

# Using Anomaly Detection Techniques for Securing 5G Infrastructure and Applications

Athanasios Priovolos, Dimitris  
Lioprasitis, Georgios Gardikis,  
Socrates Costicoglou  
R&D Department  
Space Hellas S.A.  
Athens, Greece  
{apriovolos;dlioprasitis;ggar;scostic}@  
space.gr

**Abstract**—*Key features of 5G networks include, among others, softwarisation (enabling network functions to be deployed as software components rather than hardware modules) and decentralised deployments (with compute resources pushed to the edge of the network). These features bring unprecedented benefits, yet they also widen the attack surface, making the network more prone to cyberattacks and calling for more sophisticated security controls. In this work we will show how the combined collection and processing of monitoring metrics from the RAN and the edge can be used to detect anomalies and identify attacks both to the 5G edge application as well as the infrastructure. Our approach uses Deep Learning and bidirectional LSTMs to yield quite promising results, operating in real time in a full-stack 5G testbed.*

**Keywords**—5G, Edge Computing, Cybersecurity, LSTM

## I. INTRODUCTION

5G comes with a new rich set of features and capabilities[1][2], which, in addition to their obvious technical and business value, as expected, are accompanied with various side-effects, one of the most important of which is the drastic increase in the attack surface, compared to legacy cellular network infrastructures. Some of these 5G-specific aspects, which, under certain circumstances, may introduce new vulnerabilities and increase the probability of a security incident, are: software-defined infrastructures (more prone to integrity violations compared to hardware functions); slicing and multi-tenancy (introducing new threats related to poor slice isolation); multi-actor infrastructures (implying privacy concerns) and complex, multi-tier architectures (amplifying the effect of cascading security incidents).

In 5G, not only the probability of a security incident increases, but also the expected impact and severity. The connection of more and more devices in a 5G network, many of which track personal data, while others support critical operations (as in Intelligent Transport Systems/connected cars or e-health[3][4][5]), implies that security incidents can lead to severe privacy breach and/or even life-threatening situations.

It is thus evident that the emergence of 5G calls for significantly stricter security controls compared to legacy networks. A more thorough investigation of the 5G security landscape is out of the scope of the present document and can be found in white papers and reports such as the one [6] produced by the Security Working Group of the 5G PPP. ENISA has also conducted an exhaustive survey of 5G threats and vulnerabilities [7].

5G network components, being highly heterogeneous and distributed across the network, create an enormous amount of diverse data (mostly logs and monitoring information), whose timely analysis can lead to effective inference of security incidents. This is the concept of 5G Security Analytics, which is addressed in this paper. 5G Security Analytics refers to the collection and joint analysis of massive heterogeneous data from multiple points of the 5G infrastructure for integrated monitoring, with specific focus on detecting and classifying anomalies associated with security incidents.

Big Data and Machine Learning (ML) technologies are the ideal foundations towards this goal. Big Data infrastructures enable the scalable ingestion, storage and analysis of massive data, even in real-time. At the same time, state-of-the-art ML algorithms enable the identification of incidents, which will go unnoticed using traditional rule-based detection. This enables:

- i) the detection of zero-day attacks, whose exact digital fingerprint is unknown; and
- ii) the proactive identification of threats even at their very early stages, where detection thresholds of traditional methods have not been crossed.

In this context, this paper presents an innovative approach based on the combined collection and processing of metrics from the RAN and the edge, which can be used to detect anomalies and identify attacks both to the 5G applications running at the edge as well as the infrastructure. Our approach uses deep learning and bidirectional LSTMs to yield quite promising results, as implemented and evaluated in a full-stack 5G testbed with over-the-air real-time tests.

## II. BACKGROUND AND RELATED WORK

### A. Background

The Autoencoder is a popular approach for detecting, among others, anomalies in time-series data. Autoencoders are formed by two main parts, the Encoder and the Decoder. The Encoder takes the input and compresses it in a smaller representation. The Decoder takes the compressed input and decompresses it, trying to reconstruct the original input. Between the actual and the decompressed input there is an error. Given that the model has been trained in normal data, this error must be very small. If this error is above a threshold then the given input can be considered as an anomaly.

Long Short-Term Memory (LSTM) networks are a type of recurrent neural networks (RNN) capable of learning order dependence in sequence prediction problems, such as the ones involving time-series data. Authors of [8] have shown that LSTMs are able to solve many time-series data tasks unsolvable by feed-forward networks, using fixed size time windows.

### B. Related Work

Securing 5G networks is a topic with increasing attention from the research community. The authors in [9] analyze existing solutions from both academia and industry for securing core network, radio access network, cloud infrastructure and Internet of Things (IoT). [10] addresses security monitoring and management of 5G networks. In addition, there is an evaluation of the related security measures and standards of core 5G technologies.

Some works have also dealt with securing 5G networks and infrastructures using Machine Learning and Artificial Intelligence. In [11] AI and ML driven applications for 5G network security and their implications are presented. In [12] authors propose a Convolution Neural Network (CNN) for detecting malicious network traffic. As they describe, Deep Learning based security can be very useful for detecting network attacks because of the increased network heterogeneity due to wide variety of IoT devices and sensors.

Rather than examining the network traffic, our work focuses on the processing of monitoring metrics (time-series data) from various heterogeneous network elements, which can indicate a deviation from the normal behaviour of the network. In addition, while most ML-based modules found in modern SIEMs (Security Incident and Event Management) platforms focus on behavioural analysis of individual metrics, our approach is based on the combined processing of multiple metrics across the network.

## III. TESTBED ARCHITECTURE AND COMPONENTS

### A. Overview

The architecture of our testbed consists of three nodes, as it is shown in Fig.1. The main node of 5G infrastructure is the 5G gNB (based on the Amarisoft Callbox product line in this work). This also hosts a minimal version of the 5G Core, essential for handling sessions and routing user traffic. In addition, there is an Edge compute node (based on the Dell Edge Gateway platform), where 5G edge-based applications are deployed and running. Finally, there is a third compute

node where the storage, the visualization and the anomaly detection functions are deployed.

Our anomaly detection pipeline collects metrics from two different data sources (RAN/gNB and Edge node), predicting which of these metrics are part of an anomaly and visualizes the detected anomalies in a web interface. Prometheus [13] is used as the central monitoring platform, in which all data are collected and stored in an InfluxDB [14] time-series database.

### B. Data Collection

The Data collection component consists of two data collectors, Amari Exporter, which retrieves metrics from the Amarisoft RAN (gNB) periodically through sockets and the Node Exporter, which is responsible for collecting system metrics from the Edge node. Collected metrics from both exporters are saved in Prometheus in Edge node.

Other data collectors can also be created and used as long as they send collected data in appropriate format in Prometheus. In addition, other 5G RAN vendors can be supported, if a compatible exporter is developed in order to send collected metrics to Prometheus in expected format.

The Node Exporter [15] is a Prometheus component for collecting Linux system metrics running in the Edge node. It also provides some information about the node running such as OS name, version and system architecture. Its main focus is to collect system metrics about node's CPU, the available and used memory, the number of available bytes from node's file system and swap partition and network metrics, such as the number of transmitted and received bytes from all network interfaces in the machine.

The Amari Exporter is also running in Edge node. It was developed specifically for the needs of the present work. Its main task is to collect gNB metrics, which are communicated through an implementation of Websockets protocol. The collected metrics include download & upload bitrate, RX/TX CPU usage and RX-TX delay. Amari Exporter transforms the data in appropriate format in order to send them to Prometheus.

### C. Data Storage & Visualization

The Data storage & visualization component is responsible for fetching collected data from collectors, transforms them in appropriate format for the storage and sends them to the storage platform. In addition, it provides a UI for visualization of the collected metrics and alerts.

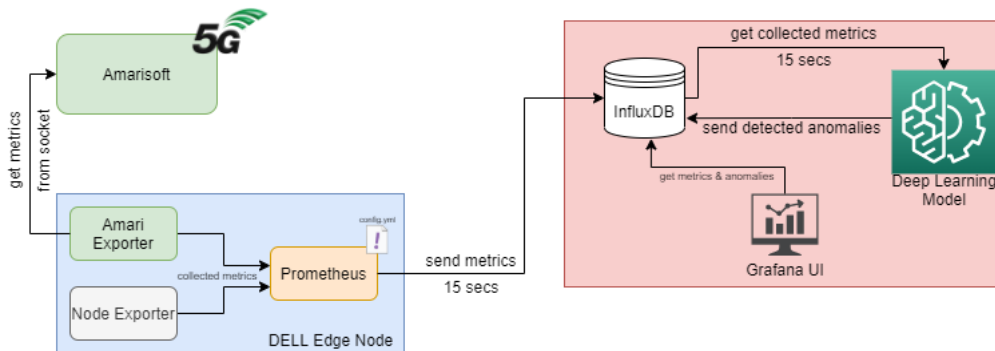


Fig. 1. Testbed architecture and components

tx_cpu_time		rx_cpu_time		dl_bitrate		ul_bitrate	
time-series	time (*)	time-series	time (*)	time-series	time (*)	time-series	time (*)
tag	__name__	tag	__name__	tag	__name__	tag	__name__
tag	instance	tag	instance	tag	cellid (*)	tag	cellid (*)
tag	job	tag	job	tag	instance	tag	instance
field	value (*)	field	value (*)	tag	job	tag	job
				field	value (*)	field	value (*)

node_cpu_seconds_total		node_memory_MemFree_bytes		node_network_receive_bytes_total		node_network_transmit_bytes_total	
time-series	time (*)	time-series	time (*)	time-series	time (*)	time-series	time (*)
tag	__name__	tag	__name__	tag	__name__	tag	__name__
tag	cpu (*)	tag	instance	tag	device (*)	tag	device (*)
tag	instance	tag	job	tag	instance	tag	instance
tag	mode (*)	field	value (*)	tag	job	tag	job
field	value (*)			field	value (*)	field	value (*)

Fig. 2.InfluxDB schema

In the implemented workflow, Prometheus is responsible for fetching, transforming and sending the collected data. After collection, Prometheus transforms and sends these data to the InfluxDB time-series database in real-time. Prometheus leverages the Influx DB API for writing data into Influx DB. It sends POST requests to the /write endpoint, including the time-series data gathered from the exporters. For every metric, a measurement is created on the InfluxDB side.

The Amari Exporter collects the percentage of gNB CPU usage, separately for the RX and TX processes. Also, it collects the bitrate of the gNB uplink and downlink. From the collected metrics for anomaly detection, the difference in rate of these metrics will be used.

The Node Exporter collects system metrics from the Edge compute node. It collects metrics about CPU, memory and network interfaces. CPU seconds are collected as separate metric for each core and each mode. This metric is a counter, so having the value in a previous timestamp and the time difference the usage of CPU in percentage can be extracted. The number of free and the number of total memory bytes are also collected. Using them, the percentage of used memory can be extracted. Finally, from all network interfaces, the total received and transmitted bytes are collected as separate metrics. These metrics are also of “counter” type, so they can be converted to bitrate knowing the values of the previous timestamps and the time difference between the two records.

From these metrics collected from the Edge node, their difference in rate is extracted and used for detecting anomalies and abnormal behaviors. Fig.2 shows the database schema used in InfluxDB, with all fields that must exist in each time series metric. The fields marked with a star are required to exist in Influx in order for the Anomaly Detection Component to work properly because they are used to extract the required features. Both the Amari Exporter and the Node Exporter are written in Go programming language.

For the visualization of collected metrics and detected anomalies, a web user interface has been created. The created UI has been created as a Grafana [16] dashboard. In this UI there are components for monitoring the Edge node in real-time (data are collected every 15 seconds). Fig. 3 shows the configuration of our Grafana dashboard. In the first row there are counters for CPU & memory usage and counters for uplink & downlink bitrate. In the second row there is a timeline about the percentage of CPU and memory and another timeline showing the upload & download bitrate for each network interface. Finally, in the bottom part of the UI, there is a table showing information about the detected anomalies, which is the outcome of our framework. Such info is the timestamp, the specific anomaly detected, the value of system metrics at this timestamp and a possible cause, showing which metric or metrics have abnormal behavior.



Fig. 3.Grafana User Interface for monitoring

#### D. Anomaly Detection Component

For the Anomaly Detection Component, the Autoencoder architecture has been selected. It consists of three parts: an Encoder, a Decoder and a Fully Connected layer, as shown in Fig.4. Encoder has 5 Bidirectional Long Short-Term Memory (LSTM) layers, which take the given data and compress them to a smaller representation. Our Decoder also has 4

```
Model: "sequential_1"
Layer (type)                Output Shape                Param #
-----
bidirectional_1 (Bidirection (None, 4, 128) 37376
bidirectional_2 (Bidirection (None, 4, 64)  41216
bidirectional_3 (Bidirection (None, 4, 32)  10368
bidirectional_4 (Bidirection (None, 4, 16)   2624
bidirectional_5 (Bidirection (None, 4, 8)    672
bidirectional_6 (Bidirection (None, 4, 16)   1088
bidirectional_7 (Bidirection (None, 4, 32)   4224
bidirectional_8 (Bidirection (None, 4, 64)   16640
bidirectional_9 (Bidirection (None, 128)    66048
dense_1 (Dense)                (None, 8)                  1032
Total params: 181,288
Trainable params: 181,288
Non-trainable params: 0
INFO:root:None
```

Fig. 4. Deep Learning Model Architecture

Bidirectional LSTM layers, which try to decompress the compressed input and recreate the original given data. We have decided to use LSTMs since our data are a batch of time series records, which are suitable for LSTMs. Also, we have chosen to use Bidirectional LSTMs, because they can utilize features from both previous and next records. Finally, at the end we have a dense layer, which maps the output of the Autoencoder in the features we want to predict. Training data have been collected from both the Node Exporter and the Amari (gNB) Exporter. Because of these two different systems, there is a small difference in timestamps for these metrics. For this reason, the training dataset is resampled every 15 seconds, to synchronize records from these two exporters. For our case, the following features have been selected: edge node CPU usage, edge node memory usage, gNB RX/TX processes CPU usage, edge node Tx/Rx rate in each network interface and gNB radio Tx/Rx rates. These features are normalized using Min-Max normalization. The min and max values that will be used for normalization are saved in a JSON file.

When the Anomaly Detection Component is deployed, it opens a connection with InfluxDB and fetches new collected metrics every 15 seconds. The collected metrics are processed in order to extract and transform the required features. We use a sliding time window which can hold a specific amount of records. The length of this time window is configurable, so as to set how many previous records will affect the prediction of the next record. When the time window is large, more historical records will affect the value of the predicted record. On the other hand, with a small time window, only the latest metrics will affect the predicted value. By default, the size of each time windows is set to 30, which means that the metrics from the last 7.5 minutes will affect the value of each predicted record.

For the training phase, we have created a “normal” dataset, for which we have used two UEs (smartphones) with a mixed Internet usage pattern, including HD streaming video viewing, Web browsing and speedtest sessions, running over approx. 30 hours. The Autoencoder is trained with this “normal” usage pattern in order to predict the next value in data time-series, given that its input data is part of the normal data. The Anomaly Detector's loss describes the Mean Absolute Error (MAE) between the predicted values of time-series points and their actual values, during training epochs.

For the training task, a Rectified Linear Unit (ReLU) activation function is used. Stochastic Gradient Descent (SGD) with Nesterov momentum has been selected as optimizer, with learning rate equal to 0.01. As validation dataset, 10% of the trainset has been used. The model is trained for 50 epochs.

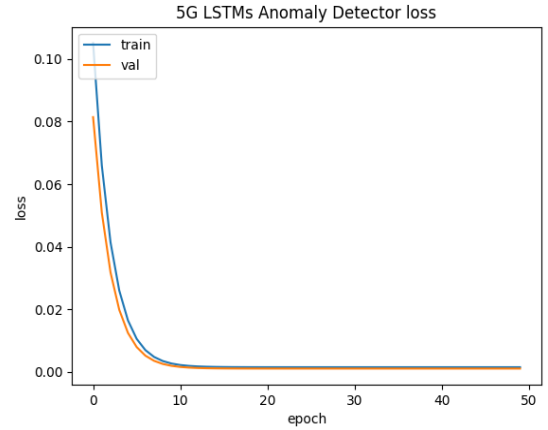


Fig. 5. Loss of Autoencoder during training

Fig.5 shows the loss of the Autoencoder during training. It is shown that during training epochs the loss is decreasing a lot, that it becomes almost equal to zero.

The Anomaly Detection Component has also a feature to propose deviation thresholds for each metric. Because the Autoencoder is threshold-based in order to identify anomalies, it is very important for the threshold to be correctly set. The 99th percentile of each feature's RMSE, calculated over the normal and also the attack datasets (see next section about the attack scenarios) is considered as threshold from the anomaly detection algorithm. The 99th percentile has been selected compared with the max value in order to avoid outliers, that may exist in these datasets. The system operator may further fine-tune this automatically proposed threshold, if so desired.

If the RMSE between the predicted and actual value is above the set threshold, this record is considered as an anomaly. This anomaly is displayed in the Grafana dashboard, along with an indication of the metric(s) which have deviated.

The Autoencoder is developed in Python using Keras with Tensorflow backend.

#### IV. EVALUATION

For evaluating the algorithm and the pipeline as a whole, in real time under actual network operation, three different evaluation scenarios are considered.

The first scenario corresponds to the normal network behaviour, with two UEs under normal user activity, as described in the previous section (web browsing/video streaming/speed tests). The second and third scenario correspond to two different types of attacks to the 5G infrastructure and services. More specifically, the second scenario corresponds to a CPU overload caused by an attacker hijacking the Edge node (Infrastructure compromise) and draining its resources to cause a DoS situation. The third scenario emulates an eavesdropping incident, caused by an attacker hijacking an edge applications (Service compromise) and modifying it to replicate the user traffic and send it to another destination. This behavior is emulated by initiating a second data stream, (produced by the iperf traffic generator) between the Edge node and the network core.

Fig.6 and Fig.7 show the results from the first scenario. In this scenario the trained model operates under normal network conditions. The size of window size is equal with the size of time window in training (i.e. 30). In Fig. 6 the prediction error (RMSE) for each of edge compute metrics is visualized. It is shown that in almost all cases the RMSE is very low. In Fig. 7, which shows the predictions' error for 5G gNB metrics, RMSE is also very low except for two sequences of records. It is shown that in almost all cases the Autoencoder does not produce false alarms, apart from two false positives observed during the entire testing time frame (30 hours).

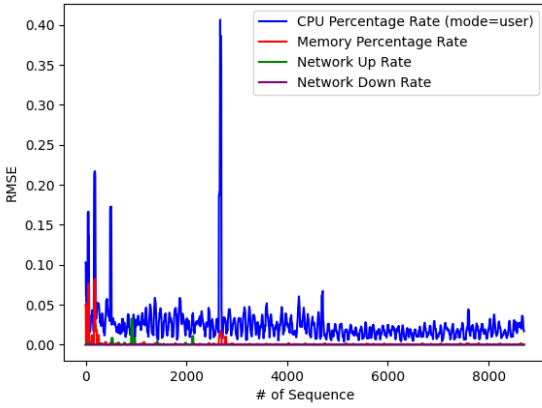


Fig. 6.Prediction Error in normal Data (Edge metrics)

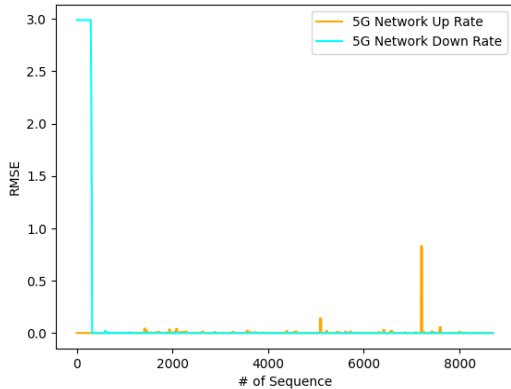


Fig. 7.Prediction Error in normal Data (5G metrics)

Fig. 8 shows the prediction error in second scenario, which is a CPU overload attack, emulating an infrastructure compromise scenario. Only the CPU prediction error is shown in this scenario, because the rest of the metrics are not affected from this type of attack. The two peaks in the plot show the

start and the end of the CPU overload attack. So, the trained Autoencoder correctly detects the attack and the corresponding anomaly entries are inserted in Influx DB and shown in Grafana UI.

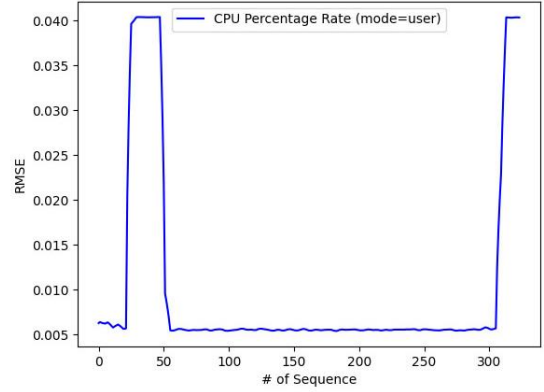


Fig. 8.Prediction Error in CPU Overload Scenario

Finally, in Fig. 9 and Fig. 10 the results of the third scenario are shown, emulating a service compromise scenario (eavesdropping). In Fig.9, the prediction error for edge network metrics are shown and in Fig.10 the prediction error for gNB network metrics is shown. It can be seen that the anomaly has been correctly detected and is temporally aligned with the actual incident.

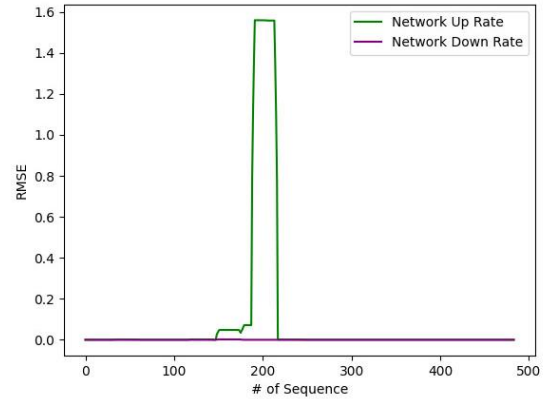


Fig. 9.Prediction Error in Eavesdropping Scenario (Edge metrics)

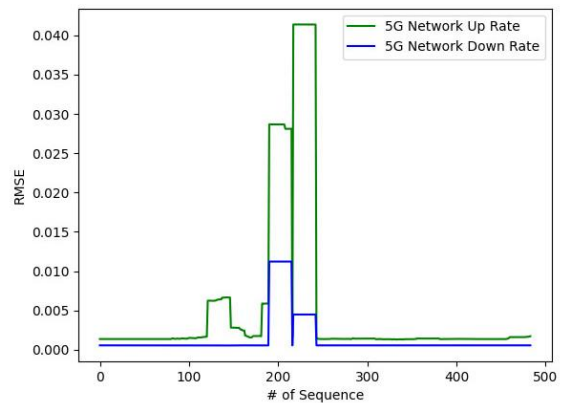


Fig. 10.Prediction Error in Eavesdropping Scenario (5G metrics)

As shown in the figures above, in most cases Autoencoder can

detect normal traffic with small error and very few false positives. In addition, it can detect anomalies in collected metrics values and properly generate the corresponding notifications.

It must be mentioned that the graphs above present a visual evidence of the proper functionality of the algorithm and the pipeline in general under two very relevant and realistic attack scenarios. The calculation of algorithm performance indicators, such as accuracy/precision/recall etc., in order to be actually meaningful, require much more extensive and diverse testing datasets, whose generation will be part of future work.

## V. CONCLUSION AND FUTURE WORK

This paper describes a proposal for an anomaly detection pipeline for 5G infrastructures, which has as its basis LSTMs following the Autoencoder architecture. This pipeline was evaluated in real time in a fully functional 5G network with over-the-air tests. It is shown that it can effectively detect anomalies against normal traffic.

The data collection, anomaly detection and visualisation modules have been released as open-source (<https://github.com/5genesis/Security-Framework>) as part of the 5G experimentation enabler framework ("Open5GENESIS") of the EU H2020 5GENESIS project.

As future steps, we plan to include more metrics from more network elements (including the 5G Core functions), as well as to conduct tests with more types of attacks.

## ACKNOWLEDGMENT

The work described in this paper has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreements No 815178 (5GENESIS) and No 883335 (PALANTIR).

## REFERENCES

- [1] E. Hajlaoui, A. Zaier, A. Khelifi, J. Ghodhbane, M. B. Hamed and L. Sbita, "4G and 5G technologies: A Comparative Study," 2020 5th International Conference on Advanced Technologies for Signal and Image Processing (ATSIP), Sousse, Tunisia, 2020, pp. 1-6, doi: 10.1109/ATSIP49331.2020.9231605.
- [2] L. F. Maimó, F. J. G. Clemente, M. G. Pérez and G. M. Pérez, "On the performance of a deep learning-based anomaly detection system for 5G networks," 2017 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computed, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/CBDCoM/IOP/SCI), San Francisco, CA, USA, 2017, pp. 1-8, doi: 10.1109/UIC-ATC.2017.8397440.
- [3] Marabissi, D. & Mucchi, Lorenzo & Fantacci, R. & Spada, Mariarita & Massimiani, Fabio & Fratini, Andrea & Cau, Giorgio & Yunpeng, Jia & Fedele, Lucio. (2018). A Real Case of Implementation of the Future 5G City. *Future Internet*. 11. 4. 10.3390/fi11010004.
- [4] F. Raissi, S. Yangui and F. Camps, "Autonomous Cars, 5G Mobile Networks and Smart Cities: Beyond the Hype," 2019 IEEE 28th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE), Napoli, Italy, 2019, pp. 180-185, doi: 10.1109/WETICE.2019.00046.
- [5] Szalay, Z., Ficzer, D., Tihanyi, V., Magyar, F., Soós, G., & Varga, P. (2020). 5G-enabled autonomous driving demonstration with a V2X scenario-in-the-loop approach. *Sensors*, 20(24), 7344.
- [6] 5G Security Landscape, June 2017, 5G-PPP, [https://5g-ppp.eu/wp-content/uploads/2014/02/5G-PPP\\_White-Paper\\_Phase-1-Security-Landscape\\_June-2017.pdf](https://5g-ppp.eu/wp-content/uploads/2014/02/5G-PPP_White-Paper_Phase-1-Security-Landscape_June-2017.pdf)
- [7] ENISA Threat Landscape for 5G Networks [Online]. Available: <https://www.enisa.europa.eu/publications/enisa-threat-landscape-for-5g-networks>
- [8] Gers, Felix & Eck, Douglas & Schmidhuber, Jürgen. (2001). Applying LSTM to Time Series Predictable through Time-Window Approaches. 669-676. 10.1007/3-540-44668-0\_93.
- [9] Shunliang Zhang, Yongming Wang, Weihua Zhou, Towards secure 5G networks: A Survey, *Computer Networks*, Volume 162, 2019, 106871, ISSN 1389-1286, <https://doi.org/10.1016/j.comnet.2019.106871>.
- [10] R. Khan, P. Kumar, D. N. K. Jayakody and M. Liyanage, "A Survey on Security and Privacy of 5G Technologies: Potential Solutions, Recent Advancements, and Future Directions," in *IEEE Communications Surveys & Tutorials*, vol. 22, no. 1, pp. 196-248, Firstquarter 2020, doi: 10.1109/COMST.2019.2933899.
- [11] Haider, Noman, Muhammad Zeeshan Baig, and Muhammad Imran. "Artificial Intelligence and Machine Learning in 5G Network Security: Opportunities, advantages, and future research trends." *arXiv preprint arXiv:2007.04490* (2020).
- [12] Lam, Jordan, and Robert Abbas. "Machine learning based anomaly detection for 5g networks." *arXiv preprint arXiv:2003.03474* (2020).
- [13] Prometheus, from metrics to insight, <https://prometheus.io/>
- [14] InfluxDB: Purpose-built, open-source time-series database, <https://www.influxdata.com/>
- [15] Prometheus Node Exporter, [https://github.com/prometheus/node\\_exporter](https://github.com/prometheus/node_exporter)
- [16] Grafana, the open observability platform, <https://grafana.com/>