Real-Time Intrusion Detection via Machine Learning Approaches

Erik Murtaj[1], Fausto Marcantoni[1], Michele Loreti[1], Michela Quadrini[1,*] and Hans-Friedrich Witschel[2]

[1]School of Science and Technology, University of Camerino, Via Madonna delle Carceri, 9, Camerino, 62032, Italy
[2]FHNW University of Applied Sciences and Arts Northwestern Switzerland, Riggenbachstrasse 16, CH-4600 Olten

Abstract

In many cybersecurity contexts, the real-time detections of hostile actions play a fundamental role in protecting network infrastructures. In this scenario, Intrusion Detection Systems (IDS), based on signature-based or anomaly detection, are widely used to analyze network traffic. The signature-based detection relies on databases of known attack signatures, and anomaly detection is mainly based on Artificial Intelligence (AI) techniques. The latter is promising to detect new kinds of cyberattacks in real time.

In this work, we propose ReTiNA-IDS, a framework that integrates the CICFlowmeter tool with Machine Learning techniques to analyze Real-Time network traffic patterns and detect abnormalities that may suggest a possible intrusion. The considered machine learning techniques, random forest and multi-layer network, are based on selected features to enhance efficiency and scalability. To select the features and train the models, we use a version of the public dataset, CSECICI-IDS2018. The framework's effectiveness has been tested in real-case scenarios by identifying different forms of intrusion. Analyzing the results, we conclude that the proposed solution shows valuable features.

Keywords

Random Forest, Feature Selection, analysis of Real-Time network traffic, Intrusion Detection Systems

1. Introduction

Intrusion Detection Systems (IDS) are relevant tools employed in cybersecurity to protect networks from possible cyber attacks.

In recent years, the world of cyber security has become more turbulent, with a rise in the number of cyber-attacks that target businesses worldwide. For this reason, always new methodologies are needed to shield vital assets from hostile actors in reaction to this expanding danger.

Recently, an increasing focus on the use of Artificial Intelligence (AI) in cyber security. As a subset of artificial intelligence, machine learning algorithms can improve danger detection and automate procedures. Organizations may examine massive volumes of data in real-time, spot patterns suggestive of malicious behaviour, and take preemptive measures to reduce risks by utilizing machine learning algorithms.

In this work, we propose ReTiNA-IDS, a framework that integrates the CICFlowmeter tool with Machine Learning techniques to analyze real-time network traffic patterns and detect abnormalities that may suggest a possible intrusion. The integrated methodology, which is based on random forest and multi-layer networks, is based on selected features to enhance efficiency and scalability. To select the features and train the models, the public dataset CSECICI-IDS2018 has been used. The framework's effectiveness has been tested in real-case scenarios by identifying different forms of intrusion. Analyzing the results, we conclude that the proposed solution shows valuable features.

The paper is structured as follows. In Section 2 related works are discussed while in Section 3 some basic background is introduced. In Section 4 the tool ReTiNa-IDS is presented, while in Section 5 some evaluation experiments are proposed. Section 6 concludes the paper.

2. Related Works

The use of machine learning approaches in intrusion detection systems to obtain real-time analysis has been exploited by many researchers. Many of them take advantage of Deep Learning (DL) approaches. ARCADE is an unsupervised DL-based approach for early anomaly detection using 1D Convolutional Neural Networks (CNNs) proposed by Lunardi et al. [1]. The approach builds a profile of normal traffic based on raw packet bytes. Kathareios et al. designed and tested a real-time net-

Ital-IA 2024: 4th National Conference on Artificial Intelligence, organized by CINI, May 29-30, 2024, Naples, Italy
*Michele Loreti
†These authors contributed equally.
erik.murtaj@studenti.unicam.it (E. Murtaj); fausto.marcantoni@unicam.it (F. Marcantoni); michele.loreti@unicam.it (M. Loreti); michela.quadrini@unicam.it (M. Quadrini); hansfriedrich.witschel@fhnw.ch (H. Witschel)
0000-0002-7779-203X (F. Marcantoni); 0000-0003-3061-863X (M. Loreti); 0000-0003-0539-0290 (M. Quadrini); 0000-0002-8608-9039 (H. Witschel)

work AD system, able to operate on encrypted and non-encrypted network packets, based on two learning stages: an autoencoder for adaptive unsupervised AD and a custom nearest-neighbour classifier to filter false positives [2]. Shuai proposed a prototype that combines big data processing frameworks like Apache Hadoop, Apache Kafka, and Apache Storm, along with ML techniques, i.e., Naïve Bayesian (NB), Support Vector Machine (SVM), and Decision Tree (DT). The proposed approach considers six features related to the IP addresses of the sender, receiver, and correspondent port without taking into account flow measurements. Ho et al. suggested an Intrusion Detection System (IDS) based on CNN that classifies all packet traffic as benign or malicious, detecting network intrusions [3]. Atefnia and Ahmadi proposed a modular deep neural network model that consists of four complete architectures that are combined with an aggregator module, each generating distinct outputs [4]. The four architectures are a Deep Feed-Forward Module (DFFM), a Stacked Restricted Boltzmann Machine Module (SRBMM), and two recurrent modules, one utilizing gated recurrent units (GRUM) and the other utilizing long short-term memory (LSTMM). Catillo et al. [5] proposed an approach based on Deep Autoencoder, and Fitni and Ramli [6] proposed a model based on decision trees that takes into account 23 features selected by Spearman's rank correlation coefficient [7]. Gamage and Samarabandu considered four DL architectures, i.e., feed-forward neural network, autoencoder, deep belief network, and LSTM [8]. Karatas et al. in [9] reviewed the implementation of a Synthetic Minority Oversampling Technique (SMOTE) [10] to balance the data by exploiting six models. Kanimozhi and Jacob presented a two-layer MLP to detect only botnet attacks that exploit a grid search for hyper-parameter optimization and a 10-fold cross-validation for mitigating the overfitting problems [11]. Huancayo Ramos et al. extended this approach by considering botnet data and Random Forests. Kim et al. also designed a model that exploits CNN for training on a single type of attack, specifically Denial of Service (DoS) attacks [12].

3. Background

In this section, we present the CICFlowMeter, an Ethernet traffic Bi-flow generator and analyzer for anomaly detection, and the Random Forest, a machine learning method used for classifying flow data and evaluating the importance of features. This classifier will then be integrated into CICFlowMeter for classifying network flows.

3.1. CICFlowMeter

CICFlowmeter is a network traffic flow generator and analyser [13, 14]. It generates bidirectional flows, where the first packet determines the forward (source to destination) and backward (destination to source) directions. The tool enables the extraction of more than 80 statistical network traffic features such as Duration, Number of packets, Number of bytes, Length of packets, etc. Such features can be calculated independently for both directions. The tool is developed in JAVA and provides a useful Graphical User Interface, shown in Figure 1 to monitor network flows in real-time. TCP flows are usually terminated upon connection teardown (by FINpacket), while a flow timeout terminates UDP flows [15].

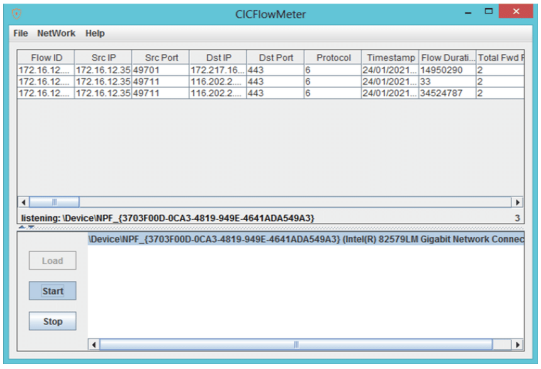

Figure 1: Example of the CICFlowmeter's GUI

The tool is developed in JAVA and provides a useful GUI (Graphical User Interface) to monitor network flows in real time.

3.2. Machine Learning Approaches and Feature Selection

The Random Forest is an ML ensemble model used for both classification and regression tasks. During training, the model creates numerous decision trees and determines the output class by either the mode (for classification) or the mean/average prediction (for regression) of the classes predicted by individual trees. Introduced by Breiman in [16], this approach combines the bagging technique with the random selection of features. Such a random selection ensures that the decision trees within the forest are uncorrelated. In the bagging phase, decision trees are constructed from bootstrap samples of the training dataset, where each sample is drawn with replacement, allowing for the possibility of repeated samples. These replicated datasets are then used to train decision trees, ensuring that each tree only sees different portions of the original dataset during training. This bagging approach is coupled with random feature selection,

which involves using distinct random subsets of the entire feature space to train each tree in the random forest. Usually, around $\sqrt{n}$ features are employed in each split for a classification task that considers $'n'$ features.

3.3. Dataset: CSE-CIC-IDS2018

The data used in this study is the CSE-CIC-IDS2018, a benchmark dataset for the evaluation of IDSs. Such data was collected by the Communications Security Establishment (CSE) and the Canadian Institute for Cybersecurity (CIC). The recorded data consists of ten days of traffic and includes seven types of attacks. Liu et al. identified some issues in such dataset related to the creation lifecycle, including attack orchestration, feature generation, documentation, and labelling and provided to reconstruct the datasets by deleting artefacts and corrected labelling logic, including corrected implementations of existing features and new features that capture valuable flow state information [17]. Table 1 reports the corrupt amount of data.

Attack Type	Corruption Rate (%)
Bot	50.06
Web - Brute Force	53.85
Web Attack - XSS	50.43
DoS Attacks	>50
DDoS Attacks	>50
FTP-Patator	100.00
Infiltration	76.84
SQL Injection	54.02
SSH-Patator	49.97

Table 1
Corruption Rate of Different Attacks on the CSE-CIC-IDS 2018 dataset [17]

3.4. Metrics

We evaluate the performance and effectiveness of the approaches by using Precision (P), Recall (R) and , defined as follows

$$P = \frac{TP}{TP + FP}$$

$$R = \frac{TP}{TP + FN}$$

$$F_1 = 2\frac{P \cdot Q}{P + Q}$$

where TP represents the number of true positive, FN denotes the number of false negative, FP represents the number of false positive, TN denotes the number of true negative.

Table 2
The first 13 attributes ordered by importance

Id	Attribute	Description
1	FWD Init Win Bytes	The total number of bytes sent in initial window in the forward direction
2	Packet Length Std	Standard deviation length of a packet
3	Packet Length Mean	Mean length of a packet
4	Bwd Packet Length Std	Standard deviation size of packet in backward direction
5	Bwd Packet Length Max	Maximum size of packet in backward direction
6	Bwd PSH Flags	Number of times the PSH flag was set in packets travelling in the backward direction
7	ACK Flag Count	Number of packets with ACK
8	Fwd Seg Size Min	Minimum segment size observed in the forward direction
9	Fwd PSH Flags	Number of times the PSH flag was set in packets travelling in the forward direction
10	CWR Flag Count	Number of packets with CWR
11	Packet Length Variance	Variance length of a packet
12	Fwd Packet Length Max	Maximum size of packet in forward direction
13	Bwd Packet Length Mean	Mean size of packet in backward direction

4. ReTiNA-IDS Approach

ReTiNA-IDS, **Re**al-**Ti**me a**n**omaly Detection IDS **A**pproach, integrates a ML model mainly based on Random Forest in the CICFlowMeter tool to detect Real-Time cyber-attacks and act as a simple IDS. The Random Forest classifier considers only 13 of the 80 features calculated by the CICFlowMeter tool. The list of features with the relative description, selected by another Random Forest model, is in Table 2. After being trained, the model has been exported in a *pmml* format with the use of the *"sklearn-pmml-model"* library from Sklearn [18]. The exported model is then imported into CICFlowMeter, which is developed in Java.

4.1. ML Pipeline

The proposed approach is based on Random Forest, described in Section 3.2, and its scheme is shown in Figure 2.

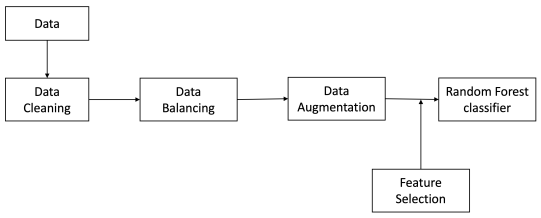

Figure 2: Pipeline of our Approach

4.1.1. Data Preprocessing

In this study, the used dataset is a revised version of CSE-CIC-IDS2018, as introduced in Section 3.3. The dataset consists of the network traffic captured on ten days, stored in 10 distinct files according to the day of data capture, as shown in Table 3.

Table 3
CSE-CIC-IDS2018 files

Id	File Name	Size
1	Wednesday-14-02-2018	3.03 GB
2	Thursday-15-02-2018	2.18 GB
3	Friday-16-02-2018	3.92 GB
4	Tuesday-20-02-2018	3.19 GB
5	Wednesday-21-02-2018	3.68 GB
6	Thursday-22-02-2018	3.23 GB
7	Friday-23-02-2018	3.17 GB
8	Wednesday-28-02-2018	3.54 GB
9	Thursday-01-03-2018	3.54 GB
10	Friday-02-03-2018	3.43 GB

The first step of the preprocessing consists of data cleaning, i.e., removing missing values, such as incomplete rows, and containing invalid (or infinite) numerical values. Moreover, many non-relevant features for spotting cyber-attacks have been eliminated, such as the IP address of the sender and receiver, the connection timestamp, the protocol type, and the destination/sender port. Furthermore, the traffic data related to Web Attacks is deleted since its volume is insufficient.

4.1.2. Data Balancing and Data Augmentation

The collected data related to network traffic is substantially unbalanced: benign traffic is more prevalent than malicious traffic. To balance the data, we have used the one step of the bootstrapping procedure, implemented in the *resample function* of Sklearn. Due to the corrupted data on the original dataset, it does not contain data related to FTP Brute Force attacks. Therefore, we have added this kind of data by collecting such data during a simulation of brute force attacks via FTP (File Transfer Protocol). The simulation involved the use of a Windows host (victim machine) and a Kali-Linux host (attacker machine), both in the same local area network (connected to the same router). The victim machine runs a FileZilla server, an open-source software utility that facilitates the transmission of files using the File FTP. It enables users to establish their own FTP servers or connect to existing FTP servers to exchange data, and the victim machine accepts connections on port 21, used to attack. When the FileZilla server on the victim machine is running, the Kali Linux host performs a brute-force attack using Patator, a multi-purpose brute-forcer tool [19]. Table 4 shows the amount of data and the relative kind of attack, after the cleaning and balancing phases.

4.1.3. Feature Selection and Classifier

To select the features, a Random Forest has been considered and implemented by setting up the depth of each decision tree and number of estimators to 16 and 20,

Table 4
Amount data per network traffic class

Class	Count
BENIGN	145904
DoS Attack	145904
BruteForce Attack	99147
PortScan Attack	49740
BotNet Attack	142921
Total	583.616

Table 5
Classification Performance Metrics Random Forest

Class	Precision	Recall	F1-score
BENIGN	1.00	1.00	1.00
Botnet Ares	1.00	1.00	1.00
BruteForce Attack	1.00	1.00	1.00
DoS Attack	1.00	1.00	1.00
Infiltration - NMAP Portscan	0.99	1.00	1.00
Accuracy		1.00	

respectively. To avoid eventually issue related to overfitting, we consider the cross-validation with 5-fold. Figure 3 shows the obtained confusion matrix.

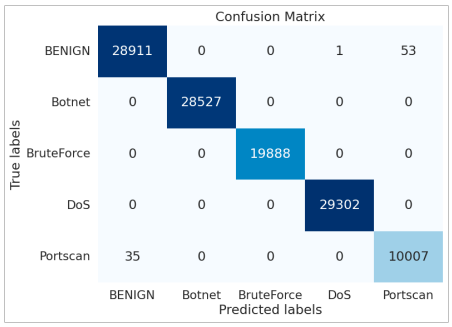

Figure 3: Confusion Matrix of the Random Forest Classifier

The performance of the model, evaluated in terms of Precision, Recall and F_1-score, is shown in the Table 5.

5. Experimental Setup

The ML models have been implemented in a Google Colab document with Python 3. The default CPU in the environment is an Intel Xeon CPU equipped with 2 virtual CPUs (vCPUs) and 13GB of RAM [20]. For this study, the configuration involved the utilization of extra RAM, resulting in a total memory capacity of 50GB (included with Google Colab Pro [20]).

For data handling, preprocessing, analysis, training, and evaluation metrics, the recommended model was built and evaluated using Numpy [21], Pandas [22], and Scikit Learn [23]. Matplotlib [24] were used to visualize the data. The testing phase for this study used a Windows operating system for the with the following specifications: an Intel Core i5-4670 CPU at 3.40GHz, 16 GB of DDR4 memory and a Nvidia GTX 1050 Ti GPU.

5.1. Testing

Retina-IDS, a tool that integrates an ML model into CICFlowMeter, analyzes data patterns and distinguishes benign traffic from malicious traffic. The testing phase of ReTiNA-IDS intends to assess the efficiency and efficacy of the machine learning model in real-world network situations. We take advantage of the Graphical Network Simulator-3 (GNS3) software, an open-source network simulation tool used for creating, modelling, and testing virtual and real networks [25], to perform the simulations. To reach the aim, we create a simple network composed of a Cisco router [26] and two generic switches, outlining two different areas of a hypothetical Local Area Network (LAN), a Windows machine and a Kali Linux machine. Figure 4shows the network infrastructure.

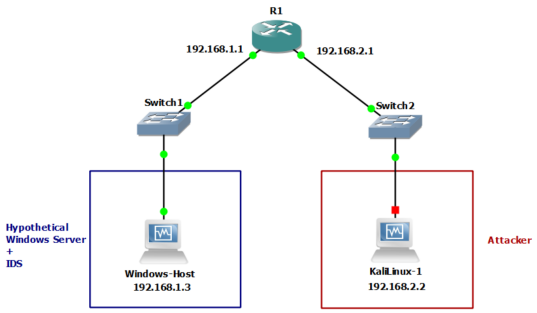

Figure 4: Network structure in GNS3 for testing simulations

The Windows machine represents the hypothetical victim running the Retina-IDS tool, acting as an IDS, while the Kali Linux machine plays the role of attacker. The victim machine is a Windows 10 host, while the used Kali Linux version is *Kali 2023.4*. Instead, the victim machine is a Windows 10 host.

Different attack simulations were performed, each one resulting in a positive detection by the tool:

- DoS attacks
- File Transfer Protocol (FTP) and Secure SHell (SHH) Bruteforce attacks
- Portscan attacks

Additionally, more tests were performed with the tool running in a normal traffic situation (without performing any cyberattack) in a local network and in the University of Camerino's network, for a total of around 5 hours of workload. The purpose of letting the tool run for hours on end was to see whether any crashes occurred during execution and to spot any false positive results. During the experiments zero false positives were identified.

6. Conclusion and Future Work

In this work, we have presented ReTiNA-IDS, a tool that integrates an ML model into CICFlowMeter, which analyzes data patterns and distinguishes benign traffic from malicious traffic in real-time. The ML model is based on a Random Forest, used to select features and to classify the data. The testing phase, performed by running the tool in a normal traffic situation (without performing any cyberattack) in a local network and the University of Camerino's network, shows that the tool does not identify false positives.

In the near future, we intend to test the approach in botnet traffic to investigate the performance of the ReTiNA-IDS. To reach this aim, we intend to create a central server to control potentially infected hosts. Moreover, we have planned to consider other machine learning models, both supervised and unsupervised. Moreover, motivated by the results obtained for modelling and verifying properties of Collective Adaptive Systems [27, 28, 29], we intend to define formal approaches to specify and verify properties of the data traffic to monitor the traffic and identify anomalous pattern in the traffic.

Acknowledgements. This work has been funded by the European Union - NextGenerationEU under the Italian Ministry of University and Research (MUR) National Innovation Ecosystem grant ECS00000041 - VITALITY - CUP J13C22000430001

References

[1] W. T. Lunardi, M. A. Lopez, J.-P. Giacalone, Arcade: Adversarially regularized convolutional autoencoder for network anomaly detection, IEEE Transactions on Network and Service Management (2022).

[2] G. Kathareios, A. Anghel, A. Mate, R. Clauberg, M. Gusat, Catch it if you can: Real-time network anomaly detection with low false alarm rates, in: 2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA), IEEE, 2017, pp. 924–929.

[3] S. Ho, S. Al Jufout, K. Dajani, M. Mozumdar, A novel intrusion detection model for detecting known and innovative cyberattacks using convolutional neural network, IEEE Open Journal of the Computer Society 2 (2021) 14–25.

[4] R. Atefinia, M. Ahmadi, Network intrusion detection using multi-architectural modular deep neural network, The Journal of Supercomputing 3571–3593 (2020).

[5] M. Catillo, M. Rak, U. Villano, 2l-zed-ids: A two-level anomaly detector for multiple attack classes,

in: Web, Artificial Intelligence and Network Applications: Proceedings of the Workshops of the 34th International Conference on Advanced Information Networking and Applications (WAINA-2020), Springer, 2020, pp. 687–696.

[6] Q. R. S. Fitni, K. Ramli, Implementation of ensemble learning and feature selection for performance improvements in anomaly-based intrusion detection systems, in: 2020 IEEE International Conference on Industry 4.0, Artificial Intelligence, and Communications Technology (IAICT), IEEE, 2020, pp. 118–124.

[7] W. W. Daniel, The spearman rank correlation coefficient, Biostatistics: A Foundation for Analysis in the Health Sciences (1987).

[8] S. Gamage, J. Samarabandu, Deep learning methods in network intrusion detection: A survey and an objective comparison, Journal of Network and Computer Applications 169 (2020) 102767. doi:10.1016/j.jnca.2020.102767.

[9] G. Karatas Baydogmus, O. Demir, O. Sahingoz, Increasing the performance of machine learning-based idss on an imbalanced and up-to-date dataset, IEEE Access PP (2020) 1–1. doi:10.1109/ACCESS.2020.2973219.

[10] B. Jason, Smote for imbalanced classification with python, 2021.

[11] V. Kanimozhi, T. P. Jacob, Artificial intelligence based network intrusion detection with hyper-parameter optimization tuning on the realistic cyber dataset cse-cic-ids2018 using cloud computing, in: 2019 international conference on communication and signal processing (ICCSP), IEEE, 2019, pp. 0033–0036.

[12] J. Kim, J. Kim, H. Kim, M. Shim, E. Choi, Cnn-based network intrusion detection against denial-of-service attacks, Electronics 9 (2020) 916.

[13] A. H. Lashkari, G. D. Gil, M. S. I. Mamun, A. A. Ghorbani, Characterization of tor traffic using time based features, in: International Conference on Information Systems Security and Privacy, volume 2, SciTePress, 2017, pp. 253–262.

[14] G. Draper-Gil, A. H. Lashkari, M. S. I. Mamun, A. A. Ghorbani, Characterization of encrypted and vpn traffic using time-related, in: Proceedings of the 2nd international conference on information systems security and privacy (ICISSP), 2016, pp. 407–414.

[15] U. of New Brunswick | UNB, Applications | research | canadian institute for cybersecurity | unb, 2017. URL: https://www.unb.ca/cic/research/applications.html.

[16] L. Breiman, Random forests, Machine learning 45 (2001) 5–32.

[17] L. Liu, G. Engelen, T. Lynar, D. Essam, W. Joosen, Error prevalence in nids datasets: A case study on cic-ids-2017 and cse-cic-ids-2018, in: 2022 IEEE Conference on Communications and Network Security (CNS), IEEE, 2022, pp. 254–262.

[18] scikit-learn: machine learning in python — scikit-learn 1.4.1 documentation, 2024. URL: https://scikit-learn.org/stable/index.html.

[19] Kali linux tools, patator, 2024. URL: https://www.kali.org/tools/patator/.

[20] Google, Google colab, 2024. URL: https://research.google.com/colaboratory/faq.html.

[21] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, T. E. Oliphant, Array programming with NumPy, Nature 585 (2020) 357–362. URL: https://doi.org/10.1038/s41586-020-2649-2. doi:10.1038/s41586-020-2649-2.

[22] T. pandas development team, pandas-dev/pandas: Pandas, 2020. URL: https://doi.org/10.5281/zenodo.3509134. doi:10.5281/zenodo.3509134.

[23] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay, Scikit-learn: Machine learning in Python, Journal of Machine Learning Research 12 (2011) 2825–2830.

[24] J. D. Hunter, Matplotlib: A 2d graphics environment, Computing in Science & Engineering 9 (2007) 90–95. doi:10.1109/MCSE.2007.55.

[25] S. Worldwide, Gns3 documentation, 2024. URL: https://docs.gns3.com/docs/.

[26] Cisco 3600 series - cisco, 2015. URL: https://www.cisco.com/c/en/us/td/docs/ios/12_2/12_2x/12_2xa/release/notes/rn3600xa.html.

[27] M. Loreti, M. Quadrini, A spatial logic for simplicial models, Log. Methods Comput. Sci. 19 (2023).

[28] N. Del Giudice, L. Matteucci, M. Quadrini, A. Rehman, M. Loreti, Sibilla: A tool for reasoning about collective systems, Science of Computer Programming (2024) 103095.

[29] N. D. Giudice, L. Matteucci, M. Quadrini, A. Rehman, M. Loreti, Sibilla: A tool for reasoning about collective systems, in: Coordination Models and Languages - 24th IFIP WG 6.1 International Conference, COORDINATION 2022, Held as Part of the 17th International Federated Conference on Distributed Computing Techniques, DisCoTec 2022, Lucca, Italy, June 13-17, 2022, Proceedings, 2022, pp. 92–98. doi:10.1007/978-3-031-08143-9_6.