ITUPublications

International Telecommunication Union
Standardization Sector

ITU-T Technical Report

(07/2024)

YSTR-IADIoT

Intelligent anomaly detection system for Internet of things

Technical Report ITU-T YSTR-IADIoT

Intelligent anomaly detection system for Internet of things

Summary

Technical Report ITU-T YSTR-IADIoT presents an intelligent anomaly detection system for the Internet of Things (IoT) which is based on the use of new technologies to detect anomalous behaviour in IoT-based systems. It is a hybrid system that responds to proven filtering security rules for known attacks. Then, through a machine learning module, new anomalous traffic can be detected, and the rules reconfigured according to the analysis and discovery made. The main objective of this Technical Report is to demonstrate the feasibility of implementing certain controls and security aspects as close as possible to IoT devices.

Keywords

Anomaly detection, IoT, machine learning, security.

Note

This is an informative ITU-T publication. Mandatory provisions, such as those found in ITU-T Recommendations, are outside the scope of this publication. This publication should only be referenced bibliographically in ITU-T Recommendations.

Table of Contents

Technical Report ITU-T YSTR-IADIoT

Intelligent anomaly detection system for Internet of things

1 Scope

This Technical Report presents an intelligent anomaly detection system for the Internet of things (IoT).

This Technical Report covers the following:

– The importance of anomaly detection;
– Advantages of using machine learning (ML) in anomaly detection;
– Challenges in the IoT application scenario;
– Requirements of the intelligent anomaly detection system for IoT;
– Anomaly detection techniques;
– Architecture of the intelligent anomaly detection system for IoT;
– Intelligence anomaly detection module related to the intelligent anomaly detection system for IoT;
– Type of anomalies to be detected.
– Test scenario.
– Machine learning tests related to the intelligent anomaly detection system for IoT.

2 References

[ITU-T X.1361] Recommendation ITU-T X.1361 (2018), *Security framework for the Internet of things based on the gateway model.*

[ITU-T Y.4000] Recommendation ITU-T Y.4000/Y.2060 (2012), *Overview of the Internet of things*.

[ITU-T Y.4101] Recommendation ITU-T Y.4101/Y.2067 (2017), *Common requirements and capabilities of a gateway for Internet of things applications.*

[ITU-T Y.4460] Recommendation ITU-T Y.4460 (2019), *Architectural reference models of devices for Internet of things applications.*

3 Definitions

3.1 Terms defined elsewhere

This Technical Report uses the following terms defined elsewhere:

3.1.1 device [ITU-T Y.4000]: With regard to the Internet of things, this is a piece of equipment with the mandatory capabilities of communication and the optional capabilities of sensing, actuation, data capture, data storage and data processing.

3.1.2 Internet of things (IoT) [ITU-T Y.4000]: A global infrastructure for the information society, enabling advanced services by interconnecting (physical and virtual) things based on existing and evolving interoperable information and communication technologies.

NOTE 1 – Through the exploitation of identification, data capture, processing and communication capabilities, the IoT makes full use of things to offer services to all kinds of applications, whilst ensuring that security and privacy requirements are fulfilled.

NOTE 2 – From a broader perspective, the IoT can be perceived as a vision with technological and societal implications.

3.1.3 gateway [ITU-T Y.4101]: A unit in the Internet of things which interconnects the devices with the communication networks. It performs the necessary translation between the protocols used in the communication networks and those used by devices.

3.1.4 thing [ITU-T Y.4000]: With regard to the Internet of things, this is an object of the physical world (physical things) or the information world (virtual things), which is capable of being identified and integrated into communication networks.

3.1.5 attack [b-ISO 13491-1]: Attempt by an adversary on the device to obtain or modify sensitive information or a service they are not authorized to obtain or modify.

3.1.6 threat [b-ISO/IEC 27000]: Potential cause of an unwanted incident, which may result in harm to a system or organization.

3.1.7 authentication [b-NIST SP 800-53]: Verifying the identity of a user, process, or device, often as a prerequisite to allowing access to resources in an information system.

3.1.8 machine learning [b-ITU-T Y.3172]: Processes that enable computational systems to understand data and gain knowledge from it without necessarily being explicitly programmed.

NOTE 1 – This definition is adapted from [b-ETSI GR ENI 004].

NOTE 2 – Supervised ML and unsupervised ML are two examples of machine learning types.

3.1.9 software-defined networking [b-ITU-T Y.3300]: A set of techniques that enables to directly program, orchestrate, control and manage network resources, which facilitates the design, delivery and operation of network services in a dynamic and scalable manner.

3.2 Terms defined in this Technical Report

None.

4 Abbreviations and acronyms

This Technical Report uses the following abbreviations and acronyms:

ADS Anomaly Detection System

AI Artificial Intelligence

CFS Correlation-based Feature Selection

CSV Comma-Separated Value

DoS Denial of Service

IADM Intelligent Anomaly Detection Module

IADS Intelligent Anomaly Detection System

IAT Packet Inter-arrival Time

IoT Internet of Things

LPWAN Low Power Wide Area Network

LXC Linux Containers

MitM Man in the Middle

ML Machine Learning

NFC Near Field Communication

PCAP Packet Capture

RFID Radio Frequency Identification

SDN Software-Defined Networking

VM Virtual Machine

Wi-Fi Wireless Fidelity

5 Introduction

The Internet of Things is a fast expanding network of smart heterogeneous objects. It refers to the physical or virtual devices that are capable of communicating with other devices. Unlike the wireless sensor networks, IoT is connected to the Internet that exposes it to global intrusion in addition to wireless attacks within an IoT network. It is protected by cryptographic and network security techniques, but they are vulnerable to internal and external attacks. The IoT devices are resource-constrained in terms of limited storage, battery, limited transmission range, and processing. A system is needed to identify the abnormal behaviour and trigger an alarm in abnormal scenario to take appropriate preventive measures. Hence, an anomaly detection system (ADS) plays an important role to prevent such cyberattacks in the IoT.

The ADS is a software and/or hardware entity to automate the detection of abnormal activities that attempt to compromise the integrity, confidentiality, or availability of a system with the following functionality [b-Debar]:

- Monitor the behaviour of the IoT systems and IoT devices;
- Automatically recognize unauthorized and malicious activities in an IoT system;
- Trigger the alarms on recognizing the malicious activity.

Anomaly detection can be done using the techniques of machine learning (ML). ML is a subset of artificial intelligence (AI) algorithms that improves the performance of the systems based on the use of different learning types such as supervised learning, unsupervised learning and reinforcement learning.

6 The importance of anomaly detection

Attack and anomaly detection in the IoT infrastructure is a rising concern in the domain of IoT. With the increased use of IoT infrastructure in every domain, threats and attacks in these infrastructures are also growing commensurately.

The first line of defence for IoT devices is based on security techniques such as cryptographic authentication and secure construction of network topology. However, some attackers still launch malicious attacks on IoT devices through data analysis. Anomaly detection is considered as the second line of defence, which plays an important role in ensuring the security of IoT devices [b-Pajouh]. Anomaly detection detects and identifies attacks and anomalies such as denial of service, data type probing, malicious control, malicious operation, spying and wrong setup which can cause an IoT system failure.

7 Advantages of using machine learning in anomaly detection

ML offers help in many aspects [b-Sciforce]:

- **Automation**: AI-driven anomaly detection algorithms can automatically analyse datasets, dynamically fine-tune the parameters of normal behaviour and identify similarities in the patterns.
- **Self-learning**: AI-driven algorithms constitute the core of self-learning systems that are able to learn from data patterns and deliver predictions or answers as required.

- **Real-time analysis**: ML solutions can interpret data activity in real time. The moment a pattern isn't recognized by the system; it sends a signal. The signals are compared with the learned patterns and the system acts according to the detected situation in a timely fashion.
- **Scrupulousness**: Anomaly detection platforms provide end-to-end gap-free monitoring to go through minutiae of data and identify the smallest anomalies that would go unnoticed by humans.

8 Challenges in the IoT application scenario

Anomaly detection system challenges in the IoT application scenario, include:

- **Dynamic threat landscape**: New IoT devices are released on a daily basis. A significant fraction of them has security vulnerabilities. Exploits targeting vulnerable devices are also being developed by adversaries at a similarly high pace. This makes the threats against IoT devices highly dynamic and ever-increasing.
- **Resource limitations**: IoT devices have limited capabilities (constrained IoT devices) including available memory, computing resources and energy often making on-device detection infeasible.
- **IoT device heterogeneity and false alarms**: The behaviours of different IoT devices are very heterogeneous, leading to frequent false alarms. Anomaly detection systems must minimize these false alarms to be useful in practice.
- **Scarcity of communications**: IoT devices generate only little traffic, often triggered by infrequent user interactions.
- **Diversity of communication technologies**: Connectivity between end devices and gateways can involve various wired or wireless technologies such as Bluetooth, Wi-Fi, and LPWAN, creating additional challenges in designing a security system.

9 Requirements of an intelligent anomaly detection system

The ADS for IoT should meet the following set of requirements:

- The deployment of ADS in IoT should not introduce new vulnerabilities to the system.
- The ADS should be self-managed to detect hardware and software changes automatically.
- It should be self-configured and adaptive to configuration changes.
- The ADS should be capable of recovering from system failures and restoring previous conditions.
- The ADS should detect anomalies with minimal usage of system resources.
- It should run continuously and remain transparent to both the system and its users.
- The ADS should monitor itself to detect if it has been compromised and protect itself from unauthorized access or attacks.
- The ADS should detect anomalies with low processing and communication overhead.
- The mobility of IoT devices should not affect detection accuracy.
- It should be scalable to efficiently process a large number of IoT devices.
- The ADS should detect a variety of anomalies with low false positive and false negative rates.
- The ADS should provide automated responses to suspicious activities without human intervention.
- The ADS should not only detect anomalies but also localize their sources.

10 Anomaly detection techniques

The anomaly detection system can be the key to solving intrusions because alterations to normal behaviour indicate the presence of intentional or unintentional induced attacks on the IoT based system.

To detect anomalies there are two techniques:

- **Rule-based**: These systems define specific rules that describe an anomaly, with thresholds and limits based on security experts' experience. They are ideal for detecting known anomalies.
- **ML-based**: Anomalies are detected using ML techniques, generally used to identify unknown security attacks.

The technique based on ML are a good alternative to detect anomalies, but they present the following problems:

- **Training data**: To use ML efficiently, a vast number of correctly labelled input samples are needed. However, even when an algorithm has been fed with a large amount of data, there is still no guarantee that it can correctly identify all new items. Therefore, human expertise and verification are constantly required. Without this process, a single incorrect input can cause a snowball effect and undermine the solution to the point of failure. The same problem arises if the algorithm only uses its own output data as inputs for subsequent learning. Errors are reinforced and multiplied, as the same incorrect results re-enter the solution in a loop and create more false positives (categorizing clean samples as malicious) and false negatives (marking malicious samples as benign).
- **Difficulty in finding training data**: A large part of the training data is synthetic and not realistic enough. Most of the datasets used are publicly available and developed by research institutes. ML-based solutions trained on those datasets alone may be outperformed by adversaries who studied the same data.
- **Imbalanced data sets**: A dataset is considered to be unbalanced when the proportions between the majority set and the minority sets are large.
- **Concept drift**: Cybersecurity is a rapidly evolving field. Models and knowledge are likely to become obsolete quickly. Volumes change due to technological advancement, new protocols and domains appear, and malicious activity also has trends and fads. To avoid continually starting from scratch, it is necessary to build ways to cope with these changes.
- **A semantic gap between initial work and practical real-world deployments**: Unlike artificial intelligence, an ML model works by recognizing deviations from what it was trained on and why. When there are issues with the training data sets, ML models generally give many false positives when implemented in the real-world environment. It is also difficult to interpret the overall results to obtain actionable intelligence, which is also known as the semantic gap.

  Even when absolutely nothing bad happens, the performance of ML models in the real world usually degrades due to a significantly larger volume of data, a much wider range of fluctuations, real-world limitations, etc. In short, although good at classification tasks, ML models are not able to recognize the context and logic underlying real-world situations. As a result, it is difficult to actually evaluate a model. Although there is no formal consensus standard on how to evaluate ML models and what metrics should be used, justifications cannot be based on accuracy rate alone.

  It is important to keep in mind that there are problems in assessing true ML performance and that model accuracy alone will not be sufficient. Let's not forget that attackers are also equipped with ML powers and can build systems to predict the behaviours of the defending models.

- **ML alone is not enough**: The ML technology is not a silver bullet for solving all cybersecurity related issues. Only a finely tuned mix of multiple layers of security that includes ML and human expertise can deliver the highest detection rates combined with low false positives.
- **Intelligent and adaptive adversary**: Another severe limitation of ML applications is the intelligent adversary. In cybersecurity, the attackers do not hesitate to bend or break rules, often changing the entire playing field without a warning. The ever-changing nature of the digital environment makes it impossible to create a protective solution able to detect and block all future threats, and ML does not change this postulate.
- **Enormous variability in input data**: In the IoT environment there are a large number of types of devices and communication technologies, as well as types of anomalies and security attacks, thereby creating a challenge when generating training datasets**.**

To mitigate these problems, both anomaly detection approaches can be combined, thus obtaining a hybrid system [b-Aslam].

11 Architecture of the intelligent anomaly detection system for IoT

The anomaly detection system will be located in the device layer of the reference architecture proposed by the ITU [ITU-T Y.4000].

As shown in Figure 1, the architecture consists of the following components:

- **IoT device**: The devices collect various kinds of information and provide it to the information and communication networks for further processing. Some devices also execute operations based on information received from the information and communication networks.

  The IoT devices for this architecture may have the following characteristics:

  – **Low connectivity capabilities**: These devices may not directly communicate with the applications or cloud services through the Internet. For this reason, they may rely on other architectural elements such as gateways for protocol translation and Internet connectivity [ITU-T Y.4460].

  – **No processing or low processing capabilities**:

    - **No processing capabilities**: Some IoT devices have no processing capabilities to execute any behaviour. They are low-cost devices with no microcontrollers. For instance, ID tags using identification technologies such as radio frequency identification (RFID) or near field communication (NFC), applied to a disposable package. It is not reasonable to embed microcontrollers that are more expensive than the package itself [ITU-T Y.4460].
    - **Low processing capabilities**: These devices have processing capabilities just sufficient for reading/writing data from/to sensors/actuators and sending/receiving those data as messages to IoT applications. They do not have sufficient processing capabilities to make decisions or run complex algorithms. For this reason, they can rely on other architectural elements like cloud services, for data storage or data processing. Usually, they are low-cost devices with very limited microcontrollers to make the product economically viable. For instance, smart lights or door sensors. It is not reasonable to embed powerful microcontrollers that are more expensive than the product itself [ITU-T Y.4460].

  – **Digital or analogue sensor**: An IoT device usually connects to digital or analogue sensors to monitor digital or analogue values (a value with precision and a physical unit), for examples, temperature sensor, pressure sensor, etc.

- **IoT gateway**: As shown in Figure 1, different devices can connect to communication networks through one or multiple IoT gateways. The connectivity between IoT devices and

IoT gateway(s) can be based on different kinds of wired or wireless technologies, such as a controller area network (CAN) bus, Bluetooth, Wi-Fi, LPWAN, etc.

The IoT gateway has the general characteristic of supporting security functions. They provide security mechanisms to support the security requirements of applications. Common security mechanisms used in an IoT gateway include those for device authentication, data encryption, confidentiality and security policy management [ITU-T Y.4101]. For the security of the IoT environment, an IoT gateway must control access to IoT devices and to itself and must protect data integrity, confidentiality, and availability from itself and end devices.

This intelligent anomaly detection system is to ensure the integrity, confidentiality, and availability of data and to detect anomalies. An open flow switch may be placed in the IoT gateway to monitor the traffic coming from end devices.

- **SDN Controller**: The SDN controller manages and configures the distributed network resources and provides an abstracted view of the network resources to the SDN applications via another standardized interface (i.e., application-control interface) and the relevant information and data models [b-ITU-T Y.3300]. This intelligent anomaly detection system may connect to SDN controllers to get optimized communication capabilities for its associated entities (such as IoT devices and IoT gateways).
- **Intelligent anomaly detection module (IADM)**: This module collects the data from the IoT gateways and/or IoT devices to check the packet for anomalies.

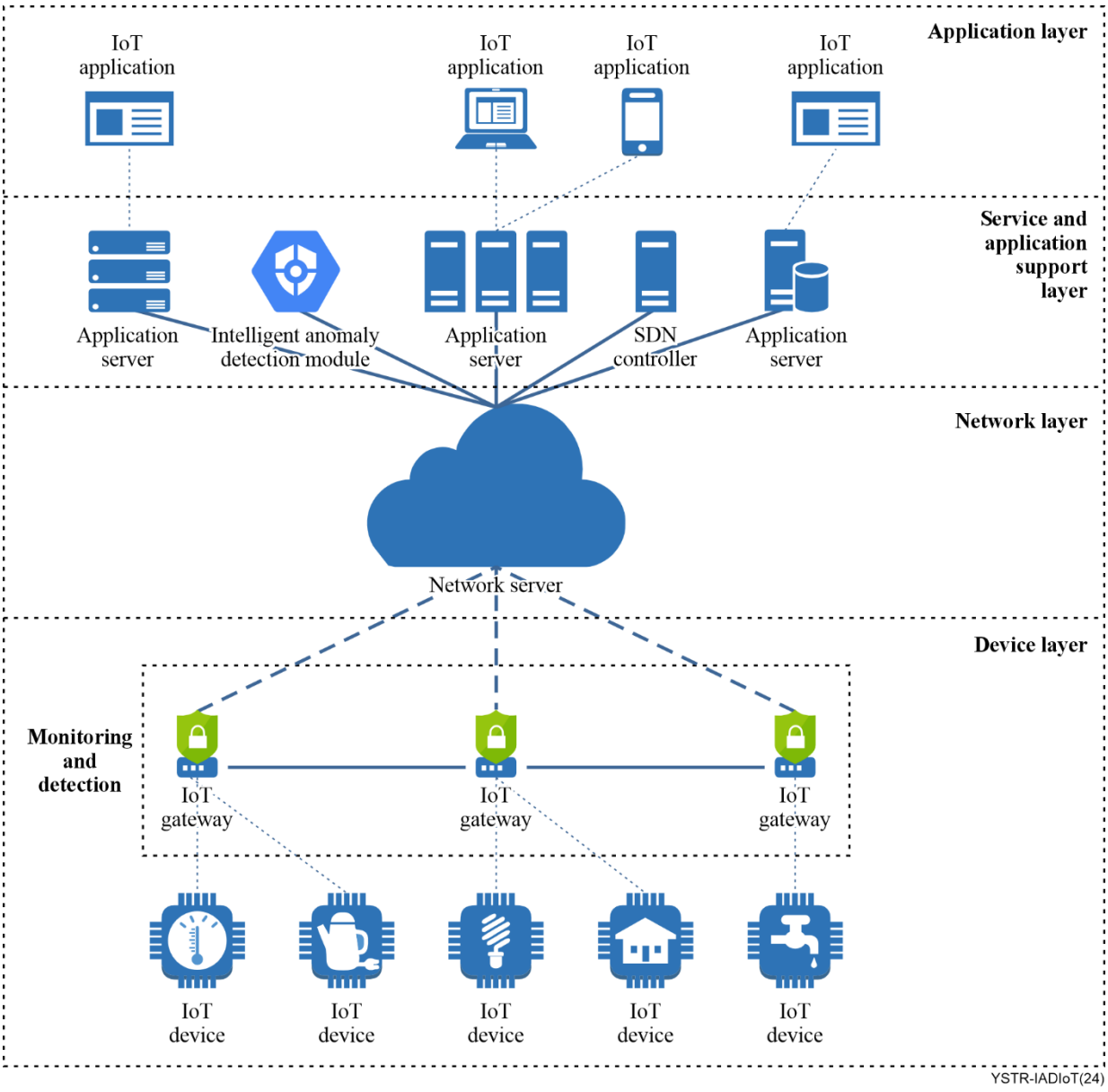

Figure 1 – Architecture of intelligent anomaly detection system for IoT

NOTE – The Intelligent Anomaly Detection Module (IADM), Software Defined Network (SDN) controller, switch and traffic monitor are separate entities that work together (interconnected). This implementation can be deployed in physical, virtualized form, or a combination of both and this places emphasis on the different levels of abstraction present in the architecture. As proposed, the Traffic Monitor resides in the lowest layer of the architecture, while the machine learning and SDN controller modules, given their nature, reside in the application and support layer.

12 Intelligent anomaly detection module (IADM)

The intelligent anomaly detection system includes the IADM, which uses rule-based and ML-based methods to process real-time data as depicted in Figure 2.

In the first part, the rule-based ADS may capture the network traffic coming from the end devices through an open flow switch in the IoT gateway, and based on some predefined rules, may classify the incoming network traffic as normal or abnormal (attack) to produce labelled training data.

After classification, the labelled training data may be stored in a database allowing that data to be used in the future for training the ML-based anomaly detector.

In the second part, the ML models are trained using the labelled training data set. After training the models, they can be used to validate the classification results performed by the rule-based anomaly detector. The final prediction of the system is obtained by comparing the results of both types of detectors. If one of the systems declares a package as a failure; it will be labelled as an attack. The validation process with ML reduces the false positives/negative rates of the rule-based system.

The overall architecture of the anomaly detection module is shown in Figure 2.

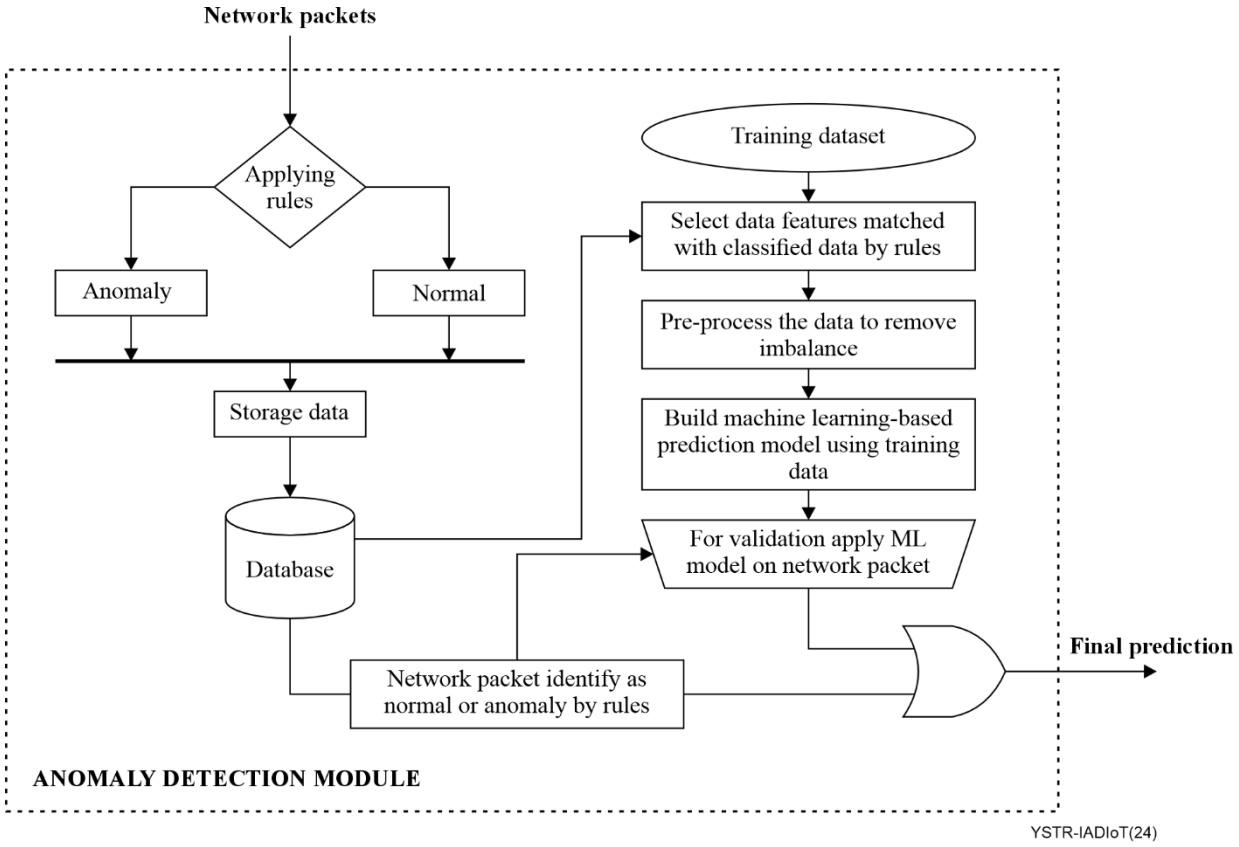

Figure 2 – Architecture of intelligent anomaly detection module

13 Type of anomalies to be detected

An anomaly is unusual behaviour in the system that may indicate a security attack or false alarm. The most relevant types of data that can be used to detect anomalies are the basic information about the packets: packet number, gateway identity (ID), timestamps, etc. Additional fields can be computed from this basic information, such as the one-way delay (OWD), which indicates the time taken by the packets from the sensors to the system. This is an important field for the detection of some attacks, such as denial of service, since these attacks tend to slow down packet reception.

The anomaly types that this system can detect are:

- Distributed Denial of Service (DDoS): The attack is a malicious attempt to disrupt the normal traffic of a targeted server, service or network by overwhelming the target or its surrounding infrastructure with a flood of requests.
- Denial of Service (DoS): The attack may be caused by having too many unwanted traffic from a single source. The attacker may send too many ambiguous packets to flood out the target and make its services unavailable to other services [b-Hasan].
- OS and Service Scan: The attacks, operating system and service scanning, are similar in nature and can be grouped into the probing attack category [b-Churcher]. This type of attack can be done passively, where the attacker collects packets from the network, or actively,

where the attacker sends traffic and logs responses. Since passive probing does not generate traffic, active probing is necessary to test the traffic. OS scans involve the attacker being able to discover the operating system used by the victim machine. This information can help an attacker identify the type of device, e.g., server, computer or IoT device. Service scans, known as port scans, involve the attacker probing a network to identify open ports on the network [b-Churcher].

- Data exfiltration attacks: This attack is the theft or unauthorized removal or movement of any data from an end device. A data exfiltration attack involves attackers gaining access to a private network and stealing data stored on the network. This type of attack can result in the theft of data such as credit card information and personal data [b-Churcher].

14 Test scenario

Monitoring is an essential concept in network management as it helps network operators to determine the network behaviour and status of its components. Anomaly detection also depends on monitoring for decision-making.

To design this system, three concepts are important, including namespace, container and connectivity network:

– **Namespace**: A namespace in computer science is an abstract container or environment created to hold a logical grouping of unique identifiers or symbols (i.e., names).

– **Containers**: Containers are packages of software that contain all of the necessary elements to run in any environment. In this way, containers virtualize the operating system and run anywhere.

  Containers make it easy to share CPU, memory, storage and network resources at the operating system level and offer a logical packaging mechanism in which applications can be abstracted from the environment in which they actually run.

– **Network connectivity with container runtime**: When installing container software on an operating system, it creates a bridge to a network, and this network connects by default all containers, unless otherwise indicated. It is necessary to configure the network understanding that it is necessary to work with different levels of abstraction. In the case of containers (they have their own namespaces), it will be necessary to connect them to the virtual machine where they are running, and this, in turn, to the host, to the Internet, and/or to other virtual machines, if they exist. Figure 3 shows in the shaded box, the concrete boundaries of the implementation of the access module to the proposed IoT architecture. It is then understood that it is implemented in the operating system that supports the IoT gateway and the traffic switching and monitoring modules.

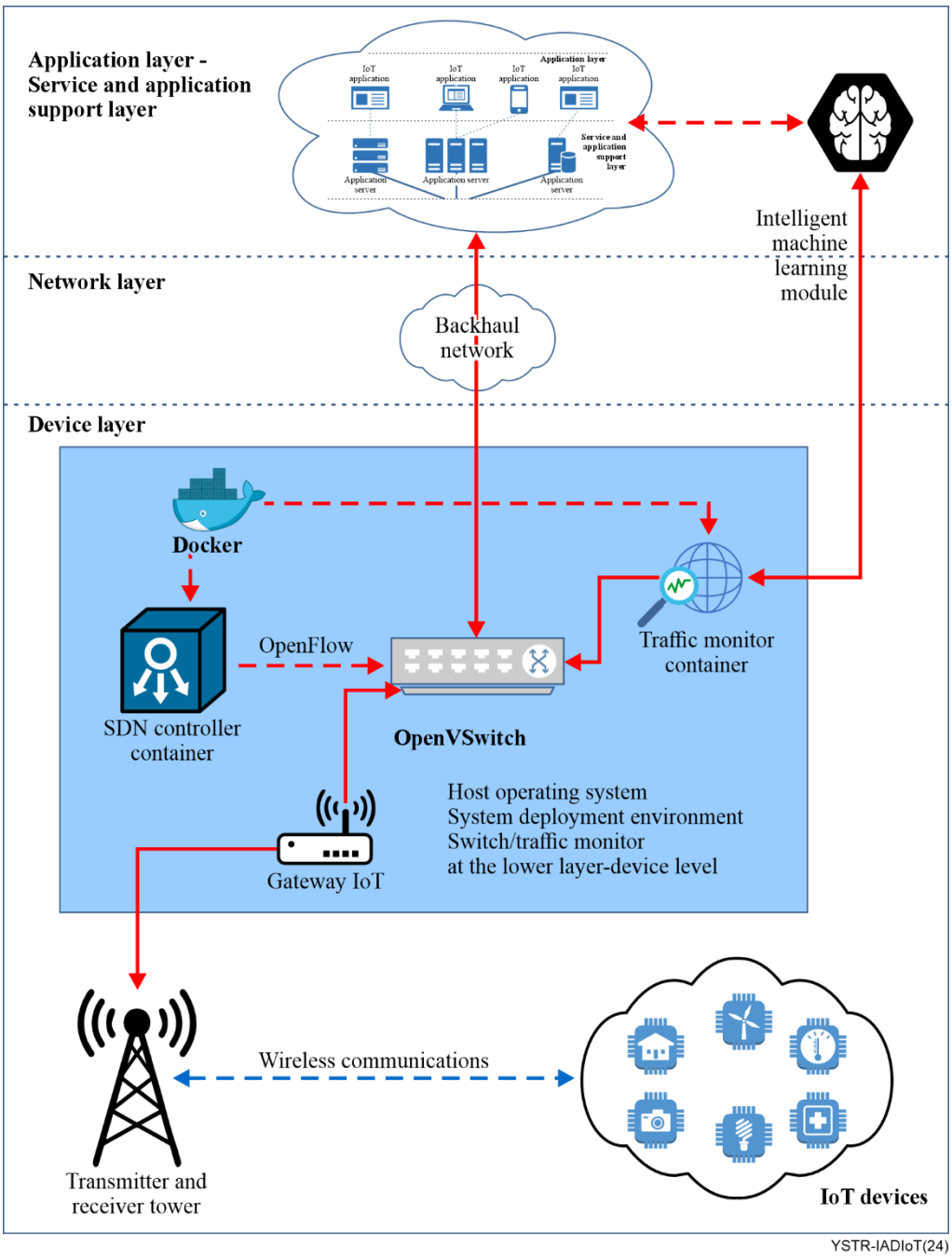

Figure 3 – Test scenario scheme

To test the components of the traffic reception and monitoring system, the following software is installed on the IoT gateway or IoT device:

– Docker.

– Open vSwitch.

This software can be used to manage traffic with an OpenFlow switch (Open vSwitch). The deployment of the test scenario is shown in Figure 4.

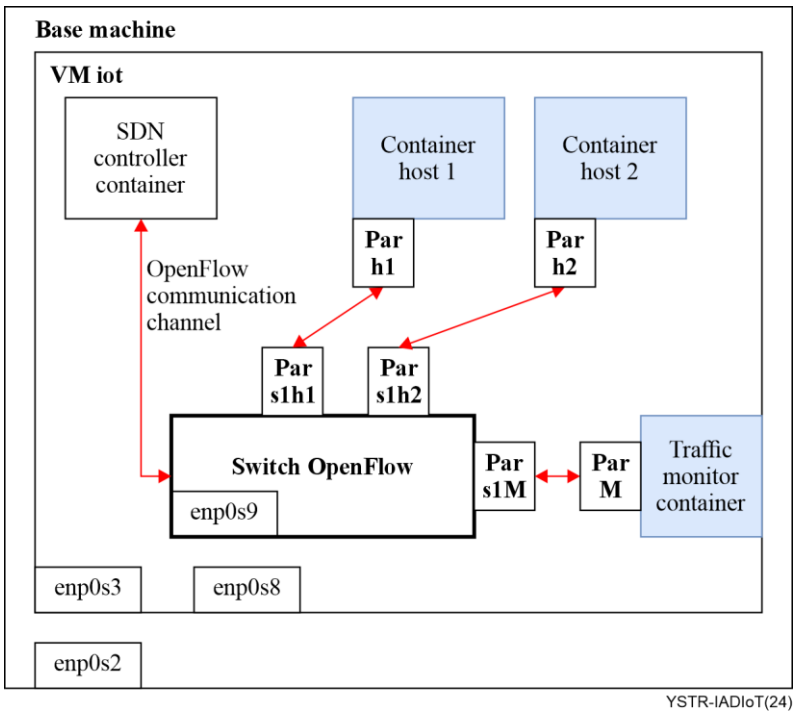

Figure 4 – Test scenario deployment

In this test scenario deployment, the components of the test scenario comprise an SDN controller, an OpenFlow switch that performs the function of managing traffic, two hosts called Host1 and Host2, which are deployed for different connectivity tests, and a container called traffic monitor container, which includes the appropriate software to capture traffic.

15 Machine learning tests related to the intelligent anomaly detection system for IoT

The IADM is responsible for running ML models, to detect anomalies in the network packets.

Methodology

- **Dataset**

  An essential step in ML is obtaining a good training dataset to train the model. The training data must be labelled to indicate whether it contains anomalies. For this first version of the ML model, the Bot-IoT 2018 dataset is created in a realistic network environment in the Cyber Range Lab of the UNSW Canberra Cyber Centre. The data includes DDoS, DoS, OS and service scan, keylogging, and data exfiltration attacks. The captured packet capture (PCAP) files are 69.3 GB, with over 72,000,000 records. Due to the size of the dataset, it was decided to use 5 per cent of the total records.

Figure 5 depicts the automatic labelling process.

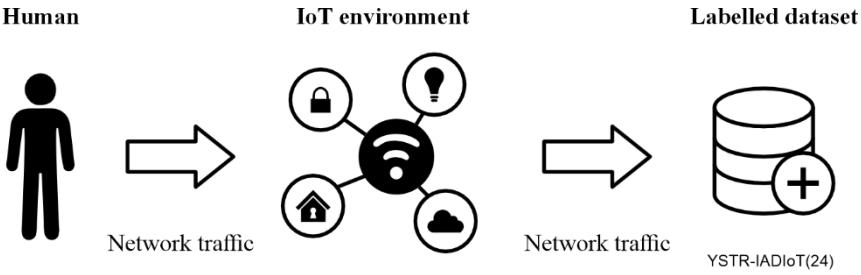

Figure 5 – Automatic labelling

Injection timing is one of the most common methods used to get labelling network traffic data sets, which consists of generating different network traces in different time windows, and then labelling all traces.

The injection timing strategy requires a controlled network environment where the user has precise information about the different applications generating traffic on the network. Figure 6 provides a simplified overview of the injection timing strategy. At ts the network has just become operational for the first time. At time tsm the user injects into the network particular malware traffic (DDOS, Botnet, port scanning, etc.)

Since the network has just become operational all the background traffic from the time window ts to tsm is labelled as normal. Beyond tsm all network traffic is labelled as malicious. When the user stops injecting the malicious behaviour at tem, all the traffic becomes labelled as normal again until the network is permanently shut down at time te.

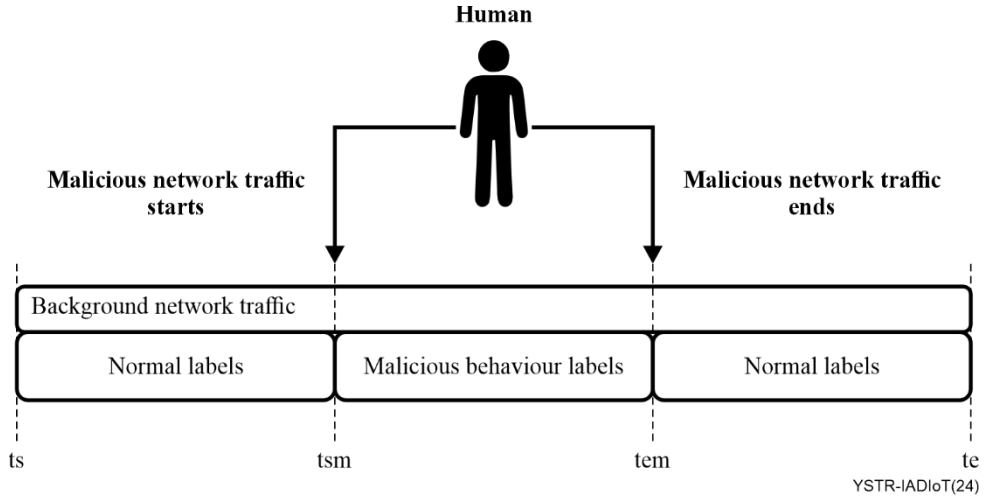

Figure 6 – Example of injection timing

Since the injection timing strategy is applied to control environments, it is possible to create labelled datasets containing observed traffic with a large number of attacks.

This feature provides the required level of authenticity for validating experimental results on the generated labelled dataset. However, since labels are obtained by merely contrasting the execution time window of each generated network trace, a strict time control mechanism is necessary to obtain accurate labels.

The first reason for separating the traffic into time windows is that it significantly improves the analysis and subsequent labelling of the traffic. When separated into different time instances, it is much easier to control each behaviour of the network.

The second reason for using time windows is that botnets tend to have temporary locational behaviour, which means that most actions remain unchanged for several minutes. Therefore, it is easier to label these network traces with higher accuracy than other types of attacks [b-Guerra].

- **Data preprocessing**

First, the duplicate and irrelevant data are removed, the syntax errors are corrected, the unwanted outliers are filtered out, and the missing data are dealt.

Data cleansing is essential because, regardless of how sophisticated the ML algorithm is, good results cannot be obtained from bad data.

Once the data is in the correct format, it is analysed by using different statistical measures (such as mean and median) and visualisation techniques (such as univariate analysis, bivariate analysis, etc.).

Exploratory data analysis is considered a fundamental and crucial step in solving any ML use case, as it helps us to identify trends or patterns in the data.

Then the variables or columns that have no impact on the result are dropped. From the 47 columns of the dataset, the following 11 columns described in Table 1 are chosen.

Table 1 – Selected features of the dataset

Feature	Data Type	Description
pkSeqID	Ordinal	Row Identifier
seq	Numerical	Argus sequence number
mean	Numerical	Average duration of aggregated records
stddev	Numerical	Standard deviation of aggregated records
min	Numerical	Minimum duration of aggregated records
max	Numerical	Maximum duration of aggregated records
srate	Numerical	Source to destination packets per second
drate	Numerical	Destination to source packers per second
N_IN_Conn_P_DstIP	Numerical	Total number of packets per destination IP
N_IN_Conn_P_SrcIP	Numerical	Total number of packets per source IP
attack	Numerical	Define if it is an attack

Finally, a dataset with 3,688,522 records is generated and saved as a CSV file for use with different ML algorithms.

- **Performance evaluation and analysis**

The results of the algorithms are discussed in this section. It includes the confusion matrix and the initial settings to build the model.

a. Evaluation of metrics

Evaluating a model involves checking its prediction accuracy, that determines how well the model is behaving on the train and test datasets.

To evaluate the results of the model, certain metrics are used which are described below.

- **Accuracy**: Used in classification problems to inform the percentage of correct predictions made by a model. The accuracy score in ML is an evaluation metric that measures the number of correct predictions made by a model in relation to the total number of predictions made. It can be calculated by dividing the number of correct predictions by the total number of predictions.

  Accuracy = (Number of correct predictions)/(Total number of predictions)

- **Precision**: Described as a measure of calculating the correctly identified positives in a model and is given by:

  Precision = (True positives)/(True positives + False positives)

- **Recall**: A measure of the actual number of positives that are correctly identified and is given by:

  Recall = (True positives)/(True positives + False negatives)

- **F1-score**: A measure of the actual number of positives that are correctly identified and is given by:

  *F1score = (Precision * Recall)/(Precision + Recall)*

- **Support score**: A measuring metric of the python library sci-kit-learn, which indicates the number of occurrences of each label where it is true.

- **Confusion matrix**: An N × N matrix used for evaluating the performance of a classification model, where N is the total number of target classes. The matrix compares the actual target values with those predicted by the ML model. This gives a holistic view of how well the classification model is performing and what kinds of errors it is making.

b. Test results for ML methods

- **Random forest**: A supervised learning algorithm that works on the concept of bagging. In bagging, a group of models is trained on different subsets of the dataset, and the final output is generated by collating the outputs of all the different models. In the case of random forest, the base model is a decision tree.

  As shown in Table 2, the overall accuracy for the random forest is 100 per cent.

Table 2 – Random forest results

Metrics	Precision	Recall	F1-score	Support
Accuracy	–	–	1.00	1467409
Macro avg	1.00	1.00	1.00	1467409
Weighted avg	1.00	1.00	1.00	1467409

Random forest is a good model, as shown in the confusion matrix (Table 3), which has high true positive and true negative values with low rates of false positives and false negatives.

Table 3 – Random forest confusion matrix

	Predicted 0	**Predicted 1**
Actual 0	True Negative 192 0.013084%	False Positive 1 0.000068%
Actual 1	False Negative 0 0.000000%	True Positive 1467216 99.986848%

To evaluate the model, cross-validation was applied with 10 folds, multiplying the average by 100, resulting in 99.9090 for F1-macro.

- **LightGBM (light gradient-boosting machine)**: A gradient boosting framework based on decision trees, designed to increase the efficiency of the model and reduce the memory usage.

  As shown in Table 4, the overall accuracy for the LightGBM is 100 per cent.

Table 4 – LightGBM results

Metrics	Precision	Recall	F1-score	Support
Accuracy	–	–	1.00	1467409
Macro avg	1.00	1.00	1.00	1467409
Weighted avg	1.00	1.00	1.00	1467409

As opposed to random forest, this framework has a higher number of false positives and false negatives. See Table 5.

Table 5 – LightGBM confusion matrix

	Predicted 0	**Predicted 1**
Actual 0	True Negative 75 0.005%	False Positive 118 0.008%
Actual 1	False Negative 107 0.007%	True Positive 1467109 99.980%

To evaluate the model, cross-validation was applied with 10 folds, multiplying the average by 100, resulting in a 49.3077 F1-score.

- **KNN**: The k-nearest neighbours algorithm, also known as KNN or k-NN, is a non-parametric, supervised learning classifier, which uses proximity to make classifications or predictions about the grouping of an individual data point. While it can be used for either regression or classification problems, it is typically used as a classification algorithm, working off the assumption that similar points can be found near one another.

  The results of the KNN algorithm with K = 5 is presented in Tables 6 and 7.

Table 6 – KNN results

Metrics	Precision	Recall	F1-score	Support
Accuracy	–	–	1.00	1467409
Macro avg	0.96	0.90	0.93	1467409
Weighted avg	1.00	1.00	1.00	1467409

Table 7 – KNN confusion matrix

	Predicted 0	**Predicted 1**
Actual 0	True Negative 155 0.010563%	False Positive 14 0.00954%
Actual 1	False Negative 38 0.002590%	True Positive 1467202 99.985894%

To evaluate the model, cross-validation was applied with 10 folds, multiplying the average by 100, resulting in a 92.7243 F1-score.

- **AdaBoost**: An ensemble learning method (also known as "meta-learning") which was initially created to increase the efficiency of binary classifiers. AdaBoost uses an iterative approach to learn from the mistakes of weak classifiers and turn them into strong ones.

  The results of the AdaBoost are presented in Tables 8 and 9.

Table 8 – AdaBoost results

Metrics	Precision	Recall	F1-score	Support
Accuracy	–	–	1.00	1467409
Macro avg	0.96	0.90	0.93	1467409
Weighted avg	1.00	1.00	1.00	1467409

Table 9 – AdaBoost confusion matrix

	Predicted 0	**Predicted 1**
Actual 0	True Negative 192 0.01%	False Positive 1 0.00%
Actual 1	False Negative 0 0.00%	True Positive 1467216 99.99%

To evaluate the model, cross-validation was applied with 10 folds, multiplying the average by 100, resulting in a 99.7365917 F1-score.

c. Result comparison

When deciding on the anomaly detection system it is important to pay attention to metrics: false positive and false negative rates.

The false negatives rate shows how many anomalies were, on average, missed by the detector.

For this reason, in this study the F1-score is considered for evaluating the models because, in an anomaly detection system, false negatives must be avoided without losing accuracy.

As shown in Table 10 random forest and AdaBoost have the highest F1-score.

All algorithms generated good results, but random forest and AdaBoost have obtained 0 false negatives, while KNN and LGBM have obtained more than 38 false negatives.

Table 10 – Result comparison

Method	F1-score
Random Forest	99.9090
LightGBM	49.3077
KNN	92.7243
AdaBoost	99.7365917

Additionally, to build the ML module, a system of rules can be used that is continuously updated by security specialists. This system will be responsible for analysing the data flowing through the network to compare it with a set of predefined rules. If it detects something that matches one of these rules, it alerts the system and the administrator.

The rules are generated manually by security experts taking into account the attacks to be detected and the features that are available in the network packets.

The main steps to write a rule are as follows:

1. **Define the protocol**: First, define the protocol to match. This can be ICMP, TCP, UDP, or other protocols.
2. **Determine the direction**: After defining the protocol, determine the direction of the traffic you want to match. For instance, if it is to match traffic from the server to the client or from the client to the server.
3. **Determine the source and destination IP addresses and ports**: To determine the source and destination IP addresses and ports for the traffic to match.
4. **Define the rule options**: Lastly, describe the rule options that will trigger the alert when traffic matches the rule.

Table 11 – Rule format example

Rule header							**Rule options**	
Action	Protocol	Source Address	Source Port	Flow	Destination Address	Destination Port	Message	Rules

Rule example

alert tcp any any -> any 80(msg:"Possible DoS Attack"; stime > 1526876545; stime <= 1528099368)

Alternatively, these rules can be generated automatically, using the J48 (C4.5) ML algorithm. It is one of the decision tree generation algorithms developed by Ross Quinlan. It is the descendant of the ID3 algorithm and can be used as a statistical classifier. It is a tree-like structure which consists of root nodes and leaf nodes derived from it. The leaf nodes may represent classes or class attributes. It is constructed by using information gain and entropy criteria and then, it generates the rules for the target outcomes. It can handle both continuous and discrete features. By using the pruning techniques, the overfitting problem can be solved. It can also be used on training data having incomplete data and different weighted features [b-Soe].

The steps to generate the rules:

1. **Generate a train dataset**: the Bot-IoT dataset can be used for this purpose.
2. **Select the features**: the correlation-based feature selection (CFS) is a simple algorithm evaluating the corresponding relations between the outputs and correlated input features. This algorithm claims that feature selection for classification in ML can be achieved on the basis of the correlation between features. A feature is redundant if one or more of the other features are highly correlated with it. The CFS algorithm is based on the fact that a good feature subset is one that contains features highly correlated with the class and uncorrelated with each other. Irrelevant or redundant features should be ignored also because they may raise the computation process and even worsen the detection accuracy.
3. **Train the model**: the decision tree algorithm, J48, with two-thirds of the dataset as training data, can be used to train the model.

4. **Generated rules**: finally, the attack signature rules are reached.

16 Conclusion and standardization suggestion

This development of the intelligent anomaly detection system for IoT has demonstrated the possibility of bringing security controls closer to IoT devices using available and robust technologies. As for the ML module, the possibility of working with data sets corresponding to traffic captures of this type of networks and the possibility of taking actions once the anomalous traffic has been detected has been proven, thereby demonstrating the feasibility of its development.

In conclusion, the implementation of an ML model for anomaly detection in IoT environments is crucial to maintain network security. Leveraging the Bot-IoT 2018 dataset, we established a foundational model capable of identifying various types of cyber-attacks. The initial model uses a fraction of the dataset to manage computational resources efficiently, while providing a foundation for future improvements. Going forward, the integration of real-time data from the actual operating environment and the adoption of automatic labelling techniques will improve the accuracy and adaptability of the model. Continuous refinement and testing are essential to ensure the model's robustness in detecting anomalies and protecting IoT networks from emerging threats.

Suggestion of future work

To address the evolving landscape of IoT cybersecurity infrastructure, the following future work is proposed:

- **Develop new Recommendations**: Initiate studies within the ITU-T study groups to develop new Recommendations focused on promoting interoperability, efficiency and security in global IoT communication networks.
- **Conduct assessments**: Identify current gaps, challenges and opportunities within IoT security systems.
- **Evaluate technologies**: Assess existing technologies, practices and standards for their applicability and effectiveness.
- **Engage stakeholders**: Collaborate with industry experts, academia and governmental bodies to gather insights and foster collaboration in the standard development process.
- **Develop framework**: Create frameworks to address identified needs and aligns with international best practices.
- **Establish guidelines**: Set guidelines for implementation, testing and compliance assessment to ensure effective adoption of the proposed standard.

Bibliography

[b-ITU-T Y.3172] Recommendation ITU-T Y.3172 (2019), *Architectural framework for machine learning in future networks including IMT-2020*.

[b-ITU-T Y.3300] Recommendation ITU-T Y.3300 (2014), *Framework of software-defined networking*.

[b-ITU-T Y.4105] Recommendation ITU-T Y.4105/Y.2221 (2010), *Requirements for support of ubiquitous sensor network (USN) applications and services in the NGN environment*.

[b-ETSI GR ENI 004] ETSI GR Experiential Networked Intelligence (ENI) V1.1.1 (2018); *Terminology for Main Concepts in ENI.*

[b-ISO 13491-1] ISO 13491-1:2016, *Financial services – Secure cryptographic devices (retail) – Part 1: Concepts, requirements and evaluation methods*.

[b-ISO/IEC 27000] ISO/CEI 27000:2018, *Information technology – Security techniques – Information security management systems – Overview and vocabulary*.

[b-NIST SP 800-53] NIST Special Publication 800-53 (2013), *Security and Privacy Controls for Federal Information Systems and Organizations*.

[b-Aslam] Aslam, U., Batool, E., Ahsan, S. N., & Sultan, A. (2017), *Hybrid network intrusion detection system using machine learning classification and rule based learning system*.

[b-Churcher] Churcher, A., Ullah, R., Ahmad, J., Ur Rehman, S., Masood, F., Gogate, M., Alqahtani, F., Nour, B., & Buchanan, W. J. (2021), *An Experimental Analysis of Attack Classification Using Machine Learning in IoT Networks. Sensors* (Basel, Switzerland), 21(2), 446. https://doi.org/10.3390/s21020446

[b-Debar] Debar H, Dacier M, Wespi A. (1999), *Towards a taxonomy of intrusion-detection systems.* Computer Networks; 31(8):805–822.

[b-Guerra] Guerra, J., Catania, C.A., & Veas, E. (2021), *Datasets are not Enough: Challenges in Labeling Network Traffic.* Comput. Secur, 120, 102810.

[b-Hasan] Hasan, M., Islam, M. M., Zarif, M. I. I., & Hashem, M. M. A. (2019), *Attack and anomaly detection in IoT sensors in IoT sites using machine learning approaches*. Internet of Things, 7, 100059.

[b-Pajouh] Pajouh, H. H., Javidan, R., Khayami, R., Ali, D., & Choo, K. K. R. (2016), *A two-layer dimension reduction and two-tier classification model for anomaly-based intrusion detection in IoT backbone networks*. IEEE Transactions on Emerging Topics in Computing.

[b-Sciforce] Sciforce (2019), *Anomaly Detection—Another Challenge for Artificial Intelligence*. Available at: https://medium.com/sciforce/anomaly-detection-another-challenge-for-artificial-intelligence-c69d414b14db

[b-Soe] Soe, Y.N., Feng, Y., Santosa, P.I., Hartanto, R., & Sakurai, K. (2019), *Rule Generation for Signature Based Detection Systems of Cyber Attacks in IoT Environments.*

[Bot-IoT 2018 dataset] M. Sarhan, S. Layeghy, N. Moustafa, y M. Portmann (2021), *NetFlow Datasets for Machine Learning-based Network Intrusion Detection Systems*, arXiv:2011.09144 [cs], vol. 371, pp. 117-135, doi: 10.1007/978-3-030-72802-1_9. Access: https://www.unsw.adfa.edu.au/unsw-canberra-cyber/cybersecurity/ADFA[1]NB15-Datasets/bot_iot.php