Simulator for analysis cyber threats to RFID based system

Dariusz Pierzchała[1,*], *Andrzej* Najgebauer[1]

[1] Institute of Computer and Information Systems, Faculty of Cybernetics, Military University of Technology, Warsaw, Poland

Abstract. The paper addresses the issues of attacks in cyberspace from the local perspective of computer network and IoT devices of RFID based office system. The rapid development in IT technology has constantly brought out revolutionary changes in the area of office services. The very promising technology for automatic identification is RFID that has changed the manual scenario of office tasks to computerized automated services. However, RFID based systems are vulnerable to various cyberattacks, from simple passive eavesdropping to well-known denial of service (DoS). One of the methods for the analysis of threats is computer modelling and simulation. M&S gives the possibility of quantitative and qualitative analysis as well as forecasting. As a part of the work, the original Java-based discrete simulation package has been developed. The simulation model and relevant software was implemented to simulate DoS attack on a protected server in restricted access administrative office. In contrast to most of existing simulation tools in the field of cyberspace, the simulation package has been designed to model and simulate heterogeneous devices which behaviours might be modified by programming Groovy/Python scripts.

1 Introduction

RFID technology (Radio Frequency Identification) is perceived to become one of the fastest growing and very promising cutting edge technologies with millions of devices (RFID tags and readers) connected into networks on a global scale. The examples focusing both on IoT and RFID in various types of problems are widely discussed, e.g. in [1], [2], [3], [18], [19]. Applications of the technology might be found in numerous branches of the economy and administration as well as in logistics, manufacturing or medical organizations. RFID offers new possibilities in automatic identification and tracking items, for instance an office document might be precisely identified, stored and verified. Moreover, it has changed the typical manual scenario of office tasks to computerized automated services.

A typical RFID system has a relatively simple structure which is similar to the Internet of Things based network. This is of particular importance for systems deployed in restricted access administrative offices (military, government etc.) where the following IoT devices are implemented as RFID components [4]:

- RFID tag equipped devices: input sluice with RFID antennas as access point to the office, cabinets with antennas, document read tunnel with RFID tags, RFID tray reader, specialized copier with RFID module,
- RFID tag reader (a software component),
- and RFID tagged document flow management system (a backend system database and software).

RFID tag equipped devices emit signals at a certain frequency that are read by RFID computer reader. Next, a reader uses a middleware software responsible for preparing recognized data and sending meaningful information to a backend document flow management system. Finally, the last component processes the information retrieved from tags it obtained via connection with reader. This type of framework system enables to recognize and record a series of events occurring in the office room [4], for instance:

- new RFID tagged documents (tags) appear in the RFID antenna zone,
- a single one or a package of tags leave from the antenna zone,
- a tagged documents change locations within the antenna zone.

Some IoT devices, like antennas and readers, are immersed in a distributed wireless environment while others, particularly tag reader and document flow management software system, are connected to the LAN network and event to the Internet. Due this fact RFID based systems are vulnerable to various cyberattacks – from eavesdropping to well-known denial of service (DoS). According to [8] the centralized security mechanism (such as firewall or VPN network) cannot be directly applied in a distributed wireless environment. Correspondingly, considering a tiny computing power and very limited storage capacity of RFID tags it seems hard to deploy cryptographic techniques in wireless protocols. On the contrary, a LAN network with document flow management system might be treated as a cyber target for the DoS cyberattack resulting either in server or network resource unavailable to its intended clients (software reader or user) or disrupting services.

* Corresponding author: dariusz.pierzchala@wat.edu.pl

Well-known and widely admitted method for threats' analysis is computer modelling and simulation which offers the possibilities for quantitative and qualitative analysis as well as forecasting. The main goal of the presented work is the original Java-based discrete simulation package applied to implement the simulation model of DoS attack on a server in restricted access administrative office. In contrast to most of existing simulation tools in the field of cyberspace, the simulation package has been designed to model and simulate heterogeneous IoT devices which behaviours or activities might be easily modified by programming Groovy/Python scripts even after building a closed ready-to-use simulator. It is the technical novelty and advantages of the work we have been conducted in the field of computer modelling and simulation.

2 Analysis of threats in cyberspace

The term "cyberspace" exists in the present world in many semantic dimensions [16]. The understanding of the word is varied for IT engineers, sociologists, politicians, typical Internet users, etc. Let cyberspace be defined as a digital space based on computer networks, where users can collect, process and exchange information. It consists of networks, which are components of the global Internet. What is important cyberspace contains information systems and services for home, business, critical infrastructure. The strong relations between modern work techniques and the hardware/software infrastructure of cyberspace makes the issues of security, reliability and effectiveness of computer networks particularly important. It is becoming more difficult because of the fast growing of IoT domain - according to Ericsson's Mobility Report the number of new Internet of Things devices in Europe is expected to grow by 400% between 2015 and 2021 (mainly due to smart meters and cars connected to the Internet).

The growing need for software tools to analyse, design, build in a proper way network systems and improve existing ones is confirmed by the systematically revealed information about effective attacks in cyberspace as well as about hundreds of incidents recorded by national CERTs (Community Emergency Response Team).

Most attacks are aimed at stealing information, fraud or disrupting the efficient functioning of institutions. Common threats are: virus and worm, Trojan horse, spyware, adware, rootkit, dialler, hoax, axploit, spamming, phishing, sniffing, backdoor or finally flooding and DoS. In the RFID area there are attacks resulting in blocking communication: Block tag reader (Shield tag- Faraday cage around reader or tag, Blocker reader, Blocker tag), Block reader backend (DoS in network) [17]. According to numerous studies, the latter one is particularly burdensome due to their easiness of execution and relatively serious consequences. The basic scheme is to overload (flood) the server by sending a huge number of false packages, each with an attempt to establish a connection. This overloads both the network connections and the servers that handle the requests. If the attack is conducted simultaneously from multiple sources (computers, IoT devices), we are faced DDoS (Distributed Denial of Service) attack. An effectively attacked computer system allocates memory resources, processors, and network bandwidth until the available resources are exhausted, what results in an interruption or suspension of system services. Hence, the basic strategies of attacks are:

- bandwidth attack: involves the server's network resources to saturate the bandwidth until the server is unavailable to the external users;
- attack on resources: involves the resources of a computer system until the responses to correct queries are blocked;
- attack on software error ("exploit"): exploits a software error until the system is blocked or under control.

With a development of the Internet of Things, the scale of mass attacks of devices connected to the IoT has significantly increased (from mobile to office and typical network infrastructure devices). The high vulnerability of these relatively unsecured devices makes them easy targets for cybercriminals. Various studies, such as Trend Micro, report this fact. Then the attacked IoT devices become remote sources of subsequent DDoS attacks. Thus the amplification degree of a DDoS attack may range from hundreds to even thousands. Today, almost every device with a CPU and a network interface is a potential, inexpensive and common tool for a DoS attack. Attack technologies can be built, acquired for free or purchased - for example, Low Orbit Ion Cannon (LOIC), a software package for DoS attacks (if multiple attackers then DDoS) to overload the attacked server with a large number of TCP/UDP packets or HTTP protocol requests. At this point, we draw particular attention to the HTTP protocol, which is most commonly used for communication between a tag reader and a backend software system. This is a critical channel of attack that can be used by attackers in the network.

The strength of DDoS attacks is huge - the attack on "KrebsOnSecurity.com" in 2016 generated 620 Gb/s of traffic. Another attack on dynamic DNS servers was reported to have a force of 1.2 Tbps and resulted in temporary blocking of access to Twitter and Spotify.

A DoS/DDoS attack can affect any network user, so it is worth and even necessary to take preventive measures to minimize the losses after its occurrence. What are the effective methods to protect against DDoS? Unfortunately, successful defence against DDoS attacks at the network level is very difficult. Preventive monitoring of infrastructure efficiency, analysis of anomalies in network traffic, use of redundant infrastructure with easy switching and DDoS traffic isolation techniques (e.g. filters in firewall devices) are commonly recommended. Properly prepared firewall rules can filter out most of the traffic generated by an attack using tools similar to the LOIC package. Contemporary firewalls enable very complex security algorithms to be implemented together with integration with IDS (Intrusion Detection System) or IPS (Intrusion Prevention System) systems, i.e. systems to detect and block attacks in a real time. In the case of RFID based

devices and systems, a number of methods have been developed – it should be mentioned according to [9], [10]: protocols utilizing hash functions, authentication using matrix multiplication or pseudorandom number generators and cyclic redundancy check, ultra light mutual authentication protocols (UMAP) e.g. the Gossamer protocol.

Any preventive action (its type and scope) should be supported by both informal heuristics and accurate analytical methods or simulation estimates. Similarly to the typical monitoring, in an analysis and modelling should be carried out as a minimum:

- the volume of incoming and outgoing traffic;
- parameters of servers and network interfaces;
- CPU and network load, number of media operations, and storage capacity.

The idea of the presented research is to use methods for modelling and simulation, as security systems should be studied and prepared comprehensively: both for large scale attacks (many easy to identify queries in the network layer), as well as for smaller ones but more sophisticated (fewer queries in the application layer, but significantly more difficult to distinguish from typical traffic). This is achieved by scenario analysis, in which we examine the modelled system for a number of hypothetical situations, including irregular conditions. Simulation experiments involving pseudo-random generators help to discover "bottlenecks" in the network and security infrastructure, which are of the highest importance for the DDoS attacks being considered.

3 Discrete simulation software package

Global trends in operational research indicate that computer simulation is one of the most common approaches to modelling cyberspace at the device, protocol, service and user levels. It increases the possibilities for quantitative and qualitative analysis and forecasting. It is a recognised alternative to experimenting with a real system or its physical prototype and to mathematical analysis. It replaces informal methods of reasoning based on experience and intuition. As a result of the development of computer systems, simulation has undergone the evolution of technologies and algorithms similar to typical IT systems: from a sequential simulation experiment performed on a stand-alone machine, through parallel, networked to distributed in virtual resources (so-called cloud computing). However, dedicated computer simulators, adequate but at the same time universal and open to modification, are essential for successful simulation. In our work, we emphasize these two features:

- openness to defining new algorithms for behaviours /activities of objects;
- universality, i.e. the ability to adapt to various types of problems and levels of modelling.

We will define computer simulation as a quantitative and qualitative method of modelling in a formal language and then implementing in a computer program the features of systems (structural and behavioural) in order to experiment with the model and then study occurring processes during the simulation time [10]. The beginning of computer simulation was the development of network models (Carl Adam Petri) and methods of system dynamics analysis (Jay W. Forrester) and then the languages and methods of continuous and discreet simulation: differential: (Lockheed - Digital Differential Analyzer Simulator DIDAS9), events (Harry Markowitz, Bernard Hausner - SIMSCRIPT), process interactions (Ole-Johan Dahl and Kristen Nygaard - Algol Simula67, Knuth and McNeley - Algol SOL), selective activity (John Buxton and John Laski - Fortran CSL) and 3-phases (Keith Douglas Tocker - General Simulation Program GSP). The milestone was the language of Simula67 which, through its concept of class and object, moved away from the structural approach commonly used at the time and becoming the protoplast of objectivity. The response to the growing simulation demand are packages extending general purpose languages (e.g. C#, Java) with simulation services (dynamic calculation of state variables values, time control, generation of random numbers, dispersion of processes in the network). Thanks to these techniques, any evolution in language also implies new possibilities in simulation applications. Simulation experiments are carried out not only on individual computer stations, not only in local networks and grids, but also in services provided from cloud computing. In the case of the Java language, the following possibilities were used in the presented research: advanced collections of objects, functional interfaces and Lambda expressions, integration through open scripts with Groovy and Python languages.

The most popular and universal type of simulation is still being constructive simulation which is based on formal models of objects and their implementation in computer programs. The original Java-based discrete simulation software package (named DisSim) has been developed in order to facilitate creating constructive simulation software – both discrete and quasi-continuous as well as hybrid. DisSim is an object paradigm based package. The basic entity is an object '*o*' with its attributes:

- $O = \{o=<id, c>\}$, $c \in C^O$, $id \in N$ - a set of simulated objects of '*c*' class that are identified by an '*id*' with a unique value;
- C^O - a non-empty set of classes of modelled objects;
- A_c - a non-empty set of attributes specified for the $c \in C^O$ object class;
- V_a^c - a set of acceptable values of the $a \in A_c$ attribute of the $c \in C^O$ objects class.

The C^O and A_c sets contain the numbers of modelled object classes and their attributes. At any time '*t*' of the simulation time each measurable feature of the system is represented by an ordered quadruple: $<o, a, v, t>$. The state of the modelled system $S(t)$ will be the vector created by all attributes of all objects at the simulation time '*t*'. In the simulation $v \in V_a^c$ values of $a \in A_c$ attributes are determined as a result of the user-defined state change function. The concept of simulation time $t \in T$ is essential for a system model - it is a non-negative and non-decreasing real variable $t \in R^+ \cup \{0\}$. For the

defined simulation moments t_i, t_j, t_k, for T as a collection of the moments when a system state changes there are applicable the following conditions:

- for a quasi-continuous passage of time: T is a subset of points in a certain range of non-negative real numbers such that $(\forall t_i, t_k)(\exists t_j)(t_i < t_j < t_k)$ – for any two moments there is a moment between them;
- for a discrete passage of time: T is a countable subset such that $(\exists t_i, t_k)(\neg\exists t_j)(t_i < t_j < t_k)$ – there are two such moments that there is no time between them.

Consequently, along with the assumptions for each object there are defined: the state vector with attributes and either the events or periodically executed equations that change the values of attributes (and finally a state of the system). The term event '*e*' from a finite set of events '*E*' is understood as the algorithmic change in object's state which is scheduled for a specific simulation moment $t \in T$: $f_e^S : T \times S \to S$ and $S(t) = S_i$ for $t \in [t_i, t_{i+1})$. On the other side we have the Runge-Kutta numerical algorithm for the differential equations calculated upon the rule "*Future state = Present state + change after time step*". A sequence of chronologically (in a sense of simulation time) ordered change in state is the process of simulation. In general, it may be perceived as a multi-dimensional stochastic process where the individual elements of a state vector describe the various parameters of the system at the time '*t*'.

The current simulation time is calculated either with the time stamp of the first event from the event calendar: $t^* = \min\left(t : e_i = \left\langle t, f_e^S \right\rangle, i = 1..2^{E \times T}\right)$ or incrementally by a constant time step: $t^* = t^{i+1} = t^i + \Delta t$.

The result of an implementation in Java of the above model are the classes of DisSim package. The main classes of the package are presented in a simplified class diagram (Fig. 1). Every $o \in O$ simulation object of the $c \in C^O$ class inherits from the abstract *BasicSimEntity* class. Its A_c attributes store the values of the state vector. Each event $e = \langle t, f_e^S \rangle$ is a state change created in objects inherited from the generic *BasicSimStateChange* class. An event is the result of a f_e^S state change function implemented as the *transition()* method of the *BasicSimStateChange* class. The event class attributes include: scheduled execution time and priority of the status change (*runSimTime*, *priority*) and an optional time step (*repetitionPeriod*) for either a discrete with a fixed step or a quasi- continuous simulation. For the latter type of simulation a special repeatable event class *RKEvent* is dedicated. After each time step (stored in *repetitionPeriod*), the values of equations describing the system state are determined in the method *transition()*. The exact form of the equations has to be determined in *Function<RKFunctionParameters, List<Double>>* by the DisSim user. The *Function* is a realization of Java functional interface thus an user is expected to implement lambda expressions and method references.

Moreover, DisSim is having classes responsible for message transmission via event bus, dynamic configuration of objects and open plug-in architecture. Additionally, as we indicate in previous section, an interpreter classes for Groovy/Python scripts have been implemented. The integration between DisSim classes and Groovy scripts has been achieved on the base of *GroovyScriptEngine* and specialized methods to load scripts (*GroovyScriptEngine(loadScriptByName(name)*), and subsequently execute scripts' code with parameters (*getDeclaredMethod(method).invoke(scriptRef, args)*).

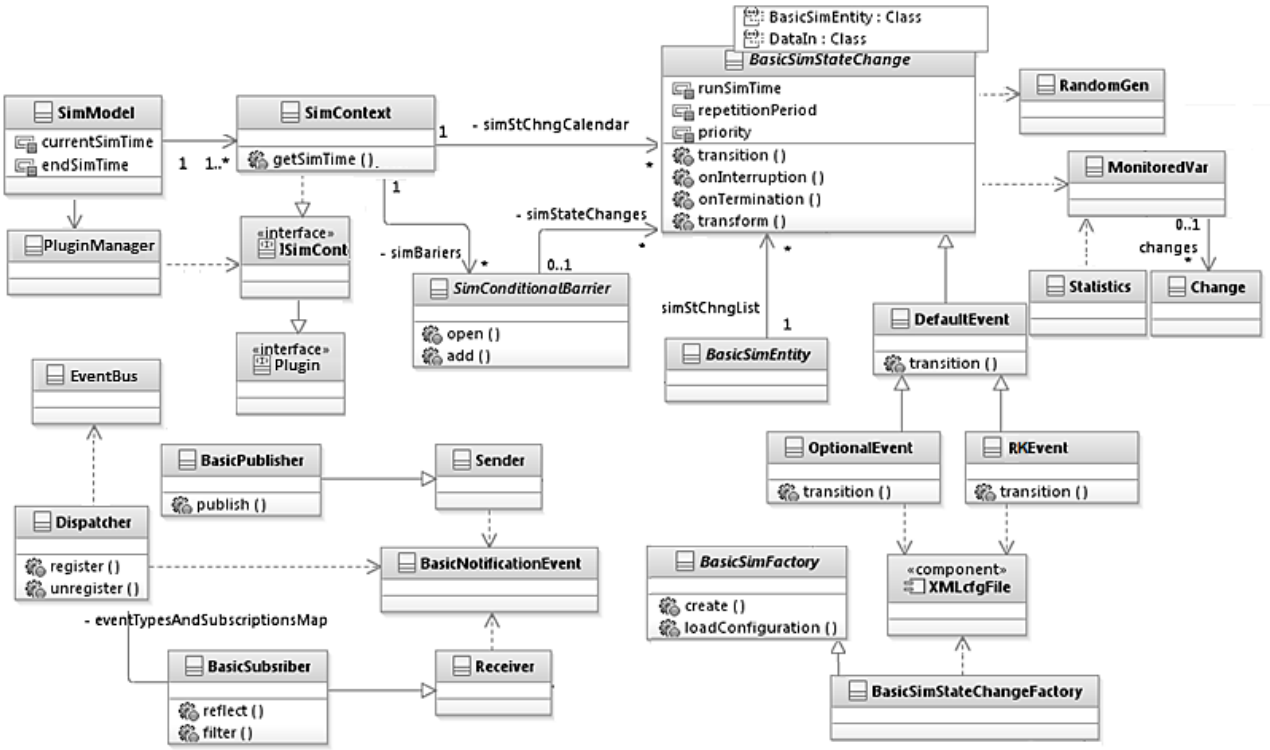

Fig. 1. Simplified class diagram of DisSim package.

For each object instance of *BasicSimStateChange* class new algorithm of the method *transition()* might be defined in a separated Groovy script file. Finally, codes from scripts will be loaded into the scripts' map in DisSim and launched by an interpreter during an execution of the *transition()*methods inside objects inherited from *BasicSimStateChange* and defined by the DisSim user.
A comparable solution has been defined and designed for scripts stored in Python-compliant files.

4 Software tool for modelling and simulation DoS threats

The popularity of constructive simulation as a tool in modelling and evaluating the security and performance of computers networks (including Internet of Things, RFID, Wireless Sensor Networks, LAN networks) has increased significantly in recent years. Ready-to-use simulators allow researchers to verify new ideas in a virtual environment without the need to use any hardware. There are several dozen of currently available simulators [4],[5],[6],[7],[12]. In the vast majority cases the simulators focus on wireless communication aspects and resources management and are restricted to hardware platforms. Following [11] they can be divided into the following groups:

- emulators - they emulate a sensor hardware or programs and usually are platform specific;
- topology control - the simulators allow to model network topology using algorithms and protocols;
- wireless environment and medium simulators – those platforms focus on simulating wireless communication and hardware configuration like antenna type, signal strength etc.;
- network and application level simulators - they are designed to simulate sensor network and mainly to test routing algorithms;
- cross level simulators - this category represents tools able to simulate devices and network at various levels of abstraction: routing algorithms, energy and resource consumption and simple applications.

The last group contains the most all-purpose tools – some representative are as follows: COOJA [13], J-Sim and Sensor Network Package [14], CupCarbon [15].
The other group is made up of simulators which base upon COTS software (Commercial Off-The-Shelf Software), like NS2 or OMNET++. Those are not WSN or IoT specific solutions, however they are mature products with great ability to simulate wireless communication and corresponding issues. Some of them have the open-source status and are supported by researcher community. Table 1 presents a list of selected software simulators from the point of view of their purpose and modelled reality.

The advantages of the Optimized Network Engineering Tool (OPNET) and the open source package OMNeT++ should be highlighted. Both packages implement a discrete model (in sense of time) and have many components ready-to-model the network. Despite such a rich set of functions, they are not complete from the point of view of IoT networks and RFID systems. In addition, modular architecture, configurability, and a wide range of device and protocol components are "costly". These packages require computers with substantial computing and storage resources. Therefore, the counterbalance to the " heavy-weight " packages are "light-weight" solutions, which are built from the scratch and customized to the needs of the end users. The other key feature to implement was the ability to change devices behaviour and activities without changing any compiled line of code, even during running simulation process. We propose the way to do that based upon Groovy/Python scripts and DisSim classes to serve them. These highlighted reasons prevailed for the decision to use the original DisSim package in order to implement simulation models specified for both the IoT and RFID domain problems.

Table 1. Software simulators comparison.

Name	Purpose	Modelling scope
OPNET	Service queuing - LIFO, FIFO; priority queues, round-robin	IP, TCP, Ethernet, 802.11, FDDI, ATM, support for wireless network
NetSim	Relative station positions in the network, realistic signal propagation modelling, collision handling and detection process	TCP/IP, Ethernet, WLAN, ATM
QualNet	Evaluation of the various protocols	LAN, WAN, wireless network
OMNeT++	Delays, jitter, packet losses	wireless network
Ns-2	Overloading, queuing and routing algorithms, multicasting	TCP/IP in LAN and wireless network
GloMoSim	Evaluation of various wireless network protocols, MAC protocols	wireless network
GTNetS	Package tracking, queuing methods, statistical methods, random number generators	Point-To-Point, Ethernet
Shunra VE	Delays, jitter, packet loss, bandwidth restrictions	TCP, WAN, Point-to-Point,

The key architectural objective of the presented software tool for modelling and simulation DoS threats to RFID based office system was easiness of its reconfiguration

and further development. Figure 2 shows its overall architecture.

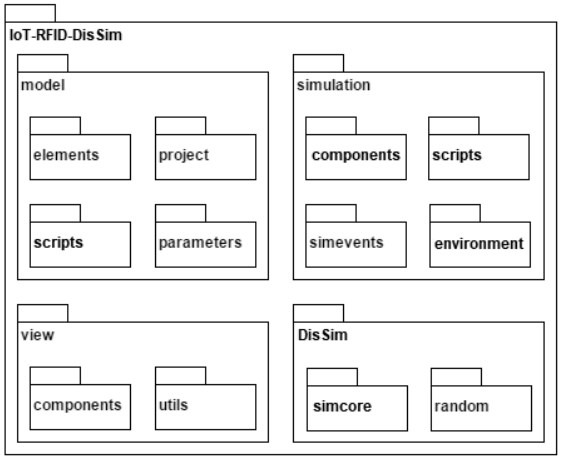

Fig. 2. Simulator architecture overview.

There are four main internal packages inside the software we have developed:

- model – the main framework for whole simulator;
- simulation – it contains classes which are a software implementation of a mathematical models of the modelled devices (structural and behavioural);
- view – it contain classes with operations which are not time-depended (within the meaning of simulation time), like adding/removing icons to/from the editor area;
- dissim – Java based simulation package with features we described earlier.

The DisSim subpackages of classes are designed for:

- the package *simcore* – the classes for controlling the whole experiment: passage of simulation time, events ordering and changes of state (both discrete and quasi-continuous);
- the package *broker* – a message bus supporting messages exchange between simulation objects (two modes: publisher-subscriber and notifier-observer);
- the package *random* – a (pseudo)random number generator with a broad range of methods responsible for different probability distributions (including template for user defined empirical distribution function);
- the package *monitors* – monitoring, collecting and statistical analysis, both in the interior *Statistics* class and in *R* statistical package (very popular software environment for statistical computing and graphics) via dedicated interface.

Each modelled object should be a specialization of the abstract class *BasicSimEntity*. The state changes (in events or after each fixed-time-step) are designed along with simulation objects on the basis of the generic class *BasicSimStateChange*. The class *SimManager* is responsible for controlling simulation experiment, e.g.: start/stop/pause an experiment as well as the passage of time (as-soon-as-possible, astronomical and scalable).

The classes from *view/components* package are dedicated to controlling user interface (GUI is implemented using JavaFX and Swing framework). According to the architectural assumptions, the user interface is loosely connected with a simulation layer, thus it is possible to run experiment without GUI (by using event bus) or even to replace GUI without the need of re-implementing the existing model and simulation classes. Figure 3 shows selected components of the GUI level.

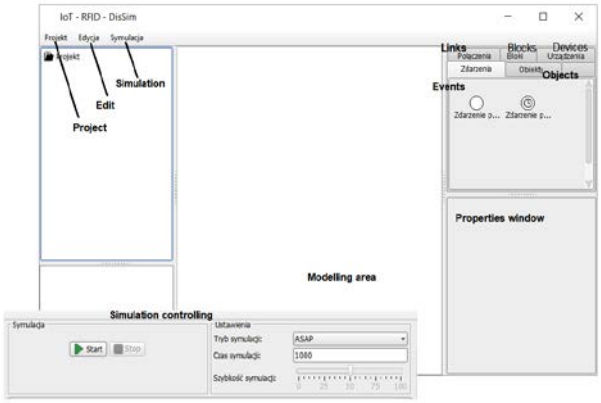

Fig. 3. Selected GUI components.

Form point of view of the IoT-RFID domain, the most important implemented layer of classes is package *simulation*. It contains models of heterogeneous devices what is the crucial part in building IoT-RFID applications. Each modelled component is independently constructed by a dedicated factory whose member is the script interpreter. By changing the script's contents it becomes possible to alter device's behaviour without changing programme code. The package *simulation* forms some kind of library the content of which is successively developed (new models of devices, new behaviour algorithms).

4 Example of DoS attack experiment

To present "in action" the developed simulation software, an experiment regarding DoS attack was carried out. Similarly to [15] we considered a "denial of service" attack case in order to simultaneously simulate hybrid model composed of bandwidth exhaustion, filtering and memory depletion sub-models. The main target of the "denial of service" attack is RFID office system is order to ultimately reduce or finally block legitimate RFID devices (RFIDDevs) from accessing services of RFID tagged document flow management system (RFIDSys). The RFIDSys is a common server for several office rooms equipped with RFID devices and RFID tag readers. We assume that in the offices there is a very intensive work with a large number of tagged documents, which is reflected in a large number of requests per unit of time. The bilateral communication between RFIDSys and readers is realized using open HTTP protocol. That fact makes the system vulnerable and open to DoS attack. In real situation, the attack usually involves different methods thus only a combined simulation model (with a variety of types of DoS attacks) is the most relevant. The other fact, that should be reflected in the simulation model, is that

contemporary servers are secured by filters in firewall devices with many isolation techniques.
For the purpose of the simulation model, we will define the following assumptions regarding the computer network and the server supporting the office RFID system (RFIDSys). Like in [15], the Queueing system theory will be adapted. Proper requests are sent from the environment to the channel according to Poisson stream with intensity "lambdaProperReq". Similarly, requests of attack arrive according to Poisson stream but with "lambdaAttackReq" intensity. The time of request's transmission in the empty transmission channel will be a random variable with exponential distribution (the parameter "lambdaProperT" for a proper request and "lambdaAttackT" for an attack one). The transmission channel supports no more than K request at the same time and rejects subsequent requests when K is already reached. After transmission, the requests are filtered by a firewall filter and the filtering result is random. We define the probability of filtering (rejecting) the proper request by "pProper" and for reporting an attack by "pAttack". The firewall processing time is insignificant for the model. Proper requests that have been rejected in the transmission channel or firewall filter may be re-submitted only by selected RFIDDevs (with probability of re-submission "pResubmission)". For each request arriving at the multi-server (with M stations) one station is reserved for the time of its operation. Request handling times are independent random variables with exponential distribution (for proper requests with "miProperServ" and "miAttackServ" for attack one). The server handles the requests with processor time sharing according to the round-robin scheduling rule (with parameter "dtRR").
The main part of the implementation with the use of IoT-RFID-DisSim package of a defined model has the form of classes with methods and corresponding scripts. An example of small code portion taken from submission event of the RFID reader class might look like at the Figure 4.

```
if (isProper){
  timeToNextReq = parent.getGenerator().
    poisson(lambdaProperReq);
  serviceTime = parent.getGenerator().
    exponential(miProperServ);
  parent.getChannel().insertIntoChannel(new
    Request(simTime(), isProper, serviceTime));
  setRepetitonPeriod(timeToNextReq);
```

Fig. 4. Sample code.

For set values of model parameters:

- K=20; M=50; pResubmission=0.4; dtRR=0.05;
- lambdaProperReq=0.4 [s]; lambdaAttackReq=0.6 [s];
- lambdaProperT=0.3 [s]; lambdaAttackT=0.1 [s];
- pProper=0.02 [s]; pAttack=0.8 [s];
- miProperServ=0.2 [s]; miAttackServ=2 [s];

and a simulation time of 600 [s], selected estimates in one experiment are as follows (Table 2).

Table 2. Selected estimates in the experiment.

	No attack on RFIDSys	**DoS attack on RFIDSys**
Average value of service waiting time for correct request [s]	0.8	10.6
Number of processed proper requests during the experiment	1405	400
Number of processed attack requests during the experiment	Not relevant	225
Number of rejected proper requests during the experiment	36	832
Number of rejected attack requests during the experiment	Not relevant	6070

The experiment shows that performing an attack, even such a short one (10 minutes), leads to significant limitations in server operation. In this case it resulted in more than 3-time reduction in the number of handled proper requests and an about 10-time increase in the waiting time for service. By choosing the values of the simulation parameters it is possible to reflect a number of different working conditions and attack scenarios and finally propose a configuration of the server ensuring the continuity of its operation.

5 Conclusions

In the paper we presented the main concepts which stand behind the software tool for modelling and simulation of DoS attack on the IoT-RFID devices of RFID based office system. This tool is compliant (in a sense of main features) with the broadly available simulators of Internet of Things. However, we focuses on other aspects than the most of the available simulators. We have tried to reflect specificity od RFID based systems and its vulnerability to cyberattacks (in particular denial of service - DoS). The DoS attacks usually involve different methods thus only a combined simulation model (with a variety of types of DoS attacks) is the most relevant. However, the known methods of attack are constantly evolving thus in contrast to most of existing simulation tools our proposition has a capability to modify behaviours or activities by programming Groovy/Python scripts even after building a closed ready-to-use software. It is the technical advantage of the work we have been conducted.

Our further goal is primarily to create a library with rich content of models of devices and behaviour algorithms.

ACKNOWLEDGMENT

This work was partially supported by the National Centre for Research and Development in Poland as a part of the project DOBRBIO4/006/13143/2013.

References

1. C. Heinrich, *RFID and Beyond: Growing Your Business Through Real World Awareness*, Wiley Publishing (2005)
2. G. D'Angelo, S. Ferretti, V. Ghini, *Modeling the Internet of Things: a simulation perspective*, Proceedings of the IEEE HPCS 2017, DOI: 10.1109/HPCS.2017.13 (2017)
3. M. Kiedrowicz, T. Nowicki, R. Waszkowski, Z. Wesolowski, and K. Worwa, *Business processes in the RFID-equipped restricted access administrative office*, DOI: doi.org/10.1051/matecconf/ 20167604003, MATEC Web Conf., **76** (2016)
4. T. Nowicki, K. Chlebicki, D. Pierzchała, K. Worwa and R. Waszkowski, *Simulator for testing hardware and software of the office system with RFID tags*, DOI: doi.org/10.1051/matecconf/201712502009 (2017)
5. Dyk M., Najgebauer A., Pierzchała D., *Agent-based M&S of smart sensors for knowledge acquisition inside the Internet of Things and sensor networks*, Int. Inf. and Database Syst., LNCS, **9012**, Subseries: LNAI, **XXXVI**, 212-223 (2015)
6. Dyk M., Najgebauer A., Pierzchała D., *SenseSim: An Agent-Based and Discrete Event Simulator for Wireless Sensor Networks and the Internet of Things*, 2015 IEEE 2nd WF-IoT Proceedings, ISBN 9781509003655 (2015)
7. M. Dyk, A. Najgebauer, D. Pierzchała, *Augmented perception using Internet of Things*, Information Systems Architecture and Technology. Sel. Asp. of Comm. and Comp. Syst., ISBN 978-83-7493-856-3, 109-118, Wrocław (2014)
8. D. Tagra, M. Rahman and S. Sampalli, *Technique for Preventing DoS Attacks on RFID Systems*, Telecommunications and Computer Networks (SoftCOM), IEEE Xplore, INSPEC Accession Number 11637601 (2010)
9. S.A. Weis, S.E. Sarma, R.L. Rivest, D.W. Engels: *Security and privacy aspects of low-cost radio frequency identification systems, Security in Pervasive Computing*, LNCS, Springer-Verlag, **2802**, 201–212 (2004)
10. D. Pierzchała, *Application of Ontology and Rough Set Theory to Information Sharing in Multi-resolution Combat M&S*, Studies in Computational Intelligence, **551**, 193–203, Springer (2014)
11. B. Musznicki, P. Zwierzykowski, *Survey of Wireless Sensor Networks Simulation Tools for Demanding Applications*. International Journal of Grid and Distributed Computing, **5**, **3** (2012)
12. F. Osterlind, A. Dunkels, J. Eriksson, N. Finne, and T. Voigt, *Cross-level sensor network simulation with COOJA*, Proceedings of SenseApp 2006, Tampa, USA (2006)
13. A. Sobeih, W.-P. Chen, J. C. Hou, L.-C. Kung, N. Li, H. Lim, H.-Y. Tyan, and H. Zhang, *J-Sim: A Simulation and Emulation Environment for Wireless Sensor Networks*, J-Sim documentation
14. Mehdi, K., Lounis, M., Bounceur, A., Kechadi., T., *Cupcarbon: A multiagent and discrete event wireless sensor network design and simulation tool*, 7th SIMUTools14), Lisbon (2014)
15. Ramanauskaitė S.,Čenys A., *Composite DoS attack model*, ISSN 2029-2341, 20-26 (2012)
16. http://ec.europa.eu/information_society/tl/help/glossary/index_en.htm#c (last enter 03.2018)
17. J. R. Vacca, *Security for the Internet of Thing*, Computer and information security, 3-rd ed. (2017)
18. M. Kiedrowicz, *Multi-faceted methodology of the risk analysis and management referring to the IT system supporting the processing of documents at different levels of sensitivity,* MATEC Web of Conferences, vol. 125, DOI: 10.1051/matecconf/201712502010, Greece (2017)
19. *M. Kiedrowicz, J. Stanik, Selected aspects of risk management in respect of security of the document, lifecycle management system with multiple levels of sensitivity*, B. Kubiak, J. Maślankowski (eds.) Information Management in Practice, pp. 231-248, ISBN 978-83-64669-05-7, Poland (2015)