Zero-touch security management for mMTC network slices: DDoS attack detection and mitigation

Redouane Niboucha, Sabra Ben Saad, Adlen Ksentini, Yacine Challal

Zero-touch security management for mMTC network slices: DDoS attack detection and mitigation

Redouane Niboucha, Sabra Ben Saad, Adlen Ksentini, *Senior member, IEEE* and Yacine Challal

***Abstract*—massive Machine Type Communications (mMTC) network slices in 5G aim to connect a massive number of MTC devices, opening the door for a widened attack surface. Network slices are well isolated, resulting in a low impact on other running slices when attackers control IoT devices belonging to a mMTC network slice (i.e., in-slice attack). However, the impact of the in-slice attacks on the shared infrastructure components with other slices, such as the 5G Core Network (CN), can be harmful, considering the massive number that can be part of mMTC slice. In this paper, we propose a zero-touch security management solution that uses Machine Learning (ML) to detect and mitigate in-slice attacks on 5G CN components, focusing on Distributed Denial of Service (DDoS) attacks. To this aim, we propose: (1) a novel closed-control loop that assists the 5G CN in detecting and mitigating attacks; (2) a ML algorithm that predicts the upper bound of expected MTC devices Attach Requests during a time interval (or an event); (3) a detection algorithm that analyzes an event and uses the ML output to compute a probability that a specific device has participated to an attack; (4) a mitigation algorithm that disconnects and blocks MTC devices suspected to be part of an attack; (5) a Proof of concept implementation on top of a 5G facility.**

***Index Terms*—Network Slicing, Machine Learning, Denial of Service, 5G, Zero-touch Service Management (ZSM), Security**

I. Introduction

In recent years, computing has evolved to support the growing market of Internet of Things (IoT), which involves pieces of hardware, such as sensors, smart door locks, smartwatches, etc., having the ability to connect to the network, transmit and receive data. A recent forecast estimates the number of IoT devices that should be connected to the network will exceed 25.4 billion in 2030 [4].

In terms of connectivity to the network, several technologies are candidates to connect these devices. We can mention LORA, SigFox, IEEE 802.14.5, and 5G. The latter is seen as an excellent alternative since it allows continuous connectivity of the IoT devices, supporting even mobile devices (i.e., embedded in cars or drones). 5G is the latest generation of mobile networks promising the support of several new emerging services, including those relying on IoT. 5G uses the concept of network slicing to efficiently manage the common infrastructure and provision network resources tailored to the network service needs. A network slice is a virtual network built using Virtual Network Functions (VNF) that run as Virtual Machines (VM) or containers on top of cloud infrastructure (central or edge) as well as Physical Network Functions (PNF) such as 5G base stations. A network slice is composed of different sub-slices that span over different technological domains, radio, core network, cloud, and transport. Sub-slices are stitched together to build the end-to-end network slice tailored to the application. Sub-slices can be specific to the owners (ex., VNF running applications or network services) or shared among slices (ex., 5G Core Network - CN, gNB).

5G envisions the running of network services relying on IoT as a network slices of massive Machine Type Communications (mMTC) type. As its name indicates, this type of network slice is intended and optimized to connect a massive number of IoT devices to a service or an application. It is worth noting that 5G also specifies two other types of network slice, namely ultra Reliable Low Latency Communications (uRLLC) and enhanced Mobile BroadBand (eMBB), which are envisioned for low-latency-demanding applications (e.g., autonomous driving, Industry 4.0) and high bandwidth-demanding applications (e.g., Virtual and Augmented Reality), respectively. Last but not least, 5G supports multi-tenancy, as network slices run in parallel, and a high degree of isolation is ensured among them; a network slice component (control or data plane) cannot access other network slices' components. In this work, we focus on mMTC network slices since they are used to deploy IoT applications and services using MTC devices.

The growing market of IoT devices has increased cyber security threats and widened attack surfaces; hence securing modern networks is a challenging task. Indeed, IoT devices are often weak when it comes to security, considering: (1) their usual low-cost and the lack of the necessary built-in security controls to defend against threats; (2) their constrained environment and limited computational capacity. Besides, they are a high-value asset to attackers. Indeed, finding a vulnerability of a type of device allows attackers to infect more devices of that type and, hence, conduct attacks from the infected equipment. For example, in 2016, a Malware called Mirai infected between 800,000 and 2.5 million devices through default passwords and used them to conduct Denial of Service (DDoS) attacks against some public web applications.

5G was designed with built-in security controls to address many of the threats found in 4G/3G/2G networks, such as enforcing mutual authentication to prevent fake base-station

R. Niboucha is with Ecole Superieur Informatique (ESI) school, Algeria.
S. Ben Saad and A. Ksentini are with communication systems department of EURECOM, France.
Y. Challal is with University of Doha for Science and Technology, Qatar.

attacks, encrypting subscriber identities, and enforcing more secure cipher suites. However, more functionality always comes with new risks and attack vectors when opening the network to IoT devices, particularly. Some of these risks can affect the performances of 5G network functions (mainly the service availability). By embracing network slicing, 5G networks can mitigate inter-slice attacks as isolation is one of the key features. Indeed, the isolation guaranteed by network slicing offers performance guarantees to the applications and ensures isolation, such that attacks (e.g., leakage, breach, Distributed DDoS - DDoS) remain contained and do not propagate to the network. However, an in-slice attack may correspond to a subset of the UEs attached to a specific network slice issuing malicious traffic towards the infrastructure services. A typical example is compromised MTC devices (i.e., IoT devices) generating a massive number of network attachment requests. These attacks need to be quickly identified and mitigated to avoid system failure. It is worth noting that 3GPP clearly stated in [1] that 5G networks should be protected from DDoS attacks, where mechanisms detecting and mitigating this type of attack are needed.

In this paper, we propose Zero-touch Security Management (ZSM) solution that addresses the challenges of in-slice DDoS attack detection and mitigation, considering the case of mMTC network slices. Generally, this type of attack targets the 5G CN elements shared among the network slices. The proposed ZSM solution relies on a closed-control loop composed of a triplet (Monitoring System - MS, Analytical Engine - AE, and Decision Engine - DE) that interacts with the 5G CN in order to detect attacks and automatically react by mitigating the attacks. The critical challenge addressed in this work is how to detect a DDoS attack initiated by a compromised set of MTC devices inside a network slice. Indeed, there is no available traces or dataset that reproduce abnormal traffic in 5G, unlike other types of networks where many datasets are available. Besides, it is very challenging to detect during an event if there is an attack and what are the involved devices in the attack. To overcome these limitations, we followed 3GPP traffic models [2] [3] for MTC to identify normal traffic and train a ML model to recognize the normal traffic and hence also abnormal traffic. The proposed solution waits until the end of an event to analyze the traffic (normal or abnormal) and deduces if an attack happened and which are the involved devices. Then, all the devices will be detached and banned from future access to the network. We believe our solution is pertinent to mitigating a DDoS attack as the latter loses its intensity if the number of involved devices is reduced by detaching and blocking them.

The paper's contributions are:

- A novel closed-control loop featuring AI/ML to achieve ZSM in 5G and beyond networks,
- A novel approach that detects MTC attach events over a time duration,
- A novel ML algorithm that leverages Gradient Boosting to learn and predict an upper bound interval for normal MTC traffic (following a $\beta(3,4)$ as per the recommendation of 3GPP),
- A novel DDoS detection algorithm that defines a detection rate of all MTC devices involved in an attack,
- A novel mitigation algorithm that detaches and blocks all MTC devices that have been suspected to be part of an attack,
- An implementation of the ZSM concept (i.e., closed-control loop) has been done on a 5G facility to prove the concept and evaluate the performances of the attack detection and mitigation algorithms.

The rest of this paper is organized as follows. In Section II, we present a review of related works. We provide general background about the used mechanisms to detect and mitigate DDoS attacks. Section III details the contributions related to attach events and attack detection, including the devised ML method. Section IV presents our attack mitigation algorithm. In Section IV, we introduce the proof of concept implementation and extensive performance results of the proposed detection algorithm, comparing it with a statistical-based solution. Finally, section V concludes the paper.

II. Background and related work

In this section, we provide the necessary background and related work to understand the concepts presented in the paper. First, we start by highlighting the 5G thread landscape. Second, we present how ML is used to detect different threats. Then, we end the section by focusing on DDoS attacks in 5G that rely on IoT devices.

A. 5G threat Landscape

A 5G network comprises four major technological domains: the Radio Access Network (RAN), CN, transport network, and inter-connect network [5]. From a security perspective, some threats affect all of these planes, and others affect only specific ones. [6]. Threats in 5G can be classified according to the parts (or the technological domains) of the network they are impacting [6]:

- User Equipment (UE) threats: Mobile botnets can launch DDoS attacks on multiple network layers, impacting the 5G infrastructure, web servers, and other UEs. The goal here is to disturb and shut-down services.
- RAN threats: Rogue base station threats for conducting Man in the Middle (MitM) attacks. This class of attacks can compromise user information, break privacy and track users, and cause a Denial of Service.
- CN threats: These attacks relate to elements of the CN, including Software Defined Networking (SDN) and VNF components, as well as Network Slicing. These attacks can lead to Denial of Service, eavesdropping, interception, or hijacking [7].
- Network slicing threats: attacks that can break the isolation between network slices [6].
- SDN threats: Separation of control and user (data) plane that allows a malicious user to attack the link between the control and the user planes. For instance, a DDoS attack could be performed, or control could be gained over network devices (ex., Topology Poisoning attacks). [8].

B. Machine Learning for threat detection

ML has gained a broad interest in many applications and fields of study, particularly in cybersecurity. With hardware and computing power becoming more accessible, ML methods can be used to analyze and classify bad actors from a huge set of available data. ML algorithms and approaches can be categorized into supervised and unsupervised learning [9]. Supervised learning approaches are made in the context of classification, where input matches to an output, or regression, where input is mapped to continuous output. Unsupervised learning is mostly accomplished through clustering and has been applied to exploratory analysis and dimension reduction. Both clustering and regression methods can be applied in cybersecurity for analyzing malware in near real-time, thus eliminating the weaknesses of traditional detection methods [9]. ML techniques have been used to detect and mitigate DDoS attacks on networks effectively. Successful approaches have always used a variety of features to achieve great results. They have targeted DDoS attacks that are either very popular (with public datasets available) or easy to reproduce (for which creating a dataset similar to real-world behavior is possible). Most of these approaches target common application protocols like TCP and UDP. Table I summarizes some of the existing research in this area.

TABLE I
COMPARISON OF SOME DDOS ATTACK DETECTION SOLUTIONS

Reference	The mitigated attacks	ML algorithms used
João Henrique Corrêa et al. 2019 [10]	TCP SYN flood	KNN and Decision Trees
Rohan DDoShi et al. 2018 [11]	TCP SYN flood, UDP flood, HTTP GET flood	KNN, SVM, Decision Trees, Random Forests and Neural Networks
Faisal Hussain et al. 2020 [12]	Various attacks on application protocols	RESNET
Xiaoyong Yuan et al. 2017 [13]	Various attacks on application protocols	CNN and RNN
Maryam Ghanbari et al. 2018 [14]	Various attacks on application protocols	CNN after SVM preprocessing
Roberto Doriguzzi-Corin et al. 2020 [15]	Various attacks on application protocols	CNN

However, it is difficult to make a comparison among these solutions as they use different datasets. Further, the accuracy levels of these solutions can be misleading since they consider that a packet is a sample in the dataset instead of a malicious host. Therefore, with the flooding nature of the DDoS attacks, even a naive decision of considering that all the traffic is malicious would yield an accuracy higher than 80%. Finally, the lack of publicly-available data and datasets for attack detection on 5G networks makes it hard to understand, reproduce and mitigate 5G attacks.

C. Detection of DDoS attacks in IoT and 5G networks

Many works relying on ML have designed algorithms and tools to detect and mitigate DDoS attacks. In [16], a ML algorithm is proposed to circumvent attacks through an automated solution in 5G networks. It has been integrated with the Deep Packet Inspection (DPI) function to detect unfamiliar attacks in the traffic of IoT devices. A solution named "Corero" has been introduced in [17]. It aims to help organizations to comprehend 5G attacks and specifically prevent DDoS attacks by using automatic and real-time detection algorithms. However, the presented approach needs to use a very high-cost infrastructure. In [19], the authors present an IoT security framework, including Azure, AWS IoT, and SmartThings. The AWS framework supports mutual authentication of IoT devices, which can be done using certificates, groups, roles, and AWS Cognito identities. The used authorization is policy-based, with the rules mapped to each certificate, allowing the owners of IoT devices to define their rules. All IoT traffic is encrypted to secure communications. In [20], the authors propose using security monitoring methods to detect and prevent massive IoT devices from being attacked or controlled. The proposed solution prevents IoT devices from being used maliciously to launch, for instance, DDoS attacks. The detection mechanism of the proposed solution is based on a randomization technique to defend against DDoS signaling attacks that emerged in the new 5G Radio Resource Control (RRC) three-state model. The work in [18] proposes a modified Network Intrusion Detection System (NIDS) that defends against DDoS attacks on the mobile edge of multi-tenant 5G networks. The proposed approach solves the problem by combining traditional NIDSs with a classifier based on Support Vector Machines (SVM) to obtain the needed information on the devices' traffic. Two filters are implemented: the first one is placed at the mobile edge computing servers to prevent spoofing attacks close to the source, and the second one is located in cloud servers to classify the complete network traffic.

Unfortunately, the few works mentioned in this section propose solutions that need new software or specific protocols, which are not standard and need a costly implementation. Besides, they do not consider the context of network slicing, where most of the addressed challenges are solved thanks to network slice isolation. Therefore, the main remaining security threats come from compromised devices running inside the slice. In this case, shared network components are vulnerable to in-slice attacks since the devices are under the tenant's control and have the green light to access the network. To the best of the authors' knowledge, our work is the first one that addresses the challenge of in-slice attack detection and mitigation in 5G considering mMTC network slices.

III. ZERO-TOUCH SECURITY MANAGEMENT FOR IN-SLICE ATTACK: ASSUMPTIONS & SYSTEM ARCHITECTURE

Our work aims to provide a solution featuring ZSM of in-slice attack detection and mitigation considering mMTC slices in 5G. The proposed solution relies on ML to detect abnormal traffic of MTC devices that could cause DDoS on the control plane of the 5G core network (by flooding the AMF with signaling messages). Hence, it will be possible to mitigate the attack by making efficient decisions to prevent flooding of the AMF with traffic and causing DDoS or deteriorating performance for legitimate users. This type of attack can be more effective on mMTC than other 5G services, assuming the

very high number of MTC devices supposed to support. In this work, we assume that a mMTC slice is composed of: (1) a shared sub-slice with other existing network slices, which runs the 5G CN (including the AMF) and gNB; (2) a specific sub-slice to run the application that collects data from the MTC devices.

It is well accepted that MTC devices generate two main types of traffic [21]. The first one is "Periodic", where the devices emit data periodically, which may correspond to the case of meteorological data. The second type is "Event-Driven". In this case, the devices emit data when a specific event occurs; for example, smoke (the signal of a possible fire) is detected. Detecting anomalous traffic in the first type is simple, as most of the anomaly detection algorithms can directly be applied to the problem. However, predicting when an event will consistently occur for the second type of traffic is very challenging. While there exist models for the traffic during activity periods, it is difficult to solve the problem of finding anomalies just by trying to learn the function the data distribution follows; if we consider time-series data. To detect malicious traffic of MTC devices, we will consider the traffic model proposed by 3GPP, which suggests that traffic of MTC devices' connection after detecting an event follows the $\beta(3,4)$ probability distribution [23]. Accordingly, we assume that (1) normal traffic follows the $\beta(3,4)$ probability distribution; (2) the detection events do not overlap (i.e., each event is independent of the other). We argue the latter by the fact that most of the devices run only one type of application, which monitors a single event.

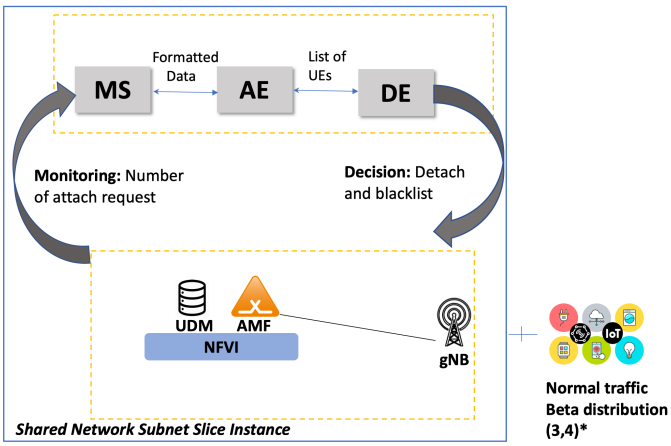

Fig. 1. The high-level vision of envisioned system architecture.

Fig. 1 illustrates the high-level vision of the system, which is composed of the closed-control loop components (MS, AE, and DE) that interact and protect the shared sub-slice components (5G CN and gNBs) against DDoS attacks. Here, we focus on protecting AMF as it is the entry point of the 5G CN and treats all the Attach Requests coming from the different gNB under its control.

The closed-control loop is composed of three entities: Monitoring System (MS), which collects information from the AMF, Analytical Engine (AE), which uses ML to predict attacks, and Decision Engine (DE), which reacts to the alert sent by AE by acting on AMF (block and blacklist UE). The control-control loop runs as software and can be co-located with the orchestrator managing the life cycle of the shared sub-slice [22]. It should be noted that AMF, via an Element Manager (EM), exposes API for an orchestrator or a manager to extract and monitor information on the AMF's functioning or to change the configuration of the latter. In the proposed framework, MS monitors the Attach Requests received by the AMF, while DE requests AMF to send Registration Reject to suspected devices. Fig. 2 highlights the interaction among the different actors involved in detecting and mitigating DDoS attacks: the mMTC network slice components (UEs and 5G CN) and the closed-control loop elements (MS, AE, and DE). It is worth noting that the closed-control loop runs in parallel to the mMTC network slice elements and only monitors Attach Requests to detect and mitigate attacks. In the considered scenario, the MTC devices (or UEs), when detecting an event or participating in an attack, first send a Attach Request to AMF. The latter must first authenticate the devices and then give them access to the network resources (register the device), mainly to the data plane, to send data to the remote application. During the authentication process, the AMF checks with the Unified Data Management (UDM) if a device is blacklisted or not. To recall, UDM is the 5G CN function, which stores subscribers' information (Subscriber Permanent Identifier - SUPI- Quality of Service -QoS- Policy, the key k, Operator key, etc.). A device is blacklisted if it has participated in an attack. Meanwhile, MS, via the EM/AMF API, monitors the Attach Requests received by AMF. MS filters the data to extract needed information, such as timestamps and SUPI. This information is communicated to AE that processes the whole event (attach period that may correspond to an attack) in order to classify if the event corresponds to an attack or not. When the event finishes, AE communicates the list of involved UEs; for each UE, a probability of being part of the attack is included. DE then mitigates the attack by requesting AMF to send a Registration Reject message to UEs having a high probability of being in the attack while adding the concerned UEs to a blacklist maintained by the UDM. The following sections will detail each component of the closed-control loop with an important focus on the AE functions, which represent the critical components of the framework.

IV. Closed-control loop: Attacks detection

A. Monitoring System (MS)

MS collects data from AMF on every Attach Request received from the MTC devices. This data is accessible through the API exposed by EM of AMF. For each Attach Request, MS extracts the device identifier SUPI and a precise timestamp. Indeed, in the 5G protocol, each UE is identified with a unique identifier called SUPI. The latter is encrypted to provide better privacy and prevent the International Mobile Subscriber Identity (IMSI) catcher attacks that were popular in the previous generation of telecommunication protocols. It should be noted that SUPI is not transferred in clear text over the RAN, which makes it challenging to identify devices from the traffic received on the radio layer. Therefore, our

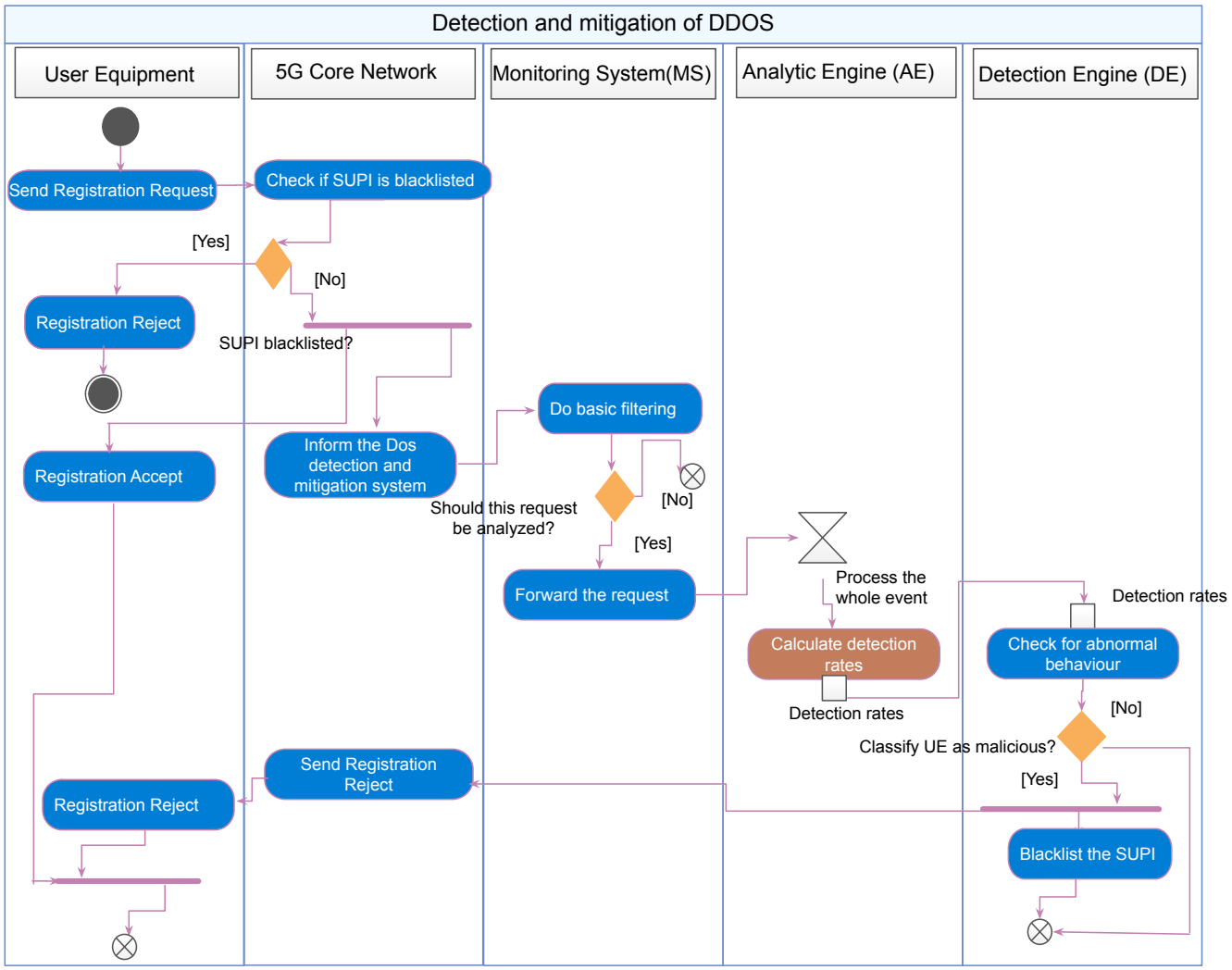

Fig. 2. Interaction of the mMTC network slice and the closed-control loop to detect and mitigate attacks

solution has to intervene at the 5G CN (i.e., AMF), which can decrypt the SUPI information. The extracted information is then forwarded to AE via a communication bus based on the Publish/Subscribe concept.

B. Analytical Engine (AE)

Fig. 3 illustrates a detailed vision of the AE components, which are: Sampler, Activity Detector, DataBase (DB), Event Detector, and Analysis components. They interact together to: (1) detect when an event starts and ends. It can correspond to MTC devices report (normal traffic) or attacks; (2) analyze the event to detect if it is normal or abnormal traffic; (3) compute the detection rate for each device (probability that a device has participated in the attack) and send a report to DE. We decided to separate the event detection from event analysis to improve performances and consider all the relevant data when running the overall attack detection algorithm. Indeed, we decided to detect activity periods (i.e., events) in the network traffic and only feed data to the ML algorithm at the end of an event, which provides the advantage that the resource-intensive component (detection analysis) runs once every event. We also consider two corner cases: (1) after a duration clearly greater than the maximum length of an event; (2) when peak traffic exceeds a limit indicating that it is definitely an attack. For both cases, we tag the devices as malicious.

The only downside of separating event detection from the analysis is that UEs participating in an attack will not get banned right away when the attack starts. However, since the damage in DDoS attacks stems from their duration in time, the devices will get disconnected and blacklisted a few seconds or minutes after the event starts by DE. The detection algorithm does not need to run in real-time, and it can look at data of the whole event.

1) Event detection: To detect activity periods, we first calculate the rate of Attach Request. To this end, we devise a new component called the Sampler, which receives data that reaches AE from MS and emits data periodically by grouping the Attach Requests in time intervals of a fixed length. Fig. 4

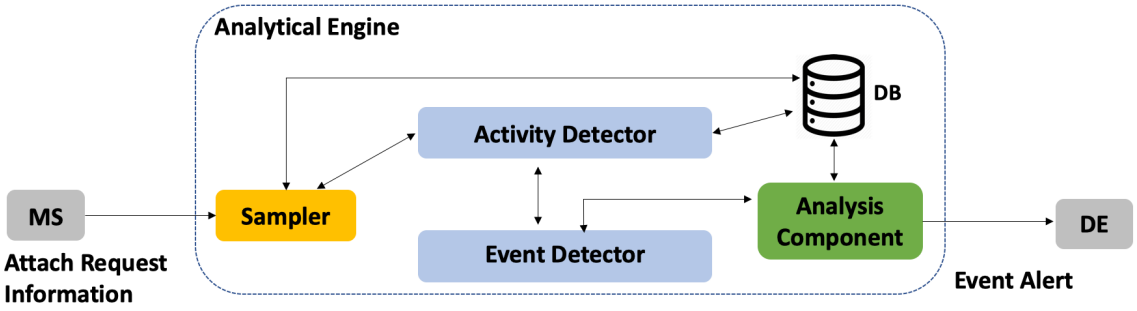

Fig. 3. AE's components

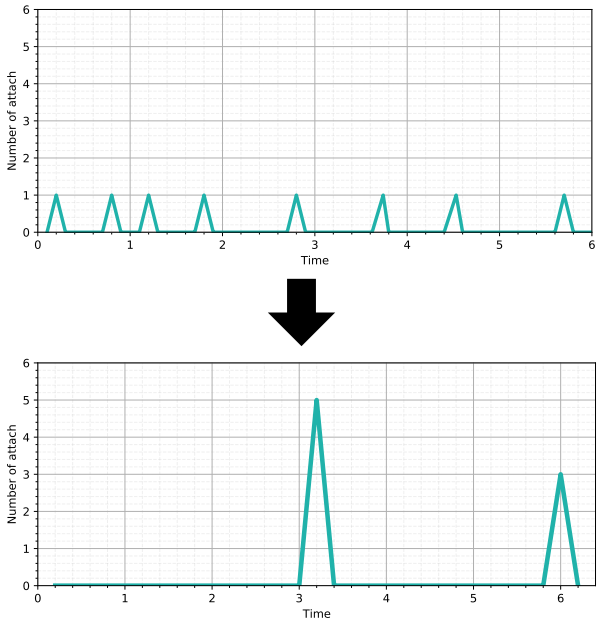

Fig. 4. Sampler: attach requests on time intervals of a fixed length

illustrates how the Sampler works. The upper part of the figure shows the real Attach Request timestamp reception, while the down part of the figure shows the output of the Sampler that groups the Attach Requests every 3 seconds. The Sampler outputs allow detecting the start and end of an event. If there is traffic, and we are not already in an event, we consider that an event has started. If we are in an event and detect that the system has not received traffic for a given duration, we assume that the event has ended. We use a DB for storing all the data relevant to the event. To have a modular system, we separate the Activity Detector (the part that detects whether there is traffic or not) and the Event Detector (the part that delimits an event's start and end timestamps).

As stated earlier, a DB is used to store information about the event, namely the number of Attach Requests received for each event, a list of SUPI values that identify the devices that emitted each Attach Request, and timestamps. The DB is accessible and used by: (1) the Sampler to store pairs (*timestamp*, *supi*), retrieve the largest timestamp stored, retrieve all the data stored, and delete all the data stored; (2) Activity and Event Detectors to store and edit a boolean value *is_event*. It is worth noting that a relational DB is not an optimal choice in terms of the DB model. A better choice is a Time Series DB, as we will store time events. After many considerations, we finally opted for a key-value database. Indeed, our processing of timestamps will still be efficient, as we will be using a sorted set, and we will be able to store the boolean value needed by the Activity Detector. This avoids the need for multiple DBs inside AE. The design of the database is highlighted in table II.

TABLE II
THE DESIGN OF OUR KEY-VALUE DATABASE

Key	Type	Used By
devices	**sorted set**	Sampler
is_event	**string**	Activity and Event Detector
last_attach_requests	**sorted set**	Activity and Event Detector

2) Event analysis: ML Algorithm: The core component of AE is the Analysis Component, which receives data about one event, including the SUPI identifiers of all the devices that sent an Attach Request; then, it calculates a percentage for each device being part of an attack. The Analysis Component output (i.e., a percentage) should be zero ideally for normal traffic, as no abnormal behavior is present. When some devices misbehave and send Attach Requests that clearly do not correspond to normal traffic, the Analysis Component output should be higher than 0. The detection rate should increase as the traffic rate increases above the normal level. We also want high detection rates (preferably 100%) for traffic likely to cause a DDoS attack.

Many algorithms can be applied to solve this problem. However, not all of them can be used as we have two important considerations:

- We already have a model for our data: the $\beta(3,4)$ probability distribution.
- New equipment can be introduced so that the event lengths can vary. Hence, the envisioned algorithm should not detect these changes as anomalies.

While malicious traffic that triggers DDoS is easily discernible from normal traffic, it is hard to evaluate the detection rates for anomalies that are not blatant attacks (i.e., when the traffic rate is just a bit above the prediction interval bound, for example).

In what follows, we enumerated a list of algorithms that can be used for this purpose.

- Statistical tests, checking the mean, median, extremums, variance, and more, and comparing them with the properties of the $\beta(3, 4)$ distribution. Note that we will compare our proposed solution against a statistical-based solution.
- k-NN, SVM, Isolation Forests, Decision Trees, Gradient Boosting, etc.
- Deep Learning algorithms based on Neural Networks (Auto Encoders, Recurrent Neural Networks).

We discarded Deep Learning based algorithms as learning from data is not something we want. It would break our second consideration, and the change in the number of connected devices could trigger wrong predictions. Further, we believe that a neural network trained on normal traffic will not give a gradually higher detection probability on outliers because it does not consider outliers as elements that are more distant from normal data. It instead works with the combination of linear and non-linear mathematical operations. We considered the algorithms that can calculate prediction intervals, which are intervals that likely include our data. Indeed, if we can get an interval that includes our data, we can use the distance from its upper bound to calculate a prediction percentage. The most popular ML algorithm for calculating prediction intervals is Gradient Boosting. It is an extension of Boosting, where the additive generation of weak models is based on the gradient descent algorithm over an objective function [25]. The Gradient Boosting decision tree algorithm has demonstrated great performances on many machine learning tasks, including global contests [26]. It produces a prediction model in the form of an ensemble of weak prediction models, often decision or regression trees. It combines weak "learners" into a single strong learner in an iterative fashion, lowering the error estimated with the chosen loss function at each stage.

3) Event analysis - Data generation and model training: We generated data for training the ML algorithm in the same format as the data we run our detection on, i.e., timestamps of Attach Requests in a simulated event. We used a 5G testbed with emulated UE to emulate an event and generate data. The details on the 5G testbed are provided in section VI.

Algorithm 1 is used for generating data. The complexity of this algorithm is $O(n^2)$, where $n * n$ operations are required for inputs, while each outer loop runs n times. We consider that the average number of equipments that would take part in an event is $\frac{duration \times 3}{7}$. We argue this by the fact that the mean of the $\beta(a, b)$ distribution is $\frac{a}{a+b}$, where a=3 and b=4. We generate random numbers following the $\beta(3, 4)$ probability distribution. These numbers represent the expected timestamps of Attach Requests. We send Attach Requests and sleep for a duration equal to the difference between the random numbers generated to simulate the event. When this step ends, the data is stored in a file, where each entry corresponds to a sample period. Let us note by S the sample vector defined as $\{s_1, s_2, s_3, \ldots, s_n\}$, where s_1 corresponds to the sample period 1, and noted in an entry of the file as (t_1 - 0) ($t_1 = \Delta$); s_i is $(t_{i+1} - 0)$ $(t_{i+1} = i + 1 * \Delta)$. The duration of the sample period is identical and obtained as follows: $\Delta = \frac{Total\_Duration\_Event}{n}$. For the training phase, we will use the data generated earlier to train a Gradient Boosting algorithm to predict for each sample s_i, the upper bound of the number of Attach Request expected during that sample period; we note it by $Predict_i$. Here we are interested in the upper bound value as it corresponds to the maximum intensity of normal traffic; hence exceeding this value means that we are most probably facing a DDoS attack. Once the training is done, we obtain a vector P that contains n predicted values corresponding to the maximum expected Attach Requests per sample period. The $Predicted_i$ values are also stored in the file with each corresponding entry. The file will be used later as an input to analyze an event and calculate the detection rate.

Algorithm 1 Data Generation

```
1:  procedure GENERATE_EVEN_DATA (int duration, int sample duration )
    ▷ The average number of equipments that would connect during an event of length duration
2:      num_equipments = floor(duration×3 / 7)
3:      timestamps is an array of num_equipments reals
4:      for i ← 1 To num_equipments do
5:          timestamps[i] ← random_β3,4() × duration
6:      end for
7:      Sort the timestamps array
    ▷ At this point, timestamps has real values in the range [0,duration] sampled is a list of integers
8:      k ← 0.0
9:      i ← 0
10:     sampled_index ← 0
11:     while k <= duration do
12:         j ← 1        ▷ Count the number of samples in the range [k, k + sample_duration]
13:         count ← 0
14:         while j < num_equipments and timstamps[j] < k + sample_duration do
15:             count ← count + 1
16:             j ← j + 1
17:         end while
18:         i ← j
19:         sampled[sampled_index] ← (k, count)
20:         sampled_index ← sampled_index + 1
21:         k ← k + sample_duration
22:     end while
23:     Output sampled
24: end procedure
```

4) Event analysis - Detection: During the detection step, the event detected by the Sampler is stored in the DB. The event is organized in sample periods equal to n with a duration Δ (the same value used for the training phase). This will allow us to normalize the number of samples of an event since each event has a different duration period, and the $\beta(3, 4)$ intensity depends on the duration. Thus, we do not need to train the model using different durations, as the normalization step will allow training on a single duration corresponding to $\beta(3, 4)$ distribution. Since the event has been organized by sample periods with the total number of Attach received during the

sample period as well as the SUPI of the UE, we use the ML model (mainly the file obtained in the precedent step) to extract the $Predict_i$ values for each sample period. Then, we derive another bound for each sample period as follows: $Predict_i \times$ (AE_DETECTION_THRESHOLD - 1); a limit above which any traffic gets a 100% detection rate and gets classified as malicious. For each sample period, we calculate the detection rate as follows:

For x_i, the number of Attach Requests in the sample period $[s_i]$:

If $x_i \leq Predict_i$, then $detection_i = 0$.

Else,

$detection_i = min(1.0, \frac{x - Predict_i}{Predict_i \times (AE\_DETECTION\_THRESHOLD - 1)})$

where $0 < detection_i < 1$.

Let us suppose that during an interval (Δ), the number of Attach Requests is greater than the predicted one, meaning abnormal traffic. In this context, to estimate the other bounds, from which detection values are 100%, we use $predicted_i \times (AE\_DETECTION\_THRESHOLD - 1)$; the predicted value is multiplied by a constant, corresponding to the rate between the distance of x from the $Predict_i$. This is needed to reduce the ML errors impact and hence reduce the False Positives. Note that if $detection_i$ is higher than zero, all the involved UEs during that sample period are considered as part of a DDoS attack with rate $detection_i$.

V. Closed-control loop: attacks mitigation

DE is the decision-maker of the closed-control loop system. It receives data from AE and decides the actions to take for UEs that emit abnormal traffic. DE gets as input a list including the suspected UEs (SUPI) and their corresponding detection rate values (i.e., $detection_i$) belonging to the attacks. We devise two versions of DE. The first solution blacklists devices if their calculated prediction is higher than $DE\_DETECTION\_THRESHOLD$ (i.e, a configurable parameter). The lower the threshold value, the higher the probability that devices are blacklisted. Therefore, the network operator would use lower values in order to be more strict, but in turn, it may increase the false positive. To avoid having high false-positive results and simplify the configuration, we introduce a second solution that relies on three thresholds. This solution considers the whole event and classifies the received list of UEs into three categories: F_1, F_2, and F_3. DE calculates how many UEs have obtained a detection rate ($detection_i$) higher than 0.8 and assigns it to the first category, namely F_1. The second category includes UEs having a detection rate between 0.3 and 0.8. This category corresponds to F_2. The remaining UEs are included in F_3. Then, DE checks if, in the event, a significant part of the devices had higher than usual detection rates. As a result, different decisions are to be taken:

- First, if $F_1 \geq F_2$ and $F_1 > F_3$, DE blacklists all the devices by adding their SUPI values to a table of blacklisted values, and disconnect them from the network.
- Second, if $F_2 \geq F_1$ and $F_2 \geq F_3$ and ($\forall x \in detection, x > 0$), DE adds the SUPI values to a table named "non-trusted devices". Each UE belonging to this table has a counter named T_{imsi}. The counter is increased by 1 each time the UE is involved in abnormal traffic that is not blatantly an attack. The counter is incremented until it reaches a value that yields to blacklist the device.
- Third, DE ignores the alert sent by AE, and it will do nothing.

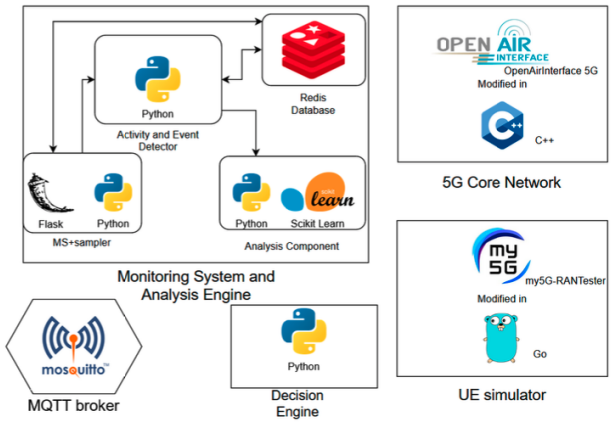

Fig. 5. Test platform and technological components

The reason behind using 3 categories is based on the AE analysis and the results accuracy of the employed ML algorithm. Indeed, we notice that when the detection values associated with the connected devices sent by AE are high, the devices' generated traffic does not follow the Beta distribution. Hence, they should be immediately blacklisted as they present abnormal behavior, justifying the need for the first category. The latter is considered a red alert, and the associated devices' SUPI are blacklisted. However, when the values are neither too high nor too low, it means that the traffic is almost close to the Beta distribution. In this case, the detected devices have malicious behavior, or the values are due to technical failure. So we introduced the second category, which means that the devices are not blacklisted; but DE memorizes the associated SUPI for future events. If the devices are classified in the second category more than 2 times, they will be considered malicious and moved to the first category to be blacklisted. The last category corresponds to the detected traffic being very close to the Beta distribution. This case can be either an error in the ML calculation or a delay in sending the Attach Request.

VI. Performance Evaluation

A. The test platform

To validate the proposed zero-touch security management system, we have used a 5G testbed deployed at EURECOM[1]. The testbed has been developed and used in many 5G European projects such as 5GEve[2] and 5G!Drones[3]. We have implemented the closed-control loop components (i.e., MS, AE, and DE) and a Element Manager (EM) on top of the AMF. The latter exposes API to (1) MS to monitor the Attach

[1]https://www.youtube.com/watch?v=90SRV9ZpPVot
[2]https://www.5g-eve.eu/
[3]https://5gdrones.eu/

Request message; (2) DE to detach and blacklist UEs involved in an attack. Fig.5 illustrates the different technologies used to implement the above-mentioned components. As a quick reminder, the roles of the different components are:

- MS and Sampler: MS is the first component to receive traffic from the 5G CN. It performs basic filtering on it. While the Sampler does sampling of the input data, it receives information on Attach Requests as they are received (with no guaranteed periodicity) and emits periodic data, with the number of Attach Requests received in time intervals of a given length.
- Activity and Event Detectors: These components receive the sampled data and should detect an event. For each event, these components only emit data at its end, with the number of requests on each time-slice along with the UE list that emitted traffic during the event.
- Analysis Component: This component runs the ML algorithm on the given data, calculating a detection rate for each time-slice (for all devices in the time-slice).
- DE: This component receives data from the Analysis Component and decides which devices should be disconnected from the network and then blacklisted.
- MQTT Broker: is used to implementing the communication bus between the different components of the closed-loop control system, and between the closed-loop control system and the AMF.

Regarding the UE, we used and updated a 5G UE emulator, my5G-RANTester[4], to be able to send a high number of UE Attach Request messages in parallel to simulate an attack or normal traffic. Indeed, my5G-RANTester is a tool for emulating control and data planes of the UEs and gNB. my5G-RANTester follows the 3GPP Release 15 standard for RAN. By using my5G-RANTester, it is possible to generate different workloads and test several functionalities of a 5G CN, including its compliance with the 3GPP standards. Scalability is also a relevant feature of the my5GRANTester, which can mimic the behavior of a large number of UEs and gNBs simultaneously accessing a 5G CN. Currently, the wireless channel is not implemented in the tool. The AMF and 5G CN components are based on OpenAirInterface (OAI).

As described earlier, AE and DE use several parameters that need to be tuned to optimize the different steps of the event analysis. These parameters are summarized in Table III and defined as follows:

Δ(Sec): Length of the sampling interval. A message is sent from the sampler every Δ seconds, including the number of UEs that connected in the last interval of time of this length.

***UNCHANGED_INTERVALS_COUNT*:** Number of intervals without activity after which an event is marked as finished. This parameter is used to detect the end of an event. For instance, if we assume its value is equal to 4 and *INTERVAL_LENGTH* is 6, then we can consider an event has ended if there is very little to no activity for 24 seconds (or four intervals) while an event was ongoing.

***REQUEST_THRESHOLD*:** If there are fewer requests than this value in an interval of time, we assume no activity.

[4]https://github.com/my5G/my5G-RANTester

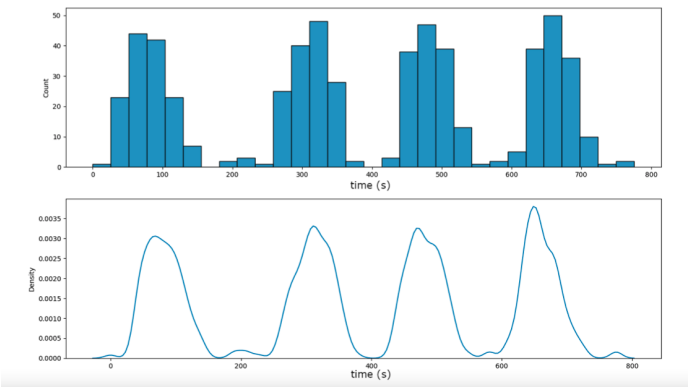

Fig. 6. Normal traffic - four events

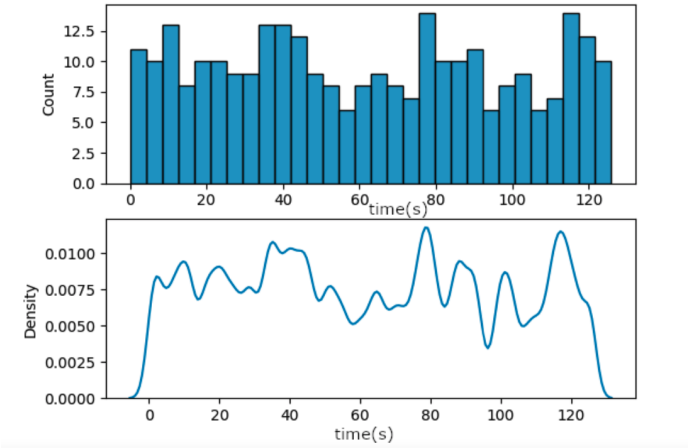

Fig. 7. Malicious traffic

***MAX_EVENT_DURATION (Sec)*:** If the end of the event is not detected after *MAX_EVENT_DURATION*, we consider it is an attack.

***DE_DETECTION_THRESHOLD*:** a threshold used by DE to determine if a device should be banned from the network or not. Any detection rate higher than this value leads the system to ban the device.

Regarding the testing conditions, as we do not have datasets that include both positives and negatives, we generate traffic corresponding to non-malicious using the $\beta(3,4)$ probability distribution, while for abnormal traffic, any other distribution can be used.

To test our system under realistic circumstances, we lever-

TABLE III
THE CONFIGURABLE PARAMETERS IN OUR SYSTEM

Parameter Name	Default Value	Component
INTERVAL_LENGTH	**6.0**	Sampler
UNCHANGED_INTERVALS_COUNT	**4**	Sampler
REQUEST_THRESHOLD	**1**	Detector
MAX_EVENT_DURATION	**600**	Detector
DE_DETECTION_THRESHOLD	**0.35**	DE

aged my5G-RANTester with a script that emulates real UE traffic (control plane) to communicate with 5G CN components. We used the emulator to simulate an event with different chosen SUPI values. The generated traffic is similar to what real UEs would generate. It follows the 3GPP specifications 15. Algorithm 2 generates the traffic featuring:

- Simulating an event: Sending Attach Requests that follow the Beta-distribution
- Simulating an attack: Sending Attach Requests that follow the uniform distribution
- Simulating a Attach Request of a single UE

Algorithm 2 is used for simulating an event. It is very similar to the algorithm that generates data. The complexity of this algorithm is $O(n)$, where n operations are required for input, and the outer loop runs n times. Here, n corresponds to the number of devices involved in an event. Figs. 6 and 7 show a visualization of traffic likely corresponding to a normal (four events and attack) and malicious one, respectively.

Last but not least, readers may see a video[5] showing a demonstration of all the components, i.e., closed-control loop as well as AMF and UEs, working together to detect DDoS attacks. The following section will provide some results on the model accuracy obtained via the experiments.

B. Test results

To evaluate the performance of the proposed framework, we focus mainly on the performance of the attack detection algorithm, which is the key function of the closed-control loop. To this end, we evaluate the Gradient Boosting algorithm to detect DDoS attacks accurately and compare its performance with a statistical method. Like the Gradient Boosting solution, the statistical method is applied at the end of the event. Based on the event duration, the statistical method defines a limit function using the mathematical function of $\beta(3,4)$ to compare the different points by the report to this limit; all the points exceeding this limit are considered as a possible attack. The main differences compared to Gradient Boosting are: (i) it needs the exact duration of the event; (ii) it uses the $\beta(3,4)$ function to deduce the limit.

[5]https://www.youtube.com/watch?v=QzCmGfwtDLAt=7s

Algorithm 2 Event simulation

```
procedure SIMULATE_EVENT(int duration)
    num_equipments = floor((duration×3)/7)
    timestamps is an array of num_equipments reals
    for i ← 1 To num_equipments do
        timestamps[i] ← random_β3,4() × duration
    end for
    currSUPI ← MCC + MNC + "0000000001"
    Sleep(timestamps[0])
    for i ← 1 To num_equipments − 1 do
    SendAttachRequest_ASYNC(currSUPI)
    currSUPI ← str(int(currSUPI) + 1)
    Sleep(timestamps[i])
    end for
    SendAttachRequest_ASYNC(currSUPI)
end procedure
```

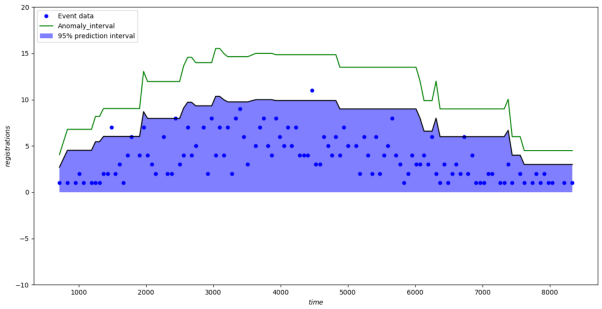

Fig. 8. The result of the detection algorithm over normal traffic

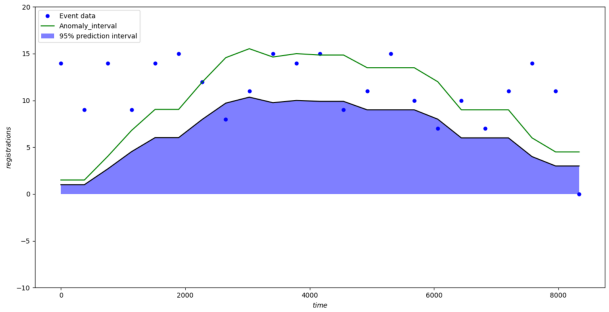

Fig. 9. The result of the detection algorithm over abnormal traffic

a) Gradient Boost: We measure the accuracy of the Gradient Boosting algorithm in front of normal and abnormal traffic. On normal traffic, the accuracy denotes how often the system yields a detection rate of UEs that is greater than zero. This does not mean that these devices will get banned, but ideally, a value of 0 should be returned for all the devices emitting normal traffic. To evaluate our model on normal traffic (True Positive (FP) = False Negative (FN) = 0), we generate data for 500 normal events and run our detection algorithm on each of them. We then counted the number of UEs for which we obtained a greater-than-zero detection rate versus the total number of UEs in all the events. We generate the event duration randomly (between 30 and 250 seconds). Regarding malicious traffic (True Negative (TN) = False Positive (FP) = 0), we also ran 500 tests, but this time, between 7 and 15 Attach Requests are received every 6 seconds, for a total duration that is random between 30 and 250 seconds.

Fig. 8 allows visualizing the results for normal traffic. The points correspond to the event data, while the green line is the anomaly interval. If a point is outside the limit (in green), it will be assumed as an attack. For normal traffic, the accuracy is computed as $1-\frac{FP}{TN+FP}$. Hence, the results show an Accuracy on normal traffic of 96.76859478052322%. We expected this result as the interval used in our training data includes around 95% of the data in the training dataset, as depicted in Fig.8.

Fig.9 illustrates the results for malicious attacks. For this case, the accuracy is computed as $1-\frac{FN}{FN+TP}$. Hence, the results show an accuracy of 83.63319140762557%. This represents an excellent result as banning a relevant part of the devices taking

TABLE IV
IMPACT OF THE AE DETECTION THRESHOLD ON THE GRADIENT BOOSTING ACCURACY

AE_DETECTION_THR.	1.2	2.0	3.0
Normal event	82.98654	96.76859	97.64853
Abnormal event	81.91305	83.6331914	61.86956

part in a DDoS attack is enough to mitigate it. We recall that this is just the detection rate calculated by the AE component, the final decision regarding the devices that should be banned is taken by the DE component.

Table IV shows the performance of the Gradient Boosting-based solution when modifying the *AE_DETECTION_THRESHOLD* value. It is worth recalling that this value is used to derive the detection rate and corresponds to a protecting gap to reduce the impact of the ML prediction error and hence reduce the FP value. We remark that the value allowing to reduce both FP and FN is 2.0. Also, when the *AE_DETECTION_THRESHOLD* value increases, FP is reduced as the FN increases; whereas when it is reduced, both FP and FN increase.

b) Statistical Method: For the sake of comparison, we used the same scenarios as for Gradient Boosting to generate normal and abnormal traffic. Then, we applied the statistical method and verified its accuracy in detecting attacks. Fig. 10 illustrates the usage of the statistical method in case of an abnormal event. The discontinue green line shows the $\beta(3,4)$ curve obtained according to the event duration. The $\beta(3,4)$ curve allows us to have a limited path from which all the outside points are considered anomalies, hence potential attacks. The statistical method's results show that 36.0% of the Attach requests are not following the β(3,4) distribution (they are out of the limit path). Therefore, they can be considered potential attacks. On the other hand, Fig. 11 presents a test of a normal event. The results show that 90.0% of the Attach Requests follow the β(3,4) distribution. The accuracy of the statistical method to detect anomaly are : $((1 - \frac{FP}{TN+FP} = 84.21052\ \%$ (normal traffic), $1 - \frac{FN}{FN+TP} = 57.142857\ \%$ (abnormal traffic))). We remark that these values are weaker than the ones obtained with the Gradient Boosting algorithm. We argue these differences by the fact that the duration estimation has a strong impact on the statistical solution. The shape of the $\beta(3,4)$ curve changes drastically according to the duration (noted D), which seriously impacts the accuracy. For instance, we change the duration by +/- $\epsilon = 2sec$, and the obtained results are summarized in Table V. We see clearly from this table the impact of the duration on the accuracy as a small error on the duration drastically yields a drop in the accuracy. Particularly, if the duration is less than the real one, many points will be out of the curve. In the Gradient Boosting algorithm, we do not have this concern, as the latter normalizes the sample period duration and uses the trained model to detect the interval.

c) Decision Engine: Regarding DE performances, we evaluated the first version that relies on a single threshold *DE_DETECTION_THRESHOLD*. Accordingly, in this section, we evaluate the *DE_DETECTION_THRESHOLD* impact on the number of blocked devices. Table VI

TABLE V
THE ACCURACY OF THE STATISTICAL MODEL CONSIDERING DIFFERENT DURATION VALUES

Method	GB	Static		
		D - Δt	D	D + Δt
Normal traffic	96.76859	45.18924	84.21052	80.24568
Abnormal traffic	83.633191	30.4156	57.142857	51.86854

TABLE VI
IMPACT OF THE DE DETECTION THRESHOLD

DE_DETECTION_THR.	0.1	0.35	0.8
Nb (Normal event)	3	1	0
Nb (Abnormal event)	21	16	10

shows the number of banned UEs for three values of the *DE_DETECTION_THRESHOLD*. As expected, we remark that lower threshold values (ex. 0.1) are very conservative, which allows blocking more UE. While a higher threshold value (ex. 0.8) may be less strict and reduces the number of banned UE. We advise two solutions to fix the *DE_DETECTION_THRESHOLD* value. The first one considers the performance limit of the element to protect against DDoS attacks. In our case, we computed the response time of the AMF to Attach Requests while increasing their number. After a certain number of Attach Requests, we noticed that the AMF started to be very slow, which can be caused by a DDoS attack. Therefore, after some tests, we found that the value of *DE_DETECTION_THRESHOLD* equal to 0.35, which avoids reaching the number of Attach Request that yields bad AMF performances. Another solution is to use a dynamic threshold value that decreases or increases over time according to the number of consecutive events classified as an attack.

VII. CONCLUSION

In this paper, we introduced a zero-touch security management framework that aims to protect mMTC network slices from in-slice DDoS attacks. The proposed framework relies on a closed-control loop that tracks Attach Requests generated by MTC devices to detect abnormal traffic and mitigate possible DDoS attacks. The mitigation process is enforced

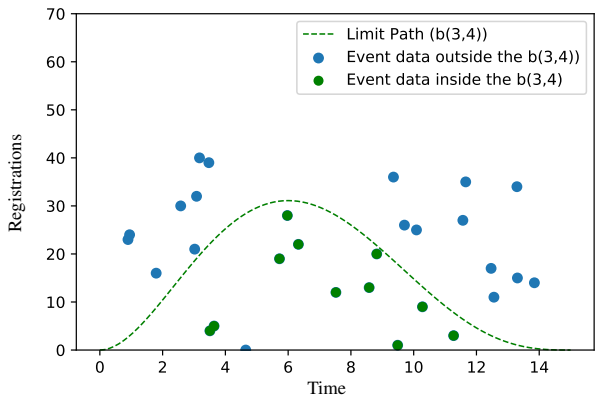

Fig. 10. The result of the statics method over abnormal traffic.

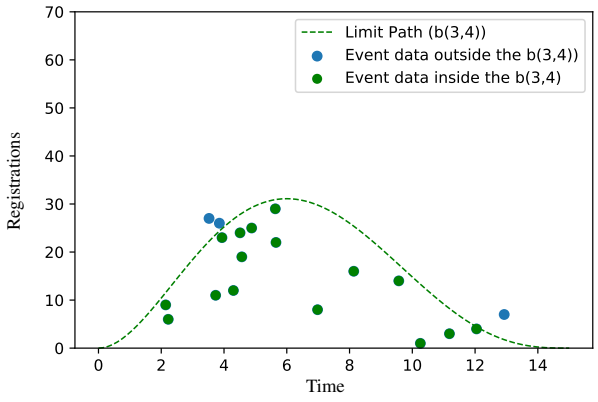

Fig. 11. The result of the statics method over normal traffic.

through disconnecting and banning suspected devices. The attack detection algorithm uses a ML technique, specifically Gradient Boost, to create a prediction interval that is likely to include normal traffic in our system. It then calculates, for every sample that is outside the interval, a metric that depends on its distance from the bound of the interval; this metric is called the detection rate. The decision engine uses this metric to take action to mitigate attacks, which consists in banning and disconnecting devices from the network to prevent them from conducting similar attacks in the future. The proposed framework has been implemented using a 5G testbed. The obtained results demonstrate the closed-control loop's ability to predict and mitigate DDoS attacks efficiently.

Acknowledgment

This work has been partially supported by the European Union's H2020 MonB5G (grant no. 871780) project.

References

[1] Architecture enhancements for 5G System (5GS) to support network data analytics services, 3GPP TS 23.288 version 16.4.0 Release 16, July, 2020.
[2] A.Amokrane, A. Ksentini et al., Congestion control for machine type communications,Proceedings of IEEE International Conference on Communications, ICC 2012, Ottawa, ON, Canada, June 10-15, 2012,
[3] O. Arouk and A. Ksentini, Multi-channel slotted aloha optimization for machine-type-communication, 17th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems, MSWiM'14, Montreal, QC, Canada, September 21-26, 2014.
[4] Arne Holst, Number of Internet of Things (IoT) connected devices worldwide from 2019 to 2030, 2021.
[5] Ericsson, A guide to 5G network security 2.0, sep, 2021.
[6] Jordan Lam and Robert Abbas, Machine Learning based Anomaly Detection for 5G Networks, mar, 2020.
[7] Grant Millar et al, D2.1: 5G Security: Current Status and Future Trends, in "Intelligent Security and Pervasive trust for 5G and Beyond, jul, 2020.
[8] Amir Alimohammadifar and Suryadipta Majumdar and Taous Madi et al, Stealthy Probing-Based Verification (SPV): An Active Approach to Defending Software Defined Networks Against Topology Poisoning Attacks, 23rd European Symposium on Research in Computer Security, ESORICS 2018, aug, 2018.
[9] Antoine Delplace and Sheryl Hermoso and Kristofer Anandita, Cyber Attack Detection thanks to Machine Learning Algorithms, jan, 2020.
[10] João Henrique Corrêa and Patrick Marques Ciarelli and Moises R. N. Ribeiro et al, On Machine Learning DDoS Attack Identification from Cloud Computing Telemetry, apr, 2019.
[11] Rohan DDoShi and Noah Apthorpe and Nick Feamster, Machine Learning DDoS Detection for Consumer Internet of Things Devices, apr, 2018.
[12] Faisal Hussain and Syed Ghazanfar Abbas and Muhammad Husnain et al, IEEE 23rd International Multitopic Conference (INMIC), IoT DDoS and DDoS Attack Detection using ResNet, 2020.
[13] Xiaoyong Yuan and Chuanhuang Li and Xiaolin Li, DeepDefense: Identifying DDoS Attack via Deep Learning, IEEE International Conference on Smart Computing (SMARTCOMP), 978-1-5090-6517-2, doi = 10.1109/SMARTCOMP.2017.7946998, 2017.
[14] Maryam Ghanbari and Witold Kinsner, Extracting Features from Both the Input and the Output of a Convolutional Neural Network to Detect Distributed Denial of Service Attacks, IEEE 17th International Conference on Cognitive Informatics & Cognitive Computing (ICCI*CC), 978-1-5386-3360-1, doi = 10.1109/ICCI-CC.2018.8482019, jul, 2018.
[15] Roberto Doriguzzi-Corin and Stuart Millar and Sandra Scott-Hayward et al, Lucid: A Practical, Lightweight Deep Learning Solution for DDoS Attack Detection, 17, 1932-4537, doi = 10.1109/TNSM.2020.2971776, feb, 2020.
[16] Michael Schachter, Allot, DDoS 5G: The Bigger the Pipe, the Stronger the Threat, June 26, 2018.
[17] Sean Newman, Enterprises Beware: Variations on the Mirai Malware Still Feeding DDoS Attacks, 2022.
[18] A. S. Mamolar, Z. Pervez, Q. Wang et al, "Towards the Detection of Mobile DDoS Attacks in 5G Multi-Tenant Networks," 2019 European Conference on Networks and Communications (EuCNC), 2019, pp. 273-277, doi: 10.1109/EuCNC.2019.8801975.
[19] Mahmoud Ammar, Giovanni Russello b, Bruno Crispo, Internet of Things: A survey on the security of IoT frameworks, Journal of Information Security and Applications, 2018.
[20] Raja Ettiane, Rachid EL Kouch, Mitigating Denial of Service Signaling Threats in 5G Mobile Networks, (IJACSA) International Journal of Advanced Computer Science and Applications, Vol. 12, No. 2, 2021.
[21] Florian Metzger and Tobias Hoßfeld and André Bauer et al, Modeling of Aggregated IoT Traffic and its Application to an IoT Cloud, Proceedings of the IEEE, 1558-2256, doi = 10.1109/JPROC.2019.2901578, mar, 2019.
[22] Hatim Chergui, Adlen Ksentini, Luis Blonco, et al, "Toward Zero-Touch Management and Orchestration of Massive Deployment of Network Slices in 6G", accepted in IEEE Wireless Magazine, special issue on 6G: The paradigm for future wireless comunications
[23] Markus Laner and Philipp Svoboda and Navid Nikaein et al, Traffic Models for Machine Type Communications, ISWCS 2013; The Tenth International Symposium on Wireless Communication Systems, aug, 2013.
[24] ETSI, 5G; Security architecture and procedures for 5G System (3GPP TS 33.501 version 15.1.0 Release 15), jul, 2018.
[25] Mason, Llew and Baxter, Jonathan and Bartlett, Peter et al,Advances in Neural Information Processing Systems, Boosting Algorithms as Gradient Descent, MIT Press, 1999.
[26] Vu, Quang and Ruta, Dymitr and Cen, Ling, Gradient boosting decision trees for cyber security threats detection based on network events logs, 5921-5928, doi = 10.1109/BigData47090.2019.9006061, 12, 2019.
[27] P. Rost and C. Mannweiler and D. S. Michalopoulos et al, Network Slicing to Enable Scalability and Flexibility in 5G Mobile Networks, 1558-1896, url = https://arxiv.org/abs/1704.02129, doi = 10.1109/MCOM.2017.1600920, IEEE Communications Magazine, 2017.
[28] The 3rd Generation Partnership Project (3GPP), Security architecture and procedures for 5G System (3GPP TS 33.501 version 15.1.0 Release 15), 2018.