computing@tanet.edu.te.ua
www.tanet.edu.te.ua/computing

ISSN 1727-6209
International Scientific
Journal of Computing

AGENT-BASED SIMULATION OF DDOS ATTACKS AND DEFENSE MECHANISMS

Igor Kotenko [1], Alexander Ulanov [2]

St.-Petersburg Institute for Informatics and Automation of Russian Academy of Sciences
39, 14th Liniya, St. Petersburg, 199178, Russia
[1] ivkote@iias.spb.su, http://space.iias.spb.su/ai/kotenko/
[2] ulanov@iias.spb.su, http://space.iias.spb.su/ai/ulanov/

Abstract: *The paper considers an approach to modeling and simulation of cyber-wars in Internet between the teams of software agents. According to this approach, the cybernetic opposition of malefactors and security systems is represented by the interaction of two different teams of software agents – malefactors' team and defense team. The approach is considered by an example of modeling and simulation of "Distributed Denial of Service" (DDoS) attacks and protection against them. The paper also describes the software environment for multi-agent simulation of defense mechanisms against DDoS attacks developed by the authors and different experiments. The main components of the software environment are outlined. One of the numerous experiments on protection against DDoS attacks is described in detail. The environment developed is based OMNeT++ INET Framework.*

Keywords: *Agents, Agent-based Modeling and Simulation, Computer network attacks, Distributed Denial of Service, Defense mechanisms*

1. INTRODUCTION

Vulnerabilities of present distributed computer systems, permanently magnified quantity, variety and complexity of cyber-attacks and gravity of their consequences highlight urgent necessity for information assurance and survivability of computer systems. Especially it is fair in connection with integration of computer systems on the basis of the Internet, permanently modified and magnified, not having state boundaries, centralized control and uniform security policy.

Experienced malefactors realize sophisticated strategies of cyber-attacks. These strategies can include:

- Information gathering about the computer system under attack, detecting its vulnerabilities and defense mechanisms;
- Determining the ways of overcoming defense mechanisms (for example, by simulating these mechanisms);
- Suppression, detour or deceit of protection components (for example, by using slow ("stretched" in time) stealthy probes, separate coordinated operations (attacks) from several sources formed complex multiphase attack, etc.);
- Getting access to resources, escalating privilege, and implementation of thread intended (violation of confidentiality, integrity, availability, etc.) using the vulnerabilities detected;
- Covering tracks of malefactors' presence and creating back doors in order to use them later.

Protection mechanisms should support real-time fulfillment of the following operations:

- Implementing the protection mechanisms appropriated to the security policy (including proactive intrusion prevention and attack blocking, misinformation, concealment, camouflage, etc.);
- Vulnerability assessment, gathering data and analysis of the current status of the computer system defended;
- Intrusion detection and prediction of the malefactors' intentions and actions;
- Direct incident response, including deception of the malefactors, their decoy with the purpose of disclosure and more precise

determining the malefactors' purposes, and reinforcement of critical protection mechanisms;

- Elimination of intrusion consequences and detected vulnerabilities, adaptation of the information assurance system to the next intrusions.

One of the most harmful classes of attacks aiming at destruction of network resources availability is "Denial of Service" (DoS). The purpose of DoS is isolation of a victim host. As a result of this attack the legitimate users can not access necessary network resources. Most of operating systems (OS), routers and network components are prone to DoS attacks that are hard to prevent.

The new type of attack arrived in the beginning this century. It is called "Destributed Denial Of Service" (DDoS). To perform DDoS attacks malefactor needs to hack a set of computers ("zombies") at first and to run on them DoS programs to attack next targets. This makes hard to detect DDoS attack and to defense from it. The DDoS domain is becoming more and more complex. We observe now the great variety of different DDoS attacks and the continuous appearance of new types that break the defense.

The presence of many attacking hosts complicates DDoS attack detection and defense. A lot of *defense issues are under solution now*:

- How DDoS attacks occur?
- What new attacks can be applied?
- Why it is so hard to resist DDoS attacks?
- How good are the present defense mechanisms for DDoS detection, prevention and reaction?
- What recommendations could be offered to create effective defense?

The main task of defense systems against DDoS is to accurately detect these attacks and quickly respond to them [26]. It is equally important to recognize the legitimate traffic that shares the attack signature and deliver it reliably to the victim [17]. *Traditional defense* include detection and reaction mechanisms [28]. Different network characteristics are used for detection of malicious actions (for example, source IP address [22], traffic volume [5], and packet content [21]). To detect abnormal network characteristics, many methods can be applied (for instance, statistical [14], cumulative sum, pattern matching, etc). As a rule, the reaction mechanisms include filtering [20], congestion control [15] and traceback [13].

But, as a result of several reasons (detection of DDoS attack is most accurate close to the victim, separation of legitimate is most successful close to the sources, etc.), adequate victim protection to constrain attack traffic can only be achieved by *cooperation of different distributed components* [17]. So, the DDoS problem requires a distributed cooperative solution [16, 17]. There are a lot of architectures for distributed cooperative defense mechanisms [1, 2, 21, 9, 27, 26, 17, etc.]. For example, the paper [1] proposes a model for an Active Security System, comprising a number of components that actively cooperate in order to effectively react to a wide range of attacks. COSSACK [21] forms a multicast group of defense nodes which are deployed at source and victim networks. The Secure Overlay Services (SOS) system [9] uses a combination of secure overlay tunneling, routing via consistent hashing, and filtering. A collaborative DDoS defense system proposed in [27] consists of routers which act as gateways. The distributed defense system described in [26] protects web applications from DDoS attacks. The DefCOM system [17] uses a peer-to-peer network of cooperative defense nodes. DefCOM nodes are classified into three categories: Alert generator nodes, Rate limiter nodes, and Classifier nodes.

On our opinion, it is possible to answer soundly on the questions about defense against DDoS attacks by *modeling and simulation of present and new DDoS attacks and defense mechanisms*.

This paper describes an approach and an environment for multiagent simulation of such mechanisms elaborated by authors.

The rest of the paper is structured as follows. *Section 2* outlines suggested common agent-based approach for modeling and simulation. *Section 3* describes the issues of modeling and simulation of DDoS attacks and defense mechanisms. The software environment developed for simulation is presented in *section 4*. *Section 5* presents one of simulation scenarios fulfilled. *Conclusion* outlines the main results of the paper and future work directions.

2. APPROACH FOR MODELING AND SIMULATION

Agent-based modeling and simulation of network security in the Internet assumes that agents' competition is represented as a large collection of semi-autonomous interacting agents [11, 12]. The *aggregate system behavior* emerges from evolving local interactions of agents in a dynamically changing environment specified by computer network model.

We assume to select at least *two antagonistic agents' teams* effecting on computer network as interconnected set of resources and each other: the team of agents that realize the DDoS attack and the defense team.

The problem of multi-agent modeling of cybernetic opposition processes is represented as modeling of antagonistic interaction of the agents-malefactors' team and the defense team.

The goal of agents-malefactors is to determine the vulnerabilities of the computer network and the defense system. Then they are to apply the given set of information security threats due to execution of distributed coordinated attacks.

The goal of defense team is to defend the network and their own components.

The agents of different teams compete to reach the opposite intentions. The agents of one team cooperate to realize the overall intention (implementing the threat or defense of computer network).

Our approach is based on *agent teamwork framework* [3, 4, 7, 10, 24, 25, 29]. It is said that the agents' team realizes teamwork, if the team members (agents) fulfill joint operations for reaching the common long-time goal in a dynamic external environment at presence of noise and counteraction of opponents. Now the research on teamwork is an area of steadfast attention in multi-agent systems.

It is offered that each team of agents is organized by the group (team) plan of the agents' actions. As result, a team has a mechanism of decision-making about who will execute particular operations. As in the joint intention theory [3], the basic elements, allowing the agents' team to fulfill a common task, are common (group) intentions, but its structuring is carried out in the same way as the plans are structured in the shared plans theory [6]. The mechanisms of the agents' interaction and coordination are based on three groups of procedures [24]:

(1) Coordination of the agents' actions (for implementation of the coordinated initialization and termination of the common scenario actions);

(2) Monitoring and restoring the agents' functionality;

(3) Communication selectivity support (for choice of the most "useful" communications).

The specification of the plan hierarchy is carried out for each role. The following elements of the plan should be described: initial conditions, when the plan is offered for fulfillment; conditions for finishing the plan execution (these conditions can be as follows: plan is fulfilled, plan is impracticable or plan is irrelevant); actions fulfilled at the team level as a part of the common plan. For the group plans it is necessary to express joint activity.

Assignment of roles and allocation of plans between the agents is fulfilled in two stages: at first the plan is arranged in terms of roles, and then the roles are put in correspondence to the agents. Agents' functionalities are generated automatically according to the roles specified.

The adversary (malefactors') team co-evolves by generation of new attack patterns to overcome defenses. On the other hand, defense team co-evolves by generating new protective actions against attacks, suppression of malefactors' team and recovery of destructed and compromised components of the computer network.

Interaction among agents can be represented as a two-player game ("game of network cats and mice"), where the agents' objective is to look for a strategy that maximizes their expected sum of rewards in the game.

To cope with the information heterogeneity and distribution of intrusion sources and agents used we apply ontology-based approach and special protocols for specification of shared consistent terminology.

The developed common ontology of DDoS attacks comprises a hierarchy of notions specifying activities of team of malefactors directed to implementation of attacks in different layers of detail. In this ontology, the hierarchy of nodes representing notions splits into two subsets according to the *macro- and micro-layers* of the domain specifications. All nodes of the ontology of DDoS attacks on the macro- and micro-levels of specification are divided into the *intermediate* and *terminal*.

The notions of the ontology of an upper layer can be interconnected with the corresponding notions of the lower layer through one of three kinds of *relationships*: "*Part of*" that is decomposition relationship ("*Whole*"–"*Part*"); "*Kind of*" that is specialization relationship ("*Notion*"–"*Particular kind of notion*"); and "*Seq of*" that is relationship specifying sequence of operation ("*Whole operation*" – "*Sub-operation*").

High-layer notions corresponding to the intentions form the upper layers of the ontology. They are interconnected by the "*Part of*" relationship. Attack actions realizing malefactor's intentions (they presented at the lower layers as compared with the intentions) are interconnected with the intentions by "*Kind of*" or "*Seq of*" relationship.

The "terminal" notions of the macro-level are further elaborated on the *micro-level of attack specification*, and on this level they belong to the set of top-level notions detailed through the use of the three relationships introduced above.

In micro specifications of the computer network attacks ontology, besides the three relations described ("*Part of*", "*Kind of*", "*Seq of*"), the relationship "*Example of*" is also used. It serves to establish the "type of object– specific sample of object" relationship.

The developed ontology includes the detailed description of the DDoS domain in which the notions of the bottom layer ("*terminals*") are specified in terms of network packets, OS calls, and audit data.

Nodes specifying a set of software exploits for generation of DDoS attacks (Trinity V3, MSTREAM, SHAFT, TFN2K, Stacheldraht, Trin00) make up a top level of the ontology fragment. At lower levels different classes of DoS-attacks are detailed, for example: "Ack flood" (sending a huge number of network packets with Ack parameter), "Land" attacks (sending an IP-packet with equal fields of port and address of the sender and the receiver, i.e. Source Address = Destination Address, Source Port Number = Destination Port Number), "Smurf" (sending broadcasting ICMP ECHO inquiries on behalf of a victim host, therefore hosts accepted such broadcasting packages answer to the victim host, that results in essential capacity reduction of a communication channel or in full isolation of an attacked network), etc.

Common formal plan of attacks implemented by team of malefactors-agents has three-level structure:

(1) Upper level is a level of intention-based scenarios of malefactors' team specified in terms of sequences of intentions and negotiation acts;

(2) Middle level is a level of intention-based scenarios of each malefactor specified in terms of ordered sequences of sub-goals;

(3) Lower level is a level of malefactor's intention realization specified in terms of sequences of low-level actions (commands).

The suggested technology for creation of the malefactors-agents' team (that is fair for other subject domains) consists in realization of the following chain of stages:

(1) Formation of the subject domain ontology;

(2) Determination of the agents' team structure;

(3) Determination of agents' interaction-and-coordination mechanisms (including roles and scenarios for roles exchanges);

(4) Specification of agents' plans as a hierarchy of stochastic formal grammars;

(5) Assignment of roles and allocation of plans between agents;

(6) Implementation of teamwork.

3. ISSUES OF MODELING AND SIMULATION OF DDOS ATTACKS AND DEFENSE MECHANISMS

The idea of DDoS attack consists in reaching the global goal – the denial of service of some resource – due to joint efforts of many components that are acting on attack side. In that way the initial goal is divided into more simple sub-goals. They are given to particular components (agents). At the same time the goal on the top level stays shared between agents. On the low level, the local goals are formed. Their achievement is targeted on solving the shared task. The agents interact with each other to coordinate local solutions. This is necessary to reach the needful quality of solution of shared goal "denial of service". In the case when the attack is controlled by a malefactor, a component for coordination of agent-attackers from the side of a malefactor is needed.

Generally, the components of DDoS attack system are the programs which have the following features: autonomy; the presence of initial knowledge about itself, interacting entities and environment; the presence of knowledge (or hard-coded algorithm) that allows to get and process the external data from environment; the presence of a goal and a list of actions to reach this goal; the communication and interaction mechanisms (protocols) to reach the shared goal. These properties let to represent every component of the system as an intelligent agent and the set of agents as the agent team.

Let us represent the DDoS attack system as an agent team. The agents aim the shared goal – the realization of attack "denial of service" for some host or network. Analyzing the present methods of DDoS realization it is possible to determine at least two types of the attack system components:

- "Daemon" – it executes the attack directly;
- "Master" – it coordinates the actions of other system components.

The analysis of present DDoS defense systems shows the following their features:

- The defense systems are built of basic components which have some local meaning but serve together for common shared goal;
- The number and functionality of defense system components depend on the place of their deployment;
- As a rule, the defense systems have a hierarchical structure, where different levels serve for particular sub-tasks of the complex defense goal.

The general approach to the DDoS defense is the following. The information about normal traffic is collected from different network sensors. Then the analyzer-component compares in real-time the current traffic with the normal traffic. The system tries to trace back the source of anomalies (due to "traceback" mechanisms) and generates the recommendations how to cut off them or how to lower the quantity of these anomalies. Depending on security administrator's choice, the system applies some countermeasure.

Let us represent the DDoS defense system as a team of intelligent agents. The agents aim the common shared goal. The goal is to defense the given host or network from DDoS attacks. In compliance with the general approach we set the following defense agent classes:

- "Sensor" - agent of initial information processing;
- "Detector" - attack detection agent;
- "Filter" - agent of attack traffic filtering;
- "Investigator" - agent of attack investigation.

The defense team consists of the given number of sensors. Sensors are deployed in the given network places to monitor the network processes and to collect the statistic data. The data received are transmitted to detectors for recognizing anomalies and DDoS attacks.

Detectors decide if there is a danger of DDoS attack and from which hosts does it come. They transmit this information to filters and (or) investigators.

Filters are deployed on the way of packets flowing to the defended host or network. Filters can use different mechanisms of filtering of malicious packets.

Investigators try to trace back the sources of DDoS attack and to neutralize them by defeating the corresponding attack agents.

4. SOFTWARE ENVIRONMENT FOR SIMULATION

To choose the simulation tool the comprehensive analysis of the following systems was made: NS2 [18], OMNeT++ INET Framework [19], SSF Net [23], J-Sim [8] and some others.

We used the following main requirements to the simulation environment:

- The detailed implementation of the protocols that are engaged in DDoS attacks. It is necessary at least to simulate the present DDoS attacks.
- The ability of writing and plugging in the personal modules. It is necessary to implement the agent approach.
- The ability of changing parameters during the simulation.
- Implementation for OS Windows and Linux (or platform-independency).
- Advanced graphical interface.
- Free for use in research and educational purposes.

We discovered that the OMNET++ INET Framework satisfies to these requirements best of all. OMNET++ is the discrete event simulator [19]. The change of state happens in the discrete moments of time. The simulation is being held by the future event list sorted by time. The event may be: the beginning of packet transmission, time-out, etc. The events occur inside the simple modules. Such modules have the functions of initialization, message processing, action (alternatively), end of work. The exchange of messages between modules happens due to channels (modules are connected with them by the gates) or directly by gates. A gate can be incoming or outgoing to receive or to send messages accordingly.

The agents were implemented as the compound modules. They contain the simple modules and the agent kernel. The simple modules are responsible for functioning of various network protocols. The agent kernel controls these modules in each agent. The agent (as the OMNET compound module) has a number of gates for connecting to standard network host from INET Framework. These gates are related to the corresponding network protocols. The connection or the deploying of the agent can take place during the simulation.

OMNET gives two alternatives to implement the module: by the message handling or by the activity. In the first case, the actions of the module are bounded to the messages arrival. The next event can happen only after the finalization of work of function that handles the messages. In the second case, the actions of the module are executed as co-routine. It allows the arbitrary embranchment into other contexts of control and the arbitrary recommencement of thread from the point of embranchment. In addition, there is the ability of describing the module actions by state machines.

The agent kernels were made as co-routines, as it is convenient for implementing the interaction protocols (on which the agent teamwork is based). The other modules were made as the handlers of events from the kernel and environment. The state machines were rejected since they make code harder to read. They also make the logics of the program

implicit sometimes. This drawback could be avoided if there were the graphical editor of state machines.

Now we are on the process of development and improvement of the environment for multi-agent simulation of DDoS attack and defense mechanisms on the basis of OMNeT++ INET Framework. We have modified the existing OMNeT++ INET Framework. For example, the following new modules have been created:

- The filtering table. It allows simulating the defense side actions on filtering network packets on the network layer.
- The "sniffer" module. It allows scanning all traffic for the given host to collect statistics. It is also used to simulate the defense side actions.

To simulate the attack and defense mechanisms, the modules responsible for sockets operating have been also changed.

The environment windows used during simulation are depicted in Fig.1.

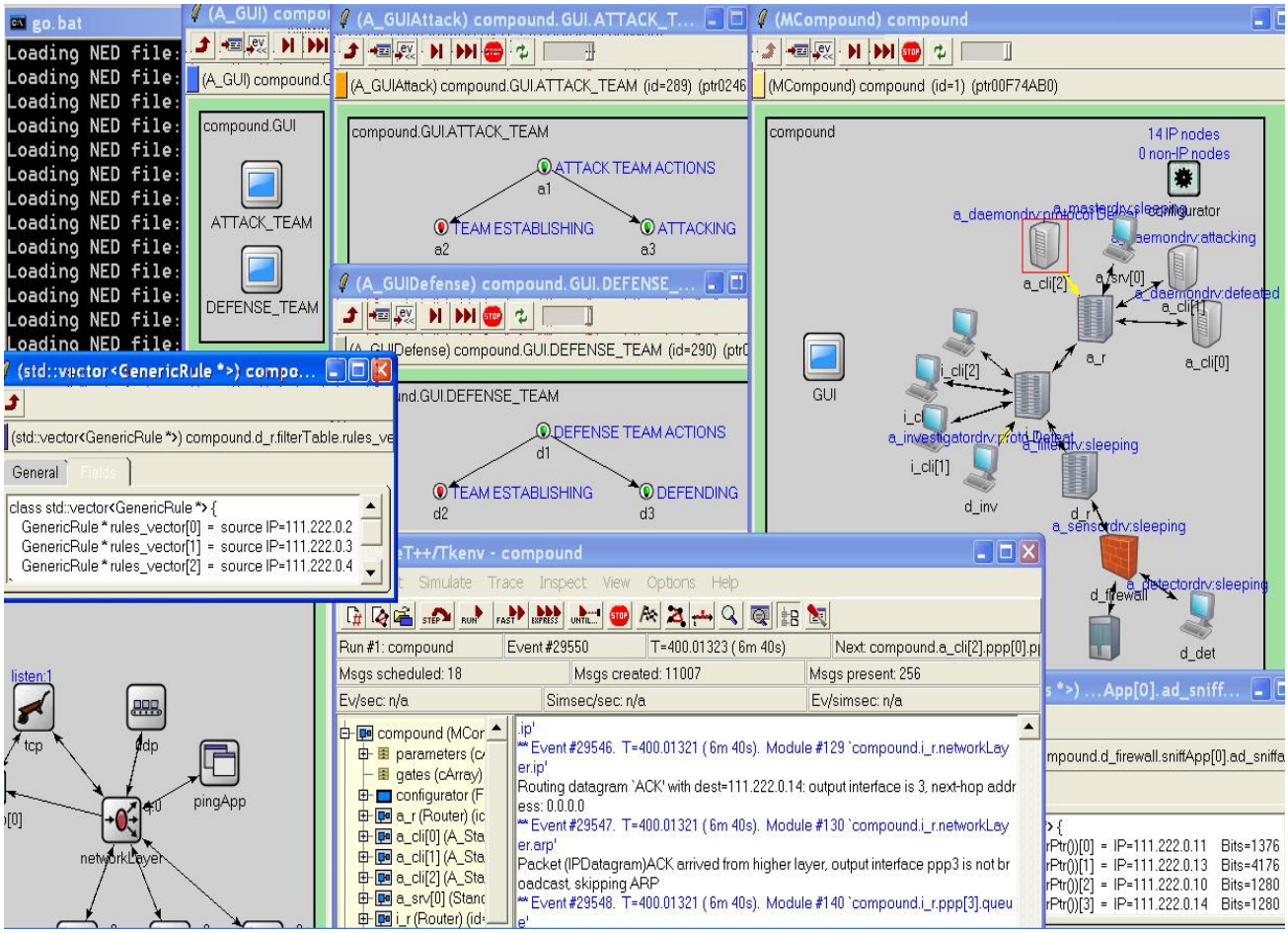

Fig.1 – Examples of representation of windows used during simulation process

At the basic window of visualization (fig.1, at upper right), a simulated computer network is displayed. The network represents a set of the hosts and channels. Hosts can fulfill different functionality depending on their parameters or a set of internal modules. Internal modules are responsible for functioning of protocols and applications at various levels of OSI model. Hosts are connected by channels which parameters can be changed. Applications (including agents) are established on hosts. Applications are connected to corresponding modules of protocols.

The window for simulation management (at the bottom of fig.1, in the middle) allows looking through and changing simulation parameters. Corresponding status windows (on top of fig.1, in the middle) show the current status of agents' teams. It is possible to open different windows which characterize functioning (the statistical data) of particular hosts, protocols and agents, for example, at the bottom left of fig.1, the window of one of the hosts is displayed.

Since all simulated processes take place in the Internet, the network model should be in the heart of simulation environment. One of the examples of computer networks for simulation is represented in Fig.2. We used different configurations of computer networks which include from 14 till 1000 and more

nodes. Each network is represented as a set of hosts connected by the channels. Each host can possess different functionalities depending on hosts parameters or the set of internal modules.

The hosts are connected with the channels. Their parameters can be changed. They are as follows:

- Delay – delay of packets propagation;
- Datarate – the speed of packets transmission.

Each network host (Fig.3) can consist of the following modules:

- ppp is responsible for the data link layer (the router can have several ppp according to the number of interfaces);
- networkLayer is responsible for the network layer;
- pingApp is responsible for applications using ICMP;
- tcp is serving for TCP;
- udp is serving for UDP;
- tcpApp[0] is the TCP application (there can be a number of them);
- notificationBoard is used for logging the events on host;
- interfaceTable contains the table of network interfaces;
- routingTable contains the routing table;
- filterTable contains the filtering table.

The applications (including the agents) are being installed on the hosts by connecting to appropriate protocol modules.

Each network for simulation consists of three sub-networks:

- The subnet of defense where the defense team is deployed;
- The intermediate subnet where the standard hosts are deployed. They produce the generic (normal) traffic in the network including the traffic to defended host;
- The subnet of attack where the attack team is deployed.

Subnet of defense (on the top of Fig.2) consists of five hosts. The following agents are deployed on the first four hosts: detector, sensor, filter and investigator. The web-server which is under defense is deployed on the fifth host. The agents and the web-server are the applications installed on the corresponding hosts. The IP-addresses are being set automatically. It is necessary to set the other application parameters.

Web-server is deployed on the host d_srv. The interaction port and the answer delay must be set. (Web-server module is from INET Framework.)

Detector is deployed on the host d_det (see Fig.2). The following parameters are used for detector: the defended host IP-address, the port for team interaction, the interval for sensor inquiry, and the maximum allowed data-rate to server (BPS, bit per second).

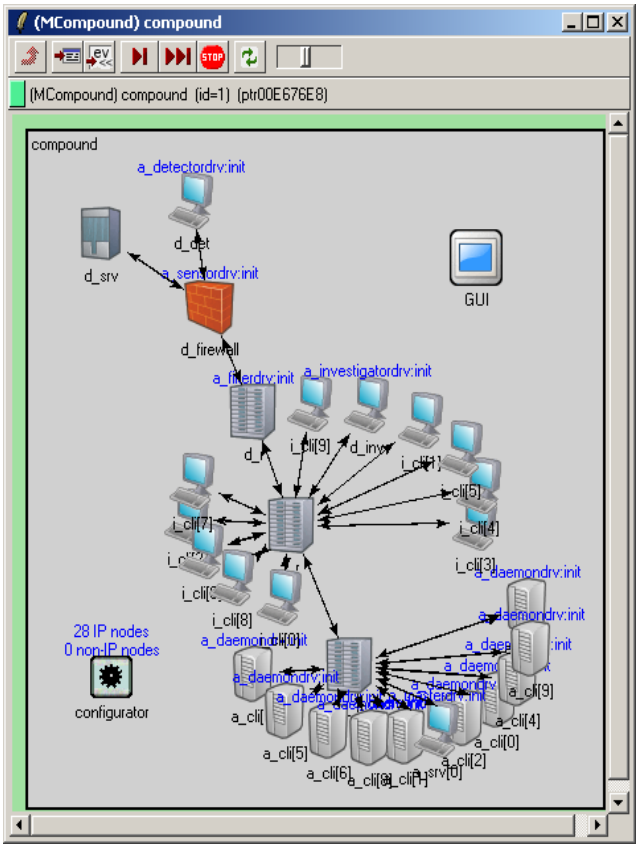

Fig.2 – Example of representation of a computer network configuration

Sensor is deployed on the host d_firewall (on the entrance to the server subnet). Filter is deployed on the host d_r (router). Investigator is deployed on the host d_inv. For each of the last three agents, the private port, detectors IP-address and the port for team interaction must be set.

The intermediate subnet (in the middle of Fig.2) consists of N hosts i_cli[…] with generic clients. They are connected by the router i_r. The number of hosts N is the modeling parameter which can be set. The following parameters of clients must be used: IP-address and port of server, the time of start of work, the quantity and size of requests while connecting to server, the size of reply and the time of reply preparation, the idle interval. (The generic client is used from INET Framework.)

The subnet of attack (in the bottom of Fig.2) consists of M hosts i_cli[…] with daemons deployed and one host with master deployed. The number of

hosts M must be set. Master has the following parameters: port for team interaction, IP-address and port of attack target, the time of start of attack and its rate (measured in packets per second). Daemon has the following parameters: the port, masters' IP-address and port for team interaction.

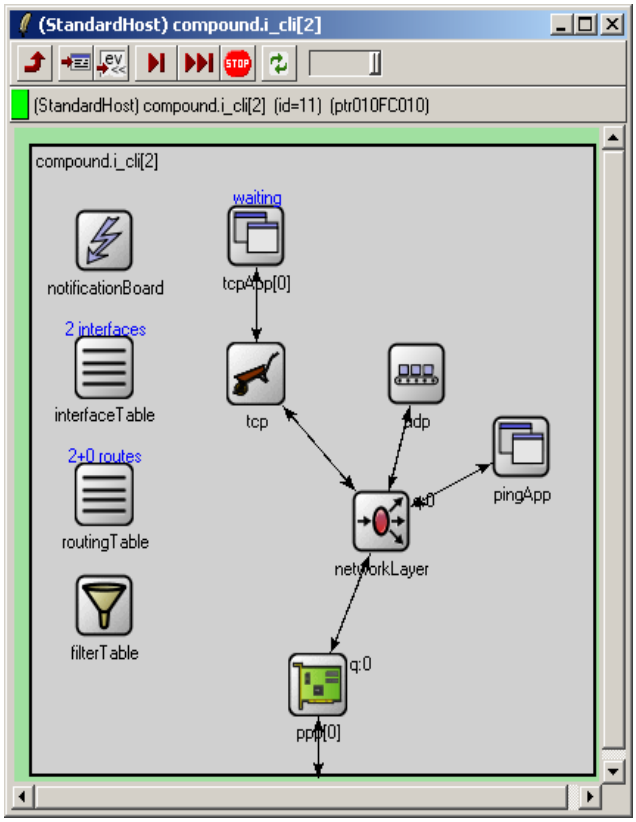

Fig.3 – Representation of generic network host

5. EXAMPLE OF SIMULATION SCENARIO

Let examine one of simulation scenarios. The network for this simulation scenario is represented in Fig.4.

The routers in this network are connected with each other with the fiberglass channels with bandwidth 512 Mbit. The other hosts are connected by 10 Mbit Ethernet channels.

In the defense subnet, server, detector, sensor, filter and investigator are deployed (in the bottom of Fig.4, see the blue signs above the corresponding hosts). The server deployed on d_srv provides some service on the port #80 with the delay of reply = 0. The parameters for detector are: defended host – d_srv, port #2000, interval of sensor poll – 60 sec, BPS=1100 bit/s. Sensor, filter and investigator have the following parameters: port for interaction #2000, detectors address and port – d_det, #2000.

Some time after the start of simulation, clients begin to send the requests to server and it replies. The packet to the server is shown in Fig.4 by the red circle. That is the way generic (normal) network traffic is generated and depicted.

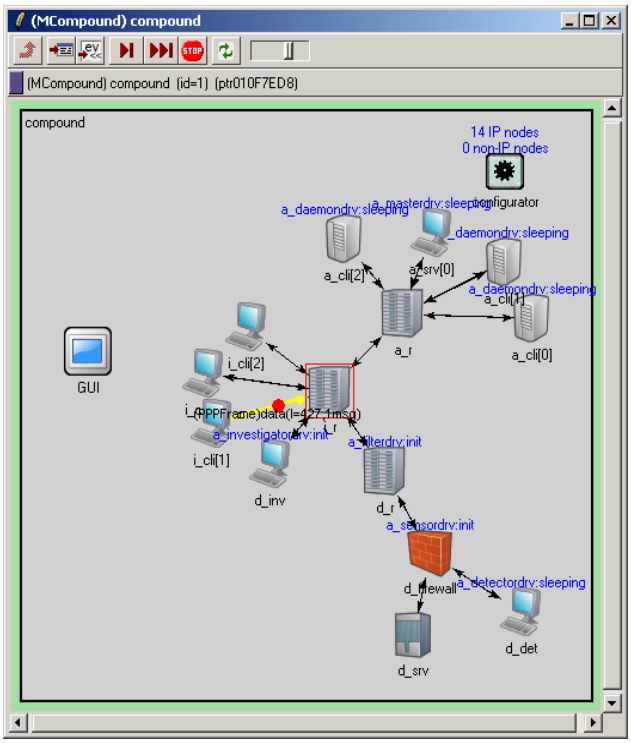

Fig.4 – Initial moment of simulation

Formation of the defense team begins some time after the start of simulation. The defense agents (investigator, sensor and filter) connect to detector. They send to detector the messages saying that they are alive and ready to work. Detector stores these messages to its memory. Formation of the attack teams happens the same way.

The fragment of master's knowledge base after the team was established is represented in Fig.5. This knowledge base contains (see Fig.5) IP-addresses and ports for message exchange between agents, as well as the state of the agents.

The defense team actions begin after this team formation. Sensor starts to collect the traffic statistics for every IP-address (the amount of transmitted bytes).

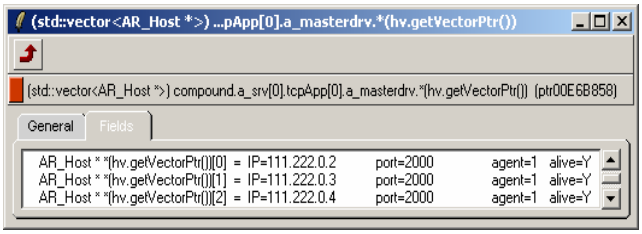

Fig.5 – Master's knowledge after the attack team formation

Detector requests data from sensor every S seconds (for example, 60 sec). It gets statistics and detects if there is an attack. Then it connects to filter and investigator and sends them the IP-addresses of suspicious hosts. While there is no attack they stay

idle.

Some time (for instance, 300 seconds) after the start of simulation, the attack team begins the attack actions. At first, master requests every daemon if it is alive and ready to work. When all daemons were examined, it occurs that they all are workable. Master calculates the rate of attack for every daemon.

The given rate (2 packets per second) is divided by the amount of workable daemons (3). The result is the individual rate of attack for each daemon. Then master sends the following attack command for every daemon: the address of attack target (d_srv), the port (#2000), the rate (0.67). Daemons start the attack by sending, for example, UDP packets to the target with the given rate. The message "attacking" is represented above the attacking daemons (Fig.8).

The regular request from detector to sensor happens approximately 100 sec later. In this moment sensor generates for each IP address the amount of transmitted bits for 60 last seconds (Fig.6).

Detector calculates BPS parameter for every host excluding the server (111.222.0.12). Obviously this parameter exceeds the maximum allowed value (1100) for the host with the following IP-addresses: 111.222.0.4, 111.222.0.3, 111.222.0.2.

Detector sends these addresses to filter and to investigator. Filter must set the rules to reject the packets from these IP-addresses. Investigator must trace the source of attack to defeat attack agents. After filter sets its rules (Fig.7) and begin to protect against the attack, the amount of traffic to server will lower.

Investigator tries to defeat the attack agents. It can be seen in Fig.8 that it succeeded to defeat one of daemons. Above it appears the message "defeated". Then investigator tries to defeat another daemon. Above investigator there is the message "Proto defeat". The path of packets from investigator to daemon is shown by yellow arrows.

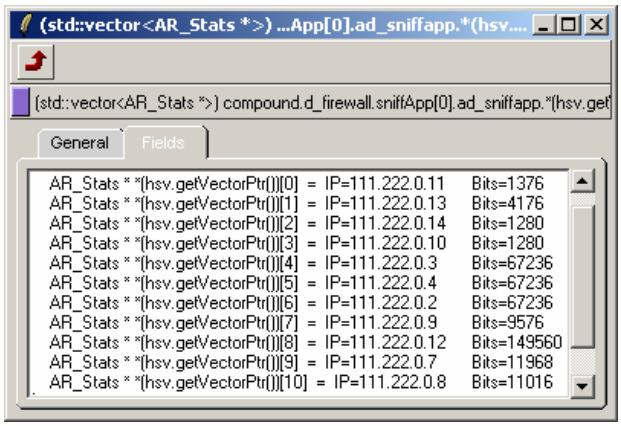

Fig.6 – Data formed by sensor during the attack

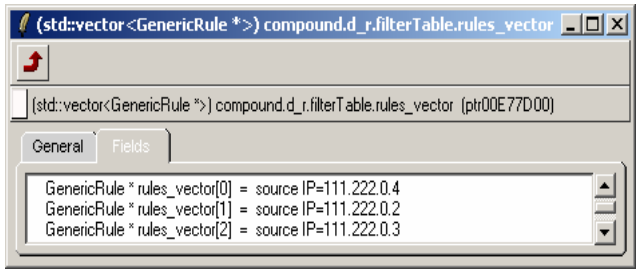

Fig.7 – Filtration rules applied by filter

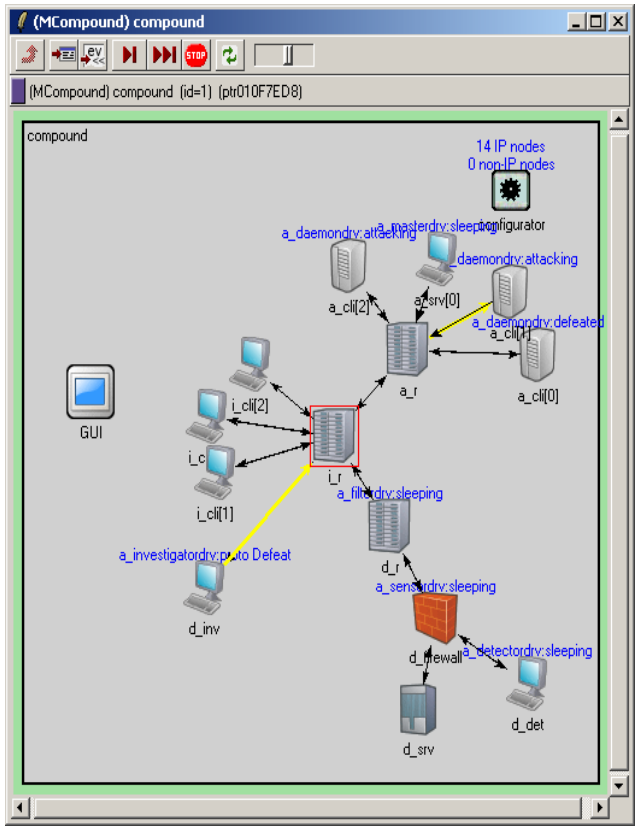

Fig.8 – Actions of agent-investigator during the attack

As the result, investigator succeeded to defeat two daemons. After this the state of traffic returns to the normal value - just like before the attack (Fig.9).

The last functioning daemon continues the attack. Master had redistributed the attack rate to him after other daemons were defeated. However attack packets do not reach the target. They are rejected on the entrance to the defended network by filter.

The main facts about this scenario are as follows: the attack was blocked 1 min 40 sec after start; three rules of filtration were applied; two attack agents (daemons) were defeated.

The graph of relationship between bits transmitted to server d_r subnet and the time is represented in Fig.10.

The main traffic in the interval 0–300 seconds was created by clients requests and servers replies. This process is represented by the vertical straights with low rate (Fig.10).

When the attack happens (the mark 300 seconds),

the intense traffic appears. The attack process is represented by the plateau from 300 to 400 seconds.

However, approximately on the 400th second the filtration rules were applied, and the malicious packets begun to be rejected on the entrance to the server subnet. After that the normal state of network returned.

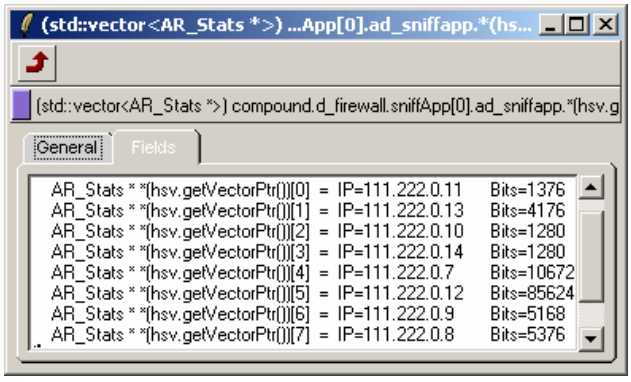

Fig.9 – Data formed by sensor after applying filtration rules

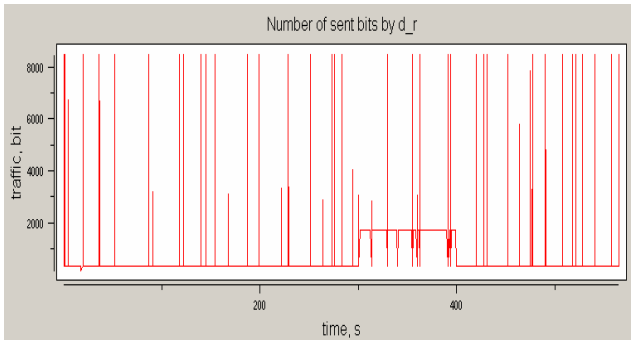

Fig.10 – Relationship of bits transmitted to server d_r from time

6. CONCLUSION

The main results of the work we described in the paper consist in developing basic ideas on multi-agent modeling and simulation of defense mechanisms against DDoS attacks and implementing corresponding software environment.

The environment developed is written in C++ and OMNeT++. It allows imitating a wide spectrum of real life DDoS attacks and defense mechanisms.

Different experiments with this environment have been fulfilled. These experiments include the investigation of attack scenarios and protection mechanisms for the networks with different structures and security policies. One of the scenarios of these experiments was demonstrated in the paper.

Future work is connected with developing formal basis for agent-based modeling and simulation of cyber agents' team competition in the Internet, building more realistic environment (including improvement of capabilities of the attack and defense agents teams by expansion of the attack and defense classes, and implementing more sophisticated attack and defense scenarios), and conducting experiments to both evaluate computer network security and analyze the efficiency and effectiveness of security policy against different attacks.

7. ACKNOWLEDGMENT

This research is being supported by grants 04-01-00167 of Russian Foundation of Basic Research and partly funded by the EC as part of the POSITIF project (contract IST-2002-002314).

8. REFERENCES

[1] R. Canonico, D. Cotroneo, L. Peluso, S.P. Romano, G. Ventre. Programming routers to improve network security. *Proceedings of the OPENSIG 2001 Workshop Next Generation Network Programming,* 2001.

[2] S. Chen, Q. Song. Perimeter-Based Defense against High Bandwidth DDoS Attacks. *IEEE Transactions on Parallel and Distributed Systems*, Vol.16, No.7, 2005.

[3] P.R. Cohen, H.J. Levesque. Teamwork, *Nous,* Vol.25, No.4, 1991.

[4] X. Fan, J. Yen. Modeling and Simulating Human Teamwork Behaviors Using Intelligent Agents, *Journal of Physics of Life Reviews*, Vol. 1, No.3, 2004.

[5] [5] T.M. Gil, M. Poletto. MULTOPS: a data-structure for bandwidth attack detection. *Proceedings of 10th Usenix Security Symposium*, 2001.

[6] B. Grosz, S. Kraus. Collaborative plans for complex group actions, *Artificial Intelligence*, Vol.86, 1996.

[7] N. Jennings. Controlling cooperative problem solving in industrial multi-agent systems using joint intentions, *Artificial Intelligence*, No.75, 1995.

[8] J-Sim homepage. www.j-sim.org

[9] A.D. Keromytis, V. Misra, D. Rubenstein. SOS: An architecture for mitigating DDoS attacks, *Journal on Selected Areas in Communications*, Vol. 21, 2003.

[10] I. Kotenko, L. Stankevich. The Control of Teams of Autonomous Objects in the Time-Constrained Environments. *Proceedings of the IEEE International Conference "Artificial Intelligence Systems",* IEEE Computer Society, 2002.

[11] I. Kotenko. Agent-Based Modeling and Simulation of Cyber-Warfare between Malefactors and Security Agents in Internet. *19th European Simulation Multiconference "Simulation in wider Europe"*, 2005.

[12] I. Kotenko, A. Ulanov. Multiagent modeling and simulation of agents' competition for network resources availability. *Second International Workshop on Safety and Security in Multiagent Systems,* Utrecht, The Netherlands, 2005.

[13] V. Kuznetsov, A. Simkin, H. Sandström. An evaluation of different ip traceback approaches. *Proceeding of the 4th International Conference on Information and Communications Security,* 2002.

[14] M. Li, C.H. Chi, W. Zhao, W.J. Jia, D.Y. Long. Decision Analysis of Statistically Detecting Distributed Denial-of-Service Flooding Attacks, *Int. J. Information Technology and Decision Making*, Vol.2, No.3, 2003.

[15] R. Mahajan, S.M. Bellovin, S. Floyd. Controlling High Bandwidth Aggregates in the Network, *Computer Communications Review*, Vol.32, No.3, 2002.

[16] J. Mirkovic, S. Dietrich, D. Dittrich, P. Reiher. *Internet Denial of Service: Attack and Defense Mechanisms*. Prentice Hall PTR, 2004.

[17] J. Mirkovic, M. Robinson, P. Reiher, G. Oikonomou. Distributed Defense Against DDOS Attacks. *University of Delaware CIS Department Technical Report CIS-TR-2005-02,* 2005.

[18] NS2 homepage. http://www.isi.edu/ nsnam/ns/

[19] OMNeT++ homepage. http://www.omnetpp. org/

[20] K. Park, H. Lee. On the Effectiveness of Route-based Packet Filtering For Distributed DoS Attack Prevention in Power-law Internet. *Proceedings ACM SIGCOMM,* 2001.

[21] C. Papadopoulos, R. Lindell, I. Mehringer, A. Hussain, R. Govindan. Cossack: Coordinated suppression of simultaneous attacks. *Proceedings of DISCEX III*, 2003.

[22] T. Peng, L. Christopher, R. Kotagiri. Protection from Distributed Denial of Service Attack Using History-based IP Filtering. *IEEE International Conference on Communications,* 2003.

[23] SSF Net homepage. www.ssfnet.org

[24] M. Tambe. Towards flexible teamwork, *Journal of AI Research*, Vol.7, 1997.

[25] M. Tambe, D.V. Pynadath. Towards Heterogeneous Agent Teams, *Lecture Notes in Artificial Intelligence*, Vol.2086, 2001.

[26] Y. Xiang, W. Zhou. An Active Distributed Defense System to Protect Web Applications from DDoS Attacks. *The Sixth International Conference on Information Integration and Web Based Application & Services,* 2004.

[27] D. Xuan, R. Bettati, W. Zhao. A gateway-based defense system for distributed dos attacks in high-speed networks, *IEEE Transactions on Systems, Man, and Cybernetics,* 2002.

[28] Y. Xiang, W. Zhou, M. Chowdhury. A Survey of Active and Passive Defence Mechanisms against DDoS Attacks. *Technical Report, TR C04/02*, School of Information Technology, Deakin University, Australia, March 2004.

[29] J. Yen, J. Yin, T.R. Ioerger, M. Miller, D. Xu, R. Volz. CAST: Collaborative agents for simulating teamworks. *Proceedings of IJCAI'2001,* 2001.

Igor Kotenko *graduated with honors from St.Petersburg Academy of Space Engineering and St.Petersburg Signal Academy. He obtained the Ph.D. degree in 1990 and the National degree of Doctor of Engineering Science in 1999. He is Professor of computer science and Leading researcher of St. Petersburg Institute for Informatics and Automation.*

Alexander Ulanov *graduated from St. Petersburg State Politechnical University (2004), received his master's degree (2004) in the area "System analysis and control". He is now PhD student in the field of agent-based modeling and simulation for computer network attacks.*