10,000 packet variable data.

Packets/s Throughput	Packets Captured	Packets Analysed	Packets Dropped	Packets Blacklisted
100	247,000	100%	0%	40.44%
500	121,334	100%	0%	82.40%
1000	106,842	100%	0%	93.39%
3000	102,444	100%	0%	97.19%
5000	101,960	100%	0%	97.72%
7000	102,255	100%	0%	97.33%
9000	104,410	100%	0%	95.75%

Table 9. Rule Test Scenario Results Summary.

	10,000 Packets/s
Analysis Runtime High/Low	19 min 03 s/2 min 50 s
Dump Runtime High/Low	16 min 40 s/10 s
Delay Runtime High/Low	3 min 15 s/1 min 57 s
Average Analysis Runtime	5 min 53 s
Average Dump Runtime	3 min 17 s
Average Delay Runtime	2 min 36 s

5. Conclusions

In this work, we presented and tested a system capable of, on the one hand, bringing transparency to IoT systems, where manufacturers declare what their devices will do (communicate) and these declarations are verifiable, and, on the other, giving IoT users control over their data privacy and security, by controlling the communication of their networked IoT devices. We believe that this contributes to establishing trust between device manufacturers and their users/consumers and contribute to a more secure and privacy-aware IoT ecosystem.

The system developed comprises three main components: rule definition, incident detection, and action according to the incident. The rule definition phase consists of defining the intents and rules the system will comply with when filtering traffic to analyse if violations of those rules are happening. The detection phase consists of analysing the contents of the packets of the network traffic being mirrored to Snort by checking for the content specified in the translated Snort rules. The action phase consists of the system's reaction when the content and destination of a packet comply with its rules, or if the intent is being violated, an alarm is triggered, according to whether the violation relates to the content or to the destination address. If a rule is being violated, the system will drop that packet, preventing it from reaching its destination.

With this mechanism, the user/consumer knows whether their devices comply with the communication intents defined by the manufacturer. The system will show them alerts if such a violation occurs, as demonstrated in the test results. With their defined traffic control rules, the user now possesses control over communication intents by monitoring exactly the amount and type of information the device is sending back to the manufacturer or some other source and by blocking information that might compromise their privacy, thus ensuring that their IoT device is secure, as demonstrated in the test results.

A known limitation of the described work is that the approach followed is only capable of handling non-encrypted network traffic. Supporting a mechanism that may act as an encryption proxy will also contribute to analysing this type of traffic. Another known limitation of the implemented system is related to the lack of knowledge the end-users will have to be able to properly define their user rules—it will be necessary to conduct further research and develop an easier way for end-users to interact with the system, to express their preferences without the technological burden.

The authors also recognise that the system could be largely improved in performance. The system was tested in a virtualised environment, and performance could change significantly when using real physical devices. Furthermore, it would be interesting to consider machine learning as a way to allow the system to improve the controlling rules and contribute to an even better IoT private and secure environment.

Author Contributions: Conceptualization, J.L., J.C. and C.S.; methodology, J.L. and C.S.; software, J.L.; validation, J.L., J.C. and C.S.; formal analysis, J.L.; investigation, J.L.; writing—original draft preparation, C.S.; writing—review and editing, C.S. and J.C.; supervision, C.S. and J.C.; project administration, C.S. and J.C. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Data used in this work was generated by Ostinato traffic generator tool. This tool generated simulated traffic that was processed and analysed in our work.

Acknowledgments: We would like to thank all the support provided by SCNL Truphone, S.A. Also, we would like to thank the research environment provided by the Information Sciences, Technologies, and Architecture Research Center (ISTAR), supported by Fundação para a Ciência e a Tecnologia (FCT), Portugal, under projects UIDB/04466/2020 and UIDP/04466/2020.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Bhosale, D.A.; Mane, V.M. Comparative study and analysis of network intrusion detection tools. In Proceedings of the 2015 International Conference on Applied and Theoretical Computing and Communication Technology (iCATccT), Davangere, India, 29–31 October 2015; pp. 312–315.
2. Nest Home Security Devices Have a Hidden Microphone. Google Calls It an "Error". Section: Security. Available online: https://www.popularmechanics.com/technology/security/a26448907/google-nest-hidden-microphone/ (accessed on 3 April 2023).
3. Calado, J.P.d.C. Open Source IDS/IPS in a Production Environment: Comparing, Assessing and Implementing. Master's Thesis, Universidade de Lisboa, Lisbon, Portugal, 2018.
4. Chakraborty, S. When Smart Gadgets Spy on You: Your Home Life Is Less Private Than You Think. Available online: https://economictimes.indiatimes.com/tech/internet/when-smart-gadgets-spy-on-you-your-home-life-is-less-private-than-you-think/articleshow/60984623.cms?from=mdr (accessed on 3 April 2023).
5. Coble, S. Xiaomi Security Camera Shows User Wrong Video Feed. Available online: https://www.infosecurity-magazine.com/news/xiaomi-camera-shows-wrong-video/ (accessed on 3 April 2023).
6. Google Nest Guard Has Microphone That Wasn't Disclosed. Available online: https://www.mercurynews.com/2019/02/20/google-nest-guard-has-a-microphone-but-it-didnt-say-that-on-the-box/ (accessed on 3 April 2023).
7. Statt, N. Amazon's Alexa Isn't just AI—Thousands of Humans Are Listening. Available online: https://www.theverge.com/2019/4/10/18305378/amazon-alexa-ai-voice-assistant-annotation-listen-private-recordings (accessed on 3 April 2023).

8. Amazon Alexa Heard and Sent Private Chat. Available online: https://www.bbc.com/news/technology-44248122 (accessed on 3 April 2023).
9. 10 Fascinating Things We Learned When We Asked The World 'How Connected Are You?' | The Mozilla Blog. Available online: https://blog.mozilla.org/en/mozilla/10-fascinating-things-we-learned-when-we-asked-the-world-how-connected-are-you/ (accessed on 3 April 2023).
10. What the Internet of Things Means for Consumer Privacy. Available online: https://impact.economist.com/perspectives/technology-innovation/what-internet-things-means-consumer-privacy-0/white-paper/what-internet-things-means-consumer-privacy (accessed on 3 April 2023).
11. Atlam, H.F.; Wills, G.B. IoT Security, Privacy, Safety and Ethics. In *Digital Twin Technologies and Smart Cities*; Internet of Things; Farsi, M., Daneshkhah, A., Hosseinian-Far, A., Jahankhani, H., Eds.; Springer International Publishing: Cham, Switzerland, 2020; pp. 123–149. [CrossRef]
12. Assiri, A.; Almagwashi, H. IoT Security and Privacy Issues. In Proceedings of the 2018 1st International Conference on Computer Applications & Information Security (ICCAIS), Riyadh, Saudi Arabia, 4–6 April 2018; pp. 1–5. [CrossRef]
13. Chanal, P.M.; Kakkasageri, M.S. Security and Privacy in IoT: A Survey. *Wirel. Pers. Commun.* **2020**, *115*, 1667–1693. [CrossRef]
14. Tawalbeh, L.; Muheidat, F.; Tawalbeh, M.; Quwaider, M. IoT Privacy and Security: Challenges and Solutions. *Appl. Sci.* **2020**, *10*, 4102. [CrossRef]
15. Khowaja, S.A.; Khuwaja, P.; Dev, K.; Lee, I.H.; Khan, W.U.; Wang, W.; Qureshi, N.M.F.; Magarini, M. A secure data sharing scheme in Community Segmented Vehicular Social Networks for 6G. *IEEE Trans. Ind. Inform.* **2022**, *19*, 890–899. [CrossRef]
16. Zhang, L.; Li, Y.; Jin, T.; Wang, W.; Jin, Z.; Zhao, C.; Cai, Z.; Chen, H. SPCBIG-EC: A robust serial hybrid model for smart contract vulnerability detection. *Sensors* **2022**, *22*, 4621. [CrossRef] [PubMed]
17. Zhang, L.; Wang, J.; Wang, W.; Jin, Z.; Su, Y.; Chen, H. Smart contract vulnerability detection combined with multi-objective detection. *Comput. Netw.* **2022**, *217*, 109289. [CrossRef]
18. Ren, Y.; Zhu, F.; Sharma, P.K.; Wang, T.; Wang, J.; Alfarraj, O.; Tolba, A. Data query mechanism based on hash computing power of blockchain in internet of things. *Sensors* **2019**, *20*, 207. [CrossRef]
19. Abdul-Ghani, H.A.; Konstantas, D. A Comprehensive Study of Security and Privacy Guidelines, Threats, and Countermeasures: An IoT Perspective. *J. Sens. Actuator Netw.* **2019**, *8*, 22. [CrossRef]
20. Shen, Y.; Vervier, P.A. IoT Security and Privacy Labels. In *Proceedings of the Privacy Technologies and Policy*; Lecture Notes in Computer Science; Naldi, M., Italiano, G.F., Rannenberg, K., Medina, M., Bourka, A., Eds.; Springer: Cham, Switzerland, 2019; pp. 136–147. [CrossRef]
21. Mohanta, B.K.; Jena, D.; Ramasubbareddy, S.; Daneshmand, M.; Gandomi, A.H. Addressing Security and Privacy Issues of IoT Using Blockchain Technology. *IEEE Internet Things J.* **2021**, *8*, 881–888. [CrossRef]
22. Bettayeb, M.; Nasir, Q.; Talib, M.A. Firmware Update Attacks and Security for IoT Devices: Survey. In Proceedings of the ArabWIC 6th Annual International Conference Research Track, Rabat, Morocco, 7–9 March 2019; pp. 1–6. [CrossRef]
23. Sun, P.; Garcia, L.; Salles-Loustau, G.; Zonouz, S. Hybrid Firmware Analysis for Known Mobile and IoT Security Vulnerabilities. In Proceedings of the 2020 50th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), Valencia, Spain, 29 June–2 July 2020; pp. 373–384. [CrossRef]
24. Srivastava, P.; Peng, H.; Li, J.; Okhravi, H.; Shrobe, H.; Payer, M. FirmFuzz: Automated IoT Firmware Introspection and Analysis. In Proceedings of the 2nd International ACM Workshop on Security and Privacy for the Internet-of-Things, London, UK, 15 November 2019; pp. 15–21. [CrossRef]
25. Government Efforts toward Promoting IoT Security Awareness for end Users: A Study of Existing Initiatives—ProQuest. Available online: https://www.proquest.com/openview/e8826900b7596e3720cbc3c9c8786ec0/1?pq-origsite=gscholar&cbl=396497 (accessed on 3 April 2023).
26. Bugeja, J.; Vogel, B.; Jacobsson, A.; Varshney, R. IoTSM: An End-to-End Security Model for IoT Ecosystems. In Proceedings of the 2019 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops), Kyoto, Japan, 1–15 March 2019; pp. 267–272. [CrossRef]
27. Jaigirdar, F.T.; Rudolph, C.; Bain, C. Prov-IoT: A Security-Aware IoT Provenance Model. In Proceedings of the 2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), Guangzhou, China, 29 December 2020–1 January 2021; pp. 1360–1367. [CrossRef]
28. Irshad, M. A Systematic Review of Information Security Frameworks in the Internet of Things (IoT). In Proceedings of the 2016 IEEE 18th International Conference on High Performance Computing and Communications; IEEE 14th International Conference on Smart City; IEEE 2nd International Conference on Data Science and Systems (HPCC/SmartCity/DSS), Sydney, NSW, Australia, 12–14 December 2016; pp. 1270–1275. [CrossRef]
29. Amar, Y.; Haddadi, H.; Mortier, R.; Brown, A.; Colley, J.; Crabtree, A. An Analysis of Home IoT Network Traffic and Behaviour. *arXiv* **2018**, arXiv:1803.0536. https://doi.org/10.48550/arXiv.1803.05368.
30. Mazhar, M.H.; Shafiq, Z. Characterizing Smart Home IoT Traffic in the Wild. In Proceedings of the 2020 IEEE/ACM Fifth International Conference on Internet-of-Things Design and Implementation (IoTDI), Sydney, NSW, Australia, 21–24 April 2020; pp. 203–215. [CrossRef]
31. Ostinato Traffic Generator for Network Engineers. Available online: https://ostinato.org/ (accessed on 3 April 2023).

32. Snort—Network Intrusion Detection & Prevention System. Available online: https://www.snort.org/ (accessed on 3 April 2023).
33. Hafeez, S.; Eng, B. Deep Packet Inspection Using Snort. Master's Thesis, University of Victoria, Victoria, Canada, 2016.