# TEAMWORK APPROACH FOR MODELING AND SIMULATION OF DDOS ATTACKS IN INTERNET

Igor Kotenko
Alexander Ulanov
St. Petersburg Institute for Informatics and Automation
39, 14th Liniya, St. Petersburg, 199178, Russia
{ivkote, ulanov}@iias.spb.su

**KEYWORDS**
Multi-agent systems, Agent-based Modeling and Simulation, Distributed Denial of Service (DDoS) Attacks, Network Security.

**ABSTRACT**

The paper considers an approach to modeling and simulation of Distributed Denial of Service (DDoS) attacks fulfilled by a group of malefactors. The approach is based on combination of "joint intentions" and "common plans" theories as well as state machines. The formal framework for modeling and simulation of DDoS) attacks is presented. The architecture and user interfaces of the Attack Simulator software prototype implemented and its evaluation results are depicted. The simulation-based exploration of the Attack Simulator prototype demonstrated its efficacy for accomplishing various DDoS attack scenarios. The framework and software prototype developed can be used for conducting experiments for evaluating computer network security and analyzing efficiency of security policy.

## 1. INTRODUCTION

Vulnerabilities of computer systems, permanently magnifying complexity of cyber-attacks and gravity of their consequences highlight urgent necessity for new approaches to information assurance and survivability of computer systems. One of the most harmful classes of attacks aiming at destruction of network resources availability is "Denial of Service" (DoS) (Mirkovic et al. 2002; Mirkovic et al. 2004). 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 thic 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 seriousness of the DDoS problem and the increased frequency of DDoS attacks have led to the development of numerous DDoS defensive mechanisms. Unfortunately, the existing theoretical basis that should support implementation of defensive mechanisms against such class of attacks is poor.

According to our opinion, among many reasons, the above is stipulated by weakness of fundamental research that consider defense against DDoS attacks as a task of adversarial competition between security systems and malefactors' attacking systems, in particular, the research intending development of an adequate formal framework for exploratory modeling and respective software architecture for simulation of DDoS attacks and distributed defensive software components of computer network (Kotenko et al. 2003).

Modeling and simulation of DDoS attacks and performing their analysis are very important for discovering computer systems prone to DDoS, formulating defense recommendations and developing effective protective methods.

The paper considers an approach to agent-based modeling and simulation of DDoS attacks fulfilled by a group of malefactors. The goals of the paper are development of agent-based formal framework for specification of DDoS attacks and implementation of a software tool making it possible to simulate DDoS attacks.

The rest of the paper is structured as follows. *Section 2* outlines suggested common approach for modeling and simulation of DDoS attacks by imitating malefactors' teamwork. *Section 3* describes the ontology of DDoS attacks and specifications of structure and common scheme of operation of agents. *Section 4* determines architecture and main user interfaces of the DDoS Attack simulator elaborated and its evaluation issues. *Conclusion* outlines the results of the paper.

## 2. DDOS COMPONENTS AS INTELLIGENT AGENTS. TEAMWORK-BASED FRAMEWORKS FOR MODELING AND SIMULATION OF ATTACKS

By the analysis of present DDoS attacks it is possible to reveal the division of DDoS software components by their roles. At first, there is a "master" program which gathers the initial information about hosts in the Internet and obtains the access to their resources for starting "daemons" programs. "Daemons" are the attack executors. They usually provide a full access to compromised host for "master". "Master" coordinates "daemons" actions: it can exchange messages with "daemons" and install on captured hosts new programs for further "daemons" propagation. "Daemon" reports to "master" about its state. As soon as the DDoS net obtains required size, "master" sends the messages about attack time

and attack target (or simply an attack signal) to all "daemons".

From analysis of DDoS attacks we can see that each attacker-program ("master" or "daemon") is an autonomous software component which has initial knowledge, can get and process data from environment, has target and list of activities to reach this target, and can interact with other components. These properties are peculiar to intelligent agents. All DDoS components form a team of agents as they fulfill joint operations for reaching the common long-time goal (Denial of service attack) in a dynamic external environment (the Internet) at presence of noise and counteraction of opponents (components of security systems).

Now the research on teamwork is an area of steadfast attention in multi-agent systems (Fan and Yen 2004). A set of approaches to formalization and simulation of the agents' teamwork is known. For the organization of teamwork of DDoS agents, we have used the base ideas stated in works on the joint intention theory, the shared plans theory and the combined theories of agents' teamwork.

In the *joint intentions theory* (Cohen and Levesque 1991) the worlds, in which the agents act, are assumed to consist of primitive events that can be associated with specific agent. Statements in which the agent is convinced are called its beliefs. States of agent that are considered by it as the most desirable are called its goals. The mutual agents' beliefs are formed from beliefs of agent group. Agent team is said to have the joint intention to complete an action if and only if all team members have joint persistent goal to complete this action.

In the *shared plans theory* (Grosz and Kraus 1996) the shared plan is believed to be the plan of joint fulfillment of some set of actions by the group of agents. The main features of shared plan are as follows: (1) the group plan demands the group (team) of agents should reach the consent to fulfill the instructions, to which they will follow in group operations; (2) the agents should take up the obligations not only on the personal operations, but also on operations of the group as a whole (personal intentions how to make operations); (3) each agent should take up the obligations on operations of other agents (approved intention); (4) the plan of the group activity can have as components the plans of the separate agents for the assigned operations, as well as plans of subgroups.

In the *combined theories* (Jennings, N. 1995; Tambe 1997; Tambe and Pynadath 2001) the notion of joint persistent goal is used for building the scheme of agents' actions coordination and agents' communication protocols. The notion of shared commitments is the basis for implementation and monitoring of team activity. This notion is used while checking the state of goal and corresponding conditions (the goal is achieved, not achieved, cannot be achieved or irrelevant in view of breaking conditions). The notion of joint intentions is used to describe the agents' team activity in terms of particular operators. The main goal in teamwork is to provide the activity of agents according to a high-level scenario where each agent knows its place. The shared plans theory supports required methods for solving this problem in form of shared plan (full or partial). This plan can specify the activity of whole team, agents' groups and particular agents and also the constraints determining agents' collaboration and communication. The joint intentions theory is used for structuring of shared plan, scenario of its execution and communication.

Considering the "master" and "daemons" specifics the most suitable approach to use is the combined theory. There must be a shared plan, because of need to provide agents work according to mentioned DDoS attack steps. Agents have a joint goal – to perform DDoS attack. However, "master" and "daemons" act each in one's own way. Individual actions and communications of agents will also be a part of shared plan.

The common (group, individual) intention and commitment are associated with each node of a general hierarchical plan. These intention and commitment manage execution of a general plan, providing necessary flexibility. During functioning each agent should possess the group beliefs concerning other team-mates. For achievement of the common beliefs at formation and disbandment of the common intentions agents should communicate. All agents' communications are managed by means of common commitments built in the common intentions. Besides it is supposed, that agents communicate only when there can be an inconsistency of their actions. It is important for reaction to unexpected changes of environment, maintenance of redistribution of roles of the agents failed or unable to execute some part of a general plan, and also at occurrence not planned actions (Tambe 1997).

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 (Kotenko et al. 2003): (1) formation of the subject domain ontology; (2) determination of the agents' team structure; (3) definition of agent interaction-and-coordination mechanisms (including roles and scenarios of an agents' roles exchange); (4) specifications of the agents' actions plans (generation of attacks); (5) assignment of roles and allocation of plans between the agents; (6) state-machine based realization of the teamwork.

Formation of the subject domain ontology is an initial stage of the agents' team creation. Modeling in any subject domain assumes development of its conceptual model, i.e. set of basic concepts of a subject domain, relations between the concepts, and also data and algorithms interpreting these concepts and relations.

The agents' team structure is described in terms of a hierarchy of group and individual roles. Leaves of the hierarchy correspond to roles of individual agents, but intermediate nodes - to group roles.

The plan hierarchy specification is carried out for each role. For group plans it is necessary to express joint activity obviously. The following elements are described for each plan: (a) entry conditions when the plan is offered for execution; (b) conditions at which the plan stops to be executed (the plan is executed, impracticable or irrelevant on conditions); (c) actions which are carried out at a team level as a part of a common plan.

The assignment of roles and allocation of plans between the agents is carried out in two stages: at first the plan is distributed in terms of roles, and then the agent is put in correspondence to each role. One agent can execute a set of roles. Agents can exchange roles in dynamics of the plan execution. Requirements to each role are formulated as

union of requirements to those parts of the plan which are put in correspondence to the role. There are also group and individual roles. Leaves correspond to individual roles. Agents' functionalities are generated automatically according to the roles.

For setting the agents' team operation in real-time a hierarchy of state machines is used. The state machines realize a choice of the plan which will be executed and a fulfillment of the established sub-plans in a cycle "agents' actions - responses of environment".

At joint performance of the scenario agents' coordination is carried out by message exchange. As the agents' team function in antagonistic environment agents can fail. Restoration of lost functionalities is carried out by means of redistribution of roles of the failed agent between other agents and cloning of new agents.

## 3. ONTOLOGY OF DDOS ATTACKS. STRUCTURE AND OPERATION OF AGENTS

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* (Kotenko and Man'kov 2003).

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.

DDoS-attack includes three stages: (1) preliminary, (2) basic and (3) final.

Main operations of the preliminary stage are investigation (reconnaissance) and installation of agents-"zombies".

The content of the basic stage is realization of DDoS attack by joint actions of agents "master" and "daemons".

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).

Algorithmic interpretation of the attack plan specified as a family of state machines. The basic elements of each state machine are states, transition arcs, and explanatory texts for each transition.

States of each state machine are divided into three types: first (initial), intermediate, and final (marker is End). The initial and intermediate states are the following:

(1) non-terminal, those that initiate the work of the corresponding nested state machines;

(2) terminal, those that interact with the host model;

(3) abstract (auxiliary) states.

Example of one of realizations of the state machine DS is represented in Figure 1. Main parameters of this realization of the state machine are defined in Table 1.

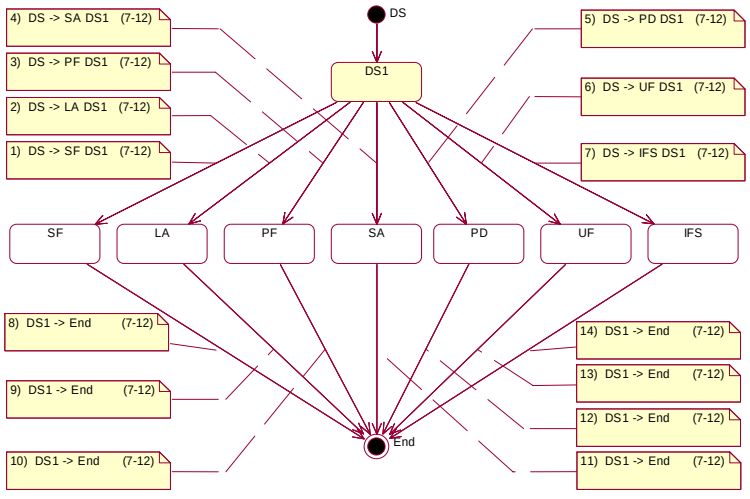


Figure 1: Diagram of State Machine DS (DoS attack)

Table 1: Main Parameters of State Machine DS

| State machine name | DS |
|---|---|
| States | DS1, SF (SYN flood), LA (Land attack), PF (Ping flooding), SA (Smurf), PD (Ping of Death), UF (UDP flooding), IFS (Storm of inquiries to FTP-server), End |
| First State | DS1 |
| Nonterminal states | - |
| Terminal states | SF, LA, PF, SA, PD, UF, IFS |
| Auxiliary states | DS1 |

We limit a team of agents by two-level structure. The team of agents consists of "master" and "daemons". Each master manages a group of "demons". "Demons" execute immediate attack actions against victim hosts.

The attack development depends on the malefactor's "skill", information regarding network characteristics, which he/she possesses, some other malefactor's attributes. An attack is being developed as interactive process, in which the network is reacting on an attack action. Computer network plays the role of the environment for DDoS agents, and therefore its model must be a part of the attack simulation tool.

So there appears need in one more agent – agent-"simulator". It simulates an attack environment – Internet. The Internet is considered as a set of connected hosts. Every host has certain characteristics, for example, ip-address and ties between hosts. All DDoS agent requests to outer world are coordinated by "simulator".

Each agent is represented as follows: $a_N = <K, B, R, P, G, C>$, where $N$ – agent identifier; $K$ – agent knowledge; $B$ – agent beliefs; $R$ – agent intentions; $P$ – a set of parameters determining the agent activity; $G = \{ L_R, f_R \}$ – a set of goals and actions, $L_R$ – hierarchy of possible goals and actions (reactions to influences), $f_R$ – of choosing the goal or action from $L_R$ according current sets $K$, $B$, $R$, $P$ and $C$; $C$ – commitments to other agents.

The knowledge ($K$) of the agent-"simulator" is the information about hosts (active or not; ip-address; e-mails list; list of open ports; OS type; ip-address of router for this host; installed e-mail client, etc.) stored in the notion "poHosts" and the information about network topology (as a table of links) represented in the notion "poIP_Links" (Figure 2).

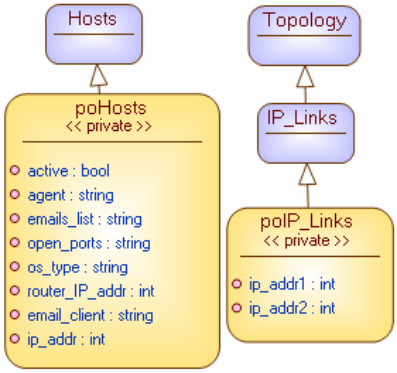


Figure 2: Fragment of Agent "Simulator" Ontology

The agent-"simulator" beliefs ($B$) represent the information about agent deployed on the current host. It is stored in the notion "poHosts" (agent name).

The "simulator" parameters are the network topology and its hosts properties.

The agent set of goals and activities ($L$) consists of responses to other agents requests (they function according to protocols). The requests are: request for determining if the host active is (ActiveHostQuery); request for scanning the hosts ports (OpenPortQuery); request for host capturing (HostCapturing); DoS attack execution (DosExecution).

The commitments ($C$) to other agents are specified and fulfilled according to the protocols of interaction between the agents with defined roles (Figure 3).

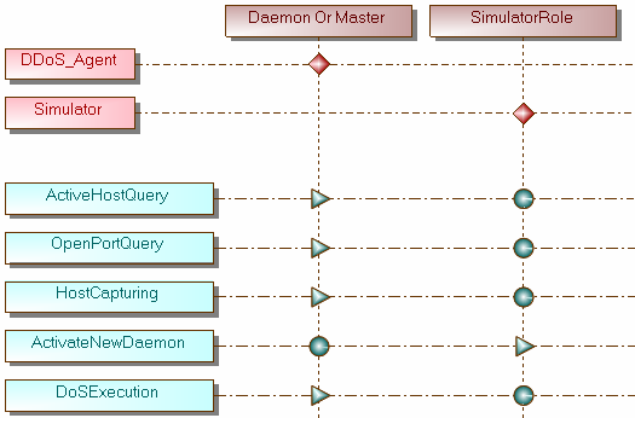


Figure 3: Agents, Their Roles and Protocols

The knowledge ($K$) of the agents "master" and "daemon" consists of information about compromised hosts and itself (one instance of the notion "poHost" and one "poAgentProps") (Figure 4).

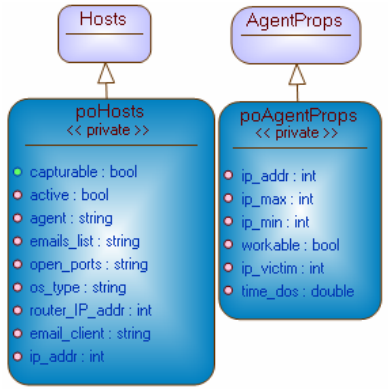


Figure 4: Fragment of Agent "Master" ("Daemon") Ontology

The agents "master" and "daemon" beliefs ($B$) are the information about environment (network topology, hosts parameters) and about other agents activity. They use the notions "poHosts" ("agent" attribute) and "poAgentProps" (host ip-address; ip-address range to scan; workable or not; victims ip-address; time to start attack). While acquiring new beliefs these agents build "the map of the world" step by step.

The agents parameters (*P*) are their activation parameters stored in the notion "AgentProps".
If the range of ip-addresses to scan is empty then the agent plays the "daemon" role. It executes the attack only. If the range is not null then the agent plays the "masters" role. It gathers the information about hosts from mentioned range, tries to capture them and also executes the attack.
The set of goals and activities (*G*) and their hierarchy (*L*) are represented by state machine representation (Figure 5).
Main goals and activities of "master" are as follows: Starting on accessible host ("DaemonActivation"); Gathering information about other hosts ("InformationGathering"), including Host activity determination ("GetActiveInfo") and Open ports determination ("GetOpenPortInfo"); Propagation by host capturing ("HostCapturing"), including Acquiring the hosts resources using the "shared resources" vulnerability ("Shared_Resources"); DoS attack execution ("AttackExecution"), for example using "Ping Of Death", "Syn overflow", "Smurf", etc. Main goals and activities of "daemon" are starting on accessible host and DoS attack execution.
The commitments (*C*) to other agents are specified and fulfilled according to the protocols of interaction between the agents with defined roles.

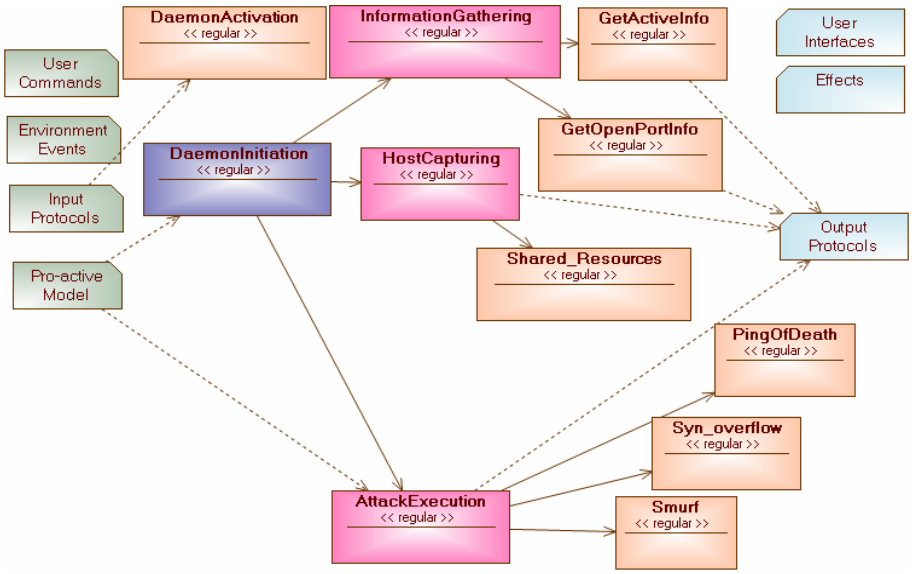


Figure 5: Fragment of "Master" ("Daemon") Goals and Activities

## 4. ATTACK SIMULATOR PROTOTYPE AND ITS EVALUATION

The software prototype of Attack Simulator has been implemented. Now it is used for validation of the accepted basic ideas, formal framework and implementation issues. The developed architecture of the attack simulator implementing the above described attack model was built as an agent of multi-agent system (MAS). The design and implementation of the attack simulator is being carried out on the basis of MAS DK – Multi-Agent System Development Kit (Gorodetski et al. 2002).
The MAS agents generated by MASDK have the same state-machine based architecture. Differences are reflected in the content of particular agents data and knowledge bases. Each agent interacts with other agents, environment which is perceived, and, possibly, modified by agents, and user communicating with agents through his interface.
The main objective of the experiments with the Attack Simulator prototype is to evaluate the tool's efficiency for different variants of attacks and attacked network configurations. These experiments were carried out for various parameters of the attack task specification and an attacked computer network configuration. The influence of the following input parameters on attacks efficacy was explored: a malefactor's intention, a degree of protection afforded by the network and personal firewall, a degree of security of attacked host, and the degree of malefactor's knowledge about a network. To investigate the Attack Simulator capabilities, the following parameters of attack realization outcome have been selected: number of terminal-level attack actions, percentage of the malefactor's intentions that are successful, percentage of "effective" network responses on attack actions, percentage of attack actions blockage by firewall, and percentage of "ineffective" results of attack actions.
Let us consider a small example of simulation of DDoS attacks. The network fragment including 7 hosts defined as the environment for DDoS is represented in Figure 6 and Table 2.

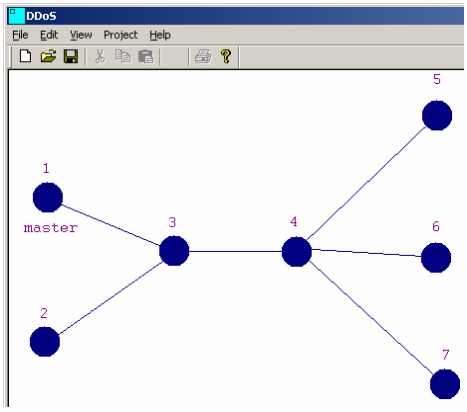


Figure 6: Graphic user interface for DDoS simulation

Table 2: Initial conditions for simulation

| № | IP address | Router IP-address | Active | Open ports | OS type | Agent |
|---|---|---|---|---|---|---|
| 1 | 1 | 3 | yes | 80 | Win | master |
| 2 | 2 | 3 | yes | 80,139 | Win | - |
| 3 | 3 | - | yes | 80 | Win | - |
| 4 | 4 | - | yes | 80,139 | Win | - |
| 5 | 5 | 4 | yes | 80 | Win | - |
| 6 | 6 | 4 | yes | 139 | Win | - |
| 7 | 7 | 4 | yes | 80 | Win | - |

Agent “master” was deployed in the initial moment on the host 1. Its parameters (*P*) were as follows: target of attack – host #7; hosts to compromise – 2-6; time to attack – 30 seconds after the start of simulation. Based on this data it was necessary to create one instance of “simulator”, one instance of “master” and six instances of “daemons”.
Every agent logs its actions to the text file to trace the DDoS simulation. A part of DoS agent log is determined below:

```
18:01:57:0492 DDoS_Agent Master - Info gathering
18:01:57:0507 DDoS_Agent Master - SendMsg Ping (Active Query) ip=6
18:01:57:0585 DDoS_Agent Master - ReceiveMsg Re_Ping (Active Query Reply) ip=6, active=1
18:01:57:0601 DDoS_Agent Master - SendMsg PortScan (Open Ports Query) ip=6
18:01:57:0679 DDoS_Agent Master - ReceiveMsg Re_IsPortOpen (Open Ports Query Reply) ip=6, open ports: 139
18:01:57:0710 DDoS_Agent Master - Capturing
18:01:57:0742 DDoS_Agent Master - Capturing: Shared Resources ip = 6
18:01:57:0851 DDoS_Agent Daemon4 - ReceiveMsg ActivateIt (Activation) ip=6 time_dos=1101135734.000000
18:02:14:0448 DDoS_Agent Daemon2 - Attack Execution method= Ping Of Death
18:02:14:0448 DDoS_Agent Daemon3 - Attack Execution method= Smurf
18:02:14:0448 DDoS_Agent Daemon4 - Attack Execution method= Syn overflow
```

In the initial moment (18:01:44) the “simulator” (see data from Table 2) and “master” (see (P) above) were initialized. Then “master” began to gather information. It try to find if the given hosts active and if they have the open ports. He tried to capture these hosts and to deploy “daemons” on them. They have waited until given time for attack execution. As a result, “master” could capture the hosts with ip-address 2, 4, 6 because they were active and had open port #139. So, only 4 agents (3 “daemons” and “master”) could start the attack on victim host (ip-address 7). The agents chose DDoS method and attacked the host #7.
The simulation-based exploration of the developed Attack Simulator prototype has demonstrated its efficacy for accomplishing various attack scenarios against networks with different structures and security policies implemented.

## 5. CONCLUSION

In the paper we developed basic ideas of the modeling and simulation of DDoS attacks by teamwork approach. We presented the structure of a team of agents, agent interaction-and-coordination mechanisms, and specifications of hierarchies of agent plans. The technology for creation of the DDoS agents’ team was suggested and described. We developed the approach to be used for conducting experiments to both evaluate computer network security and analyze the efficiency and effectiveness of security policy against DDoS attacks. Software prototype of Attack Simulator was developed. The attack simulator allows imitating a wide spectrum of real life DDoS attacks. Its software code is written in terms of Visual C++ and MASDK. Experiments with the Attack Simulator have been conducted, including the investigation of attack scenarios against networks with different security policies.
The further development of the Attack Simulator tool will consist of enlargement of its capabilities in specification of the attack tasks, expansion of the DDoS attack classes, implementing more sophisticated attack scenarios, etc.

## 11. ACKNOWLEDGEMENT

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

## REFERENCES

Cohen, P.R. and H.J. Levesque. 1991. “Teamwork”. *Nous*, 25(4).

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

Gorodetski, V.; O. Karsayev; I. Kotenko; A. Khabalov. 2002. “Software Development Kit for Multi-agent Systems Design and Implementation”. *Lecture Notes in Artificial Intelligence*, Vol. 2296.

Grosz, B. and S. Kraus. 1996. “Collaborative plans for complex group actions”, *Artificial Intelligence*, Vol.86.

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

Kotenko, I.; A. Alexeev; E. Man’kov. 2003. “Formal Framework for Modeling and Simulation of DDoS Attacks Based on Teamwork of Hackers-Agents”. *IEEE/WIC International Conference on Intelligent Agent Technology*. Halifax, Canada, Proceedings. IEEE Computer Society.

Kotenko, I. and E. Man’kov. 2003. “Agent-Based Modeling and Simulation of Computer Network Attacks”. *Fourth International Workshop “Agent-Based Simulation 4 (ABS 4)”*. Proceedings. Montpellier. France.

Mirkovic, J.; J. Martin; P. Reiher. 2002. “*A Taxonomy of DDoS Attacks and DDoS Defense Mechanisms*”. Technical report #020018. University of California, Los Angeles.

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

Tambe, M. 1997. “Towards Flexible Teamwork”. *Journal of Artificial Intelligence Research*, No.7.

Tambe, M. and D.V. Pynadath. 2001. “Towards Heterogeneous Agent Teams”. *Lecture Notes in Artificial Intelligence*, Vol.2086.

## BIOGRAPHY

**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. His e-mail address is ivkote@spiiras.nw.ru and his Web-page can be found at http://space.iias.spb.su/ai/kotenko/ .

**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. His e-mail address is ulanov@iias.spb.su and his Web-page can be found at http://space.iias.spb.su/ai/ulanov/ .