Attacks against Computer Network: Formal Grammar-based Framework and Simulation Tool

Vladimir Gorodetski and Igor Kotenko

St. Petersburg Institute for Informatics and Automation, 39, 14th Liniya, St. Petersburg, Russia
gor@mail.iias.spb.su, ivkote@iias.spb.su

Abstract. The paper presents an approach and formal framework for modeling attacks against computer network and its software implementation on the basis of a multi-agent architecture. The model of an attack is considered as a complex process of contest of adversary entities those are malefactor or team of malefactors, on the one hand, and network security system implementing a security policy, on the other hand. The paper focuses on the conceptual justification of the chosen approach, specification of the basic components composing attack model, formal frameworks for specification of the above components and their interaction in simulation procedure. The peculiarities of the developed approach are the followings: (1) malefactor's intention-centric attack modeling; (2) multi-level attack specification; (3) ontology-based distributed attack model structuring; (4) attributed stochastic *LL(2)* context-free grammar for formal specification of attack scenarios and its components ("simple attacks"); (5) using operation of formal grammar substitution for specification of multi-level structure of attacks; (6) state machine-based formal grammar framework implementation; (7) on-line generation of the malefactor's activity resulting from the reaction of the attacked network security system.

1 Introduction

Attacks against computer network form one of the many other dimensions of cyber terrorism and therefore detection of such attacks and prevention of their harmful effects have recently become the task of great concern. It is undoubtedly that substantial increase of Intrusion Detection System (IDS) efficiency could be achieved in case of using knowledge resulting from generalization and formalization of the accumulated experience with regard to computer attacks [28]. A lot of such data is hitherto accumulated and systematized in the form of taxonomies and attack languages ([2], [30], [38], etc.). Nevertheless, there are no serious attempts to generalize the accumulated data in order to develop a *mathematical model* of a wide spectrum of attacks and use this model for *attack simulation.* Perhaps this is due to the extreme complexity of the network attack and computer networks from modeling perspective: "*there is no widely accepted information physics that would allow making an accurate model*", and the network attacks are "*so complex that we cannot describe them with any reasonable degree of accuracy*" [5].

Attack formal model could be a powerful source of knowledge needed for IDSs development. It could provide for deeper study and understanding of the essence and peculiarities of attacks (intentions of malefactors, attack objects, structures and strategies of attacks, etc.). This model would play an important role in IDS learning both known and of unknown attacks if it were used as a generator of training and testing samples of attacks. Finally, formal model of attacks and attack simulation tool could be used as a testbed for security policy validation, i.e. *for testing, comparing and evaluating of IDS components and IDS on the whole.*

Development of such a model, its formal specification and implementation issues are the subjects of research in this paper. The rest of the paper is structured as follows. *Section 2* outlines conceptual aspects and general strategy of attack modeling and describes definitions of basic notions composing an attack specification. *Section 3* describes the developed ontology of the problem domain "Computer network attacks" which is considered further as a basis for consistent attack specification. *Section 4* gives an outline of the proposed formal grammar framework for specification of attacks and exemplifies such specifications for several classes of attacks. *Section 5* describes the model of the "counterparty" of attacks, i.e. model of the attacked computer network. *Section 6* presents architecture of the Attack simulation tool and its implementation issues. *Section 7* gives an overview of related works. *Section 8* summarizes the main results of the paper.

2 Attack Modeling Strategy

The computer network attacks concern to the class of complex systems possessing such features as large scale, multi-connectivity of elements, diversity of their connections, variability of structure, multiplicity of executed functions and structural redundancy. An *attack model* is understood as a formal object having a likeness in basic properties with regard to real-life attacks, serving for investigations by means of fixing known and obtaining new information about attacks. A *formal model* of attacks is a collection of mathematical dependences specifying attacks and allowing to study them formally and via simulation.

The *peculiarities of planning and execution of attacks*, influencing on choice of a formal model of attacks, are as follows:

1. *Any attack is target- and intention-centered*, i.e. it is directed against a particular object (network, computer, service, directory, file, etc.) and, as a rule, has a quite definite intention. *Intention* is understood as a goal or sub-goal a malefactor intends to achieve. We speak about malefactor's "intentions" according to the terminology used for mental concepts. Formally specified intention we call a "goal". *Examples of intentions*: reconnaissance (e.g. learning of network structure, identification of OS, hosts and/or services, etc.); penetration into the system; access to files of some directory; denial of service, etc. *Examples of targets*: IP-addresses of trusted hosts; password file; files of a particular directory; some resources of a particular host, etc. It should be noticed, that in some cases intention cannot be determined in advance. It can be accepted by malefactor in progress of attack

development as a decision made on the basis of the obtained information and successfulness or ineffectiveness of particular malefactor's actions fulfilled earlier.

2. *An attack intention* can *be represented in terms of partially ordered set of lower-level intentions.* A set of malefactor's intentions partially ordered in time is called an *attack scenario*. Intentions constituting attack scenario can be represented at different generalization levels. At the lowest level, each such intention is realized by a malefactor as a sequence of actions (network packets, commands of OS, etc.). Any malefactor's intention can be realized in multiple ways. Malefactor can vary the scenario implementing the same intention and the same attack object.

3. *Attack modeling corresponds to an adversary domain*. Attack development depends on the result of each particular step of attack, i.e. it depends on response of the attacked network. In turn, a network response depends on security policy implemented. The current attack "state" is determined by initial malefactor's information about the attacked network (or host), by information collected at preceding attack steps, and also by the successfulness or ineffectiveness of the preceding steps.

Thus, any attack development depends on many random factors and, first of all, depends on attacked network response. Therefore, even if a general malefactor's intention is determined, *the attack development scenario cannot be definitely specified beforehand*. An attack development depends on many uncertainties: (1) uncertainty in choice of the attack intention and attack object; (2) uncertainty caused by the information content with regard to the attacked network which a malefactor possesses at the beginning of attack and in progress of its development; (3) uncertainty of choice of attack scenario implementing the already selected intention; (4) uncertainty of the attacked computer network response. Let us describe conceptually the *scheme of attack generation (simulation)*.

Selection of the attack intention and attack object is a subjective act. Let the list $\boldsymbol{X}=\{ X_1, X_2, \ldots, X_N\}$ of possible attack intentions and the list $\boldsymbol{Y}=\{ Y_1, Y_2, \ldots, Y_M\}$ of attack objects be given. To select some attack intention and an attack object, it is necessary to set some formal mechanism of choice, for example, randomization mechanism. Let an intention $X \hat{I} \boldsymbol{X}$ and an attack object $Y \hat{I} \boldsymbol{Y}$ be selected.

The next component of attack modeling is a mechanism for generation of the attack given upper-level intention X and attack object Y in the terms of hierarchy of lower-level malefactor's intentions and respective sequences of actions. Let us demonstrate the basic idea of such mechanism by example. Let us suppose that the malefactor's intention X consists in getting access to files of some directory of a host. If malefactor does not possess some basic information about computer network or host then he/she has to start from reconnaissance R, which corresponds to the first intention at the level that is lower with regard to the intention X of the top level. The reconnaissance R can be fulfilled, for example, by four different sub-attacks $\{A, B, C, D\}$. Only one of them can be selected on current step of the attack development as a sub-goal (intention) of the second level. We admit, that the malefactor has selected sub-goal C. Another malefactor in the same situation could make other decision. Therefore, it is quite reasonable to specify the above selection as a randomized step. Thus, generation of an attack in terms of lower level intentions given upper-level intention X and attack object Y can be formalized on the basis of randomization of selection among $\{A, B, C, D\}$.

Let the selected sub-goal *C* be to be realized as a sequence of "commands", first of which be the command a_1. The term "command" is used here in the generalized sense. Main difference between "command" and "intention" consists in the following. The command is a concrete action; it is not a mental concept, which represents a certain abstraction in malefactor's mind. It can be a sequence of IP-packages, a command of operating system, etc. An intention is a component of the plan of actions; it is an "abstraction" represented formally at respective level of detail. In other words, malefactor "thinks and plans" in terms of intentions but acts in terms of commands.

A set of sequences of commands, by which the malefactor tries to realize his /her intention, can be selected ambiguously. Therefore it is necessary to set a non-deterministic mechanism for generation of sequences of commands. It is obvious, that it can be randomization mechanism, however, probably, not so simple, as the random-number generator with a discrete distribution. Let a_1 be the first generated command. This command is dispatched to the attacked computer network (host). The hierarchy <attack intention X, attack target Y>→<lower level intentions>→<actions>, corresponding to the considered example scheme of the initial phase of the attack generation is shown in tree-like form in Fig.1.

The formally determined process of choice can be represented as follows: *<Attack: intention X, target Y>* → *<R> <Attack continuation, detailing X>*, *<R>→<C><Attack continuation, detailing R>*, *<C>→ a_1<Attack continuation, detailing C>*. The response of the attacked system to each command can be characterized as "success" if the command is executed like the malefactor wanted, or "failure", if the attacked system reacts to the command in the way that is undesirable for the malefactor. The following commands are depend on response of the attacked object to the command a_1.

If the chosen intention *C* is failed then the attack modeling and simulation process can be stopped ("the attack is over"), or the attack can be continued starting with reselection of the choice associated with specialization of intention *R* – predecessor of the failed intention *C* in the tree (see Fig.1) in terms of the rest of the set {*A, B, C, D*}, i.e. in terms of one of the lower level intentions {*A, B, D*}. In the last case, the choice of a new alternative for the intention *R* specialization is made with the respective recalculation of the probability distribution given over the truncated set of lower-level intentions.

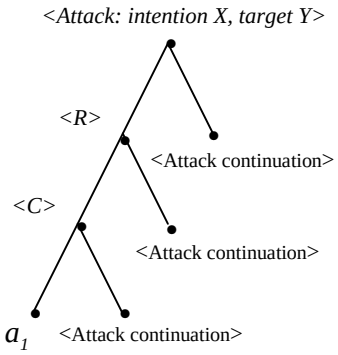

Fig.1. Attack scheme

The next step of attack generation is similar to the previous one. If in the following steps no one of intentions *A*, *B* and *D* does not result in success then the attack can be either finished or continued with the subsequent modification of the attack object. It is worth to notice, that in both above cases the probability distribution given over the set of the potentially admissible next step selections of intention alternatives should be recalculated.

If the attack with intention *C* is successful, then the attack can be stopped (if the goal is reached), or can be continued. This choice is also non-deterministic and can be simulated by a probabilistic mechanism and so on and so forth.

To an arbitrary step *n* of the attack generation (simulation) its state can be specified by a sequence of the following sort: *A*(*n*)=<*Attack prehistory*> <*Current state*> <*Attack continuation*>, where <*Attack prehistory*> is a sequence of the symbols corresponding to the preceding steps, in which each symbol is marked with a flag from a set {“success”, “failure”}. This sequence can include symbols of intentions of different levels of detail, and symbols of actions. It is supposed, that the attack can be simulated at various levels of detail of the description; <*Current state*> is a partially unfolded sequence of the current attack step symbols; <*Attack continuation*> is still unknown part of the sequence *A*(*n*), which generation is expected. In addition, current state of the attack development can also contain information collected at preceding steps.

It should be clear that it is impossible to enumerate and to specify all sequences *A*(*n*), i.e. to specify completely in declarative form the total set of attacks and variants of their development mapped to total set variants of the attacked network responses. Therefore, the only way to specify attacks if it exists at all, is procedural way, which suppose to model attack by a generation algorithm. This way is used in this research.

While describing the developed model, let us start with terminology that actually corresponds to the basic notions that will be structured and formalized below as the domain ontology. In the developed formal model of attacks, the basic notions of the domain correspond to malefactor’s intentions and all other notions are structured according to the structure of intentions. This is a reason why the developed approach is referred to as “*intention-centric approach*”.

The following *basic classes of high-lever malefactor's intentions* and their identifiers are used in the developed formal model:

1. *R* – Reconnaissance aiming at getting information about the network (host). The followings are the particular cases of intentions of this class: *IH* – Identification of the running Hosts; *IS* – Identification of the host Services; *IO* – Identification of the host Operating system; *CI* – Collection of additional Information about the network; *RE* – shared Resource Enumeration; *UE* – Users and groups Enumeration; *ABE* – Applications and Banners Enumeration.

2. *I* – Implantation and threat realization. The followings are its lower level variants of its specialization: *GAR* – Getting Access to Resources of the host; *EP* – Escalating Privilege with regard to the host resources; *GAD* – Gaining Additional Data needed for further threat realization; *TR* – Threat Realization, *TR* can be detailed at the lower level in the following terms: *CVR* – Confidentiality destruction (Confidentiality Violation Realization), for example, through getting access to file reading, *IVR* – Integrity Destruction (Integrity Violation Realization) realizing through attacks against integrity of the host resources, *AVR* – Denial of Service (Availability Violation Realization); *CT* – Covering Tracks to avoid detection of malefactors’ presence, *CBD* – Creating Back Doors.

An attack *task* specification (or a *top-level attack goal*) can be specified by the following quad: <*Network (host) address, Malefactor's intention, Known data, Attack*

object>[1]. The task specification has to determine the class of scenarios that lead to the intended result. *Known data* specifies the information about attacked computer network (host) known for a malefactor. *Attack object* corresponds to the optional variable in attack goal specification and are specified in the following ways:

(1) "**_**" – the attack object is not specified for the malefactor's intention "Reconnaissance" (*R*);

(2) If the intention corresponds to the attacks like *CVR* or *IVR* then the *attack object* is specified as follows: [*Account*,] [*Process* {<*Process name* >/< *Process mask* >},] [*File* {<*file name* >/< *file mask* >},] [*Data in transit* {< *file (data) name* >/< *file (data) mask* >}], where *Account* is object's account, *Process* is running process(es), *File* is file(s) that is the attack target(s) to get, *Data in transit* is data transmitting, where the variables in [] are optional, the repeatable variables are placed in {}, and symbol "/" is interpreted as "OR";

(*3*) "*All*" – all resources of the host (network);

(*4*) "*Anyone*" – at least one of the resources of the host (network).

3 "Computer Network Attacks" Ontology

It is well known that the development of a model of an information system must start with the development of the domain ontology. The ontology is the set of notions structured in terms of relationships existing over them. The ontology has to be abstracted from specifics of the implementation issues. A peculiarity of the particular domain is reflected in data structures and algorithms interpreting ontology notions and relationships. Although at present a lot of work is being performed in order to develop ontologies [33], but there are no such works in the network attacks domain.

The developed ontology comprises a hierarchy of notions specifying activities of malefactors directed to implementation of attacks of various classes in different levels of detail. In this ontology, the hierarchy of nodes representing notions splits into two subsets according to the *macro-* and *micro-levels* of the domain specifications. All nodes of the ontology of attacks at the macro- and micro-levels of specification are divided into the *intermediate* (detailable) and *terminal* (non-detailable).

The notions of the ontology of an upper level can be interconnected with the corresponding notions of the lower level through one through three kinds of *relationships*: (1) "*Part of*" that is decomposition relationship ("*Whole*"–"*Part*"); (2) "*Kind of*" that is specialization relationship ("*Notion*"–"*Particular kind of notion*"); and (3) "*Seq of*" that is relationship specifying sequence of operation ("*Whole operation*" – "*Sub-operation*"). High-level notions corresponding to the intentions form the upper levels of the ontology. They are interconnected by the "*Part of*" relationship. Attack actions realizing malefactor's intentions are interconnected with the intentions by "*Kind of*" or "*Seq of*" relationship. The developed ontology includes the detailed description of the network attack domain in which the notions of the bottom level ("*terminals*") can be specified in terms of network packets, OS calls, and audit data.

[1] In the software tool this quad is used for specification of simulation task by user.

Let us look at a high-level fragment of the developed ontology (Fig.2). At the upper-level of the *macro-specification of attacks*, the notion of "Network Attack" (designated by *A*) is in the "*Part of*" relationship to the "Reconnaissance" (*R*) and "Implantation and threat realization" (*I*). In turn, the notion *R* is in the "*Part of*" relationship to the notions *IH*, *IS*, *IO*, *CI*, *RE*, *UE*, and *ABE*. The notion *I* is in the "*Part of*" relationship to the notions *GAR*, *EP*, *GAD*, *TR*, *CT*, and *CBD*. In the next (lower) level of the hierarchy of the problem domain ontology, for example, the notion *IH* is in the "*Kind of*" relationship to the notions "Network Ping Sweeps" (*DC*) and "Port Scanning" (*SPIH*). At that, the notion "Network Ping Sweeps" (*DC*) is the lowest ("terminal") notion of the macro-level of attack specification, and the notion "Port Scanning" (*SPIH*) is detailed through the use of the "*Kind of*" relationship by a set of "terminal" notions of the macro-level of attack specification.

The "terminal" notions of the macro-level are further detailed at 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. Thus, for example, the notion "Network Ping Sweeps" (*DC*) is in the "*Kind of*" relationship with the notions "Network Ping Sweeps with *ping*" (*PI*), "Network Ping Sweeps with *Ping Sweep*" (*PSW*), etc., which, in turn, correspond to the names of utilities that perform "Network Ping Sweeps".

In turn, each of these notions, e.g. "Network Ping Sweeps with *Ping Sweep*" (*PSW*), is in the "*Seq of*" relationship to the "ICMP ECHO REQUEST" (*IER*) notions. The "ICMP ECHO REQUEST" (*IER*) notions correspond to network packets that are directed at the host (or the network) – the target of the attack.

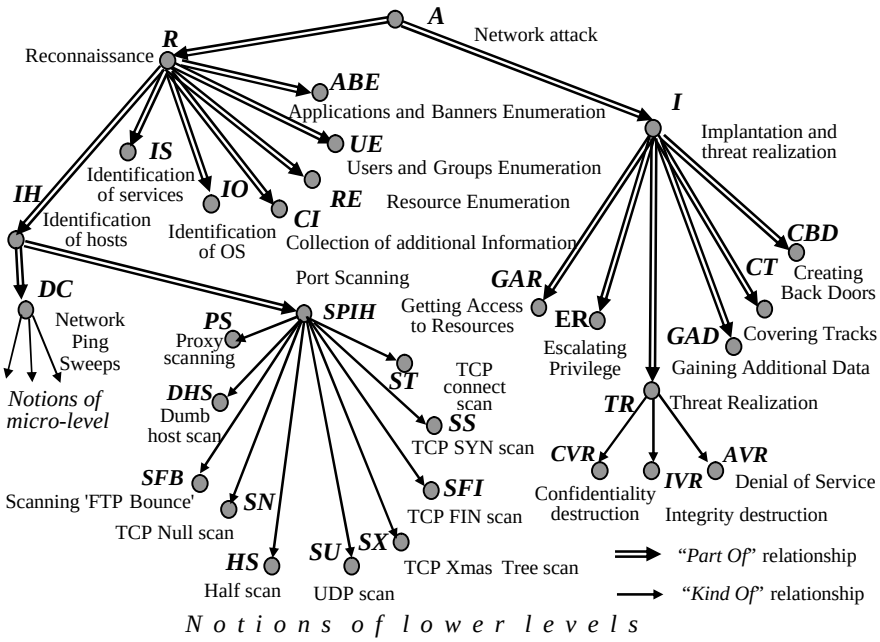

Fig.2. Macro-level fragment of the domain ontology "*Computer network attacks*"

In micro specifications of the 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. In Fig.3, this type of relationship is used to establish the connection between the echo-request of the protocol ICMP ("ICMP ECHO REQUEST") and its specific implementation specified, for example, as a message `<time> <src_addr> > <dest_addr>: icmp: echo request`, where `<time>` – time stamp, `<src_addr>` – source IP address, `<src_port>` – source port, `<dest_addr>` – destination IP address.

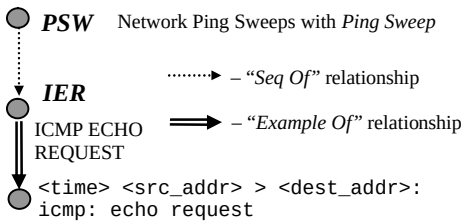

Fig.3. Micro- specifications of the domain ontology "*Computer network attacks*"

4 Formal Framework for Attacks Specification

Being based on the above explanation of the attack modeling strategy, definition of basic notions of attack specification and structure of the basic malefactors' intentions and also on the malefactors' actions, the following basic assumptions and statements are used below in the formal attack specification:

(1) Each attack intention can be considered as a *sequence of symbols* in terms of lower-level intentions. These sequences can be formally considered as "words" of a language, which can be generated by a formal grammar. Thus, each node of the ontology (see Fig.2) can be specified in terms of a formal grammar generating more detailed attack specification;

(2) Analysis of a wide spectrum of formal grammar-based specifications of attack intentions justified that attack intentions can be adequately specified in terms of *LL(2)* context-free grammar;

(3) Specification of uncertainties inherent to the attack development can be done in probabilistic terms through attributes and functions given over them. Thus, in conjunction with the above conclusions the resulting framework for attack specification can be restricted to a stochastic attribute grammar;

(4) Each node (grammar) of the ontology (like shown in Fig.2) is interconnected with the upper level node (grammar) and this interconnection can be specified through "grammar substitution" operation [15] in which a terminal symbol of the parent node is considered as the axiom of the grammar corresponding to its child node.

(5) Each malefactor's action has to be followed by an attacked network response.

The rest of this section presents the above summary in formal terms.

Thus, mathematical model of attack intentions is determined in terms of a set of *formal grammars* specifying particular intentions interconnected through "*substitution*" operations: $M_A=<\{G_i\}, \{Su\}>$, where $\{G_i\}$ – the formal grammars, $\{Su\}$ – the "substitution" operations.

Every formal grammar is specified by quintuple $G=<V_N,V_T,S,P,A>$, where G is the grammar name, V_N is the set of non-terminal symbols (that are associated with the upper and the intermediate levels of an attack scenario), V_T is the set of its terminal symbols (that designate the steps of a lower-level attack scenario), $S \hat{I} V_N$ is the grammar axiom (an initial symbol of an attack scenario), P is the set of productions that specify the specialization operations for the intention through the substitution of the symbols of an upper-level node by the symbols of the lower-level nodes, and A is the set of attributes and algorithms of their computation.

Attribute component of each grammar serves for several purposes. The first of them is to specify *randomized choice of a production* at the current inference step if several productions have the equal left part non-terminals coinciding with the "active" non-terminal in the current sequence under inference. These probabilities are recalculated on-line subject to the prehistory of attack development and previous results of attack. So, in order to specify a stochastic grammar, each production is supplemented with a specification of the probability of the rule being chosen in the inference process.

Also the attribute component is used to check *conditions determining the admissibility of using a production* at the current step of inference. These conditions depend on attack task specification, attacked computer network (host) response and also on the malefactor's previous actions. These conditions may depend on compatibility of malefactor's actions and attacked network or host properties, e.g., OS type and version, running services, security parameters, etc.

These are the examples of host parameters, which may form production conditions: (1) OS types – *Unix*, *Linux*, *Win* (all Windows OS), *9x* (*95, 98, Me*), *NT* (*NT, 2000*), *SunOS*, *Solaris*, etc.; (2) running applications – e.g., *PWS* – an initial version of Microsoft's Personal Web Server is running; (3) protection parameter – *CFP* (shared files and printers), *NS* (Null Sessions), *PA* (Password is Absent), *RR* (Remote Registry), etc.; (4) additional parameters – *AS* (Access to Segment of LAN), *THD* (Trusted Host Data), etc.

If it is necessary to specify several parameters, operations "OR" (signified by ",") and (or) "AND" (".") are used. Relationships of ownership and membership are also taken into account, e.g. $SunOS \in Unix$; $\{95, 98, Me\} \subset 9x$; $\{95, 98, Me, NT, 2000\} \subset Win$, $9x \in Win$, etc.

Thus, in general case, the grammar production is recorded as follows: $[(U)]\ X \rightarrow \alpha\ (Prob)$, where U – the condition for upholding the rule, [ ] – an optional element, X – non-terminal symbol, α –a string of terminal and non-terminal symbols, $Prob$ – the initial probability of the rule.

Let us explain by example the operation of *grammar substitution* and its role in the formal model of attacks. Let $a \hat{I} V_T(G_i)$ be a terminal symbol of the grammar G_i in the sequence of symbols generated by the grammar G_i, a is a node of the ontology mapped to the grammar $G(a)$. Symbol a denotes the name of a particular intention or attack action and $G(a)$ is the grammar generating variants of the a implementation. Let also X be the axiom of the grammar $G(a)$. Then, *operation* $Su(a)$ *of substitution* $G(a)$ in place of symbol a is specified in the form $Su(a)$*:* $\{a \rightarrow G(a)\}$. *Semantics of this operation* is that in place of symbol a in already generated sequence any "word" generated by grammar $G(a)$ can be placed. In fact, this operation corresponds to a step towards the more detailed specification of an attack scenario.

When the micro specifications are used for modeling of attacks, it is necessary to use the ontology nodes of the lowest (terminal) level and substitute specific values for the variables that determine the attack task specification.

For example, let us suppose a ping attack is being implemented using "Network Ping Sweeps with *Ping Sweep*" (*PSW*). *PSW* is in the "*Seq of*" relationship to the "ICMP ECHO REQUEST" (*IER*) network packets that are directed at the target host (network). In micro specifications of attacks the *IER* node is in the "*Example of*" relationship to its specific implementation defined as the following message: `<time> <src_addr> > <dest_addr>: icmp: echo request ,` where `<time>` – time stamp, `<src_addr>` – source IP address, `<src_port>` – source port, `<dest_addr>` – destination IP address.

The grammar that specifies *PSW* may look like this: V_N={PSW, PSW1}, V_T={IER}, S={PSW}, P={PSW → IER PSW1 (1), PSW1 → IER PSW1 (0.2), PSW1 → IER (0.8)}.

Let us suppose a ping attack with "*Ping Sweep*" is being implemented from host 244.146.4.20 on the hosts of the network 198.24.15.0 in the time interval [0:43:10.094644, 00:43:16.036735]. Let us suppose that the string "***IER IER***" was created as a result of using the *PSW* grammar. Then, based on the "*Example of*" relationship, the symbols of this string should generate two messages:

```
<time1> <src_addr> > <dest_addr>: icmp: echo request ,
<time2> <src_addr> > <dest_addr>: icmp: echo request .
```

After the parameterization `<time1>` = 00:43:10.094644, `<src_addr>`=244.146.4.20, `<dest_addr>` = 198.24.15.255, `<time2>`=00:43:16.036735, these messages should look like these:

`00:43:10.094644 244.146.4.20>198.24.15.255:icmp:echo request` and `00:43:16.036735 244.146.4.20>198.24.15.255:icmp:echo request`,

which correspond to the *icmp*-packets sent to the network hosts 198.24.15.0 (since the X.X.X.255 address is specified in the *icmp*-packets, the packets are sent to all the hosts of the specified networks).

The development of the family of grammars $\{G_i\}$ is conducted in the following order: (1) First, for each basic malefactor's intention, its own family of enclosed attributed stochastic context-free grammars is constructed; (2) Second, these families of grammars are transformed into the generalized grammars that correspond to each non-terminal node of ontology for all of the intentions.

It is assumed that if a value of the production condition is not determined at the moment of production selection all available productions may be used at the respective step of attack simulation. Also it is supposed that the terminal actions generated by productions are associated with the probabilities of successful realization of those actions (attacks) and the host response.

Let us consider, for example, the grammars for the intention "Users and groups Enumeration" (UE) can be as follows:

Level "Network Attack": V_N={A, A1}, V_T={R}, S={A},
P={A → A1 (1), A1 → R (0.7), A1 → R A1 (0.3)};
  Level "Reconnaissance": V_N={R, R1}, V_T={UE}, S={R},
  P={R → R1 (1), R1 → UE (0.7), R1 → UE R1 (0.3)};
    Level "Users and groups Enumeration"

V_N={UE, UE1, UE2, UE3, UE4}, S={UE},
V_T={DNNT, EUE, PIUD, IAUS, SNMPE, FUE, UTFTP},
$P_{for\ Windows\ 9x,Me,NT,2000}$={(Win) UE→ UE1(1), (NS)UE1→ UE2(0.65),
UE1→ SNMPE(0.25), UE1→SNMPE UE1 (0.05), (NS)UE1→UE2 UE1
(0.05), (&)UE2→CNS UE3(1), UE3→DNNT (0.2), UE3→DNNT UE4
(0.05), UE3→IAUS(0.35), UE3→EUE(0.2), UE3→ PIUD (0.2),
UE4→DNNT UE4(0.1), UE4→DNNT(0.9)},
$P_{for\ Unix/Linux}$ ={(Unix, Linux) UE→UE1(1), UE1→FUE(0.3),
UE1→SNMPE(0.2), UE1→UTFTP(0.1), UE1→FUE UE1(0.1),
UE1→SNMPE UE1(0.1), UE1→UTFTP UE1(0.2)};

Level "Identifying Accounts with user2sid/sid2user":
V_N={IAUS, IAUS1, IAUS2}, V_T={ISU, IAS}, S={IAUS},
P={(NT) IAUS → IAUS1 (1), (&) IAUS1 → ISU IAS (0.8),
IAUS1 → IAUS1 IAUS2 (0.2), (&) IAUS2 → ISU IAS (1)}.

In this set of grammars the following denotations are used: A – Network Attack; R – Reconnaissance; UE – Users and groups Enumeration; DNNT – Dumping the NetBIOS Name Table with *nbtstat* and *nbtscan*; EUE – Enumerating Users with *enum*; PIUD – Providing Information about Users with *DumpSec* (*DumpACL*); IAUS – Identifying Accounts with *user2sid*/*sid2user*; SNMPE – SNMP Enumeration with *snmputil* or *IP Network Browser*; FUE – Finger Users Enumeration; UTFTP – Use of Trivial File Transfer Protocol for Unix enumerating by stealing */etc/passwd* and (or) */etc/hosts.equiv* and (or) *~/.rhosts*; ISU – Identifying SID with *user2sid*; IAS – Identifying Account with *sid2user* using user's RID; A1, R1, UE1, UE2, UE3, UE4, IAUS1, IAUS2 – auxiliary symbols.

Algorithmic interpretation of the attack generation specified as formal generalized grammars is implemented by 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 of this state is *End*). The initial and intermediate states are the following: non-terminal, those that initiate the operation of the corresponding nested state machines; terminal, those that interact with the host model; auxiliary states. Transition arcs are identified with the productions of grammars. The model of each state machine is set by specifying the following elements: diagram; main parameters; parameters of transitions that determine the stochastic model of the state machine for different relevant intentions; executable scripts; transition conditions.

5 Formal Model of the Attacked Computer Network

The attack development depends on the malefactor's "skill", information regarding network characteristics, which he/she possesses, some other malefactor's attributes [39], security policy of the attacked network, etc. An attack is developing as interactive process, in which the network is reacting on the malefactor's action. Computer network plays the role of the environment for attacker, and therefore its model must be a part of the attack simulation tool.

The peculiarity of any attack is that the malefactor's strategy depends on the results of the intermediate actions. This is the reason why it is not possible to generate

the complete sequence of malefactor's actions from the very beginning. The malefactor's action has to be generated on-line in parallel with the getting reaction from the attacked network. The proposed context-free grammar syntax provides the model with this capability. At each particular step of inference, it generates no more than single terminal symbol that can be interpreted by the computer network model as a malefactor's action. The network returns the value of the result (success or failure). The model of attacker receives it and generates the next terminal symbol according to the attack model and depending on the returned result of the previous phase of the attack.

Model of the attacked computer network is represented as quadruple $MA = <M_{CN}, \{M_{Hi}\}, M_P, M_{HR}>$, where M_{CN} is the model of the computer network structure; $\{M_{Hi}\}$ are the models of the host resources; M_P is the model of computation of the attack success probabilities; M_{HR} is the model of the host reaction in response of attack. Let us determine the model M_{CN} of a computer network structure CN as follows: $M_{CN} = < A, P, N, C >$, where A is the network address; P is a family of protocols used (e.g., TCP/IP, FDDI, ATM, IPX, etc.); N is a set $\{CN_i\}$ of sub-networks and/or a set $\{H_i\}$ of hosts of the network CN; C is a set of connections between the sub-networks (hosts) established as a mapping matrix. If N establishes a set of sub-networks $\{CN_i\}$, then each sub-network CN_i can in turn be specified by the model M_{CNi} (if its structure needs to be developed in detail and if information is available about this structure). Each host H_i is determined as a pair $M_{Hi} = <A, T>$, where A is the host address, T is a host type (e.g., firewall, router, host, etc.).

Models $\{M_{Hi}\}$ of the network host resources serve for representing the host parameters that are important for attack simulation. Let us determine the model of the network host resources as follows: $M_{Hi} = < A, M, T, N, D, P, S, DP, ASP, RA, SP, SR, TH,$ etc.$>$, where A – IP-address, M – mask of the network address, T – type and version of OS, N – users' identifiers (IDs), D – domain names, P – host access passwords, S – users' security identifiers (SID), DP – domain parameters (domain, names of hosts in the domain, domain controller, related domains), ASP – active TCP and UDP ports and services of the hosts, RA – running applications, SP – security parameters, SR – shared resources, TH – trusted hosts.

Success or failure of any attack action (corresponding to terminal level of the attack ontology) is determined by means of the model M_P of computation of the attack success probabilities. This model is specified as follows: $M_P = \{R^{SPr}_j\}$, where R^{SPr}_j is a special rule that determines the action success probability depending on the basic parameters of the host (attack target). The rule R^{SPr}_j includes IF and THEN parts. The IF part contains action name and precondition (values of attributes constraining the attack applicability). The THEN part contains value of success probability (SPr). Examples of interpretations of the probability computation rules are as follows:

"If *action* is 'FF' (Connection on FTP and examination of bin-files in the directory /bin/ls) and *OS Type* is 'Unix, Linux' and *Service* is 'FTP' **then** *SP* is 0.7";

"If *action is 'FCA' (Free Common Access) and OS type is 'Windows 9x' and Security parameter* is 'CFP' (shared files and printers) **then** *SP* is 0.7".

The result of each attack action is determined according to the model M_{HR} of the host reaction. This model is determined as a set of rules of the host reaction: M_{HR} =

{R^{HR}_j}, R^{HR}_j: *Input* ® *Output* [*& Post-Condition*]; where *Input* – the malefactor's activity, *Output* – the host reaction, *Post-Condition* – a change of the host state, & – logical operation "AND", [] – optional part of the rule. The *Input* format: `<Attack name>: <Input message> : <Attack objects> [; <Objects involved in the attack>.` The *Output* format: `{<Attack success parameter S> [: <Output message>];{<Attack success parameter F> [: <Output message>]}.` The `Attack Success Parameter` is determined by the success probability of the attack that is associated with the host (attack target) depending on the implemented attack type. The values of attack success parameter are `Success` (`S`), and `Failure` (`F`). The part of `output message` shown in the < > is taken from the corresponding field of the host (target) parameters. The part of output message shown in quotation marks " " is displayed as a constant line. The *Post-Condition* format: $\{p_1=P_1, p_2=P_2, \ldots, p_n=P_n\}$, where p_i – i^{th} parameter of the host (for instance, *SP*, *SR*, *TH*, etc.) which value has changed, P_i – the value of i^{th} parameter.

Examples of the host reaction rules:

`SFB: Scanning "FTP Bounce" : Target host; Intermediate host (FTP-server) ® {S: <Active ports (services) of a host>; F: "It was not possible to determine Active ports (services)"};`

`IF: ICMP message quoting : Target host ® { S: <The type of operating system>; F: "It was not possible to determine the type of operating system"}.`

6 Implementation of Attack Simulator

The software prototype of the attack simulator has been implemented. Now it is used for validation of the accepted formal framework. It consists of three components: the model of attacker, the model of the attacked computer network and the background traffic generator. Background traffic is formed taking into account the model of the attacked computer network as a set of sessions between hosts of the network. The common traffic generated by integration of streams of data from these components can be an input for IDSs evaluation and learning.

Each of the components of the attack simulator 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 MASDK – "Multi-Agent System Development Kit" [16]. All MAS agents generated by MASDK have the same architecture. Differences are reflected in the content of particular agent's 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. *Receiver of input* and *Sender of output messages* perform the respective functions. Messages received are recorded in *Input message buffer*. The order of its processing is managed by *Input message processor*. In addition, this component performs syntax analysis and messages interpretation and extracts the message contents. The component *Database of agent's dialogs* stores for each input message its attributes like identifiers, type of message and its source. If a message supposes to be replied it is mapped the respective output message when it is sent.

Meta-state machine manages the semantic processing of input messages directing it for processing by the respective *State machines.* The basic agent's computations are executed by a set of *State machines*. The selection of scenario and therefore the output result depend on the input message content and inner state of the *State Machine*. In turn, inner state of this Machine depends on pre-history of its performance; in particular, this prehistory is reflected in the state of agent's *Knowledge base* and *Database.* One more state machine called "*Self-activated behaviour machine*" is changing its inner state depending on the state of the data and knowledge bases. In some combinations of these states it can activate functionality independently on input messages or state of the environment. Each agent class is provided with a set of particular message templates according to its functionalities. The developer carries out the specialization procedure with *Editor of message templates*, which, in turn, is a component of MASDK. Communication component of each agent includes also data regarding potential addressees of messages of given template.

The screen indicating generation of the intention "Gaining Access to Resources" (*GAR*) is depicted in Fig.4. In this screen the attack generation at the reconnaissance stage after execution of the action "Enumerating Users with *enum*" (*EUE*) is fixed. In the figure the information is divided on four groups: (1) the attack task specification units are mapped in the left top of the screen; (2) to the right of them the attack generation tree is visualized; (3) the strings of the malefactor' s actions are placed in the left part of the screen below the attack task specification; (4) on the right of each malefactor's action a tag of success (failure) as green (black) quadrate and data obtained from an attacked host (a host response) are depicted.

From implementation issue, a computer network attack can be considered as a sequence of coordinated actions of the spatially distributed malefactors. Each malefactor is mapped as an intelligent agent of the same architecture possessing the

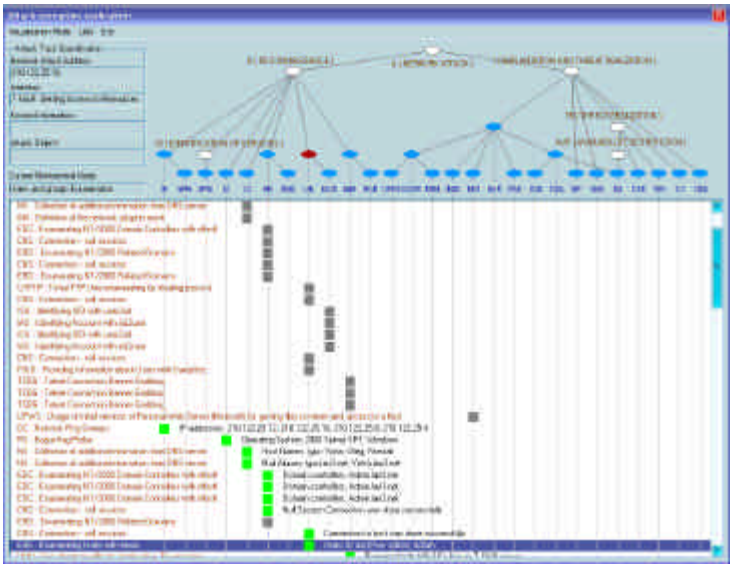

Fig.4. Visualization of the attack development: the reconnaissance stage of the attack "Gaining Access to Resources" after execution of the action *EUE*

similar functionality. While developing an attack, these agents interact via message exchange informing each other about current state and results of the attack in order to coordinate their further activity. These messages are represented in KQML that is standard of DARPA (for message "wrapper"), and XML (for message content).

We are developing[2] a teamwork interpretation of the malefactors' activity performing distributed attacks on the basis of joint intention theory [37]. When implementing the complex coordinated attacks, the special meta-agent should form the common scenario of the attack and assigns areas of responsibility to other agents based on the general attack goal constituted by simulation tool user. The agents, responsible for the particular fragments (steps) of the common scenario, can in turn "employ" other agents or realize particular operations independently. For this purpose the special scenarios of operations and protocols of messaging will be used. The concrete scenario and protocol are determined with usage of the network attacks ontology depending on a type of the realizable goal (intention) and the attacked network response. All set of the involved agents realizing the concrete scenario compose a hierarchical structure.

7 Related Works

The works relevant to attack modeling and simulation can be divided into the following groups: (1) works describing attacks and attack taxonomies, (2) works immediately coupled with attack modeling and simulation, (3) works devoted to the description of attack specification languages; (4) works on evaluating IDSs; (5) works on signature and traffic generation tools. This list is not exhaustive.

There are a lot of works in which attack cases are systematized as *attack taxonomies* (for example, [2], [7], [18], [24], [25], etc.). Based on these taxonomies *we built our own taxonomy as an ontology comprising a hierarchy of intentions and actions of malefactors directed to implementation of attacks of various classes split into macro- and micro-levels.*

In different works on *attack modeling and simulation*, as a rule, attack is considered as temporal orderings of actions ([6], [8], [22], [23], [36], etc.). In [22] the state transition analysis technique was developed to model host-based intrusions. A description of an attack has a "safe" starting state, zero or more intermediate states, and (at least) one "compromised" ending state. States are characterized by means of assertions describing different aspects of the security state. The work [6] presents an approach to simulate intrusions in sequential and parallelized forms. The paper [23] suggests formal models of both network and attacks and extends the state transition analysis technique to network-based intrusion detection in order to represent attack scenarios in networks. In [8] a simple network security model "Cause-Effect Model of Information System Attacks and Defenses" was proposed. It is composed of network model represented by node and link, cause-effect model, characteristic functions, and pseudo-random number generator. In ([39], [40]) the descriptive models of the network and the attacker's capabilities, intentions, and courses-of-action are

[2] Currently this part of work is in progress.

described. These models are used to identify the devices most likely to be compromised. Principles from economics are used to predict the attacker's behavior. Conceptual models of computer penetration were presented in [36]. The paper compares the traditional and "new" attack paradigms. Traditional attack paradigm includes phases of "information gathering", "exploitation", and "metastasis". The metastasis phase of the attack can be logically divided into sub-phases of "consolidation" and "continuation". The core of the new distributed metastasis methodology is a desire to utilize the distributed nature of network environment, and to perform an automation of the metastasis phase using a distributed agent-based approach. *We used in our formal model the temporal orderings of actions and proposed multi-agent teamwork-based approach for modeling of coordinated distributed attacks.*

In ([19], [32] and some other) attacks are modeled in a structured and reusable *"tree"-based form.* In [19] a high-level conceptual model of attack based on the intruder's intention (attack strategy) is presented. The paper determines intrusion intention as the goal-tree: the root node corresponds to ultimate goal of attack, and lower level nodes represent alternatives or ordered sub-goals in achieving the upper node goal. In [32] means for documenting attacks in a form of attack trees are described. Each attack tree enumerates and elaborates the ways that an attacker could cause the event to occur. Two structures are used for attack representation: an attack pattern (characterizing an individual type of attack), and an attack profile (organizing attack patterns to make it easier to apply them). As in [19] and [32] *we apply intension- and tree-based attack strategy representation, but "go further" using for node decomposition a formal framework based on context-free grammars implemented in terms of state machines.*

A model to evaluate survivability of networked systems after network incidents was developed in [31]. The model consists of three sub-models: the first one simulates the occurrence of incidents, the second one evaluates the impact of an attack on the system, and the third one assesses the survivability of the system. The model of incidents is determined as a marked stochastic process, where the incidents are the events that occur at random points in time, and the event type is the mark associated with an incident. *Besides attack generation model, our approach includes also the model of attacked computer network that evaluates the impact of an attack on the network hosts and generates reaction of the network. The attacked network is considered as environment that reacts on the malefactors' actions. The variance of attacks is ensured by the random choice of the grammar productions (or, what is the same, the state machine transition rules). The peculiarity of any attack is that the malefactor's strategy depends on the results of the intermediate actions.*

The paper [5] describes the cyber attack modeling and simulation methodology based on SES/MB (system entity structure and model base) framework and Discrete Event Simulation (DEVS) formalism. This simulation methodology allows classifying threats, specifying attack mechanisms, verifying protection mechanisms, and evaluating consequences. *Our approach has resembling purposes, but it uses stochastic formal-grammar-based specification of the malefactor's intentions and scenarios of network attacks.*

For attack fixing, reproduction, analysis, recognition, response, documenting, special *attack languages* are used [38]: event languages ([3], etc.), exploit languages

([4], etc.), reporting languages ([10], [13], etc.), detection languages ([12], [29], [34], etc.), correlation languages, response languages, and integrated languages ([9], [30], etc.). *Our formal language is related mostly to the exploit and event languages, because it is used to describe attack stages and the format of events generated. Our attack representation language includes parts used for description of attack preconditions, attack intentions and actions, formats of actions of terminal level, and postconditions (states of the attacked hosts).*

In all works devoted to the *evaluation of IDSs* the attack simulation issues are considered. In [35] a methodology and software tools for testing IDSs using scripts to generate both background traffic and intrusions are described. In evaluations performed by the Lincoln Laboratory at MIT ([26], etc.), investigators were given sniffed network traffic, audit data, and file-system snapshots. The report [11] discusses issues associated with the generation of suitable background traffic; attacks are obtained from a vulnerability database. In ([2], [27]) it was marked that benchmarking IDSs is not generic and systematic enough for evaluation needs. In [2] another approach is investigated. It consists of comparing and evaluating IDSs at the level of their specification rather than at the level of their implementation. *Our approach also presumes that IDSs can be evaluated and verified at different phases of their development and implementation. The more detailed level of attack representation is used in the attack model the more advanced level of IDS is evaluated.*

Now there are a lot of signature and traffic generation tools: FlameThrower, Fragrouter, Hailstorm, IDS Informer, MS WCAT, nidsbench, SmartBits, Stick, etc. But the majority of these tools are doing only simulated pseudorandom malicious packets. As Marcus Ranum noticed in discussions on focus-ids@securityfocus.com "Make sure that you're not only generating "signatures" but that they are within the context of apparently valid sessions – otherwise you're actually benchmarking an IDS' ability to detect false positives, not real attacks." On our sight Hailstorm (Cenzic) [17] and IDS Informer (BLADE Software) [21] have most interesting properties. *Hailstorm* generates traffic based on patterns specifying how a packet is to be generated over the network. *IDS Informer* has been designed to allow launch S.A.F.E. (Simulated Attacks for Evaluation) attacks. The S.A.F.E. process builds the attacks based on previously recorded real attacks. *In our approach the malicious and background traffic on the terminal levels is formed within the context of valid sessions.*

8 Conclusion

In the paper, attack is considered as competition of malefactor(s) and computer network security system, i.e. attack-modeling task is considered as an adversary domain. We proposed a formal grammar-based framework for attack modeling and considered the basic issues of the attack simulator development. Formal framework is built as a hierarchy of attribute stochastic context-free grammars interconnected via the "grammar substitution" operation. The framework makes it possible to specify and to simulate a wide spectrum of attacks at various levels of detail. Software

prototype of the attack simulator is developed. The attack simulator allows simulating a wide spectrum of real-life attacks. The respective software code is written using Visual C++ 6.0, Java 2 version 1.3.1, KQML and XML languages. The basic ideas of the modeling and simulation of coordinated distributed attacks are developed.

9 Acknowledgments

This research is being funded by European Office of Aerospace R&D (Projects #1994 P) and grant #01-01-108 of Russian Foundation of Basic Research.

References

1. Aho, A.V., Ullman, J.D.: The Theory of Parsing, Translation, and Compiling, Vol. 1, 2, Prentice-Hall, Inc. (1972)
2. Alessandri, D., Cachin, C., Dacier, M., Deak, O., Julisch, K., Randell, B., Riordan, J., Tscharner, A., Wespi, A., Wüest, C.: Towards a Taxonomy of Intrusion Detection Systems and Attacks. MAFTIA deliverable D3. Version 1.01. Project IST-1999-11583. Malicious- and Accidental-Fault Tolerance for Internet Applications (2001)
3. Bishop, M.: A standard audit trail format. Technical report, Department of Computer Science, University of California at Davis (1995)
4. Custom Attack Simulation Language (CASL). Secure Networks (1998)
5. Chi, S.-D., Park, J.S., Jung, K.-C., Lee, J.-S.: Network Security Modeling and Cyber Attack Simulation Methodology. Lecture Notes in Computer Science, Vol.2119 (2001)
6. Chung, M., Mukherjee, B., Olsson, R.A., Puketza, N.: Simulating Concurrent Intrusions for Testing Intrusion Detection Systems: Parallelizing Intrusions. Proceedings of the 18th NISSC (1995)
7. Cohen, F.B.: Information System Attacks: A Preliminary Classification Scheme. Computers and Security, Vol.16, No.1 (1997)
8. Cohen, F.: Simulating Cyber Attacks, Defenses, and Consequences. IEEE Symposium on Security and Privacy, Berkeley, CA (1999)
9. Cuppens, F., Ortalo, R.: Lambda: A language to model a database for detection of attacks. RAID'2000, Lecture Notes in Computer Science, Vol.1907 (2000)
10. Curry, D.: Intrusion detection message exchange format, extensible markup language (xml) document type definition. draft-ietf-idwg-idmef-xml-02.txt (2000)
11. Debar, H., Dacier, M., Wespi, A., Lampart, S.: An experimentation workbench for intrusion detection systems. Research Report RZ–2998 (# 93044). IBM Research Division, Zurich Research Laboratory (1998)
12. Eckmann, S.T., Vigna, G., Kemmerer, R.A.: STATL: An Attack Language for State-based Intrusion Detection. Proceedings of the ACM Workshop on Intrusion Detection, Athens, Greece (2000)
13. Feiertag, R., Kahn, C., Porras, P., Schnackenberg, D., Staniford-Chen, S., Tung, B.: A common intrusion specification language (cisl). specification draft (1999)
14. Fu, K.S.: Syntactic Methods in Pattern Recognition, Academic Press, New York (1974)
15. Glushkov, V., Tseitlin, G., Yustchenko, E.: Algebra, Languages, Programming. Naukova Dumka Publishers, Kiev (1978) (In Russian).

16. Gorodetski, V., Karsayev, O., Kotenko, I., Khabalov, A.: Software Development Kit for Multi-agent Systems Design and Implementation. Lecture Notes in Artificial Intelligence, Vol. 2296, Springer Verlag (2002)
17. Hailstorm. Users Manual, 1.0 (2000) http://www.clicktosecure.com/
18. Howard, J.D., Longstaff, T.A.: A Common Language for Computer Security Incidents, SANDIA REPORT, SAND98-8667 (1998)
19. Huang, M.-Y., Wicks, T.M.: A Large-scale Distributed Intrusion Detection Framework Based on Attack Strategy Analysis. RAID'98, Louvain-la-Neuve (1998)
20. Icove, D., Seger K., VonStorch, W.: Computer Crime: A Crimefighter's Handbook, O'Reilly & Associates, Inc., Sebastopol, CA (1995)
21. IDS Informer 3.0. User Guide. BLADE Software (2001)
22. Iglun, K., Kemmerer, R.A., Porras, P.A.: State Transition Analysis: A Rule-Based Intrusion Detection System. IEEE Transactions on Software Engineering, Vol. 21, No.3 (1995)
23. Kemmerer, R.A., Vigna, G.: NetSTAT: A network-based intrusion detection approach. Proceedings of the 14th ACSAC, Scottsdale, Arizona (1998)
24. Krsul, I.V.: Software Vulnerability Analysis, Ph.D. Dissertation, Computer Sciences Department, Purdue University, Lafayette, IN (1998)
25. Lindqvist, U., Jonsson, E.: How to Systematically Classify Computer Security Intrusions. Proceedings of the 1997 IEEE Symposium on Security and Privacy, IEEE Computer Society Press, Los Alamitos, CA (1997)
26. Lippmann, R., Haines, J.W., Fried, D.J., Korba, J., Das, K.: The 1999 DARPA off-line intrusion detection evaluation. RAID'2000, Lecture Notes in Computer Science, Vol.1907 (2000)
27. McHugh, J.: The 1998 Lincoln Laboratory IDS Evaluation: A Critique. RAID'2000, Lecture Notes in Computer Science, Vol.1907 (2000)
28. McHugh, J.: Intrusion and intrusion detection. International Journal of Information Security, No.1 (2001)
29. Me, L.: Gassata, a genetic algorithm as an alternative tool for security audit trails analysis. RAID'98, Louvain-la-Neuve (1998)
30. Michel, C., Me, L.: ADeLe: an Attack Description Language for Knowledge-based Intrusion Detection. Proceedings of the 16th International Conference on Information Security, Kluwer (2001)
31. Moitra, S.D., Konda S.L.: A Simulation Model for Managing Survivability of Networked Information Systems, Technical Report CMU/SEI-2000-TR-020 ESC-TR-2000-020 (2000)
32. Moore, A.P., Ellison, R.J., Linger, R.C.: Attack Modeling for Information Security and Survivability. Technical Note CMU/SEI-2001-TN-001 (2001)
33. http://www.ontology.org/
34. Paxson, V.: Bro: A system for detecting network intruders in real-time. Proceedings of the 7th Usenix Security Symposium (1998)
35. Puketza, N., Chung, M., Olsson, R.A., Mukherjee, B.: A Software Platform for Testing Intrusion Detection Systems. IEEE Software, Vol.14, No.5 (1997)
36. Stewart, A.J.: Distributed Metastasis: A Computer Network Penetration Methodology. The Packet Factory (1999) (Phrack Magazine, Vol. 9, Issue 55)
37. Tambe, M.: Towards Flexible Teamwork. Journal of Artificial Intelligence Research, No.7 (1997)
38. Vigna, G., Eckmann, S.T., Kemmerer, R.A.: Attack Languages. Proceedings of the IEEE Information Survivability Workshop, Boston (2000)
39. Yuill, J., Wu, F., Settle, J., Gong, F., Huang, M.: Intrusion Detection for an On-Going Attack. RAID'99, West Lafayette, Indiana, USA (1999)
40. Yuill, J., Wu, F., Settle, J., Gong, F., Forno, R., Huang, M., Asbery, J.: Intrusion-detection for incident-response, using a military battlefield-intelligence process. Computer Networks, No.34 (2000)