Computer Networks 218 (2022) 109381

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

# A generic learning simulation framework to assess security strategies in cyber-physical production systems

Moussa Koïta [a,*], Youssouf M. Diagana [a], Oumar Y. Maïga [b], Mamadou K. Traore [c]

[a] UFR SFA, Laboratoire Mathématiques-Informatique, University Nangui Abrogoua, Côte d'Ivoire
[b] University of Sciences, Techniques and Technologies of Bamako, Mali
[c] University of Bordeaux, IMS CNRS UMR 5218, France

ARTICLE INFO

*Keywords:*
Cyber-physical production system
Cybersecurity
Denial of service
Modeling and simulation
High-level language for systems specification (HiLLS)
Machine learning
Anylogic

ABSTRACT

Connected systems through computerized networks are at the heart of the Industry of the future. As they merge physical entities with cyber spaces, they fall under the paradigm of cyber-physical production systems. Cybersecurity is a key challenge for such systems, as they are subject to daily attempts of intruders to gain unauthorized access to their internal resources or to compromise their integrity. The fast increase of new attack strategies requires the rapid design and assessment of new defense strategies. It entails a complex, error-prone and time-consuming process, including the clear specification of the attack and defense strategies involved, and the design and implementation of the simulation model allowing to evaluate the performances of the defense strategy. This work intends to make such a process transparent to cybersecurity managers by limiting their workload to the sole specification of the characteristics of the system and the logic of the attack and the defense. It provides a generic hybrid simulation framework for flexible evaluation of cybersecurity policies, which is demonstrated on a SYN flooding application. Therefore, the contribution is twofold: (1) The proposed framework offers a high-level environment allowing various experts to collaborate by graphically modeling a given attack strategy and the envisioned defense strategy, without engaging in heavy implementation efforts. Then the framework's executable infrastructure, which combines simulation with machine learning to understanding the interactions between the attackers & the defender, will allow them assessing the performances of these strategies. The proposed framework differs from state-of-the-art cybersecurity simulation environments in its uniqueness to combining the expressive power of a universal simulation modeling formalism with the user-friendliness of a visual simulation tool. Therefore, it offers at one side, a very high modeling flexibility for easy exploration of various cybersecurity strategies, and at the other side, integrated learning capabilities for allowing self-adaptive user-based cybersecurity strategy design. (2) The application demonstrating the framework focuses on the most encountered and still uncontrolled threats in cybersecurity, i.e. the SYN-Flooding based Denial of Service (DoS) attack. The application targeted is not meant to propose yet another SYN flood detection algorithm or to improve the state-of-the-art in that domain, but to prove the framework operationality. The experimental results obtained showcase the ability of the framework to support learning simulation-based SYN flood defense algorithm design and validation.

## 1. Introduction

As computerized networked systems are gaining tremendous importance with the advent of the Internet of Things (IoT) and the emergence of smart systems (e.g., smart factories, supply chains, grids, buildings, cities, etc.) under the paradigm of cyber-physical production systems (CPPS, i.e., production systems that tightly integrates various physical components with computational components through communication networks), cybersecurity is becoming one of the key challenges for their administrators, managers and the mass of users and objects concerned. The potential impact of cybersecurity weakness in CPPS (towards Industry of the future) ranges from economic damage and loss of production and competitiveness, through injury and loss of life, to catastrophic nation-wide effects.

Nowadays, cybersecurity managers are under the pressure exerted by the massive increase of new attack strategies. Therefore, they need to

* Corresponding author.
*E-mail address:* koita_m70@yahoo.fr (M. Koïta).

https://doi.org/10.1016/j.comnet.2022.109381
Received 6 April 2022; Received in revised form 7 August 2022; Accepted 20 September 2022
Available online 22 September 2022

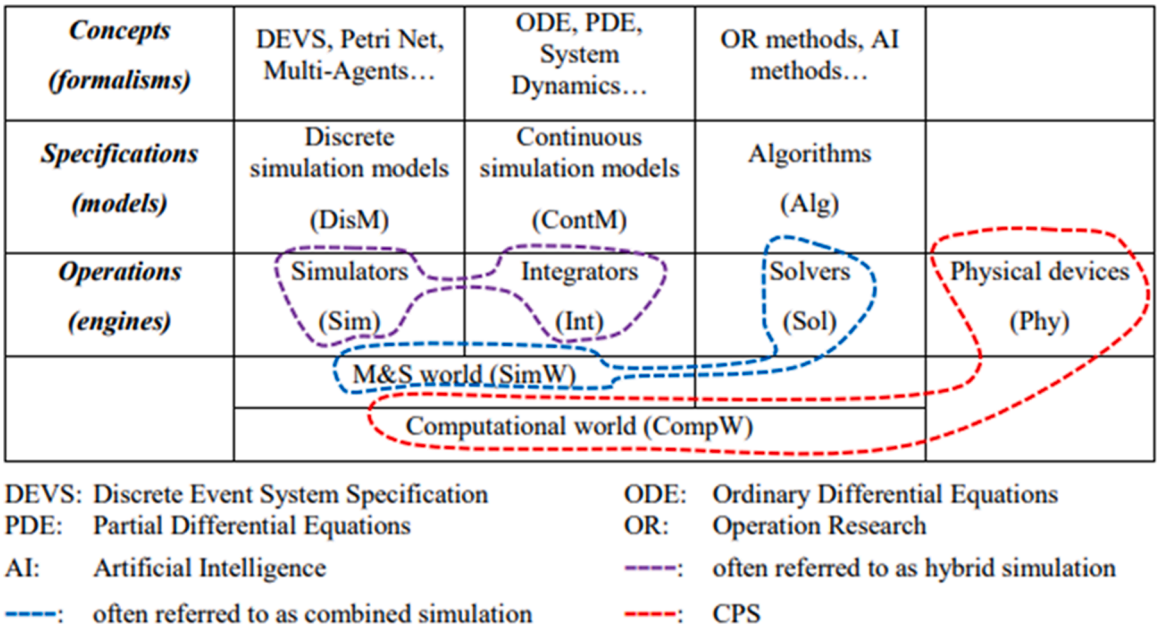


**Fig. 1.** Hybrid simulation approaches.

be more reactive in understanding the scope of a given attack strategy, and developing new defense strategies, the performance of which they must necessarily evaluate. Due to the complexity of interactions between the attackers & the defender, such an evaluation requires the use of modeling and simulation (M&S). In that perspective, one of the major challenges is the seamless communication between network experts (who provide the system knowledge) and M&S experts (who provide the methodological and technical skills to design and implement the required models). In the absence of a user-friendly analysis framework, such a communication is not obvious and its poorness may lead to the failure of the entire study. Moreover, as machine learning (ML) is widely recognized as carrying the promise of improving cybersecurity approaches, such a framework should take advantage of it. Therefore, a third level of expertise enters the scene, which the two previous types of expert involved do not necessarily have access to. It is important the analysis framework integrates learning capabilities in a way transparent to its users, especially when turning the specified attack and defense strategies into an executable simulation model.

Motivated by these needs, this work proposes a generic integrated M&S and ML framework, which facilitates the design and validation of cybersecurity strategies by making it possible to explore from a high level of abstraction, different cybersecurity strategies, using simulation as a means of evaluating the objective function, namely the effectiveness of a defense strategy against a given attack strategy, and ML as a means of guiding the defense algorithm towards optimal efficiency. A SYN flood detection application is used to showcase the applicability and efficiency of the framework. Therefore, the novelty of this work is not to find yet another SYN flood attack detection algorithm, but to offer an operational learning simulation framework for adaptive design and validation of cybersecurity approaches.

The remaining of the paper is organized as follows: Section II discusses related works. Section III presents the framework's underlying principle of hybridizing M&S with ML. Section IV gives details of the framework's structure and components. Section V applies the framework to SYN flooding and shows the experimental results obtained. Section VI concludes the paper, and gives perspectives for future work.

## 2. Related works

Due to the double contribution of the paper, the related works are divided into two sections: the first one relates to state-of-the-art cybersecurity simulation environments, while the second one relates to SYN flood-based DoS defense algorithms.

### 2.1. Cybersecurity simulation environments

Various approaches based on computer networks simulation environments have been proposed to assess the validity of cybersecurity strategies [1–6]. They all call on an in-depth knowledge of the underlying software tools such as OMNeT++ (Objective Modular Network Testbed in C++), NS2 (Network Simulator), etc., including technical skills in programming with advanced languages such as C++, TCL, and the like. A common drawback of these cybersecurity environments is the lack of a methodology layer that separate the domain and system knowledge from the technicity of designing and implementing the corresponding simulation and performance evaluation tool.

Here, we propose a framework to facilitate cooperation between cybersecurity experts and M&S experts through a high-level graphical modeling interface, as well as to save implementation efforts through a systematic model transformation mechanism. The main advantage of such a framework is the reactivity it empowers cybersecurity managers with. Anytime a new threat occurs, network experts can quickly adapt by analyzing the scope of the threat and exploring the efficiency of various defense solutions.

It is worth it to mention that the proposed framework is not dedicated to any specific cybersecurity threat, as its core metamodel suggests a structure where multiple attackers interact with a defender in a network such that the characteristics (i.e., attributes and dynamic behavior) of all stakeholders (attackers, defender, and the network) are user-defined. Moreover, the learning ability of the defender can also be user-defined or left to be the built-in default learning mechanism.

### 2.2. SYN flood-based DoS defense algorithms

Defense strategies against Syn-Flood attacks are numerous in the literature and focus on various aspects, ranging from optimizing the network configuration to improving the infrastructure or the establishment of connections, or adopting a firewall approach. Notable defense strategies are Defensive Programming [7], SYN Cache [8], Hop-Count Filtering [9], or SYN Cookies [10].

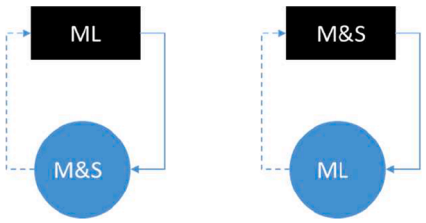


**Fig. 2.** ML-embedded M&S versus Simulation-embedded ML.

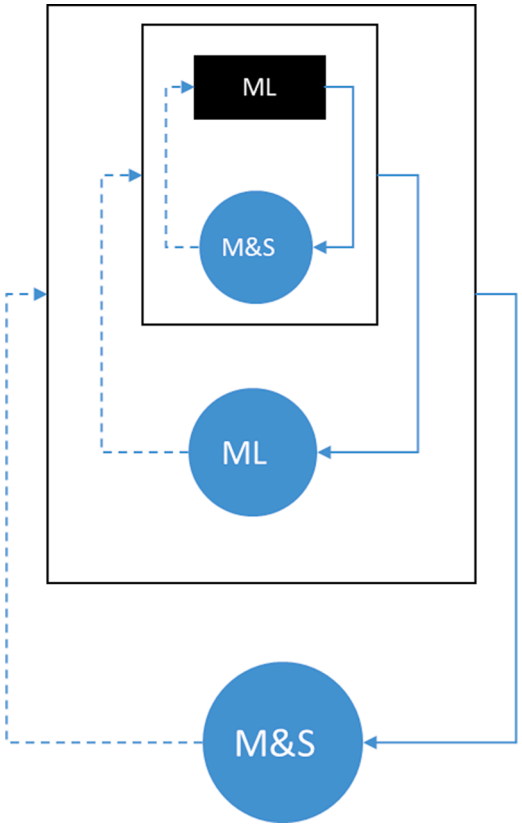


**Fig. 3.** Simulation-ML hybridization.

Recent works considering the advent and increasing security risks of Cyber Physical Systems (CPS), have used ML to improving SYN Flood detection [11–18]. More general threats in CPS not limited to SYN Flooding have been addressed with deep learning (DL) approaches [19]. The survey in [20] presents the current overall view of cyberattacks detection in CPS using DL to fully exploit cyber data.

The application used here to demonstrate the operationality of our framework, focuses on SYN flood (although in this work, we do not intend to propose yet another SYN flood detection algorithm). However, there are no limitations on the type of cybersecurity threat that can be considered.

## 3. Hybridized simulation and machine learning approach

The generic learning simulation framework we propose aims at hybridizing M&S with ML for security hardening in the context of CPPS. The envisioned approach is to allow modeling attack and defense strategies, and then using simulation empowered with ML to study the interactions between the attackers and the defender and to assess the performances of various defense strategies given an attack strategy. In this section, we present the underlying principle of hybridizing M&S with ML.

### 3.1. Level of hybridization

Modern complex systems like CPS require multiple levels of explanation be provided to achieve their various objectives, while keeping a holistic understanding of the behavioral pattern of the overall system and its interaction with the surrounding environment. As such, a hybridization of approaches that would evidently provide useful knowledge from various angles on how such systems perform at the holistic level rather than focusing on specific problems in isolation for specific solutions is an appropriate means to address their complexity. In M&S, such a hybridization can be envisioned endogenously or exogenously, and at different levels of concern.

As described in Fig. 1, the concepts level, where the universe of discourse is set (such as the notions of state, event, concurrency…), calls for formalisms and (more generally) methods to capture the required concepts for symbolic manipulation. While the M&S community traditionally distinguishes between discrete and continuous phenomena as regard to central time-related concepts, qualitative and quantitative computational approaches, such as Operation Research or Artificial Intelligence methods, rather focus on problem-solving steps and mechanisms. Hybridization comes at this level with the objective-driven need to deal with temporal considerations for the system under study while trying to find a solution to the problem under study. Such a situation happens for example when optimization techniques make use of simulation as a black box-type of evaluation function (exogenous hybridization), or when the requirement for a fine-grained understanding of the system entails both continuous and discrete phenomena be considered (endogenous hybridization).

At the specification level, the real-world system and problem under study is expressed as a model, using the universe of concepts adopted, i.e., discrete or continuous simulation model (within M&S world) or problem-solving algorithm (within the wider computational world). The literature has coined various terms to qualify the various possible hybridizations, such as DisM + ContM, or DisM/ContM + Alg (where "+" denotes a composition/mixing operation that can vary from loose to tight integration).

At the operations level, engines are built to execute the model defined at the immediate upper level. Such engines are often referred to in the M&S world as simulators and integrators (for respectively discrete and continuous operations), while solvers implement the algorithms defined in non-M&S-centered computational approaches. Operational hybridization occurs here to support the requirement for multiple execution engines, each devoted to aspects that other engines do not support.

### 3.2. Solution architecture

The framework introduced here realizes hybridization at the formalism level, where the High-Level Language for Systems Specification (HiLLS) is used for simulation modeling, and ML algorithms are integrated to the simulation metamodel defined. We then realized the corresponding operational level hybridization, where the Anylogic software is used as the simulator and the Scikit-Learn package is used as the solver.

The hybridization could be envisioned from one of the two possible perspectives shown by Fig. 2:

(1) ML-embedded M&S, where ML is a black box for M&S, i.e., simulation agents are empowered with learning capabilities, and thus can self-adapt their behavior to the situation during each simulation experiment;
(2) M&S-embedded ML, where M&S is a black box for ML, i.e., the learning algorithm is trained with data provided by simulation, instead of real-world data.

The proposed framework combines both approaches in an

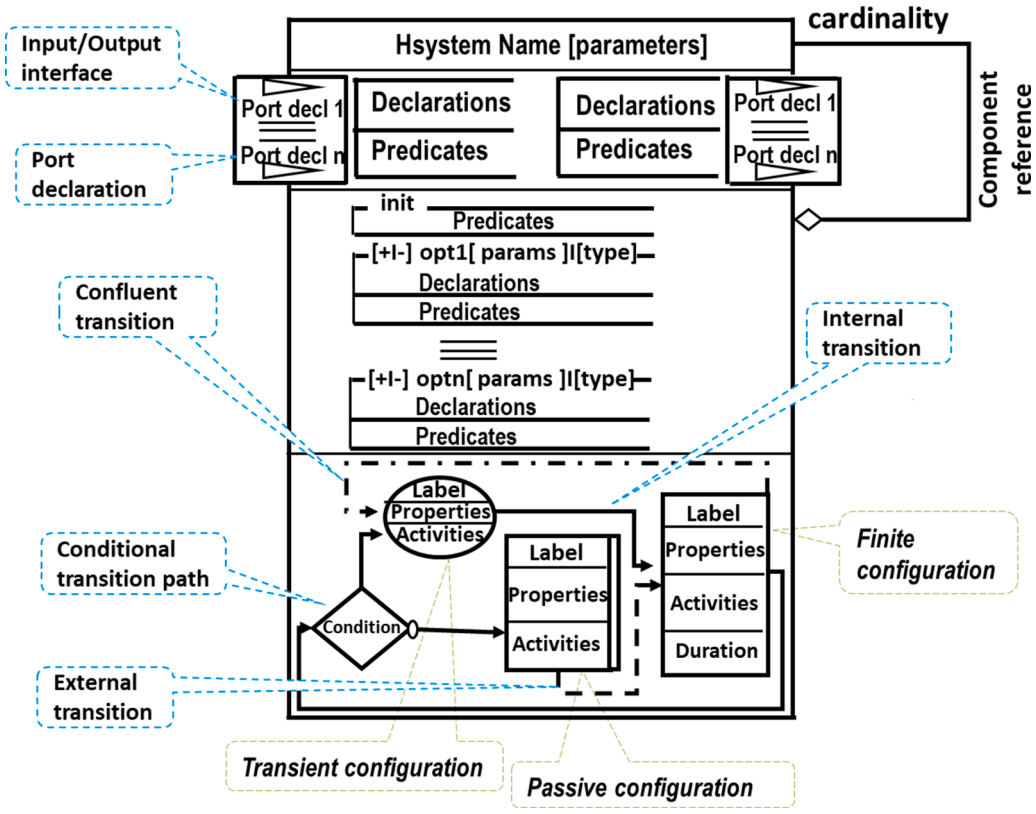


**Fig. 4.** HiLLS modeling elements.

interweaving scheme, as shown by Fig. 3, where an endomorphic integration of ML and M&S is adopted.

In other words, we first design a simulation metamodel (i.e., a generic simulation model, that a user can instantiate to derive a custom simulation model for a specific case study) that embeds ML by construction. We secondly build a training scheme, where the simulation model (which ML component is disabled) is used to train a chosen ML algorithm. The optimized algorithm is then used to enable the ML component embedded in the simulation model, and the resulting model is simulated in various experiments to assess the performances of the trained model versus the ones of the untrained model.

## 4. Framework's structure and components

In this section, we describe in details each of the two legs of our solution, i.e., the M&S (metamodel-based) component and the integrated ML component.

The framework proposed is based on a metamodel, which is specified using the High-level Language for Systems Specification (HiLLS), a graphical modeling formalism amenable to both simulation, formal analysis, and enactment [21]. Therefore, any model instance resulting from this metamodel can be used to:

(1) evaluate the performances of the model over a period of simulated time,
(2) perform rigorous logical investigation of the model's consistency through an exhaustive exploration of the fulfillment of certain given requirements, and
(3) execute the model for real-time verification when interactions with a real-world environment are involved, including humans in the loop.

### 4.1. HiLLS modeling

A dynamic entity is modeled in HiLLS as an HSystem, while a passive entity is modeled as an HClass. An HClass is described exactly the same way a class is described in object-oriented modeling (i.e., as a three-compartments box, where the first compartment shows the name of the class, the second compartment shows the declaration of attributes, and the third compartment shows the declaration of methods), while an HSystem is a four-compartments structure (as shown by Fig. 4) with input/output ports (i.e., interface to interact with its environment by message exchange), name and parameters in the first compartment (to abstract all assumptions on the context in which the model is a representation of the system of interest), state variables in the second compartment (to catch its attributes), operations in the third compartment (to catch its functional capabilities), and the configuration transition diagram in the fourth compartment (to capture its timed behavior). Fig. 4 shows how all elements of an HSystem are represented. Predicate-based schemas are used for the specification of attributes and operations. An HSystem can be composed of other HSystems coupled together. Therefore, HSystems can be hierarchically be composed to form larger HSystems. The composition relationship is represented by the object-oriented aggregation relationship (a line with a diamond shape at the side of the composed system, and the cardinality indicated at the component side to specify the number of such component involved in the composition relationship).

The configuration transition diagram is made up of different types of configuration and different types of transition. A configuration can either be a finite configuration (which is denoted by a four-compartments box, respectively indicating the configuration's label, properties, activities, and duration), or a passive configuration (which is similar to a finite configuration except that a vertical stripe is attached to its right side as an indication of its infinite duration), or a transient configuration (which is similar to a passive configuration except that it is shaped as a circle to indicate that its duration is zero in the simulated

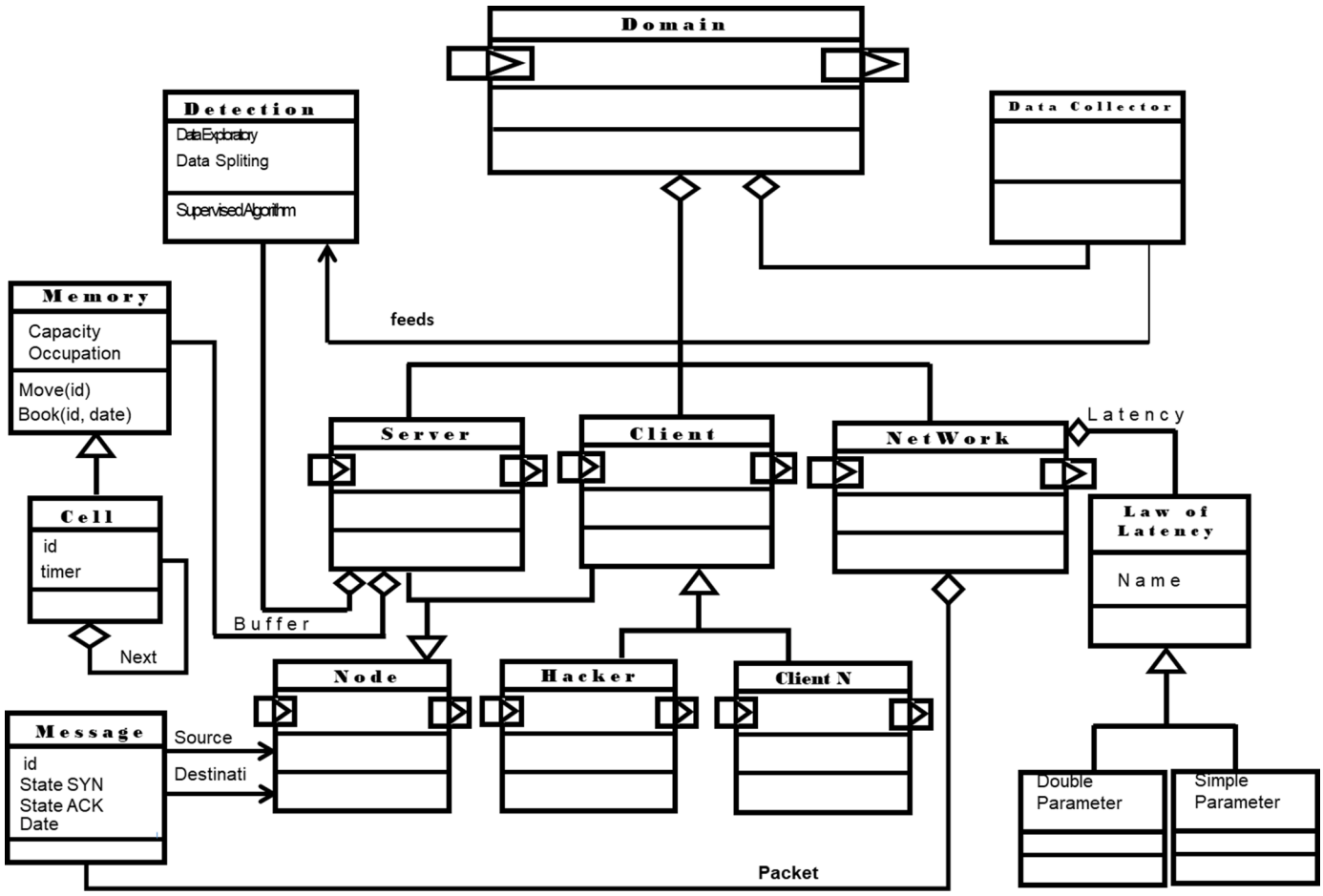


**Fig. 5.** HiLLS-specified simulation metamodel.

**Table I**
Hills to Anylogic transformation rules.

| HiLLS | ANYLOGIC |
|---|---|
| HSystem | Agent without port |
| HClass | Agent with ports |
| Attribute | Agent's variable |
| Operation | Agent's function |
| Input port | Agent's receiving port |
| Output port | Agent's sending port |
| Finite configuration | Agent's state with a delayed transition |
| Transient configuration | Agent's state with a zero delay |
| Passive configuration | Agent's state without delayed transition |
| Internal transition | Transition with associated delay |
| External transition | Transition on receipt of message |
| Confluent transition | Transition with delay and condition |
| Composite HSystem | Agent population |
| HSystems' interaction | Communication link between agents |

time). A transition can either be an internal transition (a straight-line arrow, which indicates the next configuration of the system when the duration of the current configuration elapses without any input received by the system on any of its ports), or an external transition (a dotted-line arrow, which indicates the next configuration of the system when the system receives an input on one or many of its ports before the duration of the current configuration elapses), or a transient transition (a dot-and-straight-line arrow, which indicates the next configuration of the system when the system receives an input on one or many of its ports exactly at the moment the duration of the current configuration elapses). Some transitions can be grouped as a single one, with selection conditions associated (a diamond shape is used to visualize the condition and associated paths).

### 4.2. Framework's core metamodel

The simulation metamodel of the proposed framework is shown by Fig. 5. The entire simulation domain is composed of the server under assessment, multiple clients, and the network linking them (including routers, gateways, etc.). The network's characteristics are aggregated into a law of latency (with simple or double parameters), which can be tuned for simulation experiments under various scenarios. Clients can be normal users or hackers; the behavior of the latter is specified to capture the attack strategy under study. The server's main characteristics are captured by its memory, a set of sequential cells which management is described in the behavior of the server to capture the defense strategy under study. The server also has a learning feature, a detection system that embeds the ML algorithm considered. Both the server and the clients are nodes that exchange messages through the network. The domain's data collector is the component that keeps historical data collected during simulation experiments (therefore, it is the one feeding the ML algorithms).

### 4.3. Framework implementation

The framework proposed is implemented in the Anylogic M&S environment [22]. AnyLogic is a software whose expressive power in M&S is now indisputable. It supports multiple modeling abstractions (agents, processes, differential equations, and automata). It provides a visual notation to automatically generate the Java code for simulation execution. The agent-based paradigm is used to embed and glue all these heterogeneous abstractions through communication links. The Anylogic simulation is enabled with animation features (2D and 3D), and it's possible to incorporate Geographic Information Systems (GIS) [23] as

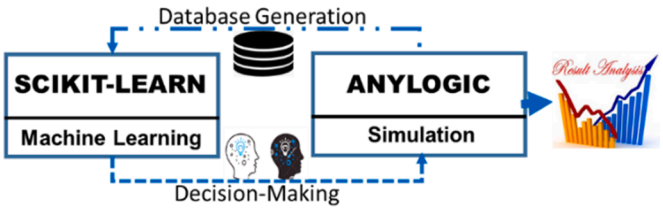


**Fig. 6.** Integration between simulation and machine learning models.

part of the simulation environment.

We implemented the simulation semantics of the HiLLS-specified metamodel of the framework into AnyLogic executables, using transformation rules defined in Table I.

### 4.4. Machine learning integration

The simulation metamodel embeds a learning feature through the ML-based detection system. While the simulation model is carried out with the Java-written AnyLogic software [24], the learning feature is provided by the Python-written Scikit-Learn package that integrates a wide range of advanced ML algorithms for medium-scale supervised and unsupervised problems [25]. The Java-Python interoperability is achieved using Pypline, a library of custom extensions for AnyLogic to connect and communicate with a local installation of Python, which is exportable in the form of jar files. Data generated by the simulation model are stored in an Anylogic database, which the ML algorithm uses for training and decision-making, as captured by Fig. 6. Additionally, other packages like Numpy, Matplotlib, and Pandas are used to perform preprocessing, data analysis, and visualization tasks.

### 4.5. Learning simulation process

When the detection system is deactivated, the system behaves without any learning capability. When the detection system is activated, the learning process requires three steps to be fully operational, as shown by Fig. 7, i.e.:

(1) *Data collection*: in evaluating the performance of the defense strategy against the adopted attack strategy, normal clients and hackers allow the generation of benign and malicious traffic for data collection. Data recorded include the source address of packets, destination address of packets, packet size, and flag and timestamp of all packets sent from the agents.

(2) *Data preparation*: this is a three-stage step, with data preprocessing as the first step, feature selection as the second one, and data splitting as the last one. Preprocessing is an important step, either getting a clean dataset from the raw input data or reducing its dimensionality by retaining the relevant information. It also helps improve efficiency, reduce computation time, and ease the data mining process, depending on the behavior of the data. The main data preprocessing tasks performed are: data cleaning, transformation, normalization and finally the data formatting process. For feature selection, functionalities for each packet are explored according to the domain knowledge, in order to analyze what is relevant to differentiate a normal traffic from an attack. There are static features (such as packet size and interval between packets), and dynamic features (such as packet ratio and frequency of destination address). Finally, data splitting is used to split data into training data set for model fitting, and test data set for final model assessment.

(3) *ML algorithms application*: learning algorithms used in the proposed framework and capable of differentiating normal traffic from attack traffic with great precision include the K-nearest neighbors (KNN), random forests (RF), decision trees (DT), and vector-supported machines (SVM) [26].

(4) *Model evaluation*: after the model is built, its evaluation is called for, to see if it contributes to correctly predicting the target in the context of new data to come. As future instances have unknown target values, we must then generally check the evaluation metrics and especially the precision metric of the ML model on data for which the target response is already known, and then use this evaluation as an indicator of precision predictive of future data. ML provides a standards-compliant precision metric for binary classification models called Area Under the (Receiver Operating Characteristic) Curve (AUC) and returns a decimal value between 0 and 1. This metric measures the fitness of the model to predict a higher score for positive examples compared to negative examples. AUC values close to 1 indicate an ML model that is very accurate. Values close to 0.5 indicate an ML model that is no better than guessing at random. Values close to 0 are unusual and generally indicate a problem with data. Basically, an AUC value close to 0 indicates that the ML model has learned the right trends, but uses them to make predictions reversed from reality (i.e., 0 is predicted as 1, and vice versa).

(5) Implementation for real-time detection.

### 4.6. Framework instantiation process

The HiLLS specifications of the behavior of the major components of the simulation metamodel are given in Annex A. The flexibility of the framework comes from the ease of modification one can bring to any of these specifications. More specifically, it is possible to model any given attack strategy, any given defense strategy, any given network configuration, and any learning algorithm, for the framework to automatically assess the resulting performances, thus allowing a large exploration of

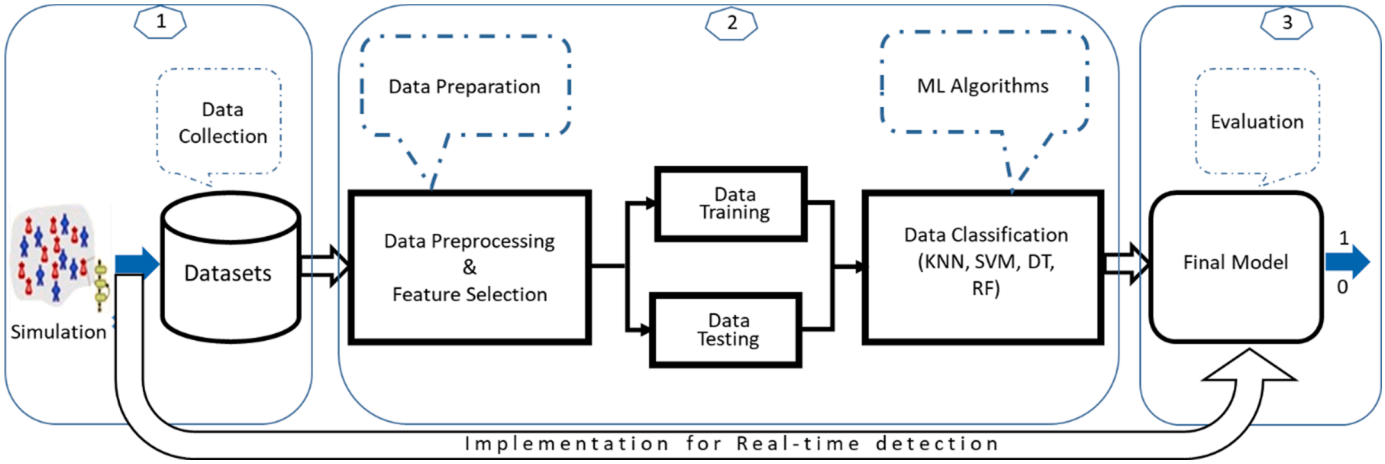


**Fig. 7.** ML integration process.

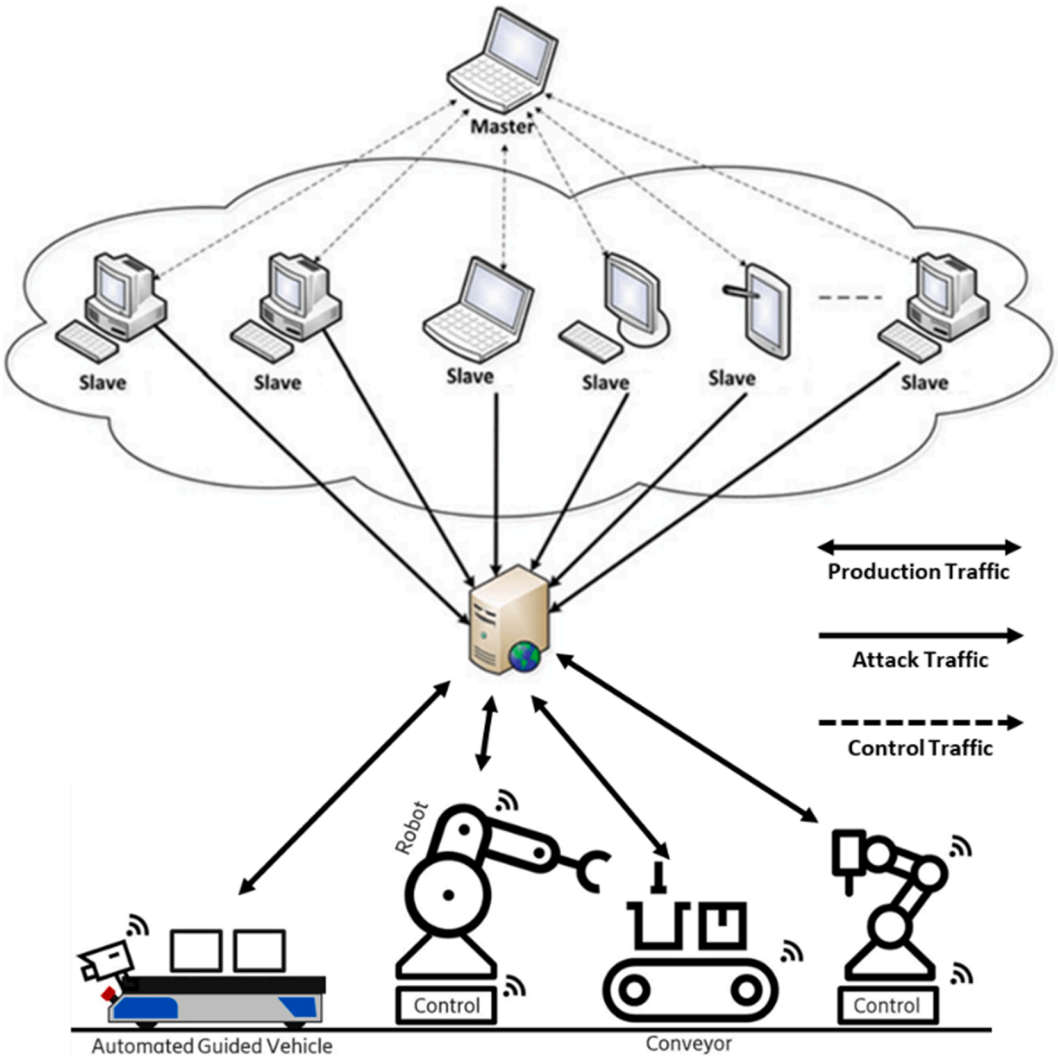


**Fig. 8.** DDoS attack in the context of the Industry of the future.

attack and defense strategies, as well as the observation during simulation experiments of the relationship between hackers and the server's defense. The framework application is achieved in the following eight steps:

(1) Instantiation of the metamodel into a specific model with a given number of normal clients and hackers.
(2) HiLLS specification of the network characteristics (latency) and behavior.
(3) HiLLS specification of the attack strategy through the specification of the behavior of hackers (either by using the default behavior, or by modifying it).
(4) HiLLS specification of the defense strategy through the specification of the characteristics (memory) and the behavior of the server (either by using the default characteristics and behavior, or by modifying them).
(5) Anylogic simulation of this model (with deactivated learning), and evaluation of the performance of the defense strategy against the adopted attack strategy (Data Collector collects the learning data).
(6) Ignition of model learning (model with activated learning on a given algorithm), where Data Collector feeds Detection System with a dataset extracted from the collected data.
(7) Test-based validation of learning, where Data Collector compares the results of Detection System with the rest of the data collected.
(8) Anylogic simulation of this model (with activated learning), and evaluation of the performance of the defense strategy against the adopted attack strategy.

## 5. Application to SYN flooding

In this section, experimental results are presented from the application of our framework to SYN Flood management. The application follows the eight steps previously presented. This section first introduces the principles of SYN Flooding, and then shows how the eight steps of our framework application process can result in improving a basic defense strategy after exploring the potential of some integrated learning capabilities.

### 5.1. SYN flooding denial of service

One of the most common, and yet not completely solved cybersecurity issues is known as Denial of Service (DoS). A DoS attack is a type of single-source attack on a network structure that prevents a server from serving its clients It consists of sending millions of requests to a server, from an invalid or usurped IP address (often called robot, zombie, bot, etc.) in an attempt to slow it down [27], and even to lead it to total inaccessibility. A distributed denial of service (DDoS) attack is a DoS attack variant, in which multiple distributed bots are aggregated as one (and called a botnet), and are used to cause a DoS. Fig. 8 illustrates

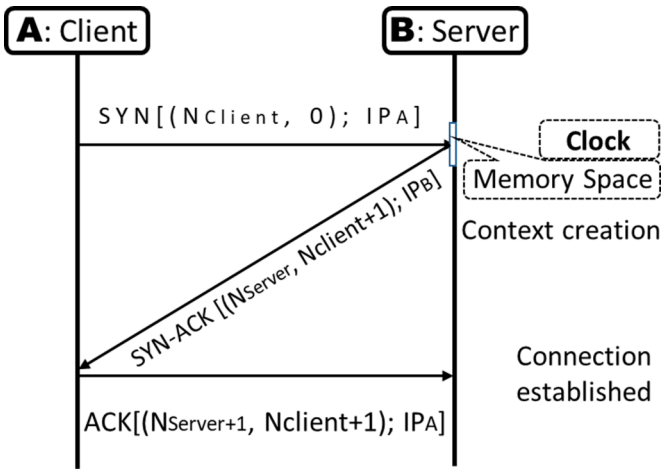


**Fig. 9.** Successful handshaking.

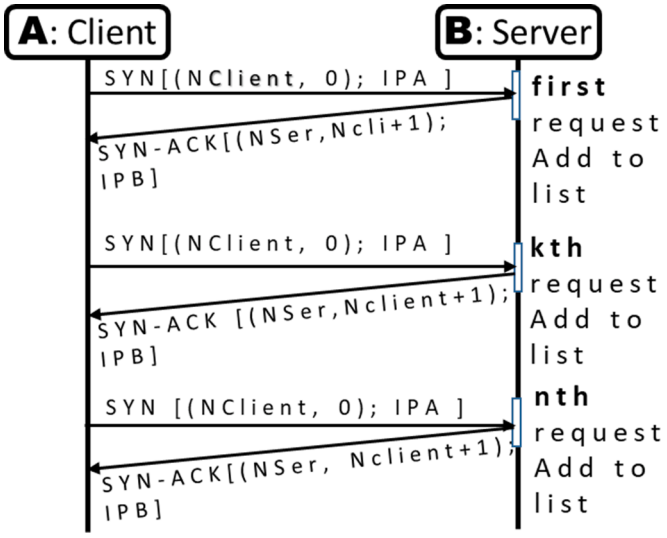


**Fig. 10.** unsuccessful handshaking.

**Table II**
Scenarios explored.

| | % of hackers | % of regular users |
|---|---|---|
| Scenario 1 | 25 | 75 |
| Scenario 2 | 50 | 50 |
| Scenario 3 | 75 | 25 |

the working model of a typical DDoS attack where the botnet is driven by a bot Master, in the context of CPPS. The Master bot takes control of an army of slave bots (without the knowledge of their owners), usually by infecting them with a Trojan or backdoor program, and use them simultaneously to attack a target server using the public Internet infrastructure. Each bot will then engage in a SYN flooding attack, a technique that exploits the vulnerability of the Transmission Control Protocol (TCP) used by Internet [28].

This vulnerability is due to the "Three-way Handshaking" principle of the TCP, which when successful takes place in three stages, as illustrated by Fig. 9:

- During the initialization of a TCP connection between a client and a server, the client sends a SYN message to the server, which is a vector (#seq, ACK flag, SYN flag) where #seq is randomly chosen by the client, and ACK flag = 0, and SYN flag = 1.
- Then, the server allocates memory space for the Transmission Control Block (TCB) and sends a SYN ACK message, a vector (#rseq, #ACK, ACK flag, SYN flag) where #rseq is randomly chosen by the server, and #ACK = #seq+1, and ACK flag = 1, and SYN flag = 1.
- Then, the client sends (#ACK, #rseq + 1, 1, 0). The handshaking ensures that both parties are ready to transmit data to each other.

A bot will ignore the last step and not respond with the ACK message, resulting in an accumulation of unnecessarily allocated memory spaces by the server, as illustrated by Fig. 10. Such accumulation leads to a crash.

Traditionally, a timer is triggered whenever a SYN message is sent. If the SYN ACK response is slow to arrive (> wait time), the connection is dropped and the server releases the memory space allocated for the TCB. Such a strategy, though necessary to avoid the permanent allocation of memory space that will never be used, also generates false positive. Indeed, the delay of receiving a SYN ACK response can be due to the latency of the network while the client is not a bot.

Various defense algorithms have been elaborated against this type of attack, with different performances obtained depending on the network configuration, the server capacity, the frequency of attacks, etc. However, as discussed in this paper, these strategies are built ad-hoc, and there is a lack of methodological and operational framework for performance-based design and exploration of defense strategies, given an attack strategy [29]. This work proposes such a framework, with the possibility for network experts and simulation experts to jointly explore the effectiveness of new defense strategies in a landscape of multiple attack configurations, including the structure and characteristics of the network (such as latency, number of users, etc.), as well as the structure and behavior of the botnet (e.g., variable number of hackers, variable frequency of hacking attempts, variation of the hacking strategy). Such a feature is a very useful decision-support tool for CPPS cybersecurity administrators.

### 5.2. Framework application

We consider a metamodel instantiation with 100 clients, and we consider 3 different scenarios, each corresponding to a different percentage of hackers over the total number of clients, as shown by Table II. At this stage, it is also important to identify the structure of the data set to be collected. Thus, a correlation matrix (see Fig. C.1 in Annex C) has been used to eliminate strongly correlated variables. The only variables we retained to discriminate between a normal client and a hacker are (Inter-packets, Duration, Count_SYN), where Inter-packets is the frequency of packet emission by the client, Duration is the response time to the client's request, and Count_SYN is the number of SYN requests sent by the client.

### 5.3. Specification of the network

We consider a Markovian network's latency (i.e., which distribution follows a Poisson law), with the parameter set to 0.5 seconds. The To feed the Data Collector, the hackers for the DoS traffic, and the normal clients for the normal traffic (non-DoS) interacted for a certain duration in order to constitute a dataset of 36,000 requests. Data is made up of two types, data labeled as normal traffic data and data labeled as SYN Flood attack data, obtained from the header of packets.

### 5.4. Specification of the attack strategy

The HiLLS specification of the Hacker system is given by Fig. A.3 in Annex A. The hacking attempt is tried periodically, where each attempt simply consists in sending out a SYN request, while ignoring any message received back.

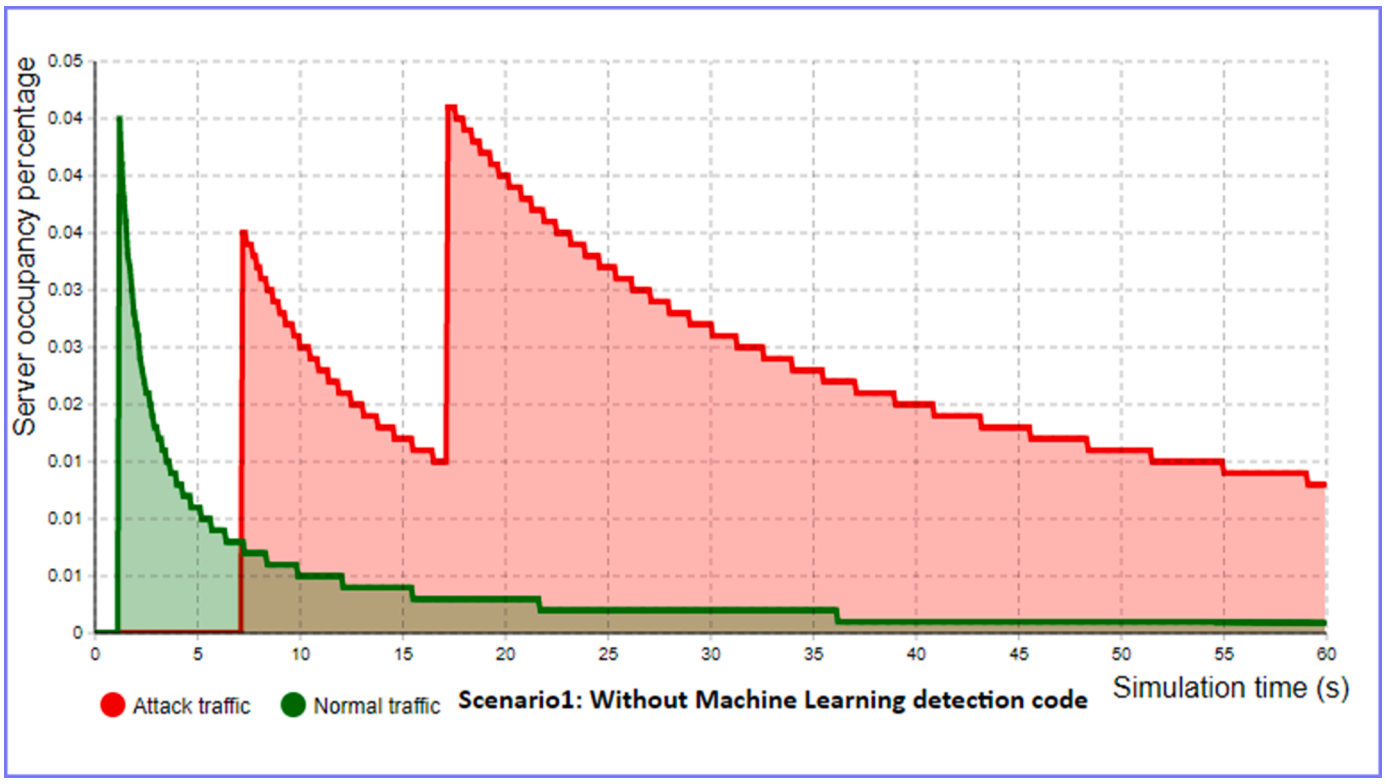


**Fig. 11.** Server occupancy in Scenario1 without ML.

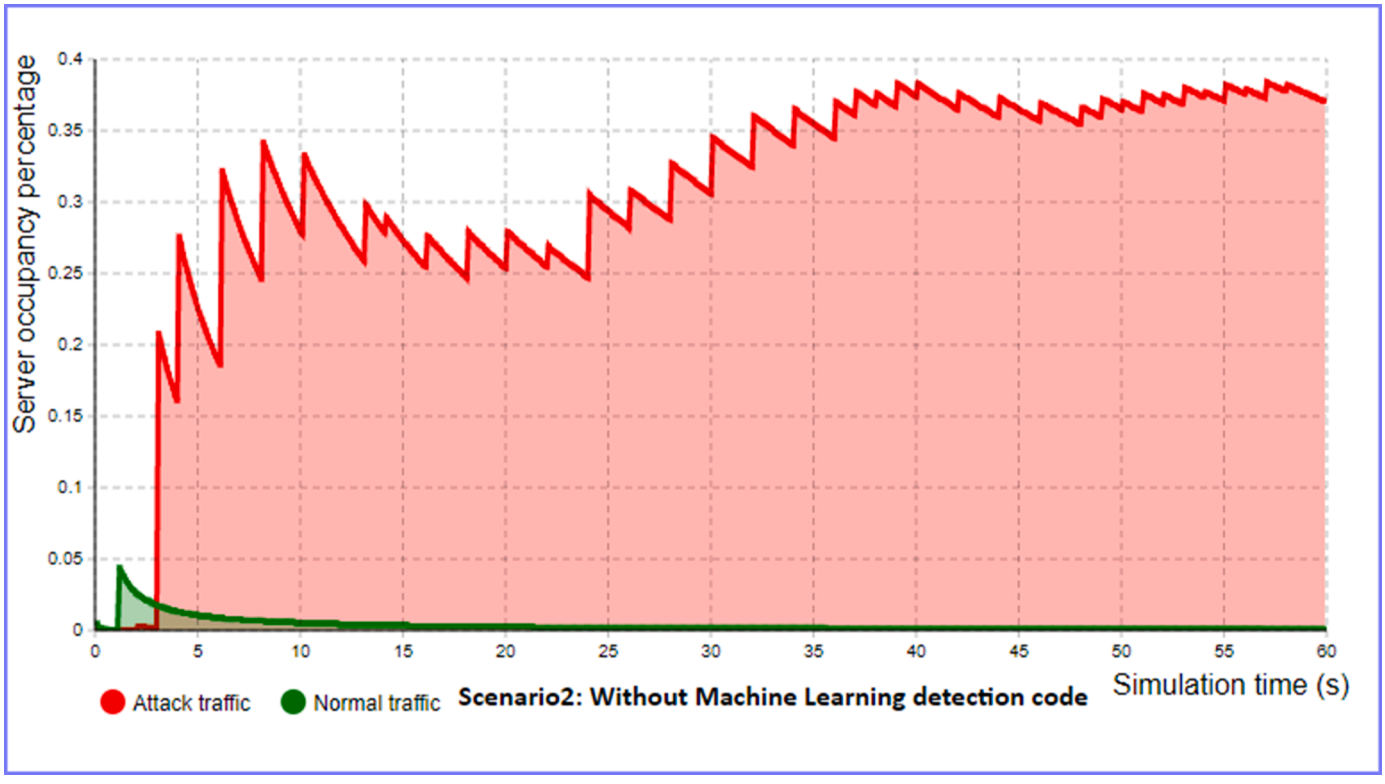


**Fig. 12.** Server occupancy in Scenario2 without ML.

### 5.5. *Specification of the defense strategy*

The HiLLS specification of the Server system is given by Fig. A.3 in Annex A2. The defense strategy relies on the awareness of the server. Each SYN request received is checked, and ignored if identified as a hack or treated otherwise. Each SYN request's treatment books a slot in the memory buffer, as long as it is possible. SYN requests received while the buffer is full are ignored. ACK requests free the buffer. An unaware

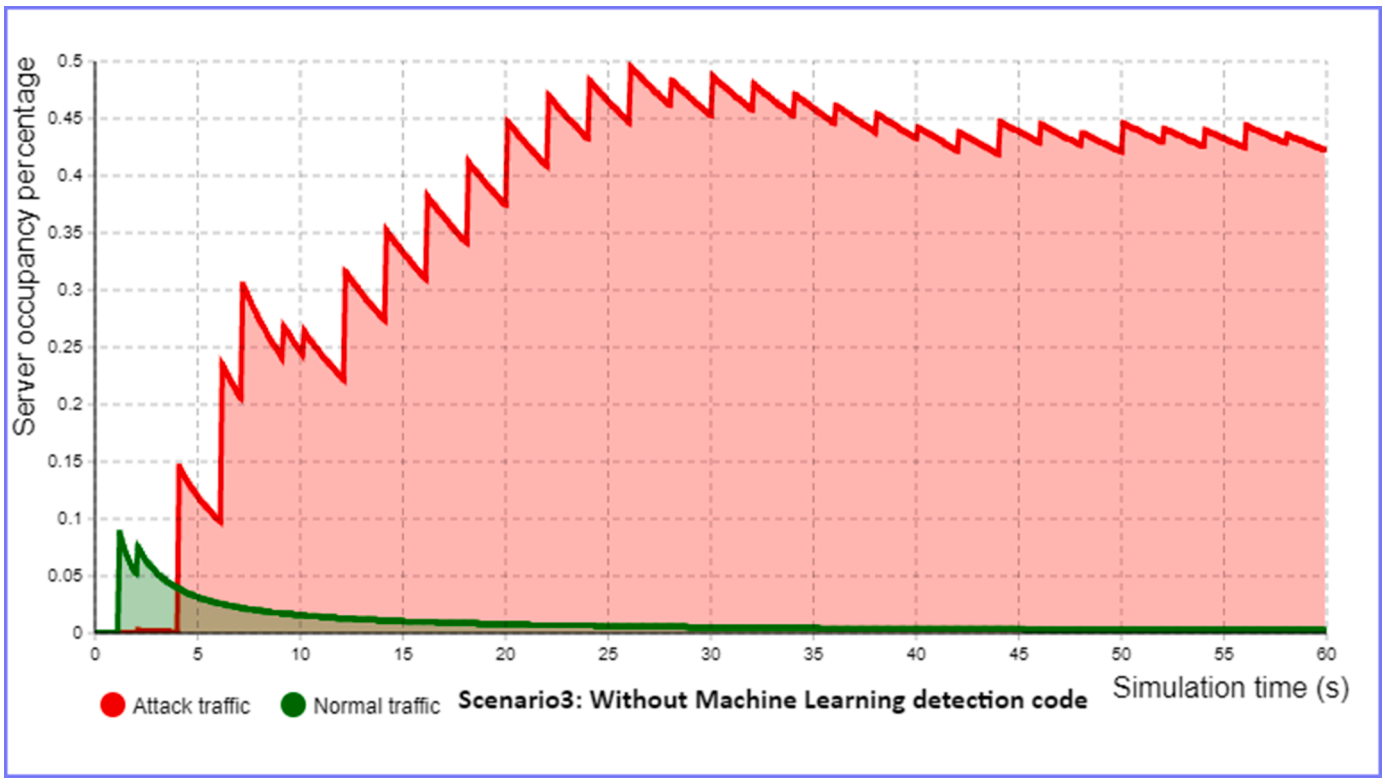


**Fig. 13.** Server occupancy in Scenario3 without ML.

**Table III**
Confusion matrix value.

| | | Predicted Result Positive | Negative |
|---|---|---|---|
| Actual Result | Positive | TP | FN |
| | Negative | FP | TN |

server has no functionality for checking requests, therefore all SYN requests received are treated equally.

### 5.6. Simulation of the unaware model

The simulation is carried out according to the values in accordance with Table II, where PartH and PartN respectively indicate the percentage of hackers and regular users in the population. Metrics evaluated are the percentage of server occupancy due to both regular and attack requests [30]. Figs. 11, 12 and 13 respectively show for Scenario 1, 2 and 3, the timely evolution of the server occupancy (red for what is due to the attack traffic, and green for the normal traffic).

### 5.7. Learning model

For learning purpose, 70% of the entire data set are used for training, and the remaining 30% for testing. The following metrics are used to measure the predictive accuracy of the model, as widely adopted [31]:

- *Confusion matrices*, which are used to evaluate the performance of each classifier. Each classifier makes a binary decision for each SYN flood or normal traffic; so, a confusion matrix ($2 \times 2$) is used as shown in Table III, where TP is True Positive, TN is True Negative, FP is False Positive, and FN is False Negative.
- *Accuracy*, which the correct prediction (positive or negative) made overall, in a classification problem. As a reminder, the effectiveness of the proposed detection module is measured in terms of how accurately it identifies how well it classifies the upcoming packet as normal or attack. The accuracy of the detection approach is calculated using Eq. (1).
- *Precision*, which is used to measure the proportion of positive data instances that a model has classified as positive. The precision metric ignores a model's ability to recognize negative classes. The precision of the detection approach is calculated using Eq. (2).
- *Recall*, which is a proportion of the pattern of true positives that has been identified. As such, a model that gives no FN has a well-being-worth callback. Precision and recall are normally inversely proportional to each other. Which means that if one is improved, the other is degraded. The recall of the detection approach is calculated using Eq. (3).
- *False Positive Rate* (FPR, i.e., rate of normal clients wrongly identified as hackers), which is minimized by a high precision.
- *False Negative Rate* (FNR, i.e., rate of hackers wrongly identified as normal clients), which is minimized by a high recall.
- *True Positive Rate* (TPR), which measures the classifier's ability to correctly classify test data.
- *Receiver Operating Characteristics* (ROC), which curve is a graph of FPR versus TPR.

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{FN} + \text{TN}} \tag{1}$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{2}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{3}$$

### 5.8. Learning validation

We set up a reproducible scoring system which allowed us to test a series of learning algorithms by evaluating their performance. Since detection issues can often be tagged, supervised ML techniques are

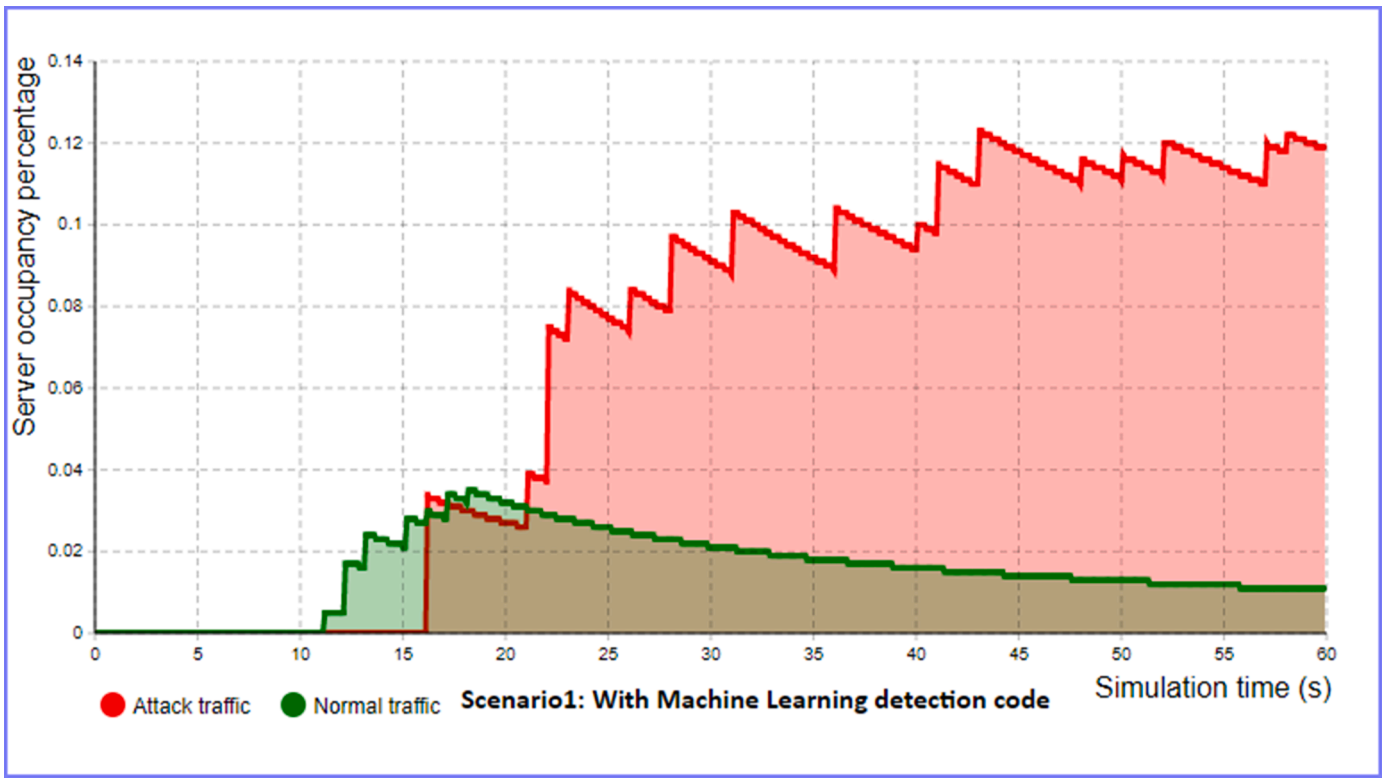


**Fig. 14.** Server occupancy in Scenario1 with ML.

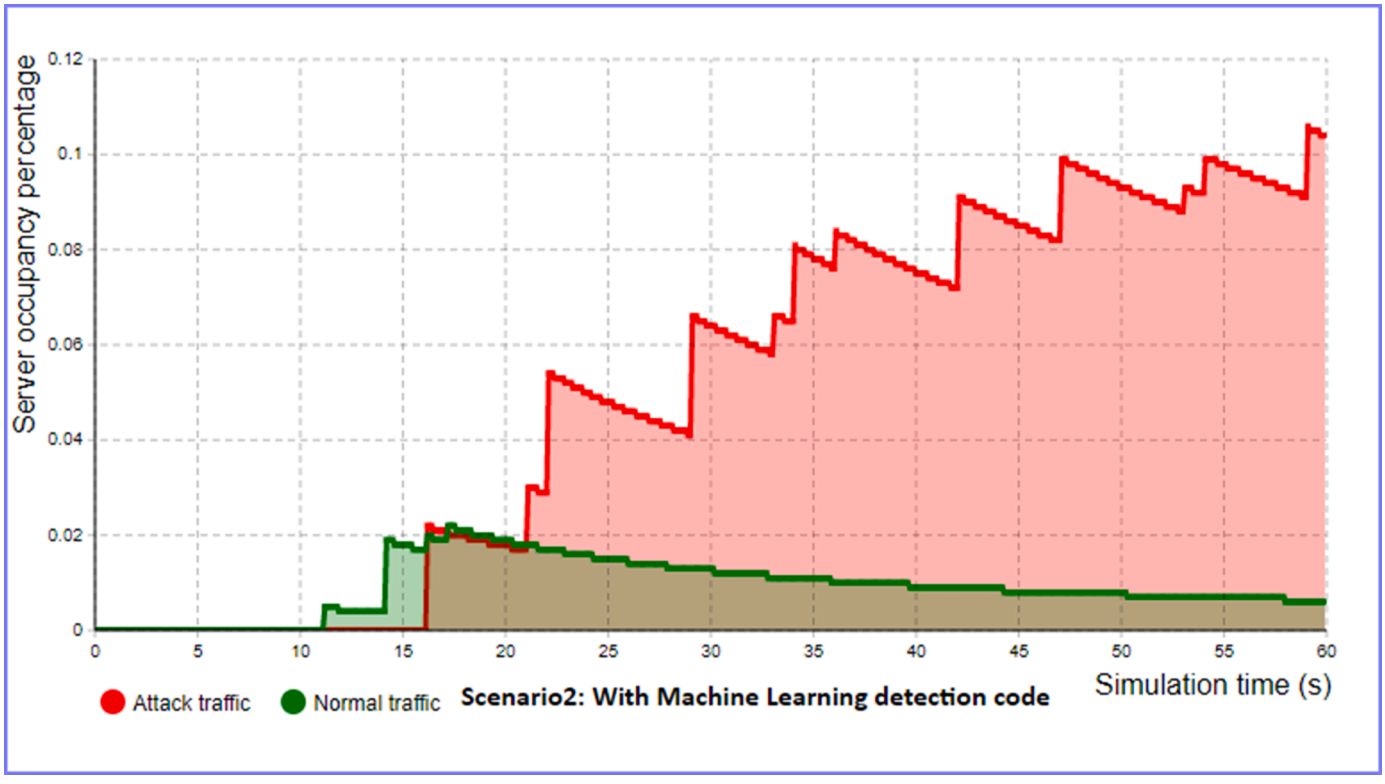


**Fig. 15.** Server occupancy in Scenario2 with ML.

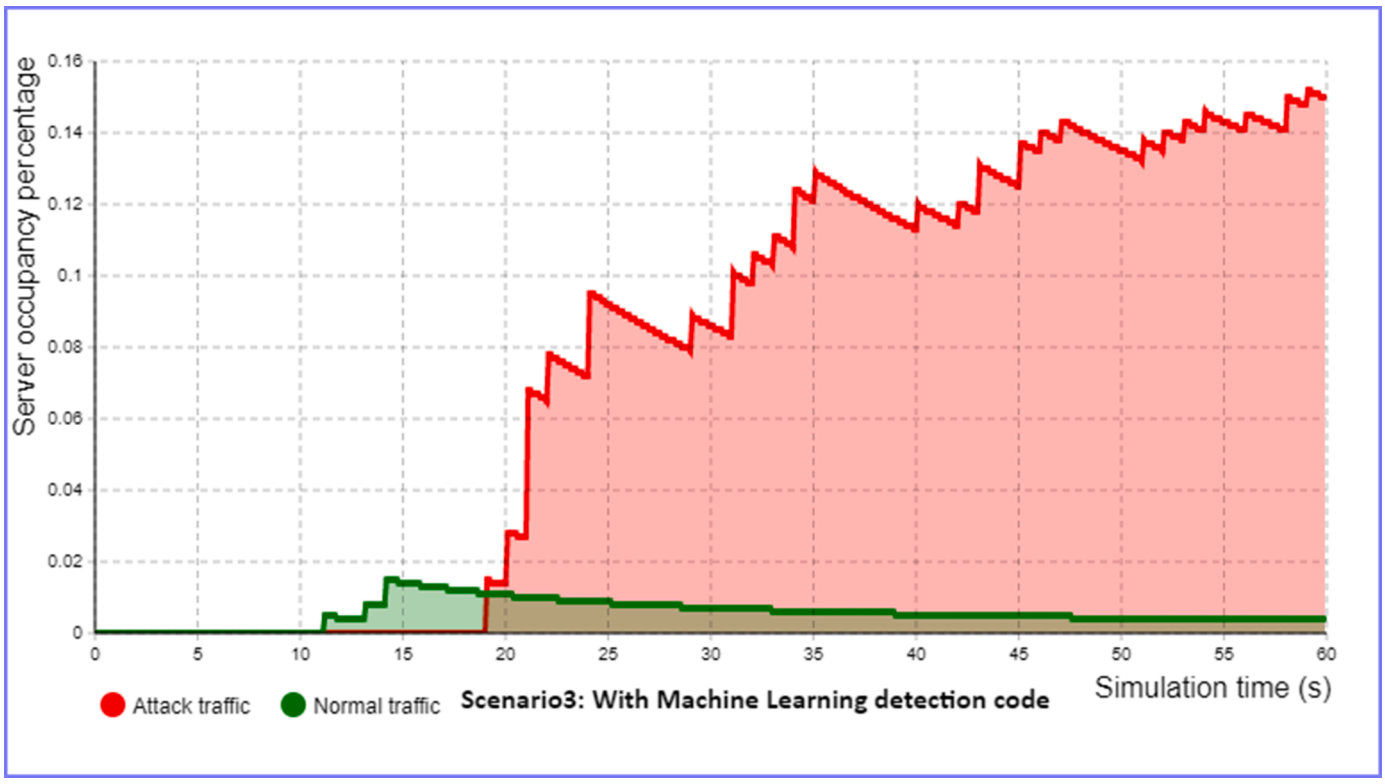


**Fig. 16.** Server occupancy in Scenario3 with ML.

therefore recommended for detection activities. Discrete or continuous labels can be managed by classification or regression algorithms, respectively [32]. We obtained the following results with KNN as the best score: *DT Accuracy = 0.98133; RF Accuracy = 0.99027; KNN Accuracy = 0.99144; SVM Accuracy = 0.98231.*

The KNN algorithm is known to be the most common classification algorithm in cases where there is no prior knowledge of the data distribution**.** This has been confirmed here by the ROC curve obtained (see Fig. C.2 in Annex C) for the deduction of AUC = 0,99000 (a sufficiently high value).

### *5.9. Simulation of the aware model*

Figs. 14, 15 and 16 respectively show for Scenario 1, 2 and 3, the timely evolution of the server occupancy (red for attack traffic, green for normal traffic) with the trained model.

It is clear that the percentage of server occupancy due to the attack traffic (red areas) is significantly reduced in all scenarios, but not null. This is because a good proportion of attack packets corresponding to true positives has been detected and only false negatives continues to flood the server.

## Conclusion

This work presents a simplified and flexible framework, based on a hybrid M&S and ML approach, for the evaluation of attack and defense strategies in CPPS. This framework has the advantage of allowing security experts without simulation knowledge to easily explore and validate their defense strategies against various attack strategies. It differs from state-of-the-art approaches in its uniqueness to simultaneously offering a very high modeling flexibility, and integrated learning capabilities for allowing self-adaptive strategy design.

Although the application in this paper focuses on the SYN-Flooding based denial of service attack, there is no limitation on the type of cybersecurity threat that can be considered. Also, any simulation platform can be used instead of the Anylogic environment, due to the expressive power of the High-Level Language for Systems Specification used for simulation modeling (HiLLS).

Such a framework is a decision support to address key questions like: (1) How can one capture the dynamic between the attacker and the defender? (2) How can one select security countermeasures that ensure self-adaptation to changing attacker strategies? And (3) How can one integrate this analysis into a cybersecurity decision support process?

In this work, we do not intend to propose yet another SYN flood detection algorithm. In our future efforts, we plan to use our framework to exploring and improving state-of-the-art SYN flood defense strategies against various attack strategies under diverse network configurations.

## Authorship contributions

Category 1 Conception and design of study: M. Koita, O.Y. Maiga, M. K. Traore; Acquisition of data: M. Koita, M.K. Traore; Analysis and/or interpretation of data: M. Koita, M.K. Traore, Y.M. Diagana; Category 2 Drafting the manuscript: M. Koita, M.K. Traore, O.Y. Maiga; Revising the manuscript critically for important intellectual content: M.K. Traore, Y. M. Diagana Category 3 Approval of the version of the manuscript to be published: M. Koita, Y.M. Diagana, M.K. Traore, O.Y. Maiga;

## ANNEXES

*ANNEX A: hills specification of the simulation metamodel's components*

Figs. A1, A2, A3, A4, A5, A6

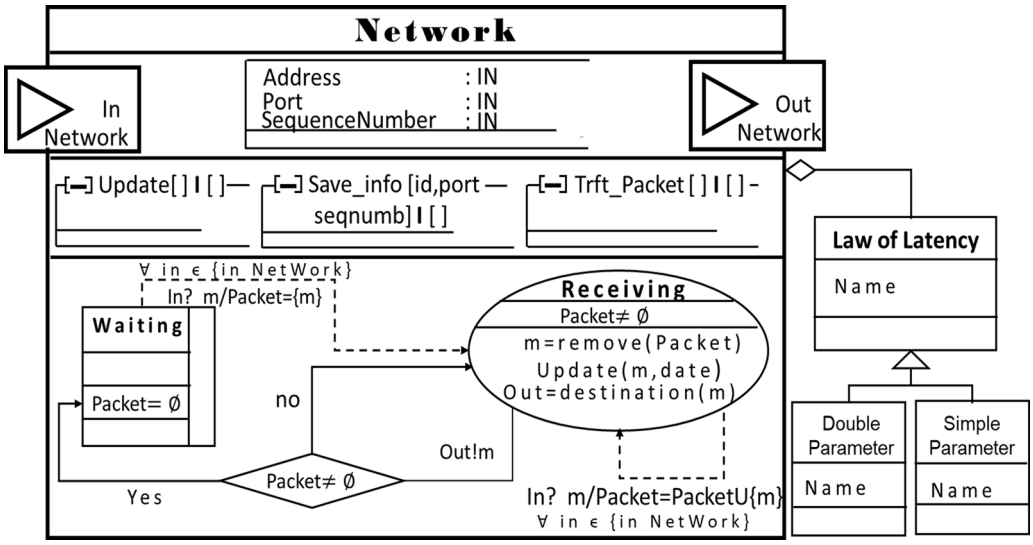


**Fig. A.1.** Network HSystem.

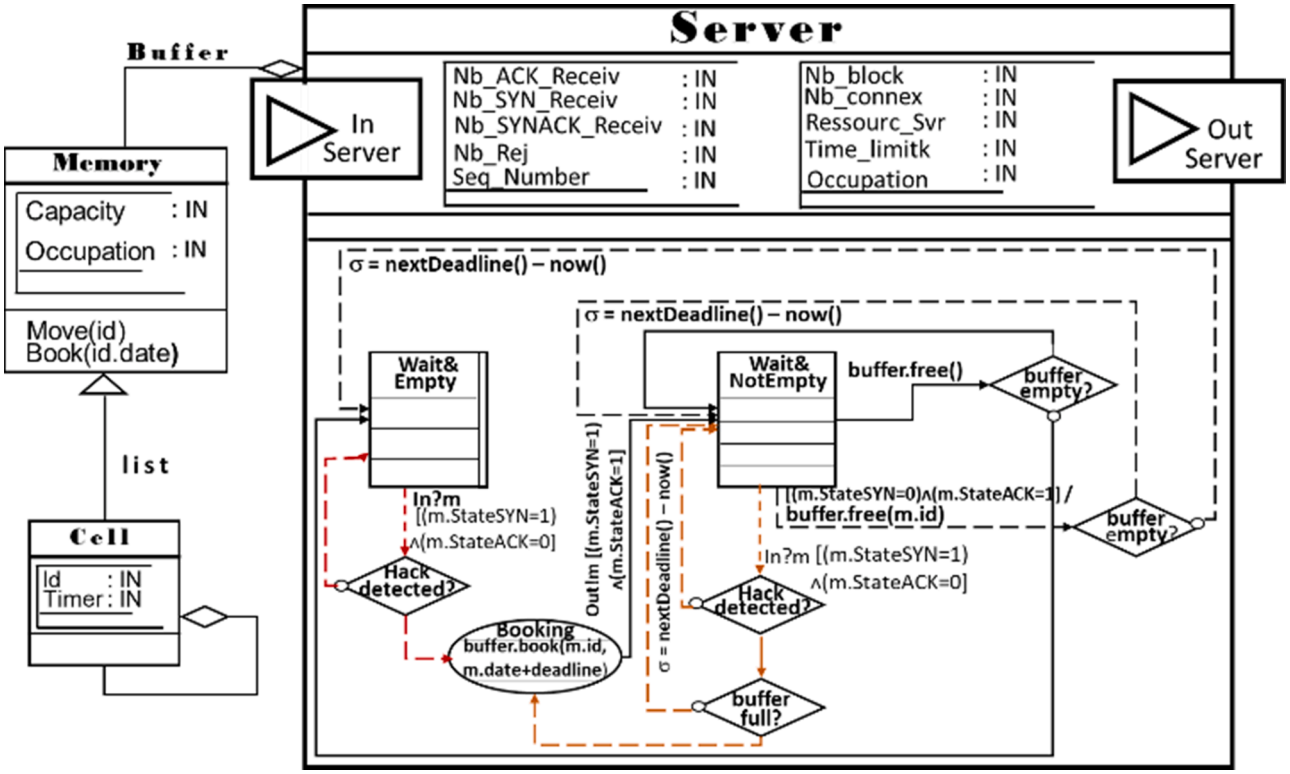


**Fig. A.2.** Server HSystem.

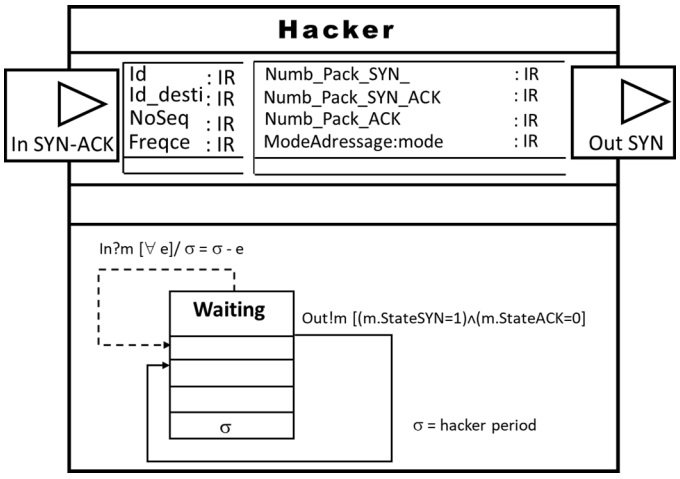


**Fig. A.3.** Hacker HSystem.

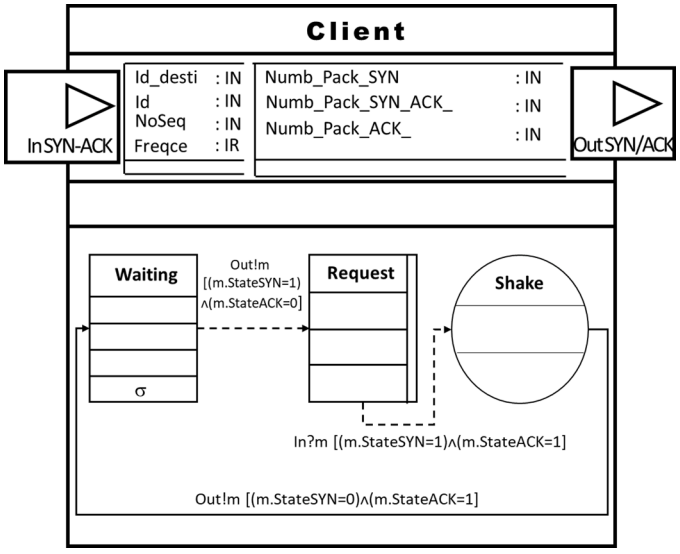


**Fig. A.4.** Client HSystem.

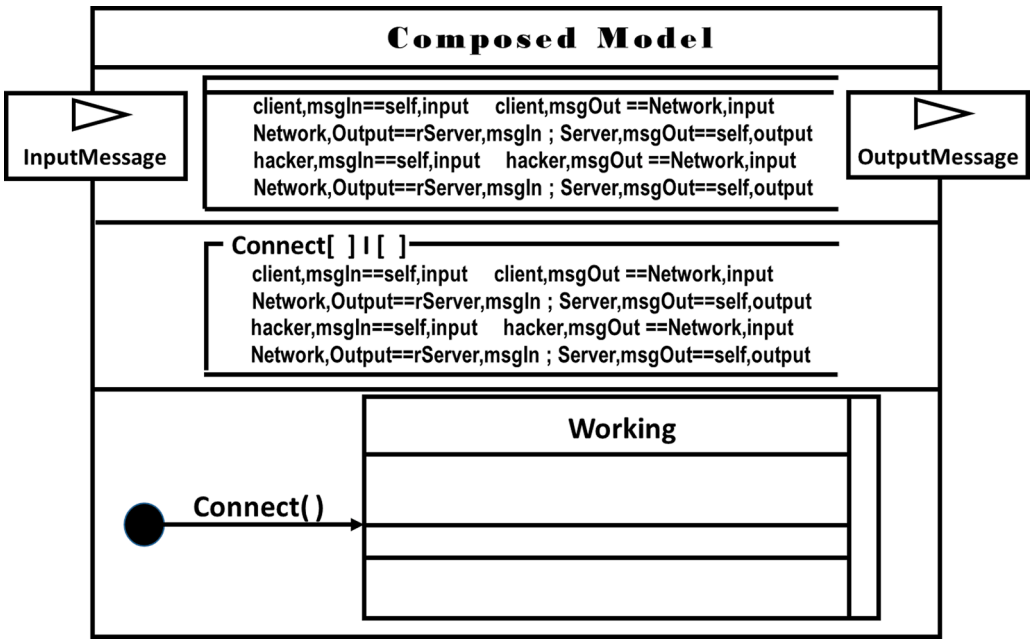


**Fig. A.5.** Domain HSystem.

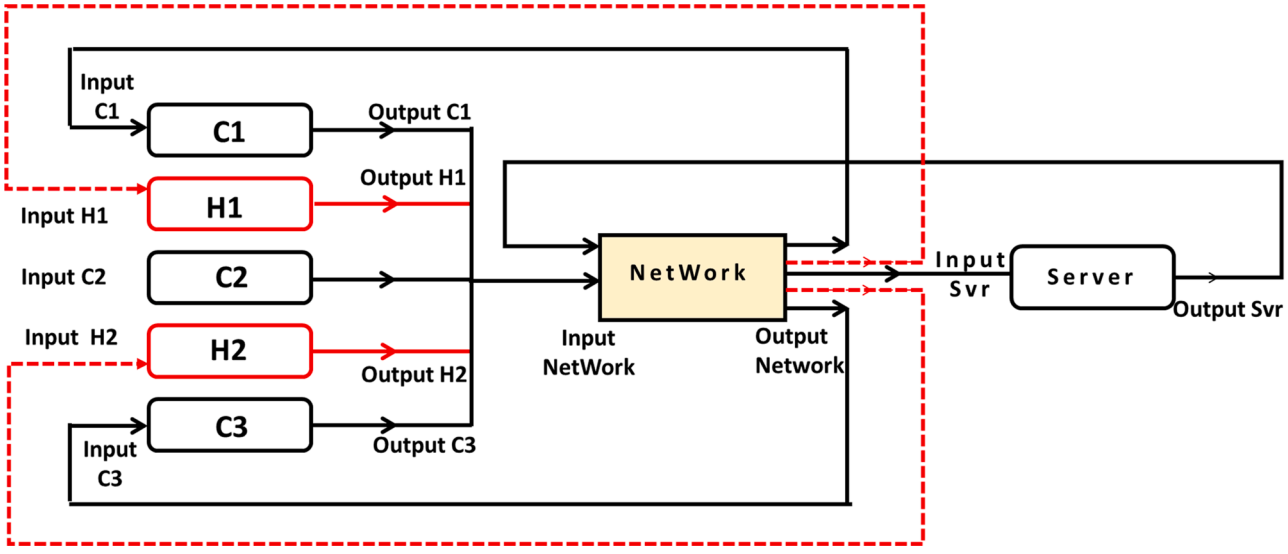


**Fig. A.6.** Coupling diagram of the domain's components.

*ANNEX B: anylogic agents of the simulation metamodel*

Figs. B1, B2, B3, B4

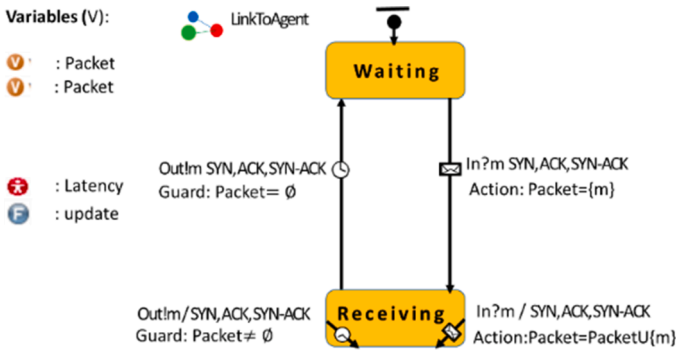


**Fig. B.1.** AnyLogic agent of the Network HSystem.

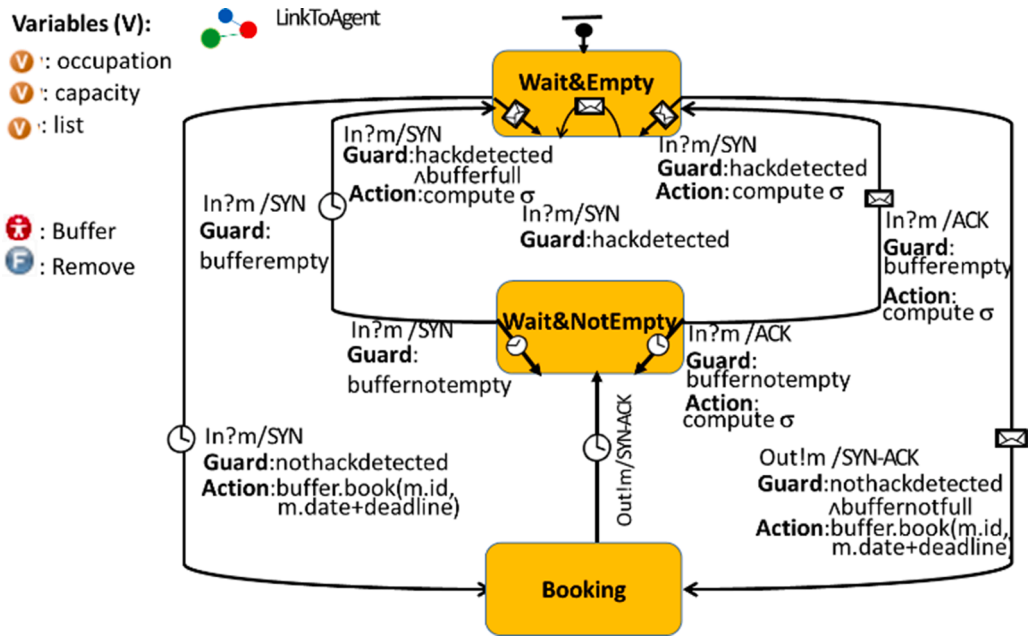


**Fig. B.2.** AnyLogic agent of the Server HSystem.

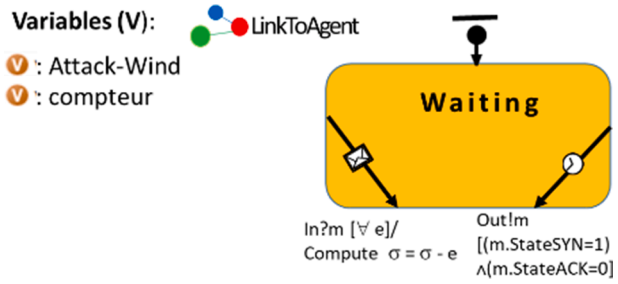


**Fig. B.3.** The AnyLogic equivalent of the HSystem Hacker.

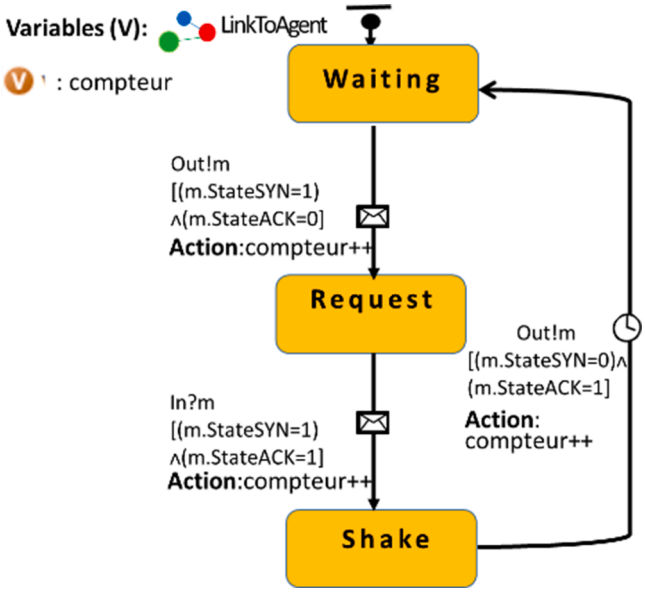


**Fig. B.4.** AnyLogic agent of the Client HSystem.

*ANNEX C: performance metrics*

Figs. C1, C3

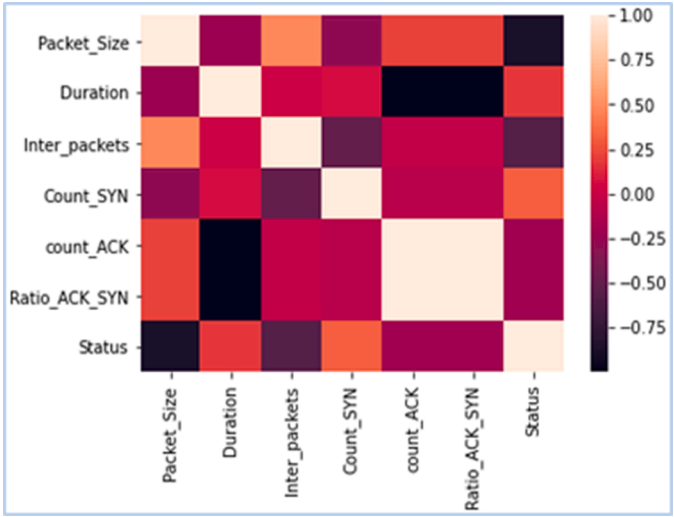


**Fig. C.1.** Correlation matrix between features.

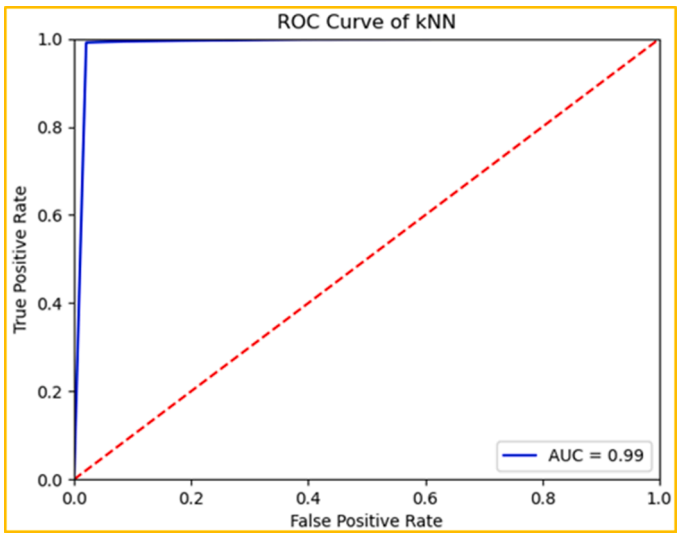


**Fig. C.3.** ROC Curve and AUC metric of the learning model.

## Declaration of Competing Interest

1. All authors have participated in:(a) Conception and design, or analysis and interpretation of the data; (b) Drafting the article or revising it critically for important intellectual content; and (c) Approval of the final version. 2. This manuscript has not been submitted to, nor is under review at, another journal or other publishing venue.3. The authors have no affiliation with any organization with a direct or indirect financial interest in the subject matter discussed in the manuscript.

## Data availability

The data that has been used is confidential.

## References

[1] A. Varga, R. Hornig, An overview of the OMNeT++ simulation environment, in: Proceedings of the 1st International Conference on Simulation Tools and Techniques for Communications, Networks And Systems & Workshops, ICSTInstitute for Computer Sciences, Social-Informatics and Telecommunications Engineering, 2008, p. 60. March. 2008.
[2] S. Bajaj, L. Breslau, D. Estrin, K. Fall, S. Floyd, P. Haldar, and S. Kumar, "Improving simulation for network research." (1999).
[3] K. Borisenko, I. Kholod, A. Shorov, Modeling framework for developing and testing network security techniques against DDoS attacks. SEKE, 2015, p. 715. July.
[4] A.S Jauhari, A.I. Kistijantoro, INET Framework modifications in OMNeT++ simulator for MPLS traffic engineering, in: 2014 International Conference of Advanced Informatics: Concept, Theory and Application (ICAICTA), IEEE, 2014, pp. 87–92. August.
[5] T. Gamer, M. Scharf, Realistic simulation environments for IP-based networks, in: Proceedings of the 1st international conference on Simulation tools and techniques for communications, networks and systems & workshops, 2008, pp. 1–7. March.
[6] L. Sánchez-Casado, R.A. Rodríguez-Gómez, R. Magán-Carrión, G. Maciá-Fernández, NETA: evaluating the effects of network attacks. MANETs as a case study, in: International Conference on Security of Information and Communication Networks, Springer, Berlin, Heidelberg, 2013, pp. 1–10. September.
[7] X. Qie, R. Pang, L, Peterson defensive programming: using an annotation toolkit to build DoS-resistant software, ACM SIGOPS Oper. Syst. Rev. 36 (SI) (2002) 45–60.
[8] J. Lemon, Resisting SYN flood DoS attacks with a SYN cache, in: BSDCon, 2002, 2002, pp. 89–97. February.
[9] C. Jin, H. Wang, K.G. Shin, Hop-count filtering: an effective defense against spoofed DDoS traffic, in: Proceedings of the 10th ACM conference on Computer and communications security, ACM, October 2003, pp. 30–41.
[10] D.M. Divakaran, H.A. Murthy, T.A. Gonsalves, Detection of SYN flooding attacks using linear prediction analysis, in: 2006 14th IEEE International Conference on Networks 1, IEEE, 2006, pp. 1–6. September.
[11] K. Shaukat, T.M. Alam, I.A. Hameed, W.A. Khan, N. Abbas, S. Luo, A review on security challenges in internet of things (IoT), in: 2021 26th International Conference on Automation and Computing (ICAC), IEEE, 2021, pp. 1–6.
[12] H. Qiao, J. Peng, C. Feng, J.W. Rozenblit, Behavior analysis-based learning framework for host level intrusion detection, in: 14th Annual IEEE International Conference and Workshops on the Engineering of Computer-Based Systems (ECBS'07), IEEE, 2007, pp. 441–447.
[13] I.C. Lin, C.C. Chang, C.H. Peng, An Anomaly-based IDS framework using centroid-based classification, Symmetry 14 (1) (2022) 105.
[14] A. Prasad, S. Chandra, VMFCVD: an optimized framework to combat volumetric DDoS attacks using machine learning, Arab. J. Sci. Eng. (2022) 1–19.
[15] O. Bamasag, A. Alsaeedi, A. Munshi, D. Alghazzawi, S. Alshehri, A. Jamjoom, Real-time DDoS flood attack monitoring and detection (RT-AMD) model for cloud computing, PeerJ Comput. Sci. 7 (2022) e814.
[16] M. Najafimehr, S. Zarifzadeh, S. Mostafavi, A hybrid machine learning approach for detecting unprecedented DDoS attacks, J. Supercomput. 78 (6) (2022) 8106–8136.
[17] L. Hou, J. Zhang, N. Jin, M. Zhu, Y. Li, Digital substation cyber security analysis with SYN-flood attack as a simulation case, in: 2016 Chinese Control and Decision Conference (CCDC), IEEE, 2016, pp. 4467–4472.
[18] G. Settanni, F. Skopik, A. Karaj, M. Wurzenberger, R. Fiedler, Protecting cyber physical production systems using anomaly detection to enable self-adaptation, 2018 IEEE Ind. Cyber-Phys. Syst. (ICPS) (2018) 173–180. IEEE].
[19] K. Shaukat, S. Luo, S. Chen, D. Liu, Cyber threat detection using machine learning techniques: a performance evaluation perspective, in: 2020 International Conference on Cyber Warfare and Security (ICCWS), IEEE, 2020, pp. 1–6.
[20] J. Zhang, L. Pan, Q.L. Han, C. Chen, S. Wen, Y. Xiang, Deep learning based attack detection for cyber-physical system cybersecurity: a survey, IEEE/CAA J. Automat. Sin. 9 (3) (2021) 377–391.
[21] H.O. Aliyu, O. Maïga, M.K. Traoré, The high-level language for system specification: a model-driven approach to systems engineering, Int. J. Model. Simul. Sci. Comput. 7 (01) (2016), 1641003.
[22] B. B. Thiago, These Thiago Barros Brito "agent-based simulation for yard management in container terminal operations," 2016.
[23] M. T. García, M. A. Barcelona, M. Ruiz, L. García-Borgoñón, and I. Ramos, "A discrete-event simulation metamodel for obtaining simulation models from business process models". In Information.
[24] A. Borshchev, Multi-method modeling, in: Proceedings of the 2013 Winter Simulation Conference: Simulation: Making Decisions in a Complex World (WSC '13, IEEE Press, 2013, pp. 4089–4100.
[25] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, E. Duchesnay, Scikit-learn: machine learning in Python, J. Mach. Learn. Res. 12 (2011) 2825–2830.
[26] N. Li, H. Kong, Y. Ma, G. Gong, W. Huai, W, Human performance modeling for manufacturing based on an improved KNN algorithm, Int. J. Adv. Manuf. Technol. 84 (1-4) (2016) 473–483.
[27] K. M. Elleithy, D. Blagovic, W. K. Cheng, and P. Sideleau, "Denial of service attack techniques: Analysis, implementation and comparison", (2005).
[28] M. Kumar, A. Panwar, A. Jain, An analysis of tcp syn flooding attack and defense mechanism, Int. J. Eng. Res. Technol. (IJERT) 1 (5) (2012) 1–6.
[29] J. Mirkovic, A. Hussain, B. Wilson, S. Fahmy, P. Reiher, R. Thomas, S. Schwab, Towards user-centric metrics for denial-of-service measurement, in: Proceedings of the 2007 Workshop on Experimental Computer Science, ACM, 2007, p. 8. JuneJune.
[30] S. Abbasvand, S.N.S. Hashemi, S. Jamali, Defense against SYN-flooding attacks by using game theory, Indian J. Sci. Technol. 7 (10) (2014).
[31] G. Kumar, Evaluation metrics for intrusion detection systems-a study, Evaluation 2 (11) (2014) 11–17.
[32] M. Ribeiro, K. Grolinger, M.A. Capretz, Mlaas: Machine learning as a service, in: 2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA), IEEE, Dec. 2015, pp. 896–902.

**Moussa Koïta** is a PhD student in Computer Science at the Nangui Abrogoua University of Abidjan (Ivory Coast), where he got his MSc in Computer Science. His current research is on security of cyber-physical systems.

**Youssouf M. Diagana** is full Professor at the Nangui Abrogoua University of Abidjan (Ivory Coast). He got his PhD in Mathematics at the Félix Houphouet Boigny University of Abidjan, Ivory Coast, in 1994. His current research is in cryptography for cybersecurity.

**Oumar Y. Maïga** is Associate Professor at the University of Sciences, Techniques and Technologies of Bamako, Mali. He got his PhD in Computer Science at Université Blaise Pascal, Clermont Ferrand, France in 2015. He published 20+ papers in Modeling and Simulation-related international journals and conferences. His current research is in methodologies for Modeling and Simulation of complex systems.

**Mamadou K. Traoré** is full Professor at Université de Bordeaux (France). He got his PhD in Computer Science at Université Blaise Pascal, Clermont Ferrand, France in 1992. He published 100+ papers in Modeling and Simulation-related international journals and conferences and 10+ books and proceedings. His current research is in hybrid M&S and AI for production systems engineering. He is an ACM senior member, a member of SCS, and an ASI fellow.