State of the Art in Modeling of Computer Attacks

Igor KOTENKO

Computer Security Research Group,

St. Petersburg Institute for Informatics and Automation of Russian Academy of Sciences

Outline

- Introduction
- Works describing attacks and attack taxonomies
- Works directly coupled with attack modeling and simulation
- Works devoted to descriptions of attack specification languages
- Works on evaluating security systems
- Formal grammar and state machines based approach
- Agent based and packet level simulation approach
- Conclusion

Outline

- Introduction
- Works describing attacks and attack taxonomies
- Works directly coupled with attack modeling and simulation
- Works devoted to descriptions of attack specification languages
- Works on evaluating security systems
- Formal grammar and state machines based approach
- Agent based and packet level simulation approach
- Conclusion

Significance of the Problem

- Vulnerabilities, variety and complexity of cyber-attacks and gravity of their consequences highlight urgent necessity for information assurance and survivability of computer systems.
- Now we see in the Internet the next step of counteraction between the means of assault and the means of defense
- Traditionally the attackers have advantage over defenders
- *Hackers characterize the current state of counteraction of malefactors' systems to security systems as "a game of network cats and mice".*
- *Modeling and simulation has become fundamental to computer science, including computer security area*

What is it "Computer Attack"?

"Any computer attack … is a **complex phenomenon** involving mixes of human behavior and interactions of complex interdependent systems. There is no widely accepted information physics that would allow to make an accurate model, and the sizes of the things we are modeling are so large and complex that we cannot describe attacks with any reasonable degree of accuracy." ([Chi et al-01]).

Computer Attack Trends (1)

Source: CERT/CC (A. Householder, K. Houle, C. Dougherty, etc.)

- **Increasing the level of automation and penetration speed**
  - Main phases of attacks: scanning for potential victims and vulnerabilities, compromising, propagating, and coordinated management
  - Now, attack tools exploit vulnerabilities as a part of the scanning activity, which increases the speed of propagation
  - Attack tools can initiate new attack cycles themselves
  - Coordination functions are very advanced …
- **Increasing speed of vulnerabilities discovery**
  - Intruders are able to discover new vulnerabilities before the vendors are able to correct them. Time to patch is increasingly small

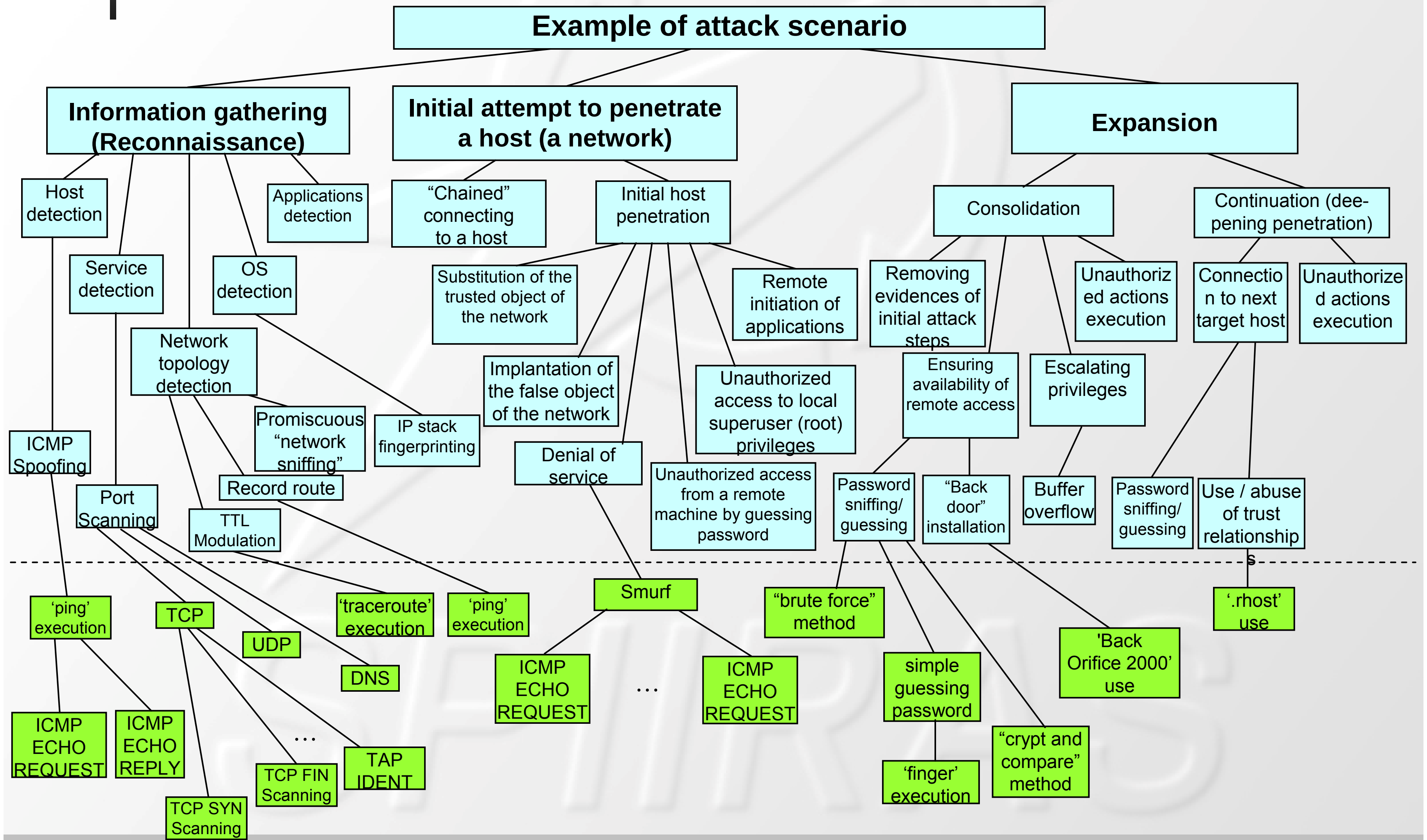
Computer Attack Trends (2)

- **Using global Internet security policy lacks**
  - a single attacker can relatively easily employ a large number of distributed systems to launch attacks
  - Infrastructure attacks (DoS, worms, DNS, and router attacks)
- **Increasing sophistication of attack tools (difficult to discover)**
  - Anti-forensics. Analysis often includes laboratory testing and reverse engineering
  - Dynamic behavior (based on random selection, predefined decision paths, or through direct intruder management)
  - Technologies are being designed to bypass typical firewall configurations (IPP (the Internet Printing Protocol), WebDAV (Web-based Distributed Authoring and Versioning)) …
- **Wars between malefactors' teams**
- …

Strategies of cyber-attacks

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

Sophisticated Attack Scenarios



Why Do We Need to Simulate Attacks?

- It could **help in deeper study of the essence and** features of different attacks (intentions of malefactors, attack objects, structure of attack scenario, strategies of multi-phase attack realization, etc.) ;
- It could be used for **active vulnerability assessment** (penetration testing) to validate implemented security systems, in particular, Intrusion detection systems (IDS);
- For **investigation of protection mechanisms**;
- To use an artificially generated sample of attacks as training and testing data sets for **security tools learning**.

Security Evaluation Areas

- **Impact assessment** for determining how security measures affect system and application properties (performance, reliability, etc.) [D.Nicol, S.Smith, M.Zhao-04 ; S.Kent, C.Lynn, K.Seo-00 (Secure BGP); M.Zhao, S.Smith, D.Nicol-05; etc.]
- **Emulation**, in which real and virtual worlds are combined to study the interaction between malware and systems, and probe for new system weaknesses [G.Bakos, V.Berk-02 (Worm activity by metering ICMP); M. Liljenstam et al-03 (Simulating worm traffic); etc.]
- **Cyberattack exercises** and training scenarios [M. Liljenstam et al-05 (RINSE); B. Brown et al-03; etc.]
- **Risk assessment** based on known vulnerabilities, exploits, attack capabilities, and system configuration [R. Ortalo, Y.Deswarte, M.Kaaniche-99; Sheyner et al-02; V.Gorodetski, I.Kotenko-02 (Attack Simulator); B.Madam, K.Goseva-Popstojanova-02; E.Pogossian, A.Javadyan-03; etc.]

Source: DAVID M. NICOL

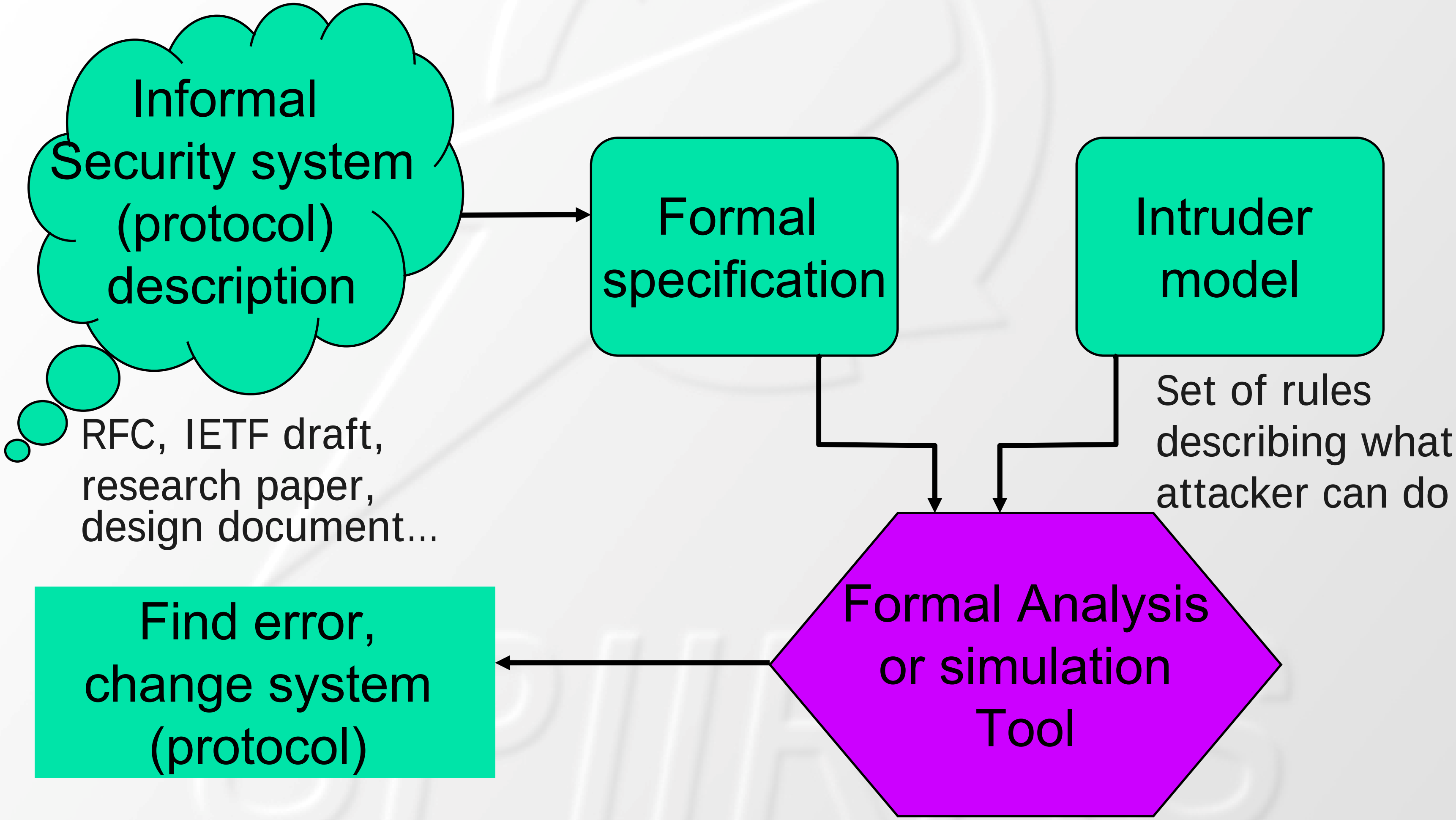
Security Analysis

1. Model system
2. Model adversary
3. Identify security properties
4. See if properties preserved under attack

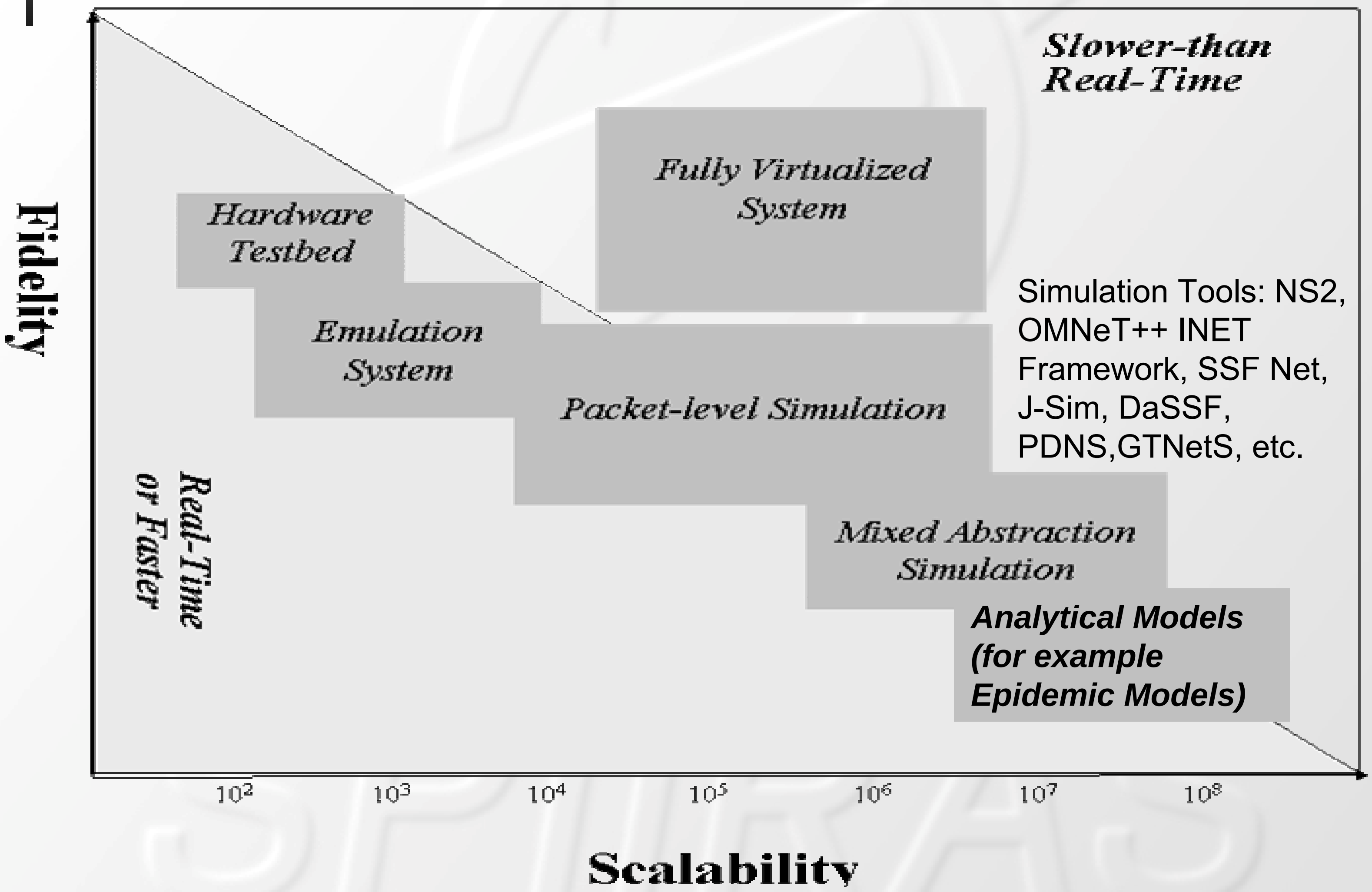
- Result
  - Under given assumptions about system, no attack of a certain form will destroy specified properties

/Vitaly Shmatikov/

Explicit Intruder Method



Range of Modeling and Simulation Alternatives



Source: [K.Perumalla, S.Sundaragopalan-04]

Works Related to Attack Modeling

- Describing attacks and attack taxonomies
- Describing particular classes of attacks
- Directly coupled with attack modeling and simulation
- Devoted to descriptions of attack specification languages
- On evaluating security systems
- Devoted to signatures and traffic generation tools
- ……………………

Outline

- Introduction
- Works describing attacks and attack taxonomies
- Works directly coupled with attack modeling and simulation
- Works devoted to descriptions of attack specification languages
- Works on evaluating security systems
- Formal grammar and state machines based approach
- Agent based and packet level simulation approach
- Conclusion

List of main works

- Lists of attack terms ([Cohen-95], [Icove *et al*-95], [Cohen-97], [Howard *et al*-98]);
- Lists of attack categories ([Cheswick *et al*-94], [Ranum-97]);
- Attack results categories ([Cohen-95], [Russell *et al*-91]);
- Empirical lists of attack types ([Lackey-74], [Neumann *et al*-89], [Amoroso-94], [Lindqvist *et al*-97]);
- Vulnerabilities matrices ([Amoroso-94], [Landwehr *et al*-94]);
- Action-based taxonomies [Stallings-95];
- Security flaws or vulnerabilities taxonomies ([Beizer-90], [Saltzer *et al*-75], [Hogan-88], [Aslam-95], [Dodson-96], [Krsul-98]);
- Taxonomies of intrusions based on the signatures [Kumar-95];
- Incident taxonomies [Howard *et al*-98],
- etc.

Lists of attack terms

[Cohen-95]

Trojan horses	*Toll fraud networks*	*Fictitious people*	*Infrastructure observation*	*E-mail overflow*
Time bombs	*Get a job*	*Protection limit poking*	*Infrastructure interference*	*Human*
engineering				
Bribes	*Dumpster diving*	*Sympathetic vibration*	*Password guessing*	*Packet insertion*
Data diddling	*Computer viruses*	*Invalid values on calls*	*Van Eck bugging*	*Packet watching*
PBX bugging	*Shoulder surfing*	*Open microphone listening*	*Old disk information*	*Video viewing*
Backup theft	*Data aggregation*	*Use or condition bombs*	*Process bypassing*	*False update disks*
Input overflow	*Hang-up hooking*	*Call forwarding fakery*	*Illegal value insertion*	*E-mail spoofing*
Login spoofing	*Induced stress failures*	*Network services attacks*	*Combined attacks*	

[Icove *et al*-95]

Wiretapping	*Dumpster diving*	*Eavesdropping on Emanations*	*Denial-of-service*	*Harassment*
Masquerading	*Software piracy*	*Unauthorized data copying*	*Degradation of service*	*Traffic analysis*
Trap doors	*Covert channels*	*Viruses and worms*	*Session hijacking*	*Timing attacks*
Tunneling	*Trojan horses*	*IP spoofing*	*Logic bombs*	*Data diddling*
Salamis	*Password sniffing*	*Excess privileges*	*Scanning*	

Computer and Network Incident Taxonomy [Howard *et al*-98]

Incident

Attack

Event

Attackers	**Tool**	**Vulnerability**	**Action**	**Target**	**Unauthorized Result**	*Objectives*
Hackers	Physical Attack	Design	Probe	Account	Increased Access	Challenge, Status, Thrill
Spies	Information Exchange	Implementation	Scan	Process	Disclosure of Information	Political Gain
Terrorists	User Command	Configuration	Flood	Data	Corruption of Information	Financial Gain
Corporate Raiders	Script or Program		Authenticate	Component	Theft of Resources	Damage
Professional Criminals	Autonomous Agent		Bypass	Computer	Denial of Service	
Vandals	Toolkit		Spoof	Network		
Voyeurs	Distributed Tool		Read	Inter-network		
	Data Tap		Copy			
			Steal			
			Modify			
			Delete			

Outline

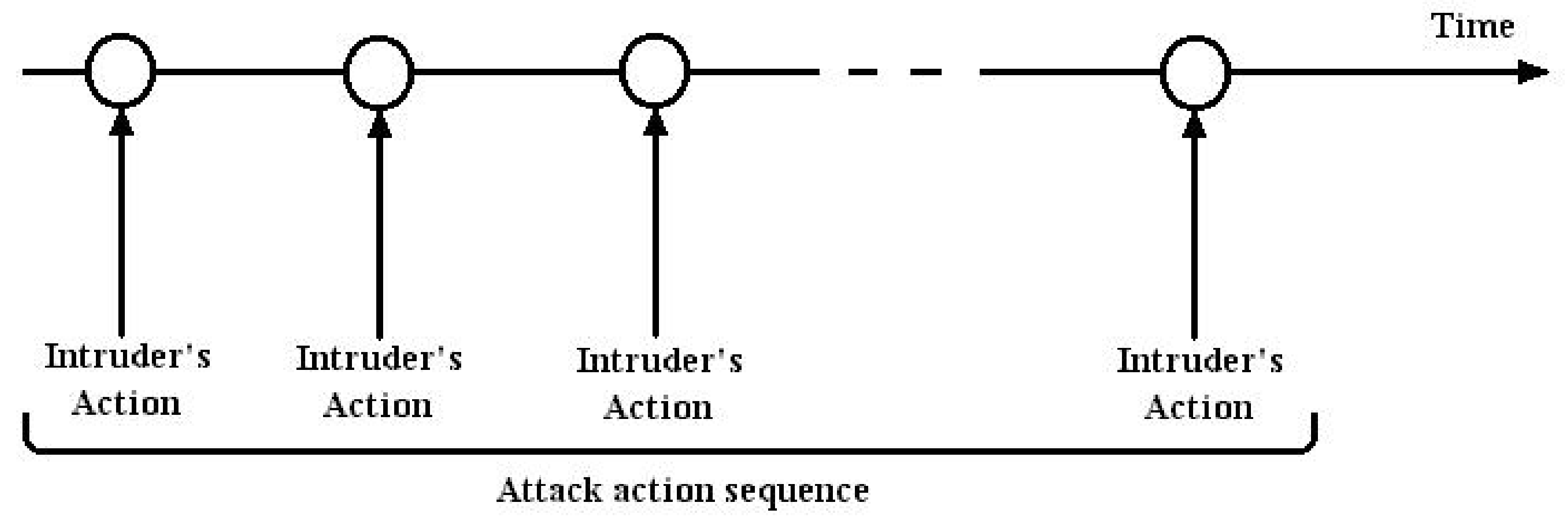
- Introduction
- Works describing attacks and attack taxonomies
- **Works directly coupled with attack modeling and simulation**
- Works devoted to descriptions of attack specification languages
- Works on evaluating security systems
- Formal grammar and state machines based approach
- Agent based and packet level simulation approach
- Conclusion

List of main works

- Temporal model of intrusion [Amoroso-99]
- Using Colored Petri Nets [Kumar *et al*-94]
- State transition analysis technique [Iglun *et al*-95], [Kemmerer *et al*-98]
- Conceptual models of computer penetration ([Cohen-99],[Stewart-99])
- Descriptive models of attackers [Yuill *et al*-00]
- "Tree"-based models of attacks ([Huang *et al*-98], [Schneier-99], [Moore *et al*-01], [Dawkins *et al*-02])
- Modeling survivability of networked systems [Moitra *et al*-01]
- Object-oriented Discrete Event Simulation [Chi *et al*-01]
- Situation calculus and goal-directed procedure invocation [Goldman-02]
- Using and building attack graphs for vulnerability analysis ([Swiler *et al*-01], [Ortalo *et al*-01], [Sheyner *et al*-02], [Jha *et al*-02])
- Game-theoretic models [Lye and Wing-03]
- Multi-stage attack analysis [Dawkins, Hale-04]
- Modeling and inference of attacker [Liu, Zang-05], etc.

Temporal Model of Intrusion [Amoroso-99]

- In a temporal model of attack realization, an intruder begins with some initial action, and this action is followed by supporting actions, etc.
- Response and other actions may also be involved, and the security officer, normal users, other intruders, and so on may initiate these actions.
- The resultant sequence of actions models the exploitation of vulnerabilities to bring about the unauthorized security threat.



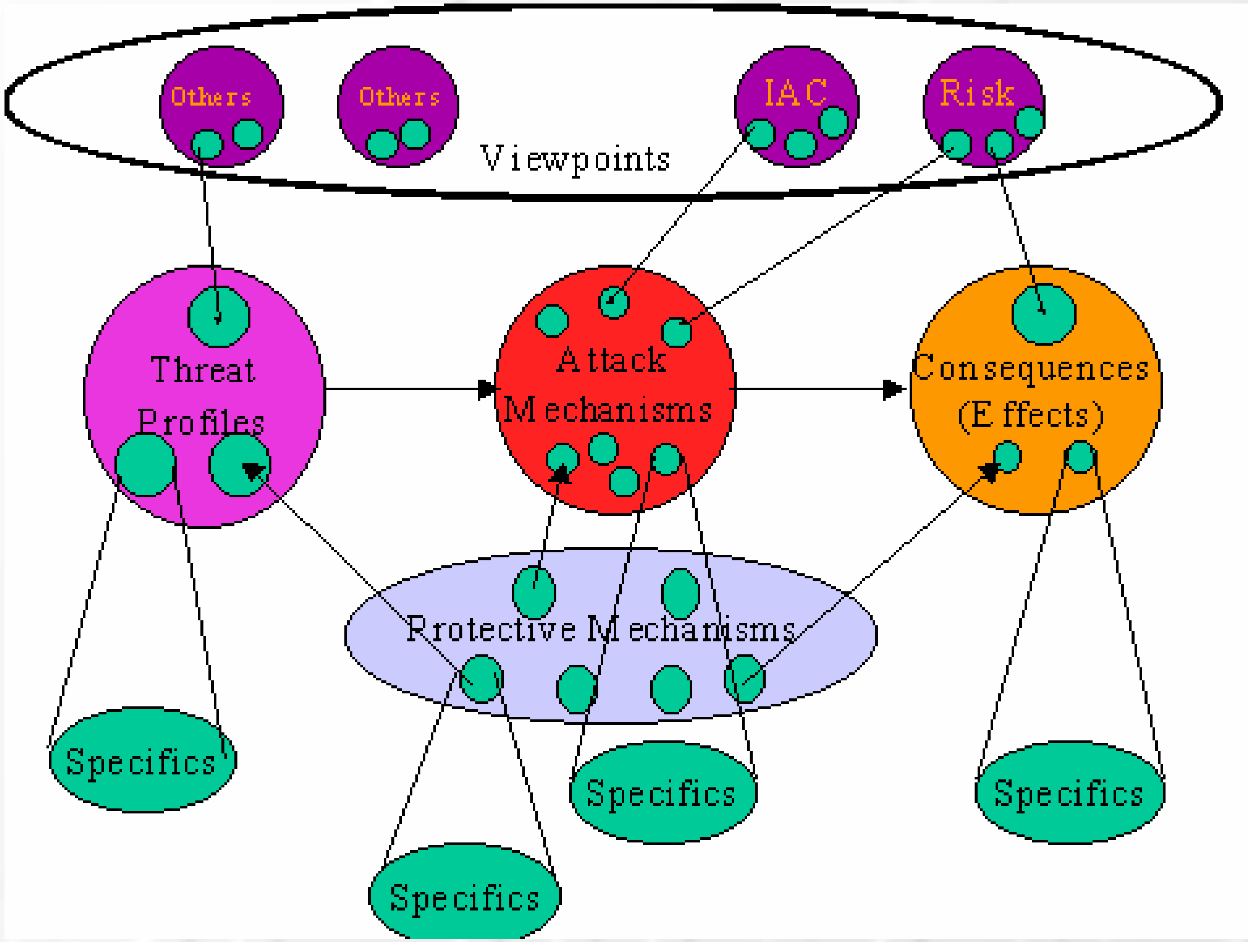
Using Colored Petri Nets [Kumar *et al*-94]

- Each intrusion signature is expressed as a pattern that represents the relationship among events and their context.
- The notions of start and final states, and paths between them determine the set of event sequences.
- Intrusion patterns have preceding conditions and following actions associated with them.

State transition analysis technique [Iglun *et al*-95]

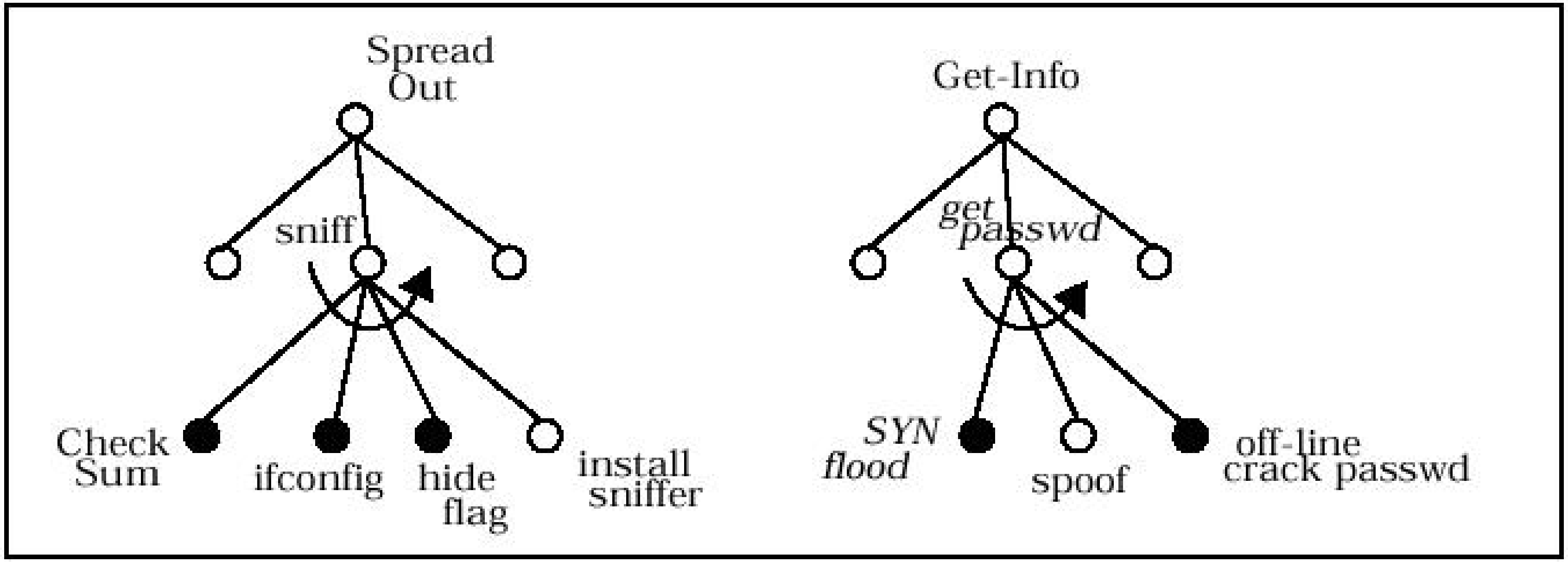
- Computer penetrations are described as sequences of actions that an attacker performs to compromise the security of a computer system.
- Attacks are described by using state transition diagrams.
- The description of an attack has a “safe” starting state, zero or more intermediate states, and (at least) one “compromised” ending state.
- States are characterized by means of assertions describing aspects of the security state (file ownership, user identification, user authorization).

Cause-effect model of cyber attack [Cohen-99]



High-level Attack Model based on Intruder's Intent [Huang et al-98]

- Intrusion intention is determined as the goal-tree.
- The ultimate goal of intrusion corresponds to the root node.
- Lower level nodes represent alternatives or ordered sub-goals in achieving the upper node/goal.
- The "OR", "AND", and "Ordered-AND" constructs are used for representation of temporal sequences of intrusion intentions.



Representation of flooding/spoofing/sniffing sequences

Attack trees [Schneier-99]

- "*AND*" and "*OR*" nodes are used in attack trees.
- *OR* nodes are alternatives.
- *AND* nodes represent different steps toward achieving the same goal.

“Tree”- based Approach [Moore et al-01]

- “An enterprise typically has a set, or forest, of attack trees that are relevant to its operation. The root of each tree in a forest represents an event that could significantly harm the enterprise’s mission.
- Two structures are used for attack representation:
  (1) *attack pattern* (characterizing an individual type of attack),
  (2) *attack profile* (organizing attack patterns to make it easier to apply them).
- Each attack pattern contains: the overall goal of the attack, a list of preconditions for its use, the steps for carrying out the attack, a list of post-conditions that are true if the attack is successful.
- Attack profiles contain a common reference model, a set of variants, a set of attack patterns, and a glossary of defined terms and phrases.

Modeling survivability of networked systems [Moitra *et al*-01]

- The model consists of three sub-models.
  - The first one simulates the occurrence of attacks or incidents.
  - The second one evaluates the impact of an attack on the system depending on the attack type and the protection system maturity.
  - 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. Each occurrence time *tk* of the *k*-th incident in a temporal point-process has a mark *jk* associated with it, where *jk* will have values in a specified space. The mark has to take into account the severity of the incident and the possibility of single, or multiple and simultaneous attacks.

Situation calculus and goal-directed procedure invocation [Goldman-02]

- The suggested computer network attack model uses an action representation based on the Golog *situation calculus* and *goal-directed* procedure invocation.
- Goldman has designed components of a stochastic attack simulator which can simulate some goal-directed attacks on a network.
- Using the situation calculus, the developed attack simulator can project the results actions with complex preconditions and context-dependent effects.
- The goal-directed invocation permits to express attacker plans like “first attain root privilege on a host trusted by the target, and then exploit the trust relationship to escalate privilege on the target”.

Technique for generating and analyzing attack graphs ([Sheyner et al-02], [Jha et al-02])

- The technique is based on symbolic model checking algorithms ([Clarke *et al*-00], [SMV], [NuSMV]), letting construct attack graphs automatically and efficiently. The authors implemented the technique in a tool suite and tested it on a small network example.
- Authors suggested applying this technique and the tool suite for vulnerability analysis of a network. A typical process for vulnerability analysis proceeds as follows.
  - First, vulnerabilities of individual hosts (using scanning tools) are determined.
  - Using this local vulnerability information along with other information about the network, such as connectivity between hosts, they then produce attack graphs. Each path in an attack graph is a series of exploits, which they call atomic attacks, that leads to an undesirable state.
  - Then further analyses (such as risk analysis, reliability analysis, or shortest path analysis) are performed.

Outline

- Introduction
- Works describing attacks and attack taxonomies
- Works directly coupled with attack modeling and simulation
- **Works devoted to descriptions of attack specification languages**
- Works on evaluating security systems
- Formal grammar and state machines based approach
- Agent based and packet level simulation approach
- Conclusion

Attack languages and their classification

- *Attack languages* are used with the purpose of attack recognition, analysis of the relations between various attacks, response on them and documenting of intrusions. Besides, attack languages can be used for fixing the scenarios and prehistory of attacks, and also for reproduction of attacks with the purposes of testing intrusion detection systems ([Vigna *et al*-00], [Eckmann *et al*-00]).
- Attack languages are classified using various tags. In particular, in [Vigna *et al*-00] the classification of the attack description languages is offered, according to which six types of languages are entered:
  - event languages;
  - exploit languages;
  - reporting languages;
  - detection languages;
  - correlation languages;
  - response languages.

Types of attack languages

- *Event languages* ([BSM-91], [Jacobson et al-00], [Bishop-95], etc.) describe the format of events used during the detection process.
- *Exploit languages* ([CASL-98], [Deraison-99], etc.) are used to describe the stages to be followed to perform an intrusion.
- *Reporting languages* ([Feiertag et al-99], [Curry-00]) describe the format of alerts produced by the IDS.
- *Detection languages* ([Kumar *et al*-95], [Paxson-98], [Roesch-99], [Turner et al-00], [Eckmann *et al*-00], [Me-98]) allow the expression of the manifestation of attacks.
- *Correlation languages* permit analysis of alerts provided by several IDS.
- *Response languages* are used to express countermeasures to attacks.

List of main works

	Event languages	Exploit languages	Reporting languages	Detection languages	Correlation languages	Response languages
Tcpdump [Jacobson *et al*-00]	+					
Bishop [Bishop-95]	+					
CASL [CASL-98]		+				
NASL [Deraison-99]		+				
CISL [Feiertag *et al*-99]			+			
IDMEF [Curry-00]			+			
Kumar [Kumar *et al*-95]				+		
BRO [Paxson-98]				+		
Snort [Roesch-99]				+		
SNP-L [Turner *et al*-00]				+		
STATL [Eckmann *et al*-00]				+		
GasSATA [Me-98]				+		
LAMBDA [Cuppens *et al*-00]				+	+	
AdeLe [Michel *et al*-01]				+	+	+

STATL

- **STATL** is an extensible attack language designed to support intrusion detection [Eckmann et al-00].
- The STATL provides constructs to represent an attack as a composition of states and transitions.
- States are used to characterize different snapshots of a system during the evolution of an attack.
- A transition has an associated action that is a specification of the event that may cause the scenario to move to a new state.

AdeLe

- **AdeLe** is designed to model a database of known attack scenarios [Michel et al-01]. An ADeLe description looks like a function in C programming language with name and parameters.
- The description body is made up of three parts: exploit part, detection part, and response part.
- The exploit part represents the attacker's point of view. It is composed of three sections: pre-condition, attack, and post-condition.
- The pre-condition section expresses the requirements for launching the attack. These are data about the target operating system, installed software, the vulnerabilities, the level of privilege needed by the attacker to launch a successful attack, etc.
- The attack section determines the source code of the attack that can be expressed in different languages (“*C*”, “*C++*”, “*Perl*”, “*Casl*”, “*Nasl*”, etc.).

Outline

- Introduction
- Works describing attacks and attack taxonomies
- Works directly coupled with attack modeling and simulation
- Works devoted to descriptions of attack specification languages
- **Works on evaluating security systems**
- Formal grammar and state machines based approach
- Agent based and packet level simulation approach
- Conclusion

List of main works

- Methodology and software tools for testing IDSs ([Puketza *et al*-96], [Puketza *et al*-97], [Debar *et al*-98], [Alessandri *et al*-01], [McHugh-00]);
- Evaluations of IDSs of MIT ([Lippmann *et al*-98, 00, 02]);
- Real-time test bed of AFRL [Durst *et al*-00];
- Dependability models for evaluation security [Nicol *et al*-04];
- Penetration testing of formal models of networks for estimating security metrics [Sheyner *et al*-02];
- Model checking for analysis of network vulnerabilities [Ritchey, Ammann-00];
- Global metrics for analyzing the effects of complex network faults and attacks [Hariri *et al*-03];
- Natural-deduction for automatic generation and analysis of attacks against IDS [Rubin *et al*-04];
- Knowledge-based approach to network risk assessment [Shepard *et al*-05], etc.

Outline

- Introduction
- Works describing attacks and attack taxonomies
- Works directly coupled with attack modeling and simulation
- Works devoted to descriptions of attack specification languages
- Works on evaluating security systems
- **Formal grammar and state machines based approach**
- Agent based and packet level simulation approach
- Conclusion

Lessons Learnt from Study of Attacks Peculiarities

1: Formal model of a distributed attacks implemented by team of malefactors has to have at least three-level structure:

Upper level - intention-based scenarios of malefactors' team.

Middle level - intention-based scenarios of each malefactor.

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

2: Attack is being developed dynamically and depends on the attacked network response and on effectiveness of the malefactor's actions (like any adversary domain).

3: Formal model of attack against computer network should be capable to represent many uncertainties inherent to the real-life practice of attacker and computer network security system response.

Ontology of Attacks: Fragment of Ontology at Macro-Level

• The **ontology of attacks and defense mechanisms** comprises a hierarchy of notions specifying activities of the team of information warriors who aim to implement attacks and protection against them at different levels of detail.

• In this ontology, the hierarchy of nodes representing notions can be divided into two subsets according to the *macro-* and *micro-levels* of the domain specifications.

• The notions of the ontology of an upper level can be interconnected with the corresponding notions of the lower level through one of the following kinds of *relationships*:

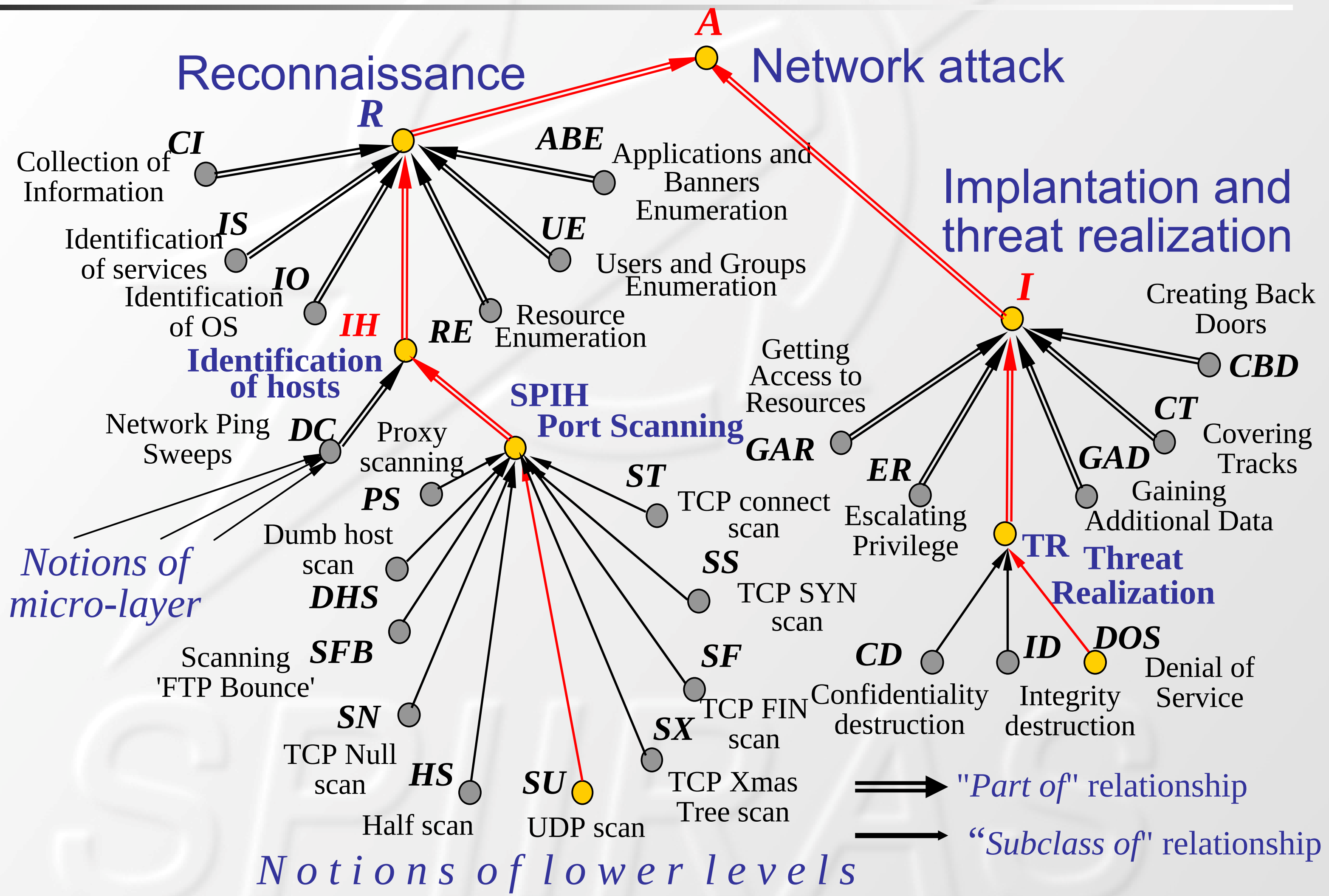
(1) "*Part of*" (decomposition);

(2) "*Kind of*" (specialization);

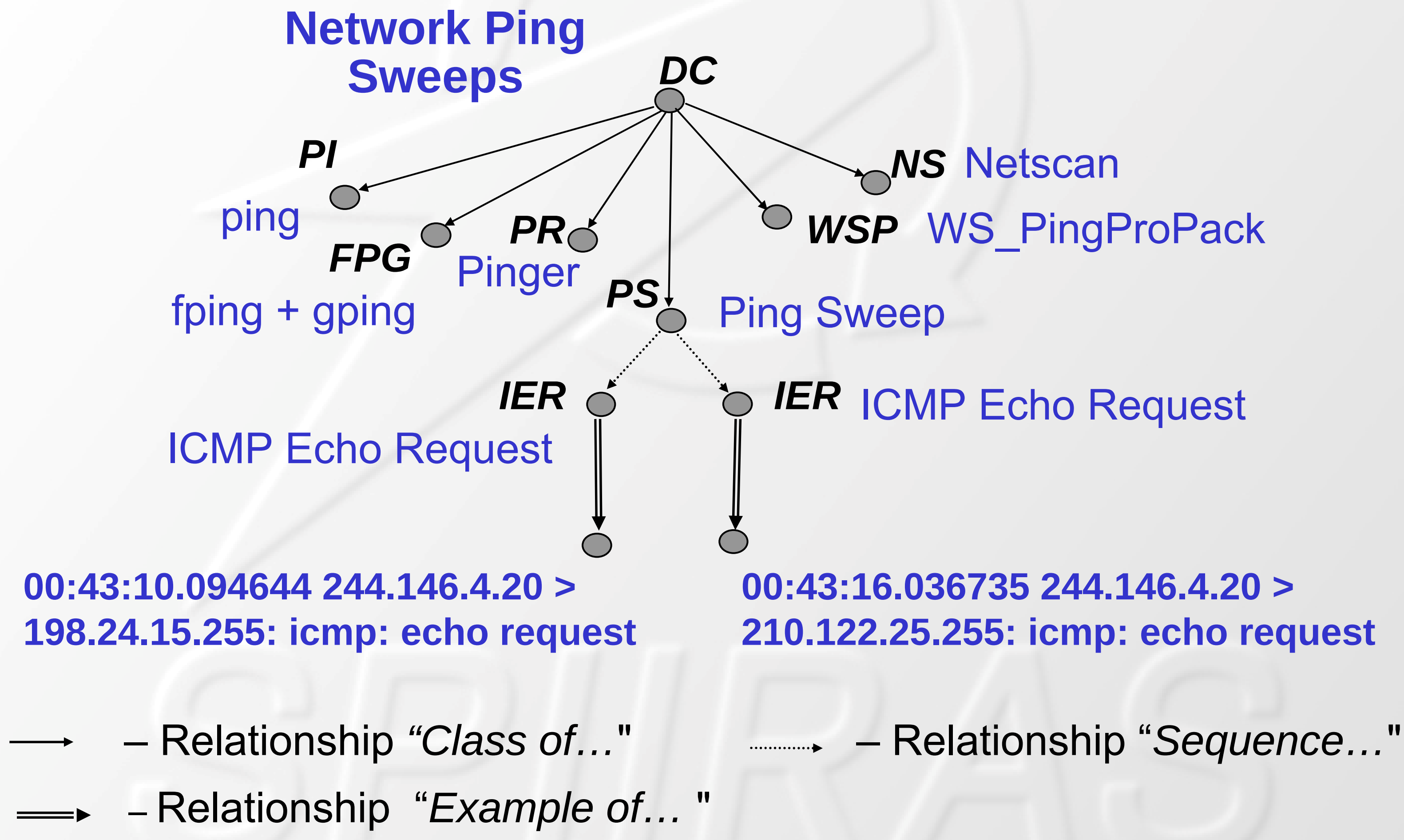
(3) "*Seq of*" (sequence of operation).

(4) "*Example of*" ("type of object – specific sample of object").

Ontology of Computer Network Attacks: Fragment of Ontology at Macro-Level



Ontology of computer network attacks: Fragment of Ontology at Micro-Level



Basic Malefactors' Intentions

Intention-centric approach to the specification of malefactor's activity: basic notions of the domain correspond to the malefactor intentions and all other notions are structured according to the structure of intentions.

List of Basic Classes of High-level Malefactor's Intentions:

R – *Reconnaissance:*
- *IH* – *Identification* of the running *Hosts*
- *IS* – *Identification* of the host *Services*
- *IO* – *Identification* of the host *Operating* system
- (*CI* – *Collection* of additional *Information*)
- *RE* – shared *Resource Enumeration*
- *UE* – *Users* and groups *Enumeration*
- *ABE* – *Applications* and *Banners Enumeration*

I – *Implantation* and threat realization:
- *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*
  - *CD* – *Confidentiality Destruction*
  - *ID* – *Integrity Destruction*
  - *DOS* – *Denial of Service*
- *CT* – *Covering Tracks*
- *CBD* – *Creating Back Doors*

Formal Grammar Framework for Specification of Hackers' Plans (1)

Higher level formal model of attack generation:

$$M_A = <\{ G_i \}, Sub>,$$

where M_A – meta-grammar, $\{G_i\}$ – set of (attribute stochastic) grammars,

Sub – "substitution" operation.

Each grammar of the set $\{G_i\}$ corresponds to a node of the ontology.

Each terminal symbol of an upper level grammar is mapped to the name of the axiom (grammar) of a lower level grammar.

Use of substitution operation semantically corresponds to more detailed specification of an attack scenario.

Formal Grammar Framework for Specification of Hackers' Plans (2)

Formal grammar: $G_i = \langle V_N, V_T, S, P, A \rangle$,

where G_i – formal grammar name (it coincides with the name of attack and the name of its axiom); V_N – the set of non-terminal symbols; V_T – the set of terminal symbols; $S \in V_N$ – formal grammar axiom;

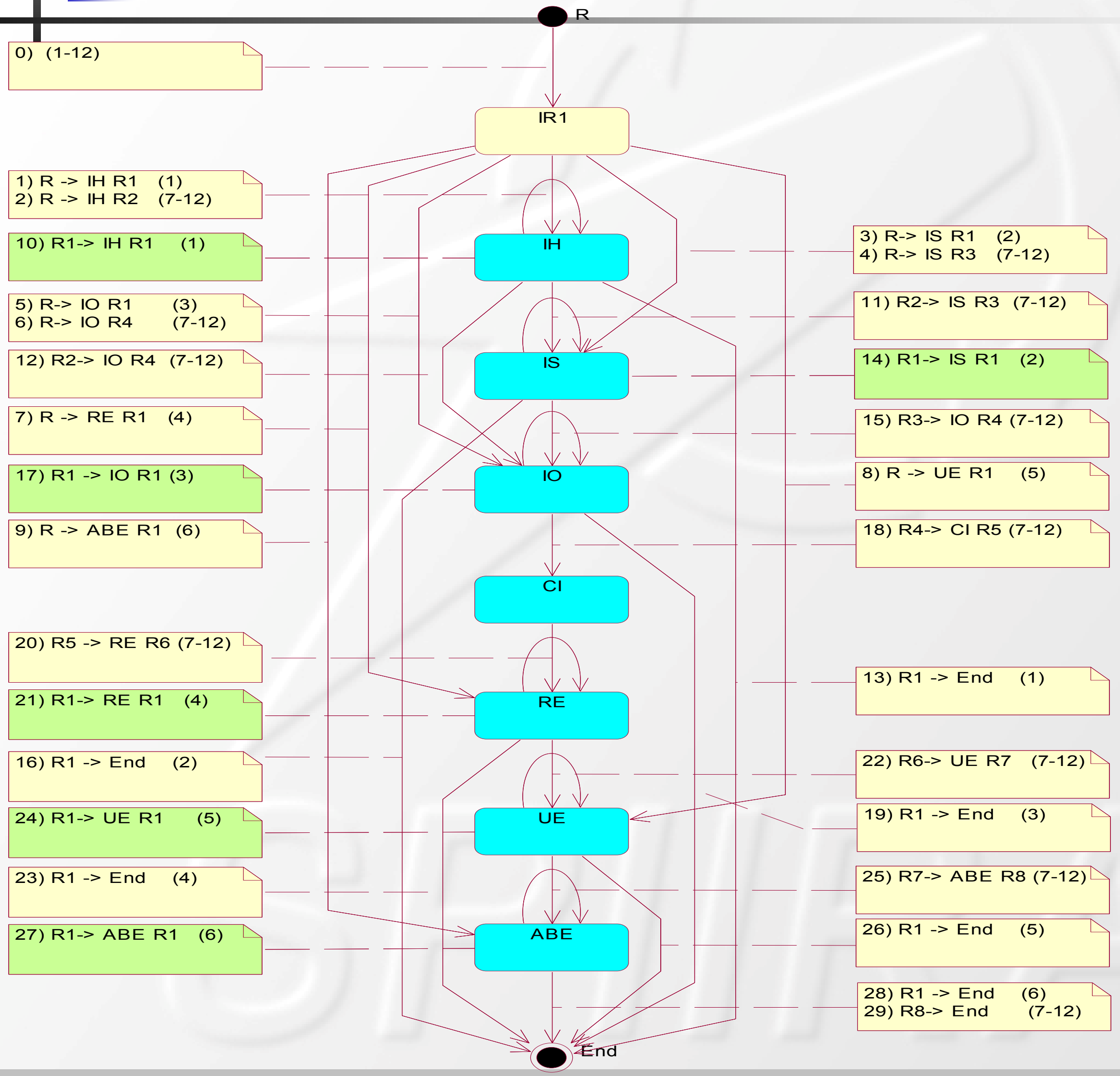
P – the set of productions which look like follows:

$$(U)\ X \rightarrow \alpha\ (Prob),$$

where $X \in V_N$, $\alpha \in (V_T \cup V_N)^*$, U – precondition of the production application; $Prob$ – probability of the production application;

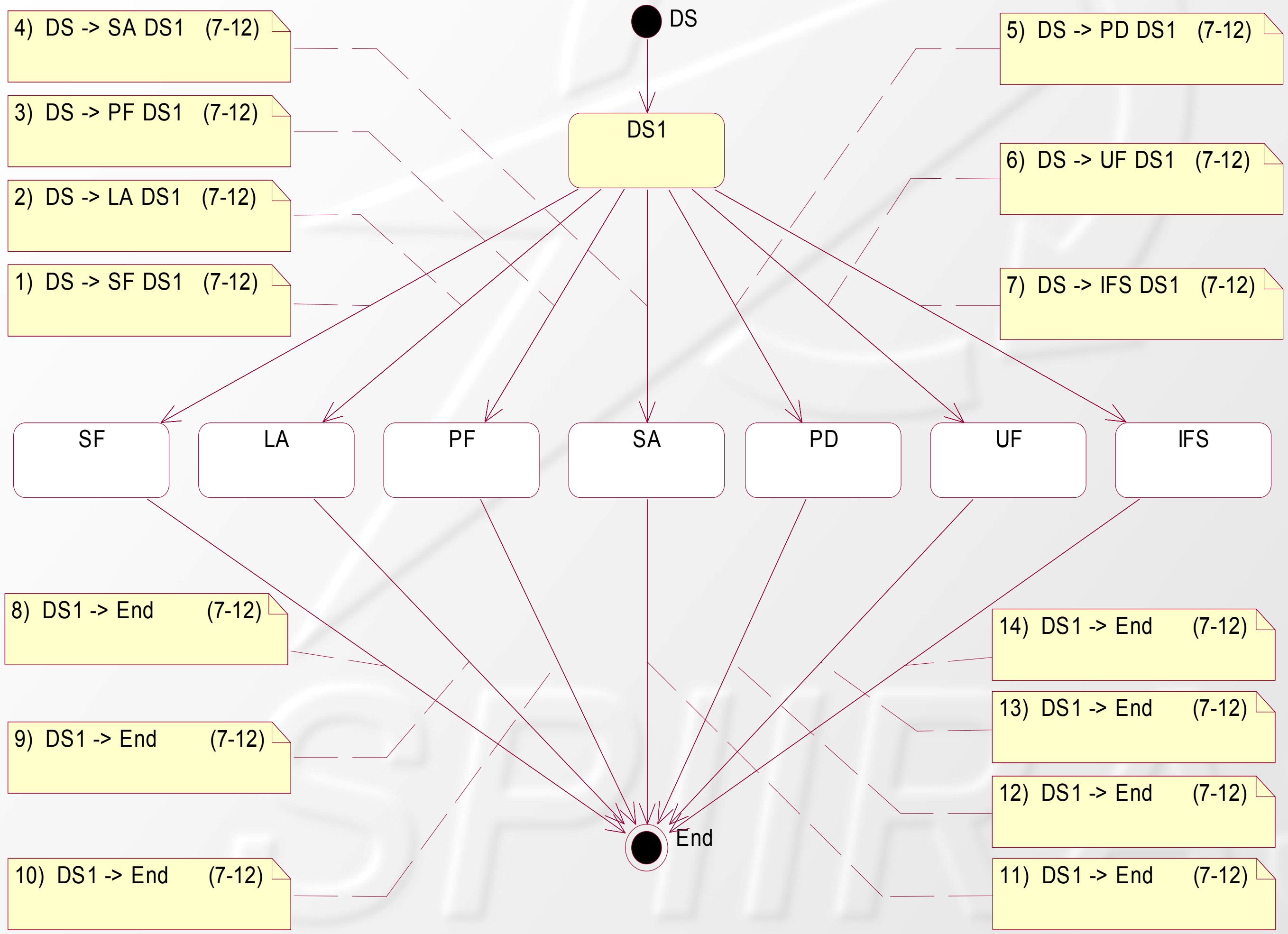
A – the set of attributes and their dependencies (functions having attributes as variables).

Implementation Issue: State Machine-based Representation of "Reconnaissance"



IR1 – Intermediate state
IH – Identification of running Hosts
IS – Identification of Services
IO – Identification of OS
CI – Collection of Information
RE – Resource Enumeration
UE – Users and Groups Enumeration
ABE – Applications and Banners Enumeration

Implementation Issue: State Machine-based Representation of DoS Attack



DS1– Intermediate state,

SF – SYN flood,

LA – Land attack,

PF – Ping flood,

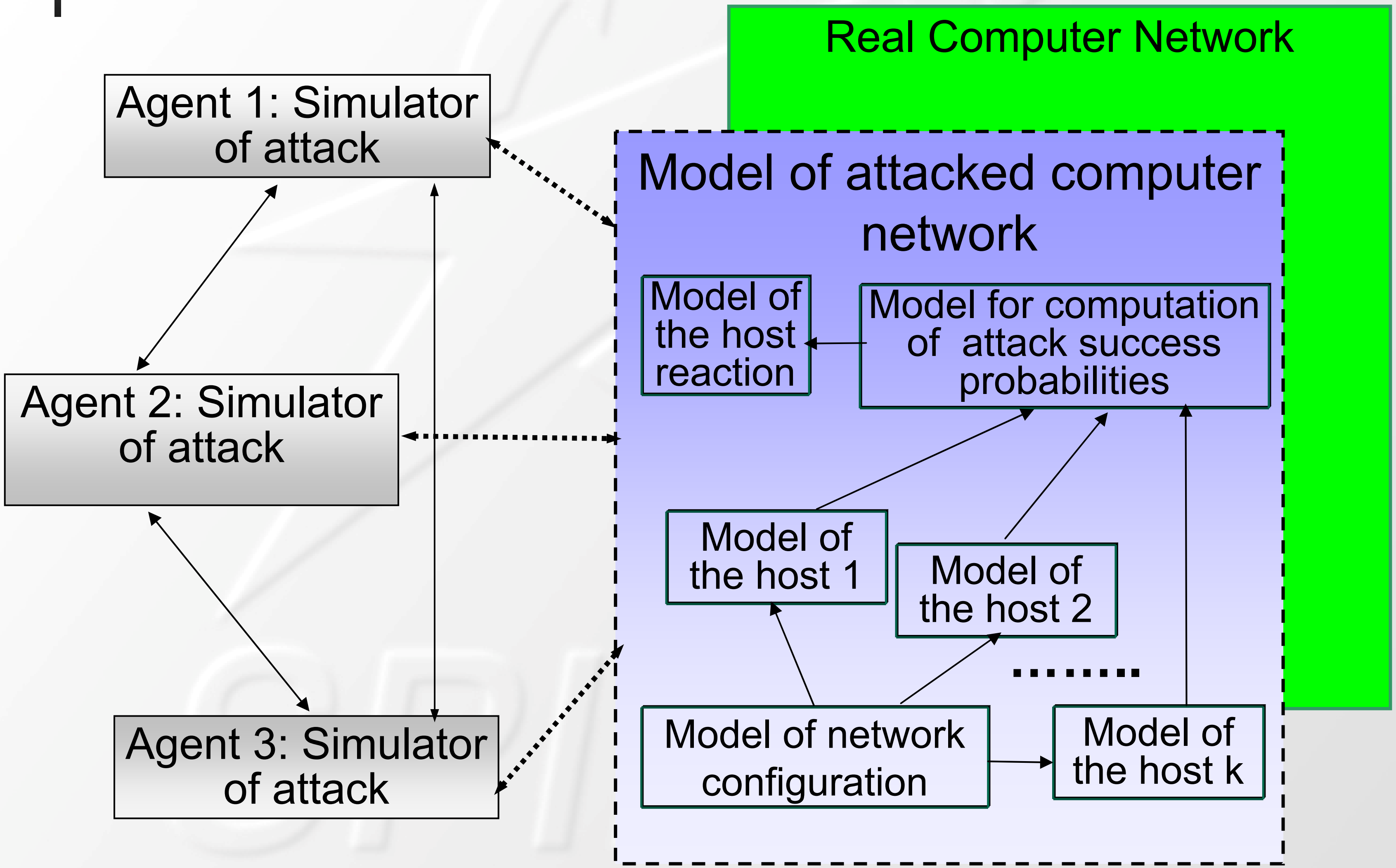
SA – Smurf attack,

PD – Ping of Death,

UF – UDP flood,

IFS – Storm of inquiries to FTP-server

Interaction of attack agents and computer network model



Model of computer network configuration

$$M_{CN} = < A, P, N, C > ,$$

where

A – computer network address;

P – the set of network protocols;

N – the set $\{CN_i\}$ of sub-networks of the computer network **CN** and/or the set $\{H_i\}$ of hosts of **CN**;

C – model of connections between sub-networks and/or hosts given in the form of matrix of connections.

Each of $\{CN_i\}$ (if any), in turn, is specified formally by the model M_{CNi}. in the form (1).

Host Model

***M_{Hi}* =<A,M,T,N,D,P,S,DP, ASP, RA, SP, SR, TH, … >**

- ***A*** – address,
- ***M*** – sub-network masks,
- ***T*** – types and versions of Operation Systems,
- ***N*** – users' identifiers,
- ***D*** – domain names,
- ***P*** – passwords
- ***S*** – secure users' identifier (*SID*),
- ***DP*** – domain variables,
- ***ASP*** – running services and ports of the host,
- ***RA*** – running applications,
- ***SP*** – security parameters,
- ***SR*** – shared resources,
- ***TH*** – trusted hosts, etc..

Model of "security policy"

Model of Computation of Probabilities of an attack success: examples of computation of the probabilities

Attack action		Pre-condition (Host attributes constraining an attack applicability)				Proba-bility
Attack ID	Name of attack	Operation System		Service, version	Other Attributes	
		Type	Version			
STIH	*TCP connect scan*	.	.			0.9
SFI	*TCP FIN scan*	Unix, Linux	.			0.9
CNS	*Connection "null sessions"*	Win	.	NetBIOS		0.5
LA	*Land attack*	.	.			0.3

Example of computation of the success probability:

*If action is "**SFI** (TCP FIN scan)" and Type of OS = "Unix,Linux" then probability of success is 0.9".*

Model of the Host Reaction

"Input → Output & Post-Condition".

Format of Input (represented in KQML+XML languages):

<Attack name>:<Message>:<Attack objects>(<Objects involved into attack>).

Output message depends on success or ineffectiveness of an attack.

Format of Output message (KQML+XML):

*< Result {Success (**S**), Failure (**F**)} > : <message> .*

Examples of output messages:

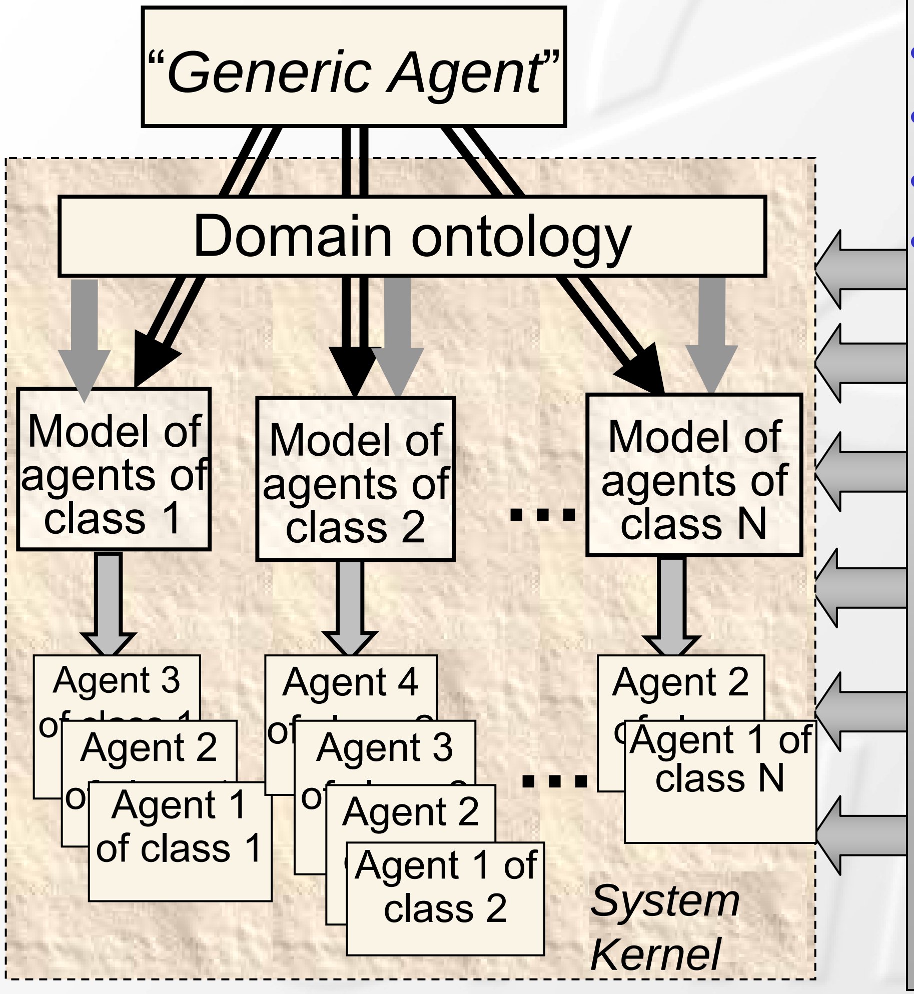
Input	Output
***TZ*: Telnet connection and analysis of the host message header concerning OS: <Target host >(<*Telnet-server*>)**	***S*: <Type of OS>** ***F*: "Type of OS not detected"**
***TS*: Telnet connection and sending command SYST<(for *Unix/Linux*): <Target host>(<*Telnet*-server>)**	***S*: <Type of OS>** ***F*: "Type of OS not detected"**
☞☞:FTP connection and analysis of bin-files in */bin/ls* (for *Unix/Linux*): <Target host> (<*FTP*- server>)	***S*: <Type of OS>** ***F*: "Type of OS not detected"**
***RF*: Exploration by *FIN*-packet: <Target host>**	***S*: <Type of OS>** ***F*: "Type of OS not detected"**

Simulation Tool Implementation: Technology of MAS Design and Implementation

1. Detailed specification of MAS in terms of the developed specification language resulting in design of a so-called "*System kernel* ";
2. Generation software code of the application and its installation in the network computers.

Both these steps are carried out by a MAS developer(s) starting from "*Generic agent* " using "*Multi-agent system development kit*" that is a software tool for MAS system design and implementation on the basis of "*Library of domain classes*".

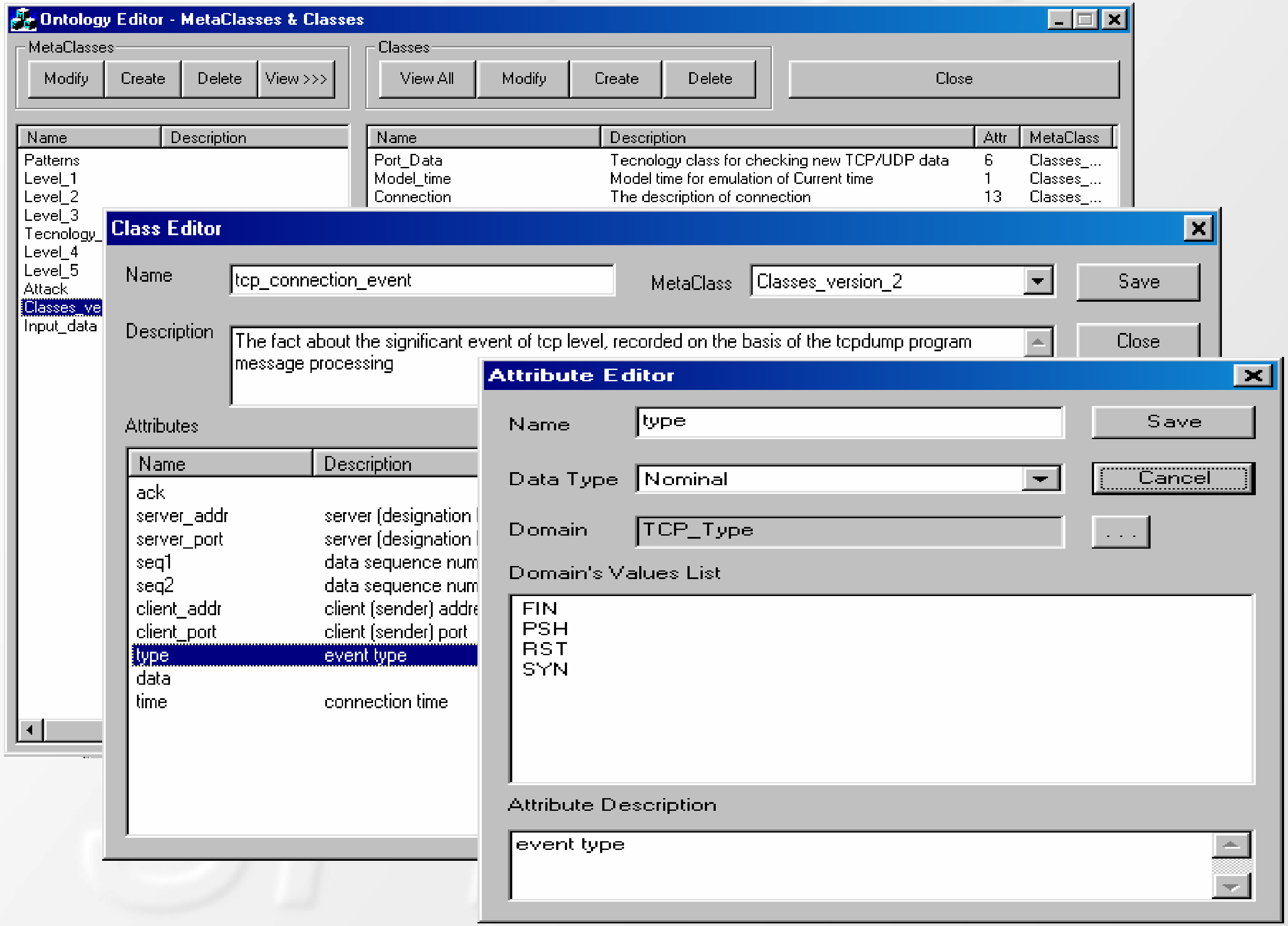
Simulation Tool Implementation: Technology of MAS Design and Implementation



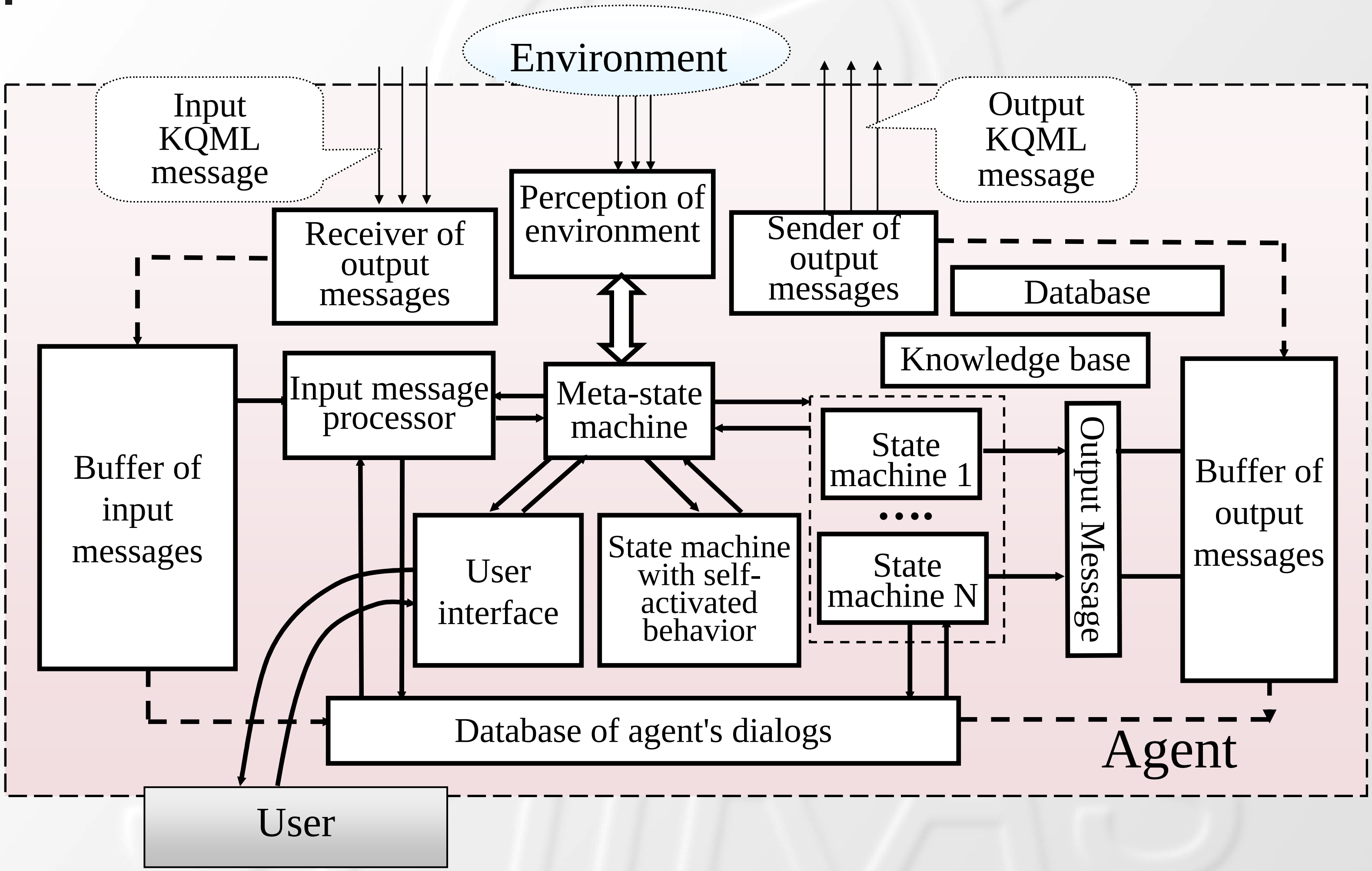
MAS DK

- Editor of System Kernel;
- MAS Ontology editor;
- Cloning System editor;
- Editors of Agent's class components:
    - Editor of agent's ontology;
    - Agent class instances generator;
    - Message templates editor;
    - Editors of notions of agent's class ontology;
    - Generator of agent class DB;
    - Three editors of state machines;
    - Editor of behavior scripts
    - Meta-state machine editor

Simulation Tool Implementation: Dialog Windows of Ontology Editor



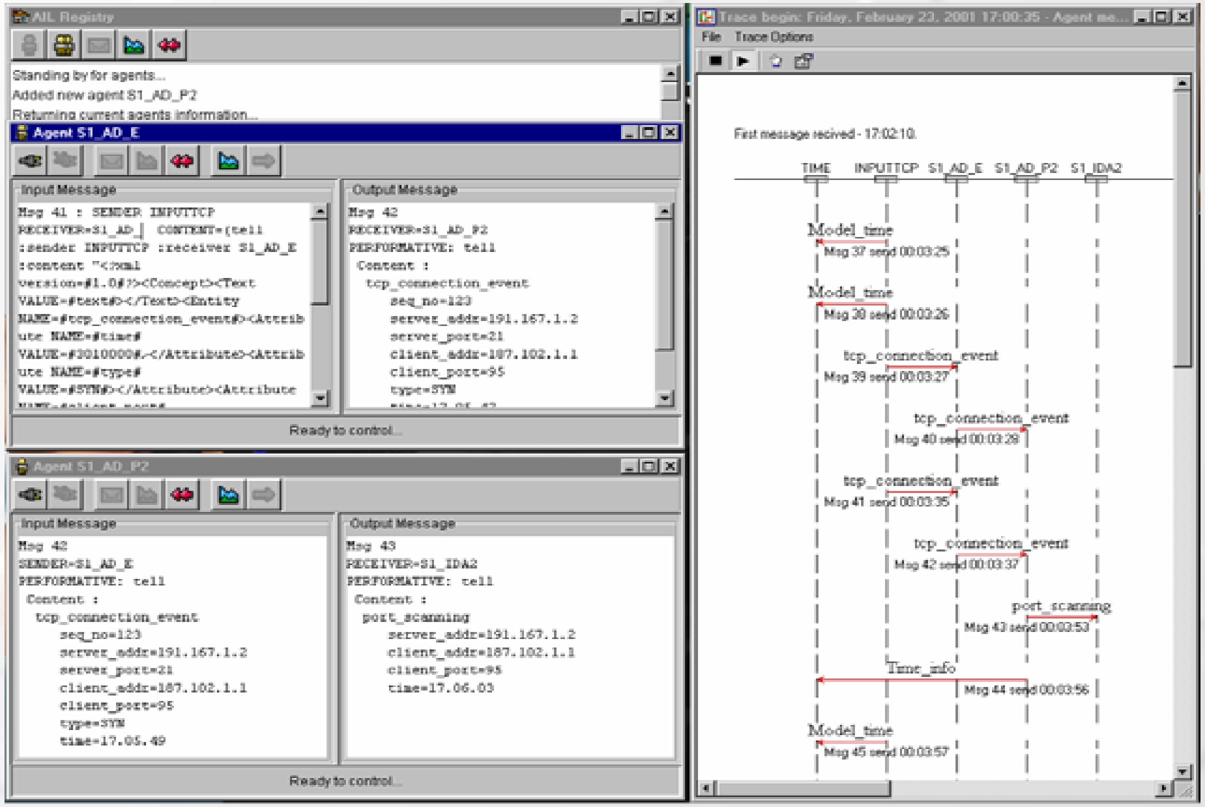
Simulation Tool Implementation: Standard Agent Architecture



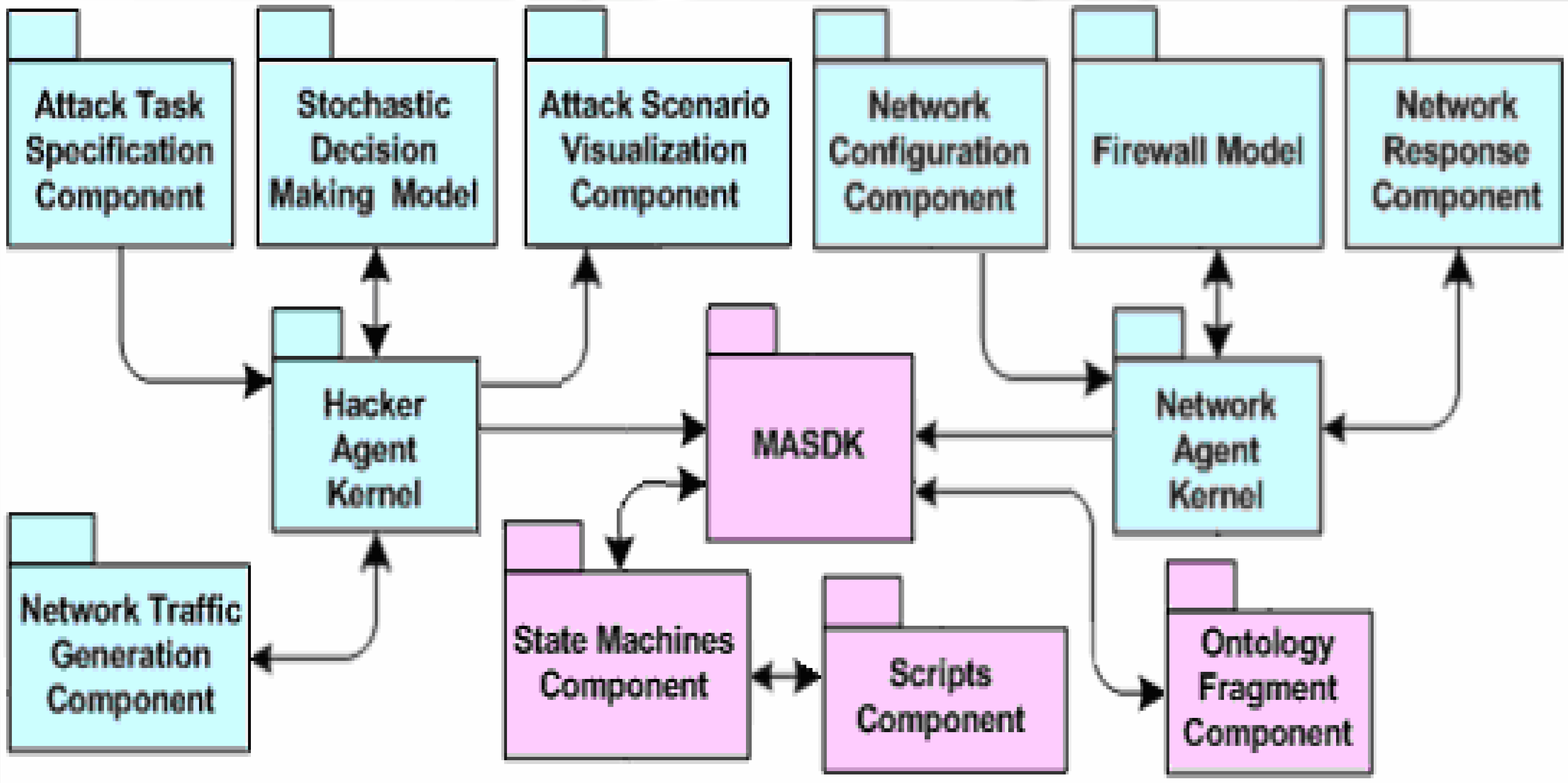
Simulation Tool Implementation: Agents' Communication

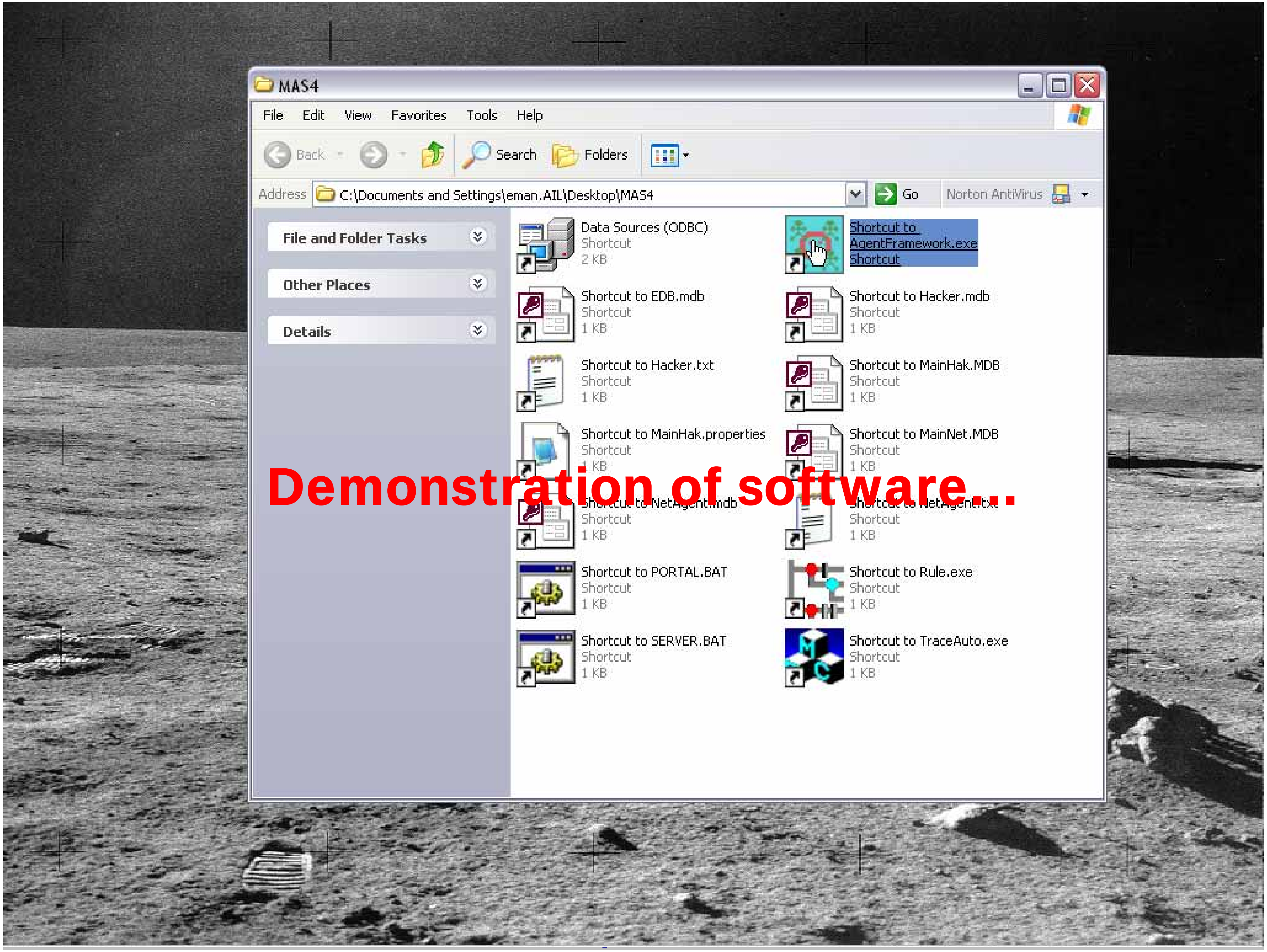
Message templates are specified in KQML language and message content is specified in XML language (RDF, DAML)

Visualization of message exchange

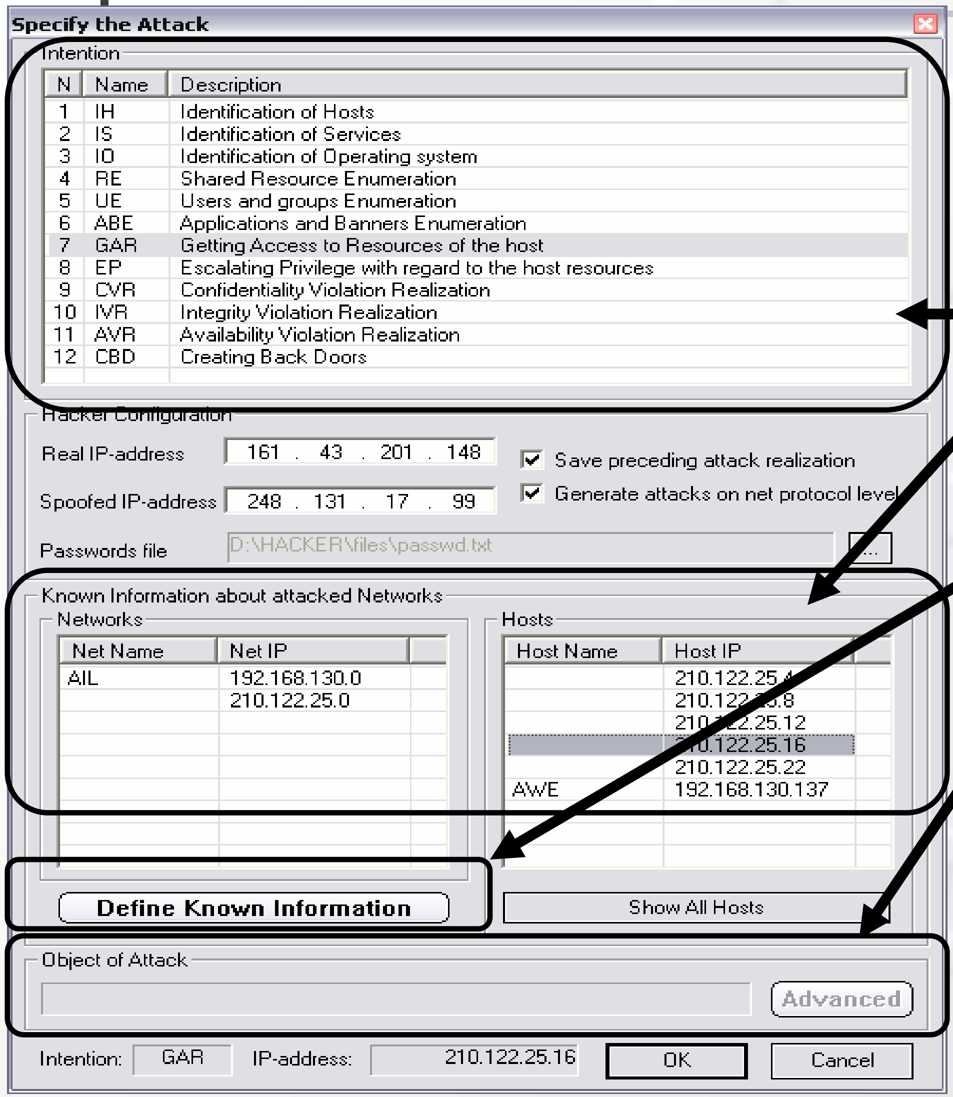


Component Models of Network Agent and Hacker Agent




MAS4
File
Edit
View
Favorites
Tools
Help
Back
Search
Folders
Address
C:\Documents and Settings\eman.AIL\Desktop\MAS4
Go
Norton AntiVirus
File and Folder Tasks
Other Places
Details
Data Sources (ODBC)
Shortcut
2 KB
Shortcut to AgentFramework.exe
Shortcut
Shortcut to EDB.mdb
Shortcut
1 KB
Shortcut to Hacker.mdb
Shortcut
1 KB
Shortcut to Hacker.txt
Shortcut
1 KB
Shortcut to MainHak.MDB
Shortcut
1 KB
Shortcut to MainHak.properties
Shortcut
Shortcut to MainNet.MDB
Shortcut
1 KB
Shortcut
1 KB
Shortcut
1 KB
Shortcut to PORTAL.BAT
Shortcut
1 KB
Shortcut to Rule.exe
Shortcut
1 KB
Shortcut to SERVER.BAT
Shortcut
1 KB
Shortcut to TraceAuto.exe
Shortcut
1 KB
Demonstration of software...

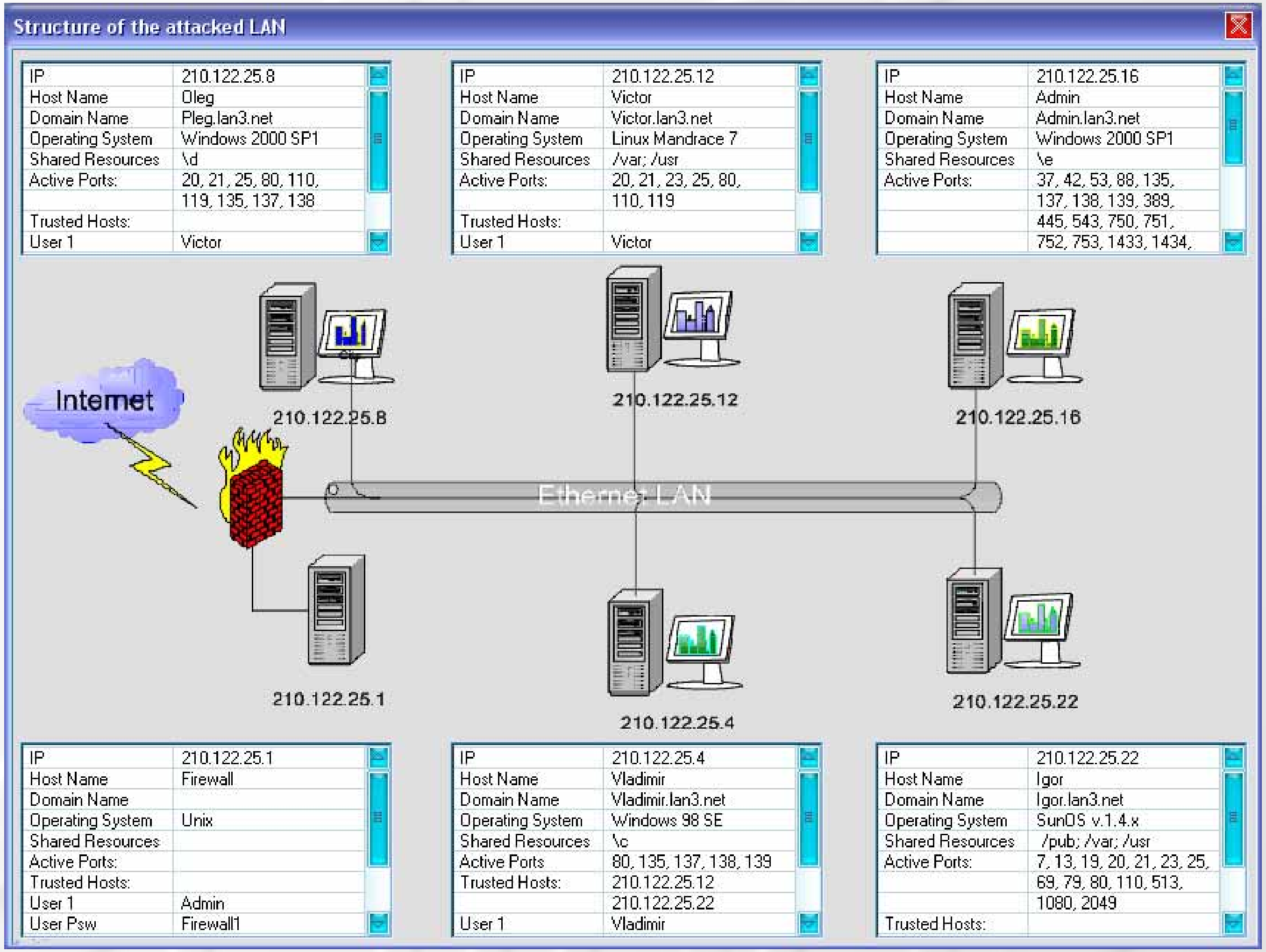
Simulation Tool Implementation: User Interface for Attack Specification



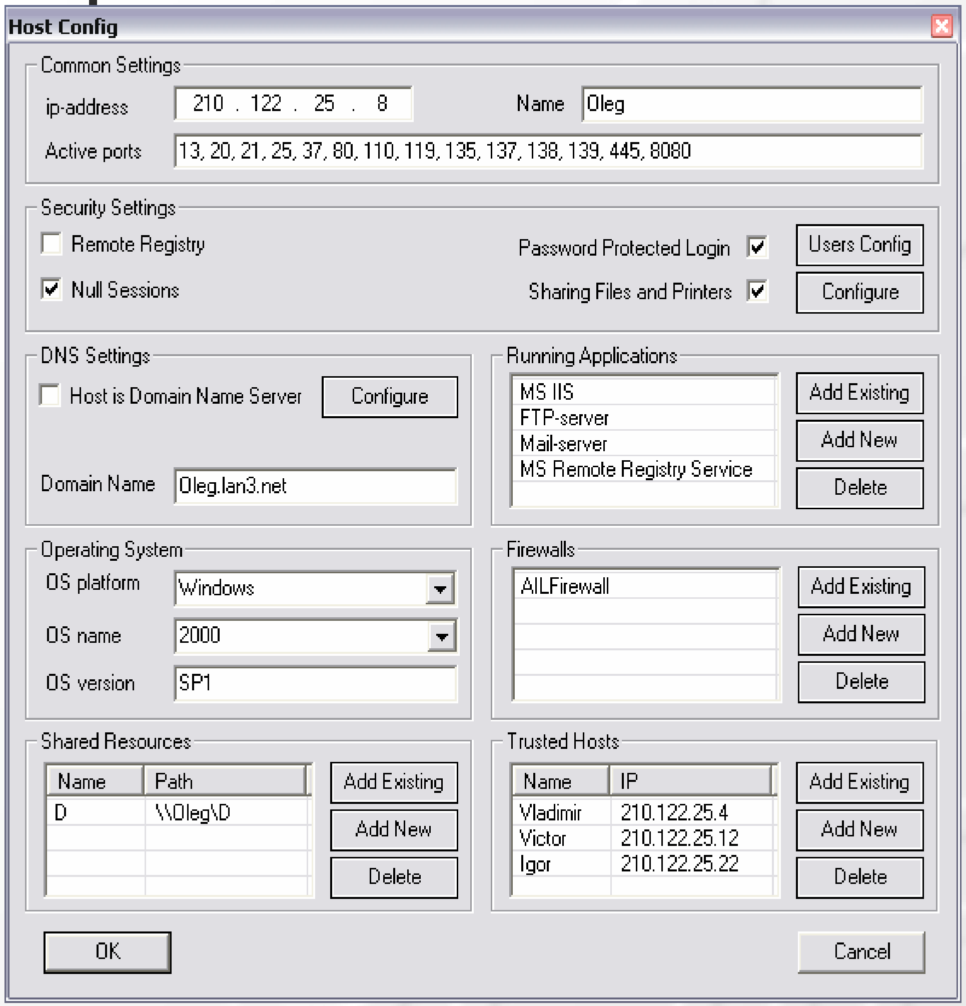
Main elements of attack specification:

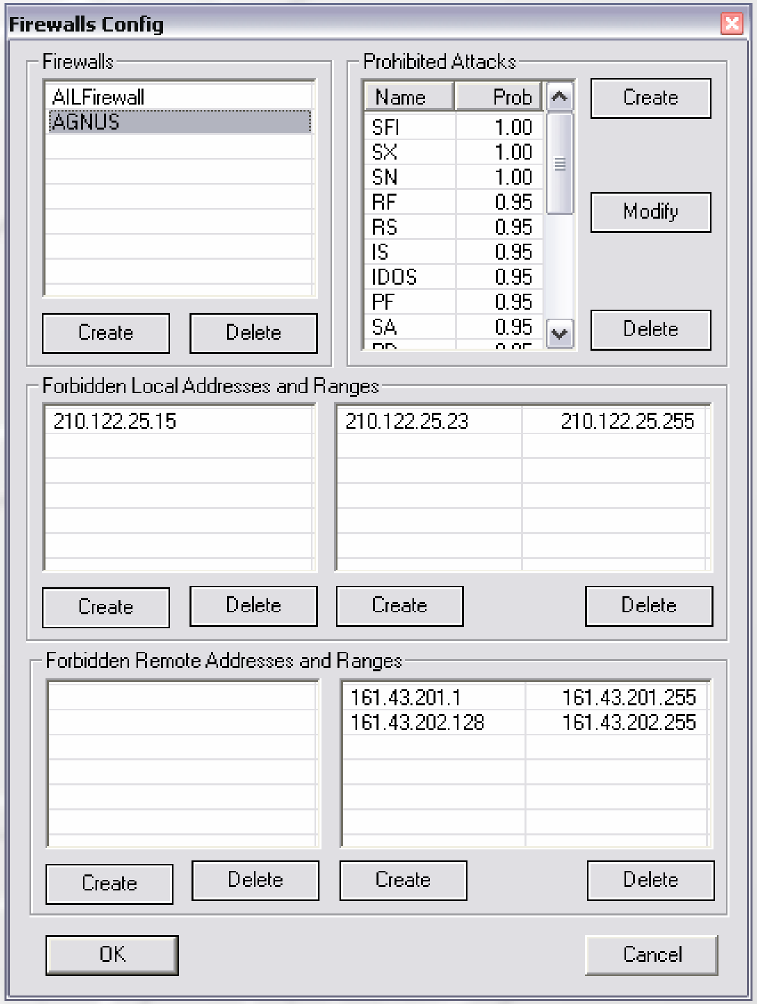
1) Malefactor's intention (1-12);
2) Address of the attacked host or network;
3) Available information about attacked host;
4) Attack object (file name, user account, resource, etc.);

Visualization of the Attacked Network Model



Visualization of the Attacked Network Model





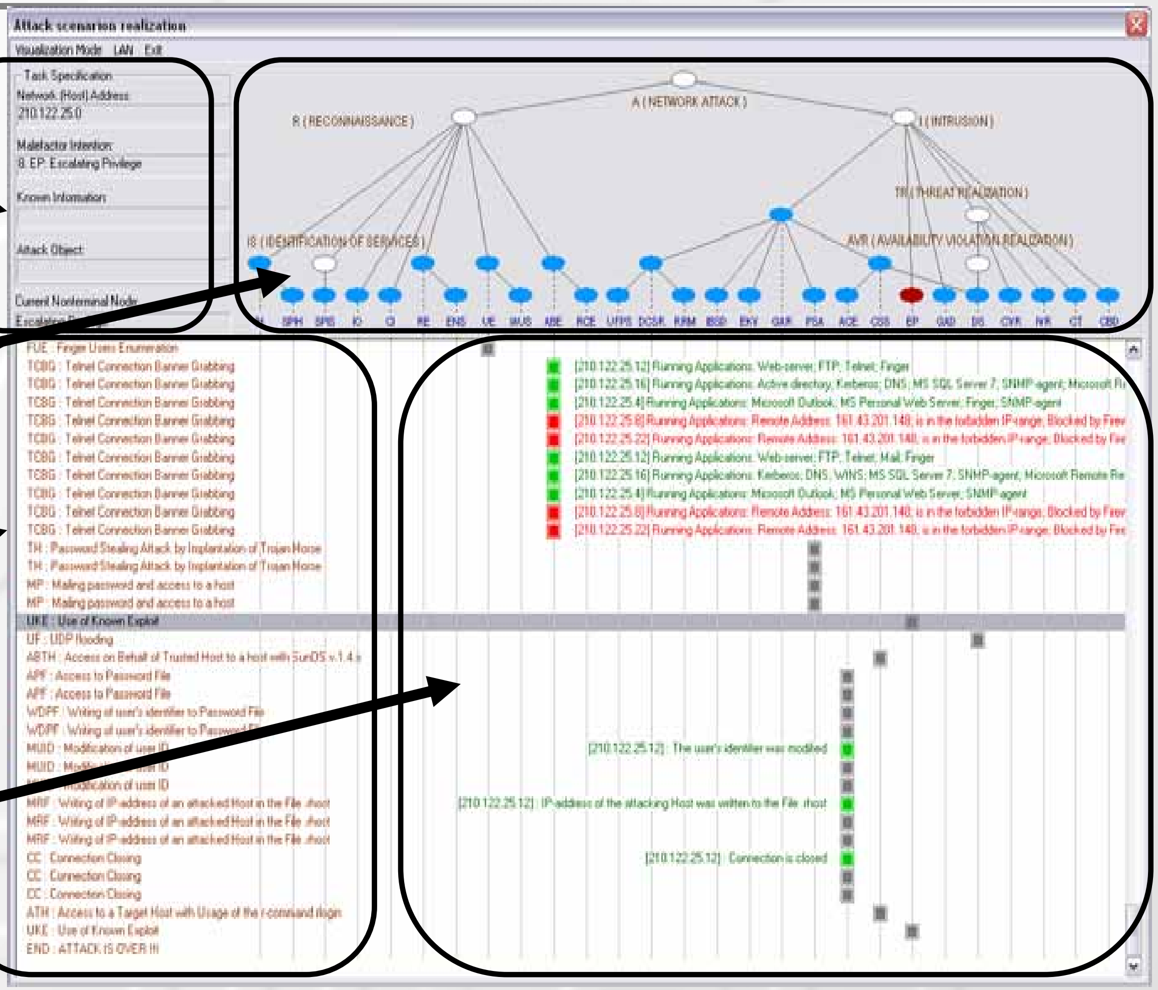
On-line Visualization of an Attack Development on Macro-Level

Attack task specification

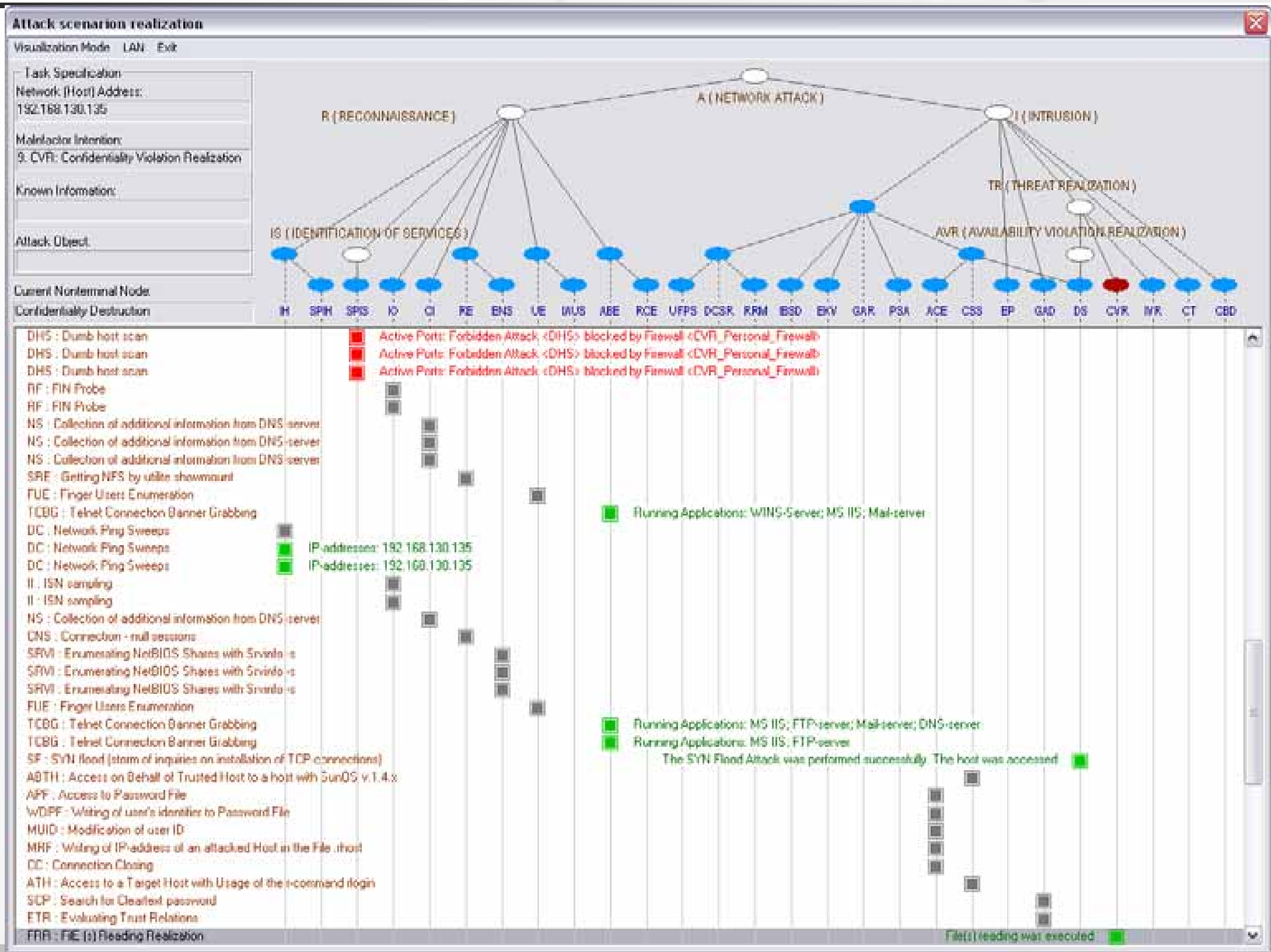
Attack generation tree

Malefactor’ s actions

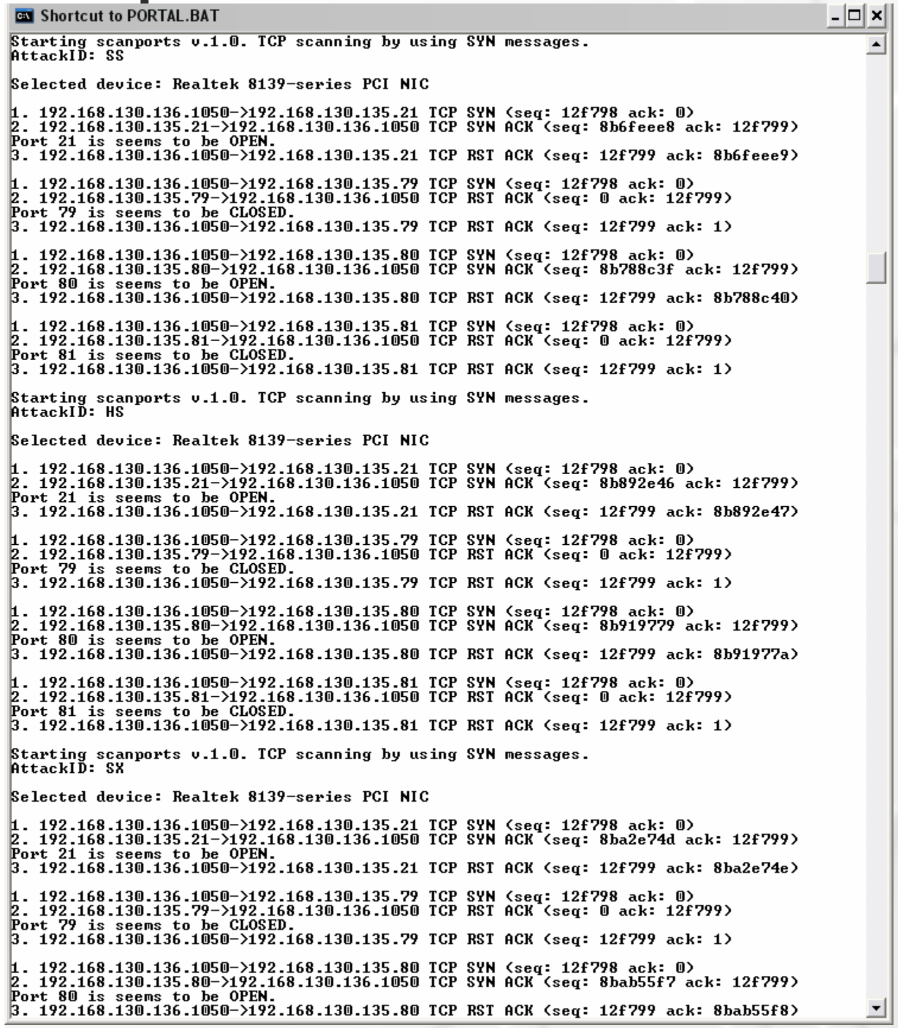
A tag of success (failure) and and data obtained from an attacked host (a host response)

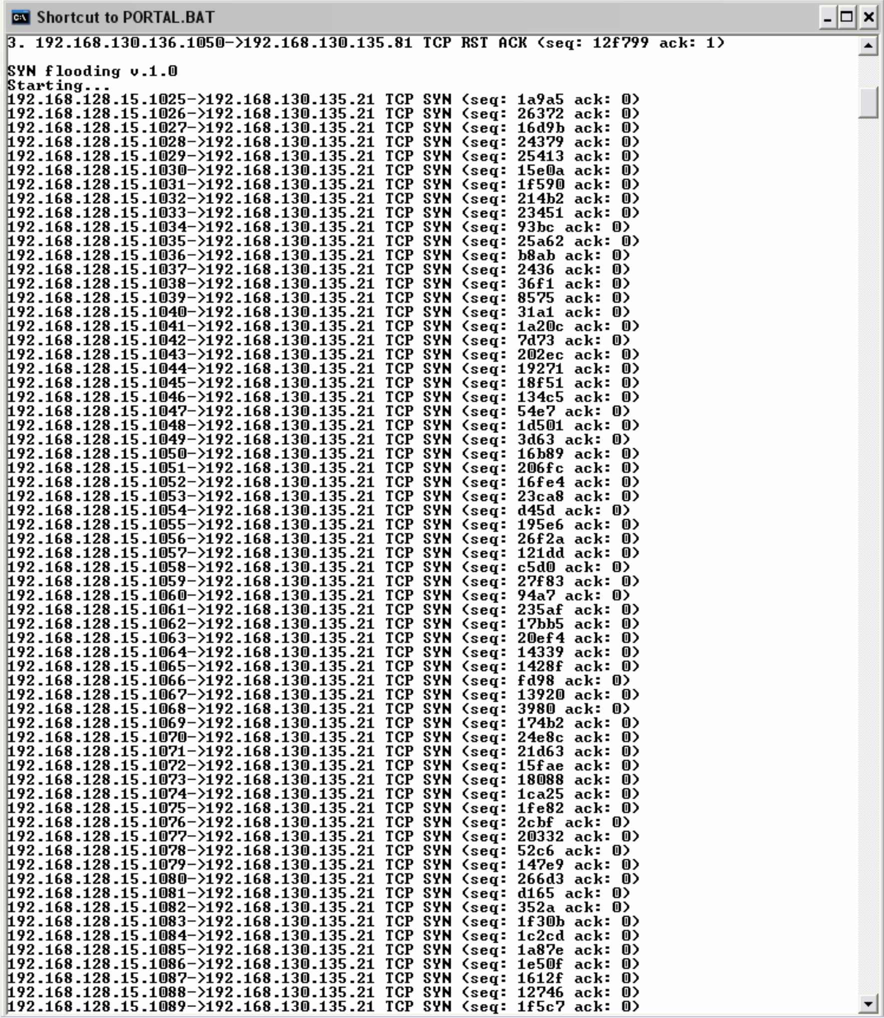


On-line Visualization of an Attack Development on Macro-Level



On-line Visualization of an Attack Development on Micro-Level





Classes of Experiments with Attack Simulator

Therefore all experiments have been divided into two classes:

(1) Experiments on simulation of attacks on macro-level (generation and investigation of malicious actions against computer network model);

(2) Experiments on simulation of attacks on micro-level (generation malicious network traffic against a real computer network).

Input Parameters

For intention "*Reconnaissance*":

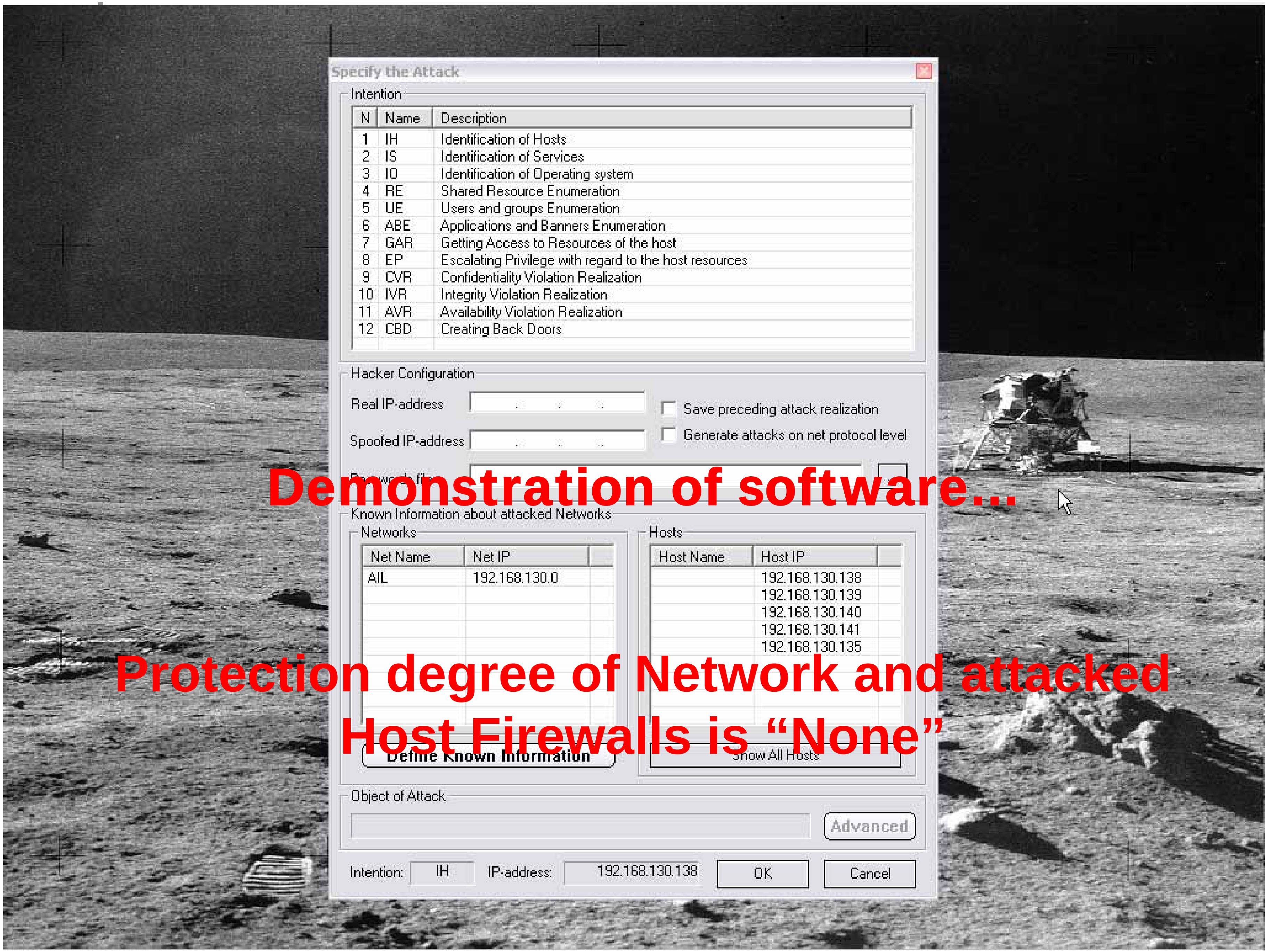
- Configurations of network firewall (**NF**):
  1 – "Strong" (if firewall can protect from 60-90% of implemented attacks);
  2 – "Medium" (if firewall can protect from 20-50% of attacks);
  3 – "None" (if firewall does not protect or is absent).

For intention "*Implantation and threat realization*" :

- protection degree of Network Firewall (**NF**) and attacked Host Firewall (**HF**):
  1 – "Strong" (if firewall can protect from 60-90% of attacks);
  2 – "None" (if firewall does not protect or is absent);
- protection Parameters of attacked Host (**PH**):
  1 – "Strong" (60-90% of security parameters have secure values,
  for example, strong password, absence of sharing files and printers,
  and other resources, absence of trusted hosts, etc.);
  2 – "Weak" (security parameters are weak);
- Hacker's Knowledge about a network (**HK**):
  1 – "Good" (hacker knows about 50-80% of information about network);
  2 – "Nothing" (hacker knows nothing about network).

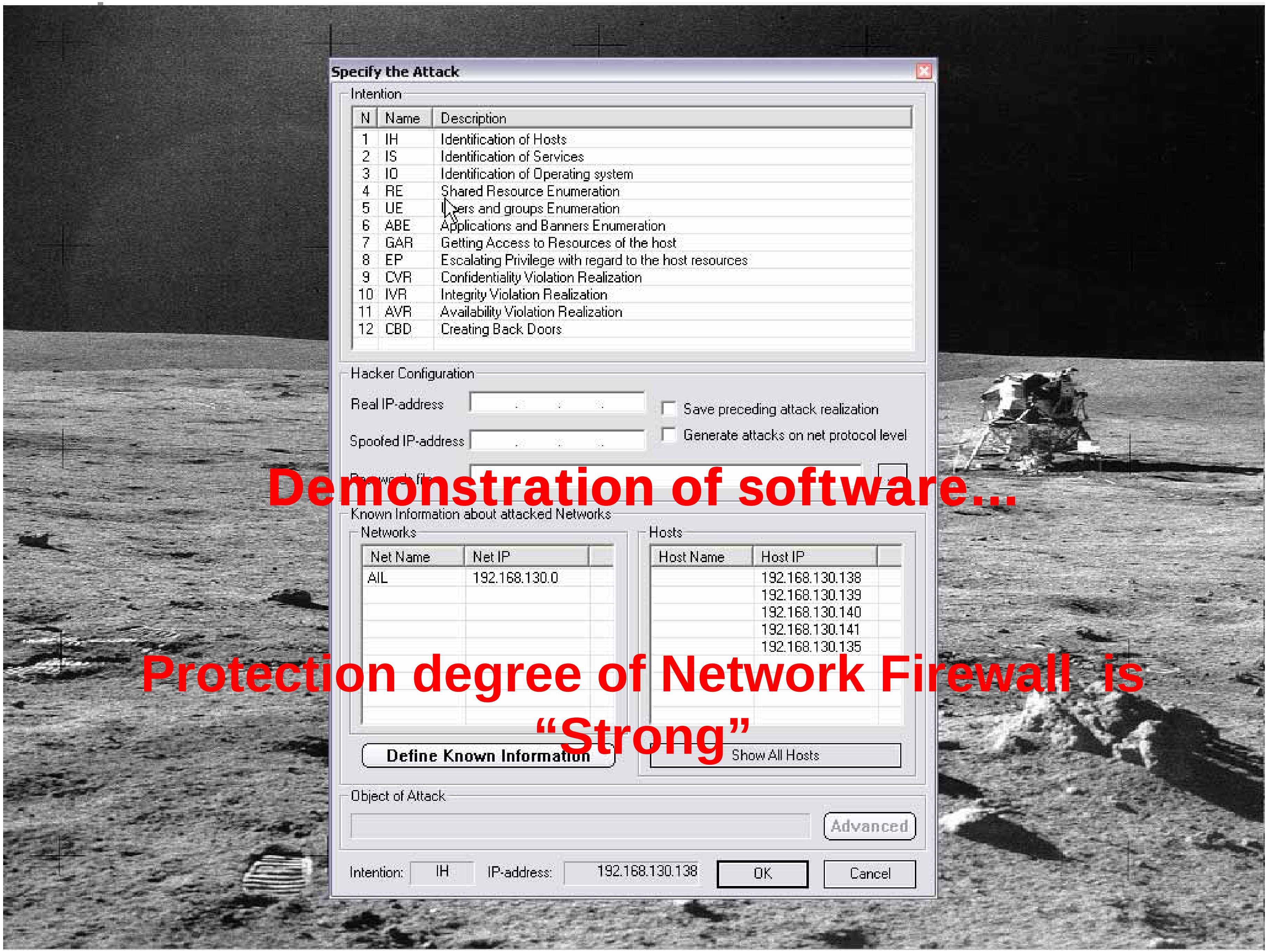
Parameters of attack realization outcome

- **NS (Number of attack Steps) –** number of terminal level attack actions;
- **PIR (Percentage of Intention Realization) –** percentage of the hacker's intentions realized successfully (for "Reconnaissance" it is a percentage of objects about which the information has been received; for "Implantation and threat realization" it is a percentage of successful realizations of the common attack goal on all runs);
- **PAR Percentage of Attack actions Realization –** percentage of "positive" messages (responses) of the Network Agent on attack actions (the "positive" messages are designated in attack visualization window by green lines);
- **PFB (Percentage of Firewall Blockage) –** percentage of attack actions blockage by firewall (red lines in attack visualization window);
- **PRA (Percentage of Reply Absence)** - percentage of "negative" messages (responses) of the Network Agent on attack actions (gray lines in attack visualization window) .

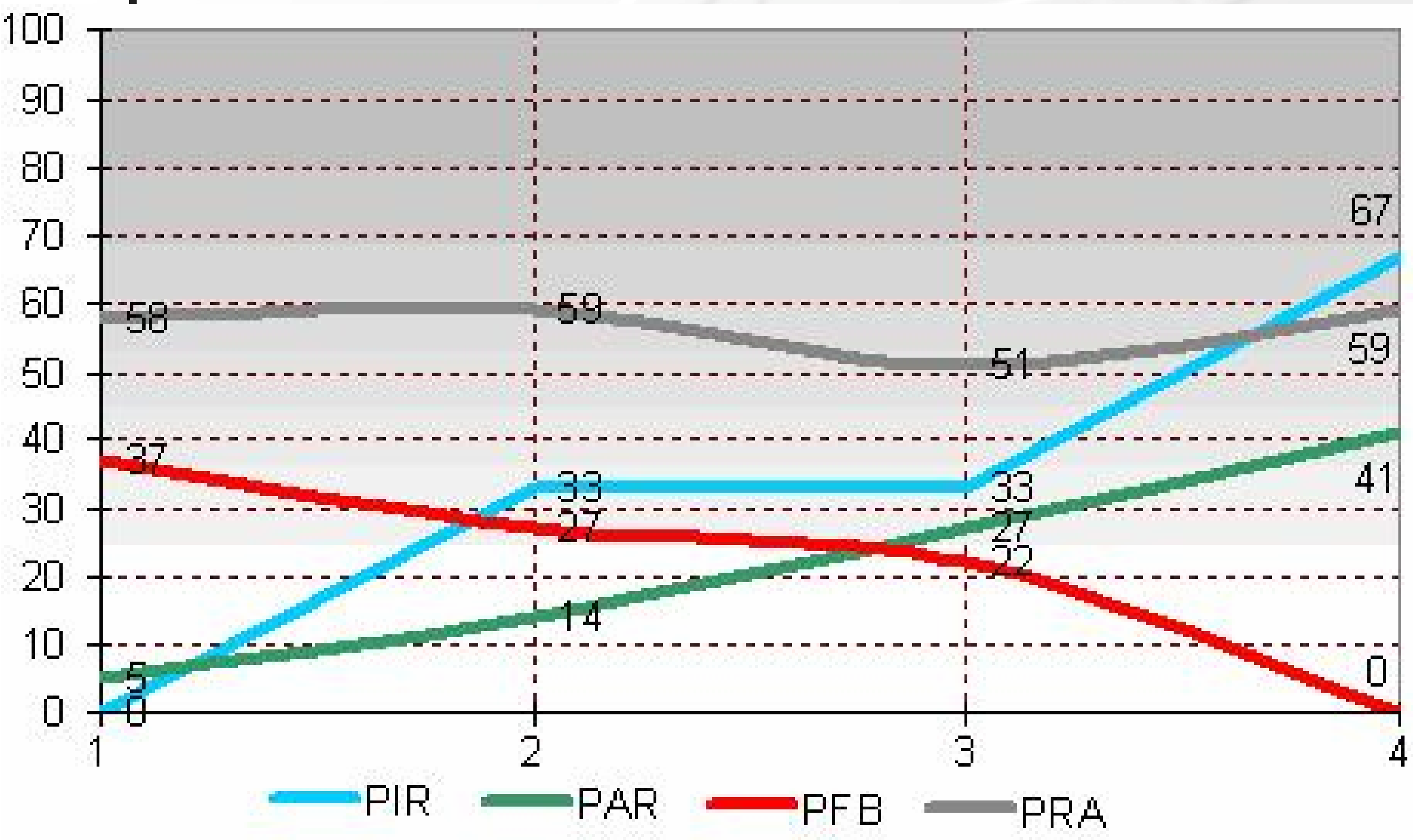


Demonstration of software…

Protection degree of Network and attacked Host Firewalls is “None”


Specify the Attack
Intention
N Name Description
1 IH Identification of Hosts
2 IS Identification of Services
3 IO Identification of Operating system
4 RE Shared Resource Enumeration
5 UE Users and groups Enumeration
6 ABE Applications and Banners Enumeration
7 GAR Getting Access to Resources of the host
8 EP Escalating Privilege with regard to the host resources
9 CVR Confidentiality Violation Realization
10 IVR Integrity Violation Realization
11 AVR Availability Violation Realization
12 CBD Creating Back Doors
Hacker Configuration
Real IP-address
Spoofed IP-address
Save preceding attack realization
Generate attacks on net protocol level
Demonstration of software…
Known Information about attacked Networks
Networks
Net Name Net IP
AIL 192.168.130.0
Hosts
Host Name Host IP
192.168.130.138
192.168.130.139
192.168.130.140
192.168.130.141
192.168.130.135
Protection degree of Network Firewall is “Strong”
Define Known Information
Show All Hosts
Object of Attack
Advanced
Intention: IH
IP-address: 192.168.130.138
OK
Cancel

Results of experiments for intention GAR ("Gaining Access to host Resources")

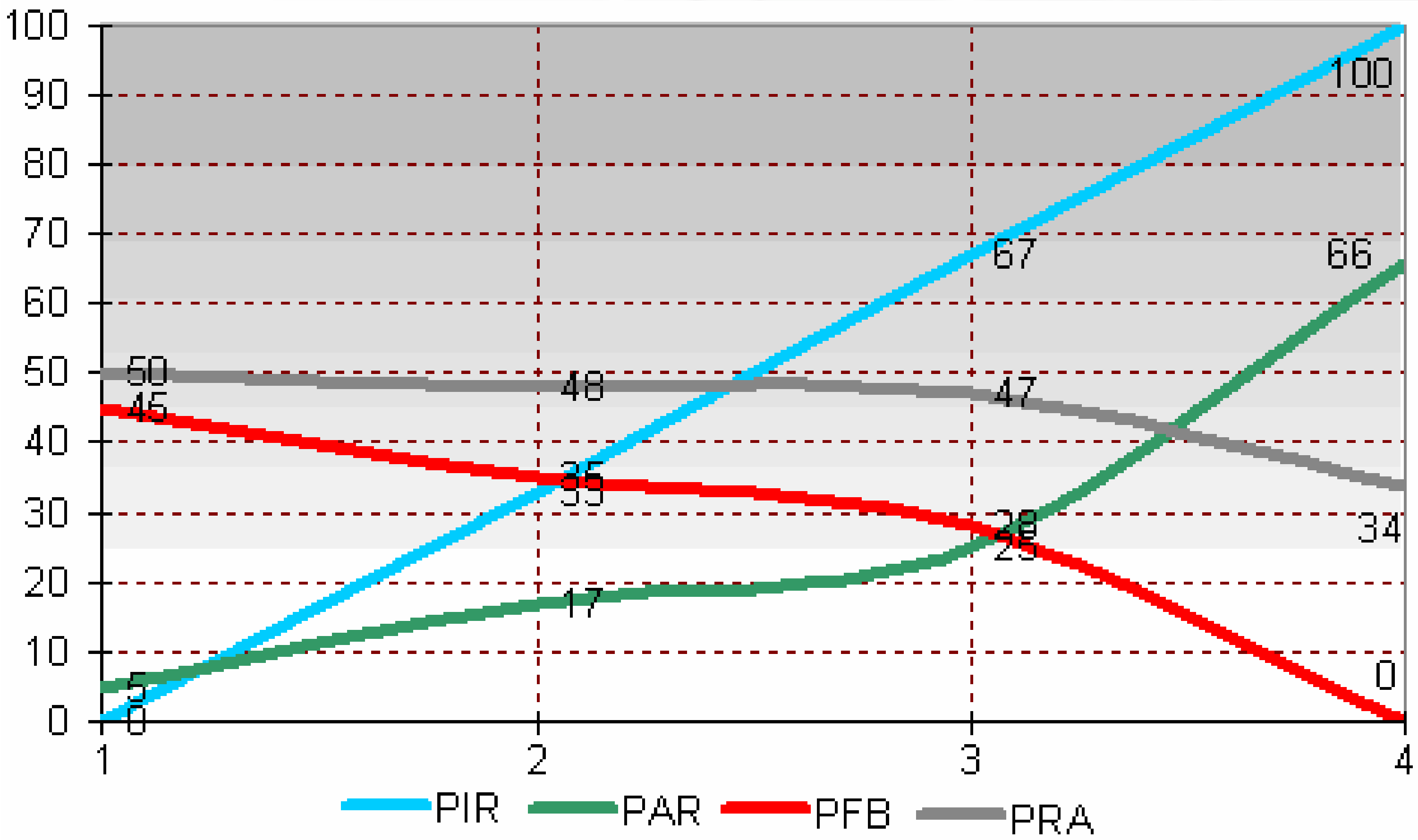


PH=Strong,

HK=Good

Configurations of firewalls: 1 - Both Net & Personal firewalls are active; 2 - Only Net firewall is active; 3 - Only Personal firewall is active; 4 - None of firewalls is active

Results of experiments for intention GAR ("Gaining Access to host Resources")



PH=Weak,

HK=Good

Configurations of firewalls: 1 - Both Net & Personal firewalls are active; 2 - Only Net firewall is active; 3 - Only Personal firewall is active; 4 - None of firewalls is active

Outline

- Introduction
- Works describing attacks and attack taxonomies
- Works directly coupled with attack modeling and simulation
- Works devoted to descriptions of attack specification languages
- Works on evaluating security systems
- Formal grammar and state machines based approach
- **Agent based and packet level simulation approach**
- Conclusion

Basic Assumptions

- Cyberwarfare is represented as a large collection of semi-autonomous interacting agents.
- The aggregate system behavior emerges from evolving local interactions of agents in a dynamically changing environment specified by computer network model.
- We assume to select two agents' subsystems (teams):

(1) Adversary attacking system - a team of malefactor's agents (for automatic generation of distributed coordinated attacks);

(2) Security (defense) system - a team of security agents (for intrusion protection, data sensing and information fusion, intrusion detection, adversary intentions and actions prediction, and incident response).

- Agents of different teams compete to reach opposite intentions. Agents of the same team cooperate to achieve common intention.

Teamwork Approaches and Procedures for Teamwork Support

- **The agents' team realizes teamwork**, if the team members fulfill joint operations for reaching the common long-time goal in a dynamic external environment at presence of noise and counteraction of opponents.
- The teamwork is something greater, than simply coordinated set of personal actions of individual agents. It is accepted to speak, that in teamwork the agents collaborate.
- The collaboration is a special sort of a coordinated activity of the agents, in which they jointly solve some task or fulfill some activity for reaching a common goal.

Teamwork Approaches and Procedures for Teamwork Support

- The general intentions of agents are determined in a hierarchical reactive plan.
- This plan describes actions of the team as well as the actions of particular agents.
- The coordinated tasks are carried out due to installation of constraints on agents' roles.

Basic procedures for teamwork support [Tambe, 97] :

- maintenance of actions coordination;
- monitoring and restoration of agents' functionality;
- maintenance of communication selectivity.

Related Works on Teamwork Approaches (1)

Main Agents' Teamwork Approaches:

- The **Joint intention theory** [Cohen et al., 91];
- The **Shared Plans theory** [Grosz et al., 96];
- **Combined approaches** ([Jennings,95], [Tambe,97], [Tambe et al.,01], etc.).

Important teamwork frameworks and systems:

GRATE* [Jennings,95] is an implementation of teamwork using the Joint Responsibility model. This model includes concepts of common goals and instructions (recipes). The individual commitments determine how an agent should operate in a context of teamwork.

OAA (Open Agent Architecture) [Martin, et all., 99] uses a blackboard-based framework that allows individual agents to communicate by means of goals posted on blackboards controlled by facilitator agents.

Related Works on Teamwork Approaches (2)

Important teamwork frameworks and systems:

CAST (Collaborative Agents for Simulating Teamwork) [Yen, et all., 01] supports teamwork using a shared mental model. The mental model includes team processes, team structures and the capability of each teammate.

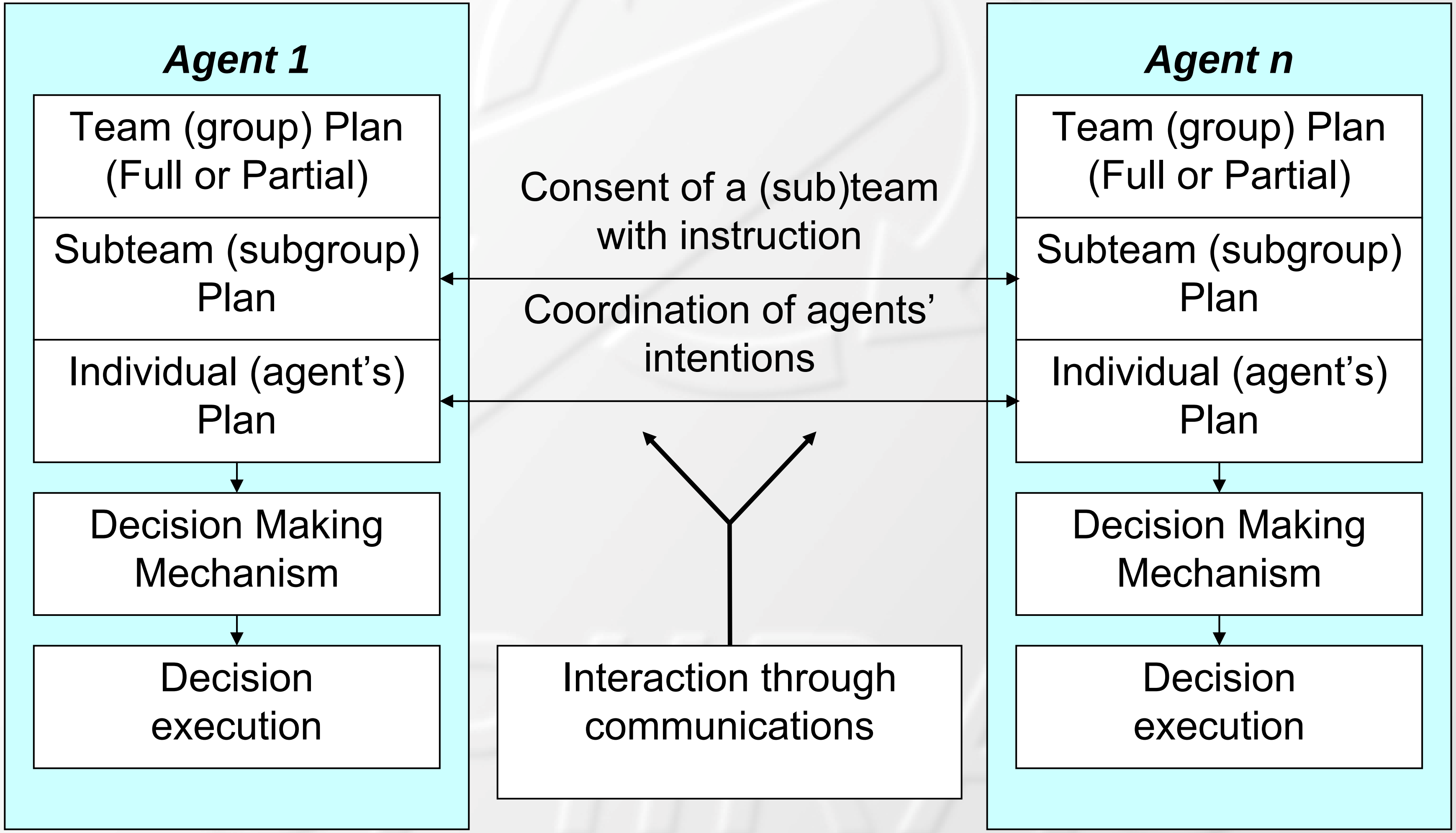
In **RETSINA-MAS** [Giampapa, Sycara, 02], agents have own copy of a common partial plan. Each agent estimates its opportunities to the requirements of the team goal.

In "**Robocup Soccer**" [Stone, Veloso, 99], agents have common knowledge operating their cooperative behavior.

COGNET/BATON [Zachary, Mentec, 00] is a system for simulation of teamwork of people with use of intelligent agents.

Team-Soar [Kang, 01] is a model implemented for testing a theory of team decision making.

Common Agents' Teamwork Scheme



Technology for Creation of Agents' Team

Main stages of creation of agents' team

(1) formation of the **subject domain ontology**;

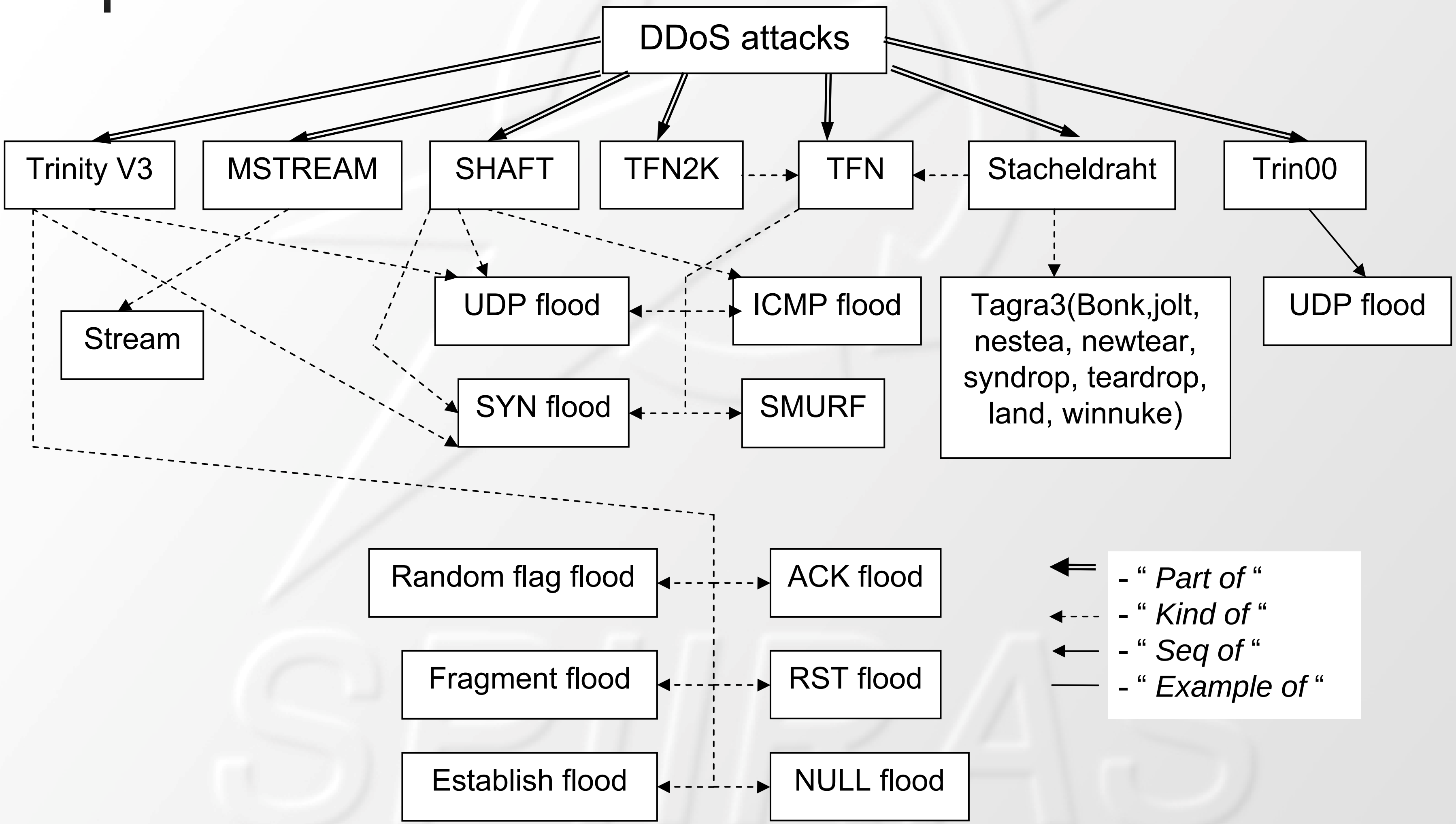
(2) determination of the **agents' team structure and mechanisms of their interaction and coordination** (including roles and scenarios of an agents' roles exchange);

(3) specifications of the **agents' actions plans** (generation of attacks) as a hierarchy of attribute stochastic formal grammars;

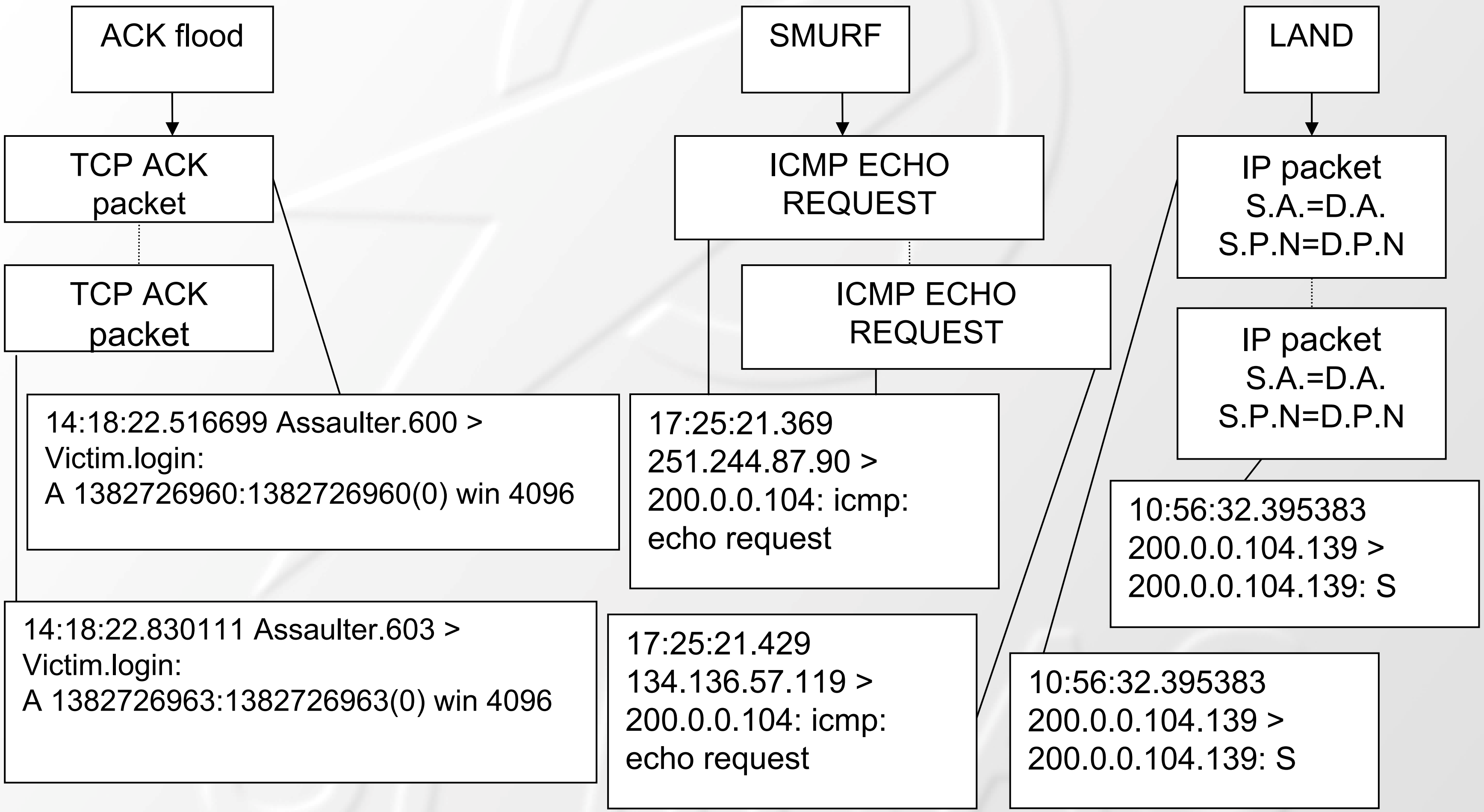
(4) assignment of roles and allocation of plans between the agents;

(5) state-machine based **interpretation** of the teamwork.

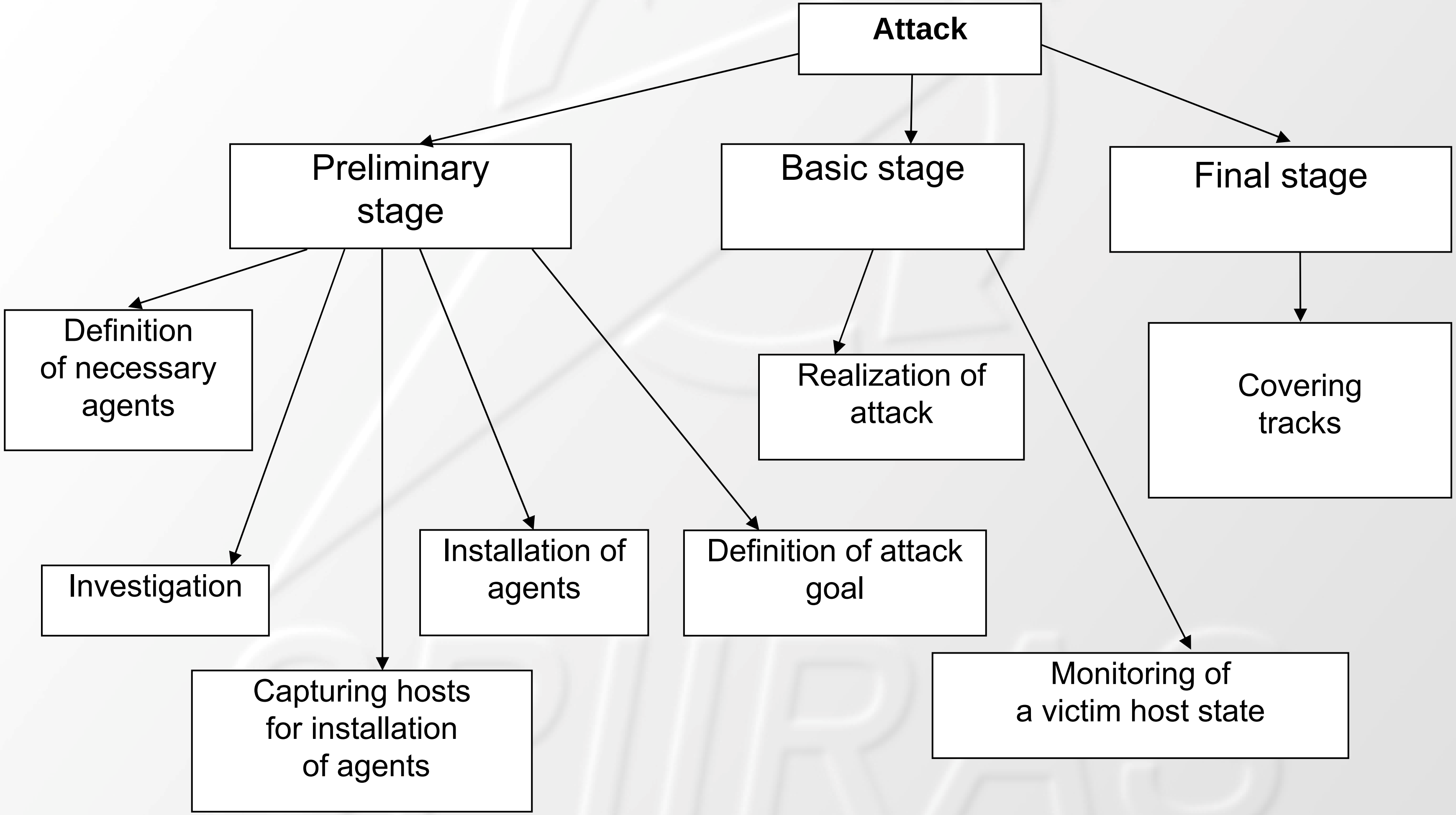
Ontology of DDoS Attacks: Fragment of Ontology at Macro-Level



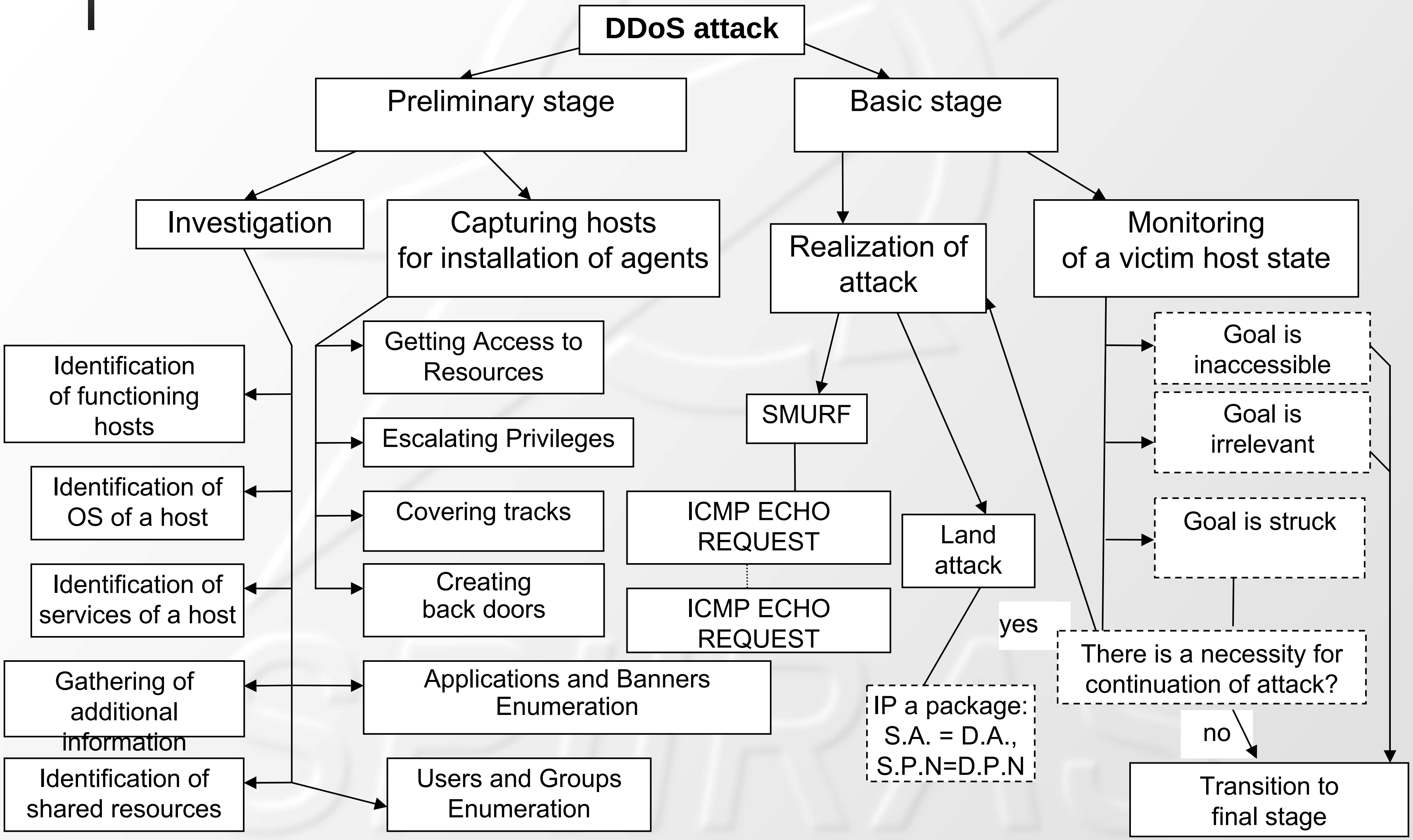
Ontology of DDoS Attacks: Fragment of Ontology at Micro-Level



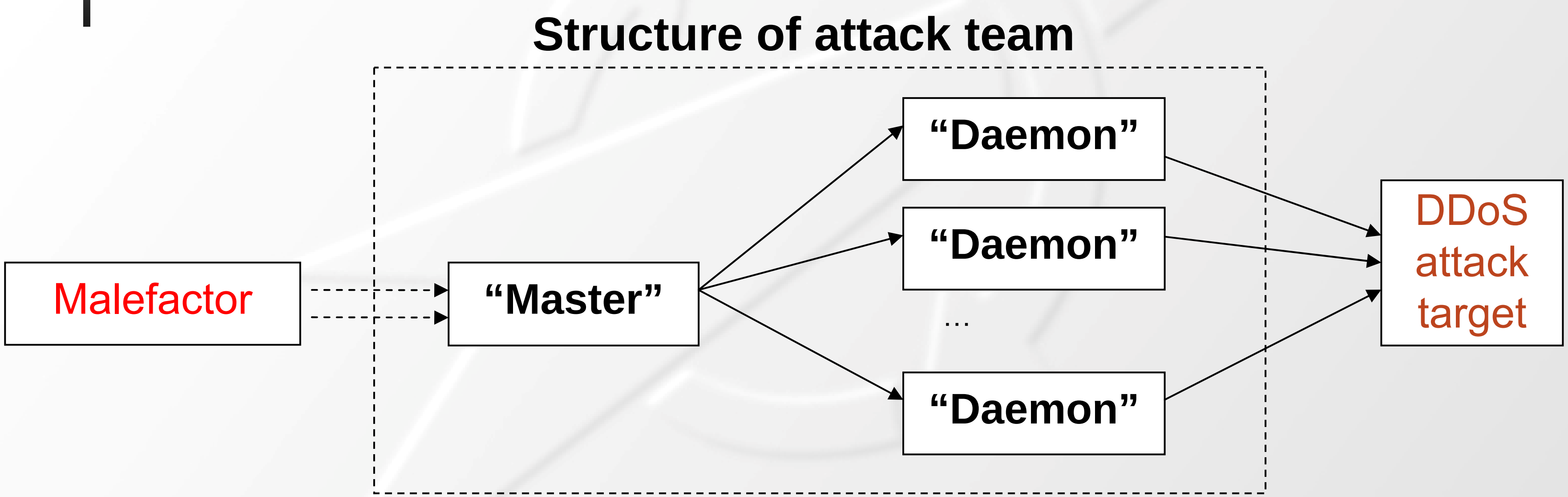
Upper Level of Hierarchy of Agent Plans for DDoS Attacks



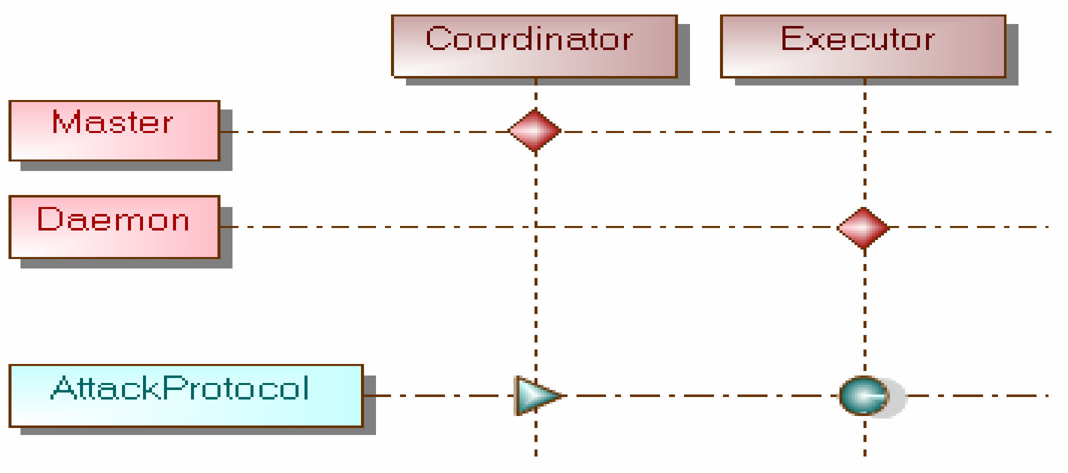
Fragment of Upper and Middle Level of Hierarchy of Agent Plans for DDoS Attacks



Structure and model of attack team

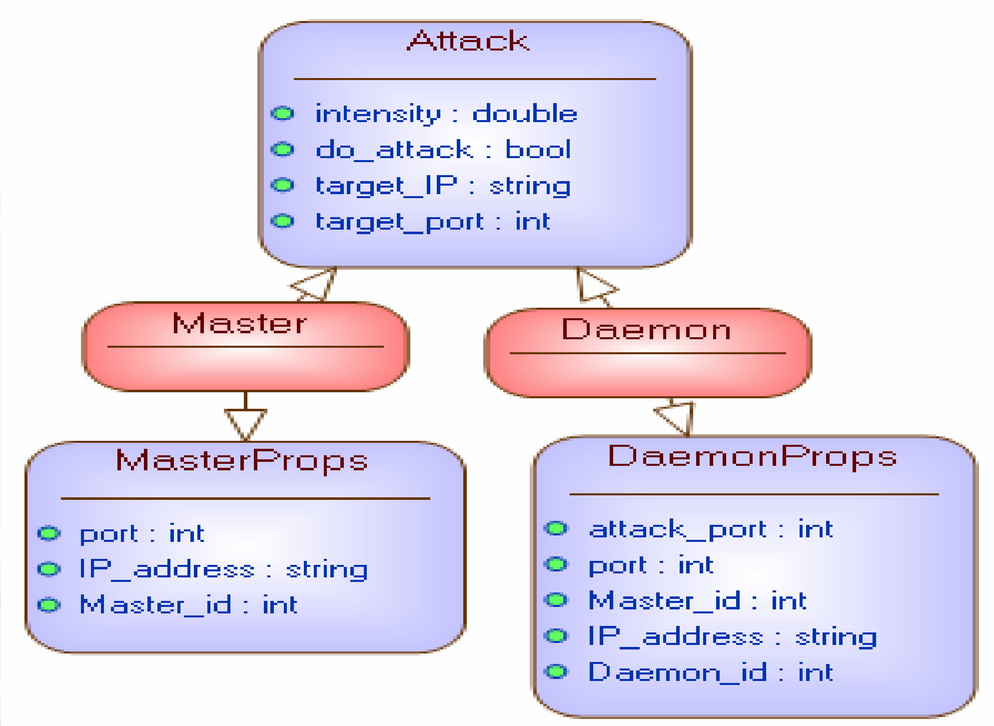


Meta-model of attack team (screenshot of MASDK meta-model editor)



Fragment of ontology

The low-level fragment of attack ontology (the screenshot of MASDK ontology editor)

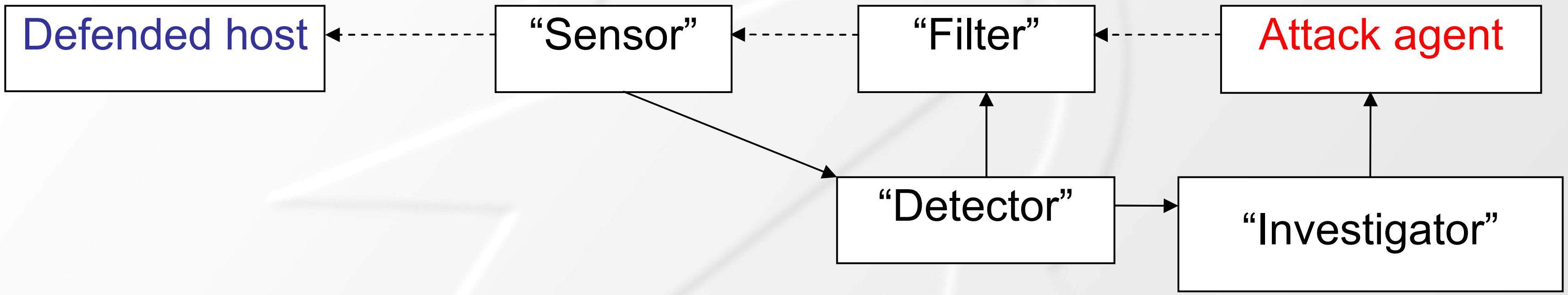


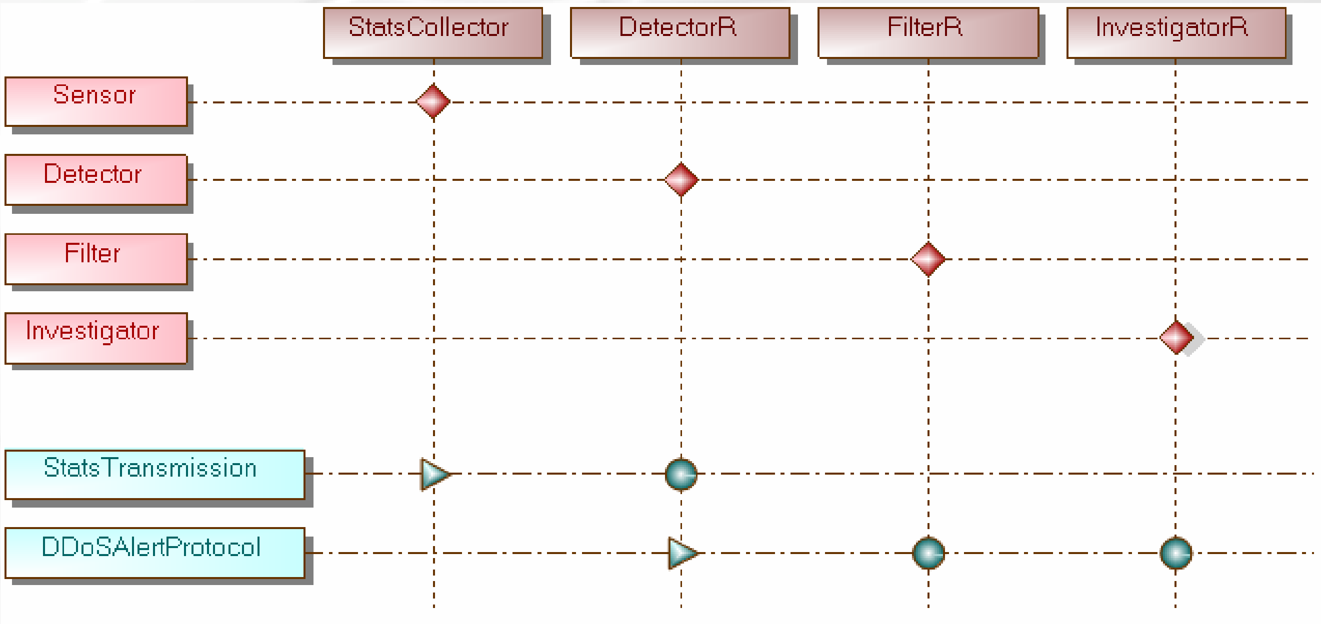
Format of the message from the masters to daemon

Start the attack: yes/no	IP address of attack target	Port of attack target	Intensity of attack (in packets per second)

Structure and model of defense team

Structure of defense team

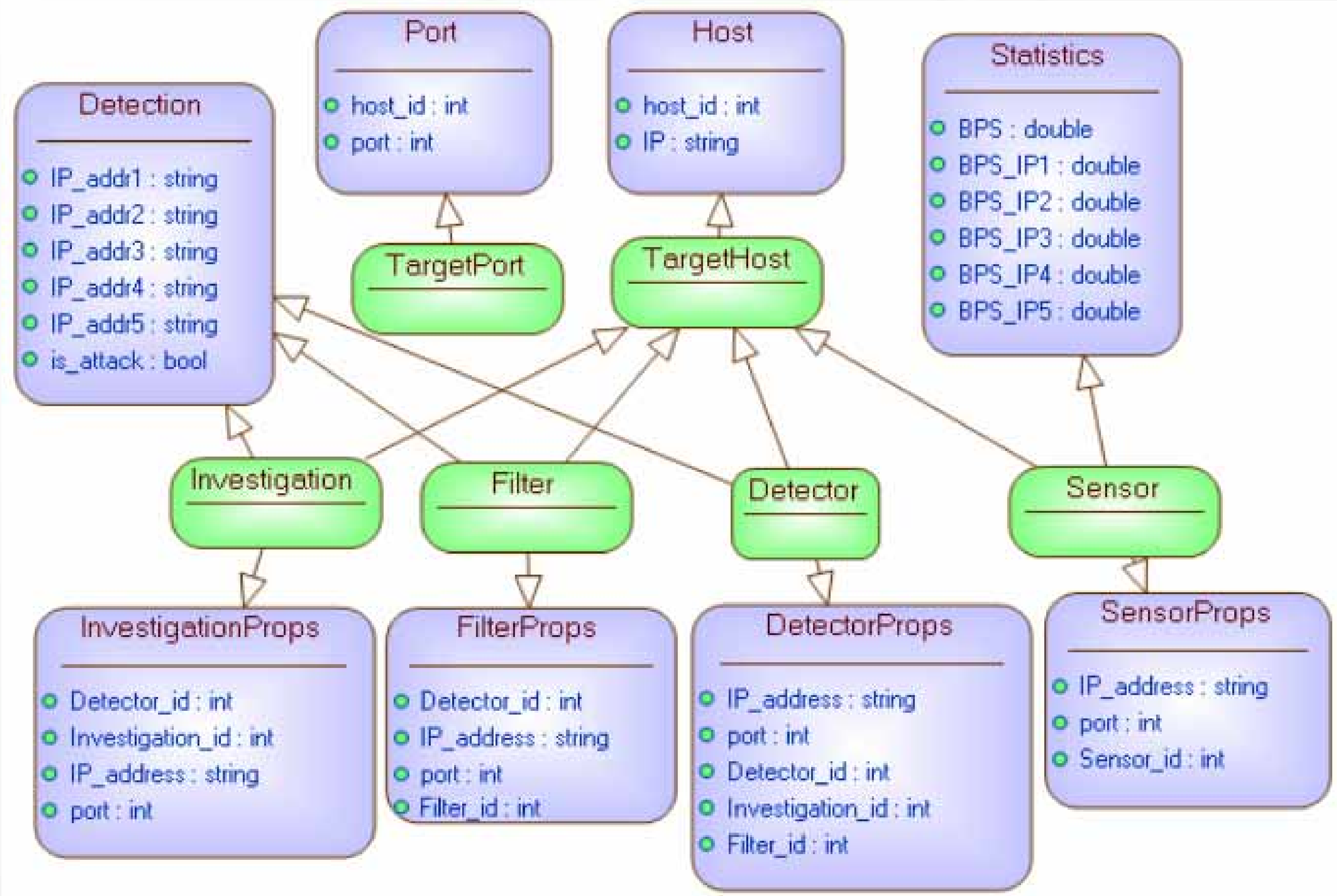




Meta-model of defense team (screenshot of MASDK meta-model editor)

Fragment of ontology

The low-level fragment of attack ontology (the screenshot of MASDK ontology editor)



Main Classes of Attack and Defense Parameters. Parameters of Defense Efficiency

Attack module

- *Victim type*
- *Attack type*
- *Impact on the victim*
- *Attack rate dynamics*
- *Persistent of agent set*
- *Possibility of exposure*
- *Source address validity*
- *Degree of automation*

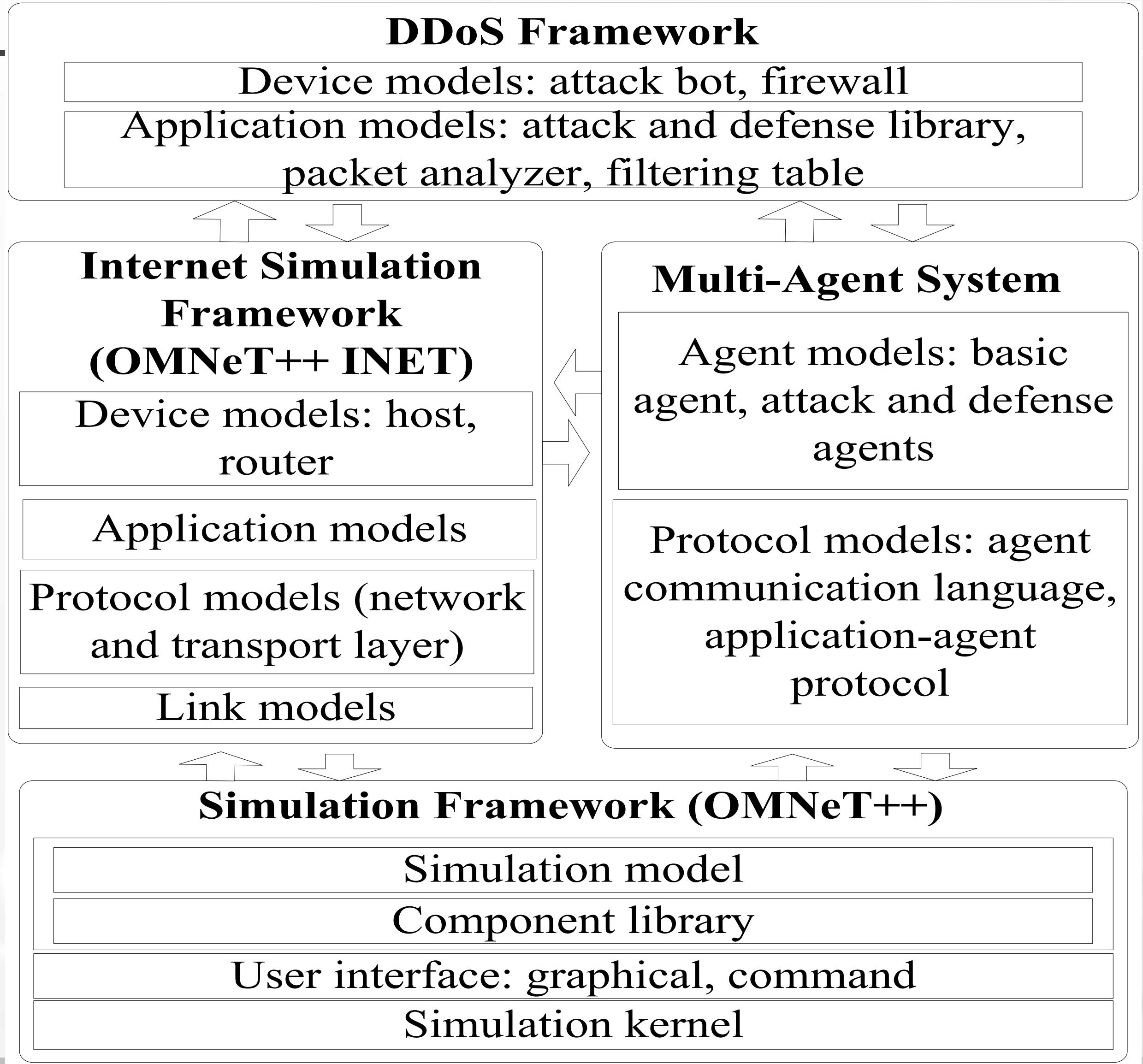
Defense module

- *Deployment location*
- *Mechanism of cooperation*
- *Covered defense stages*
- *Attack detection technique*
- *Attack source detection technique*
- *Attack prevention/counteraction technique*
- *Model data gathering technique*
- *Determination of deviation from model data*

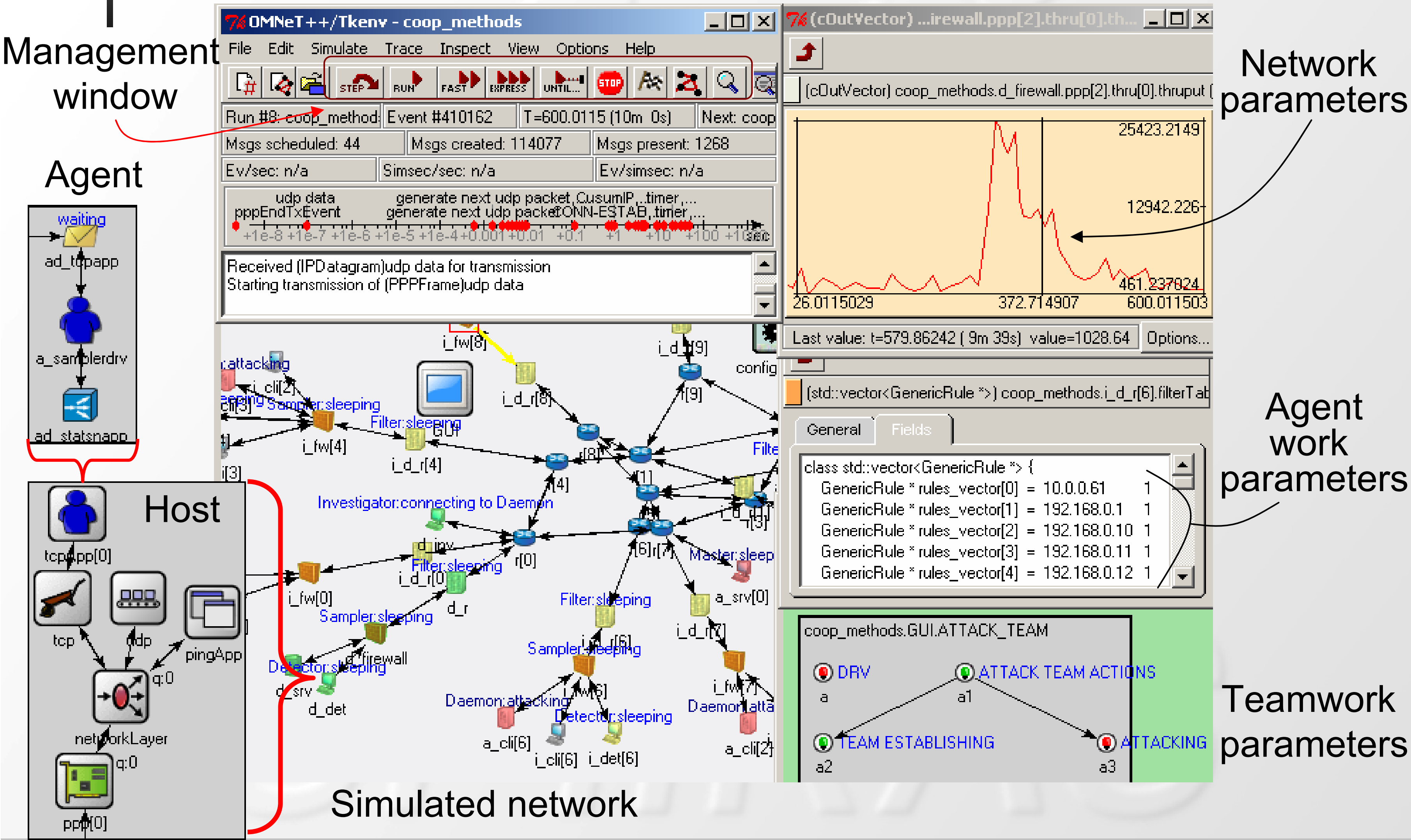
Efficiency Parameters:

- List of detectable attacks
- Volume of the input traffic before and after filters
- Percent of the normal traffic and the attack traffic on entrance to attacked network
- Rate of dropped legitimate traffic (false positive rate)
- Rate of admitted attack traffic (false positive rate)
- Attack detection and attack reaction times
- Computational complexity
- etc.

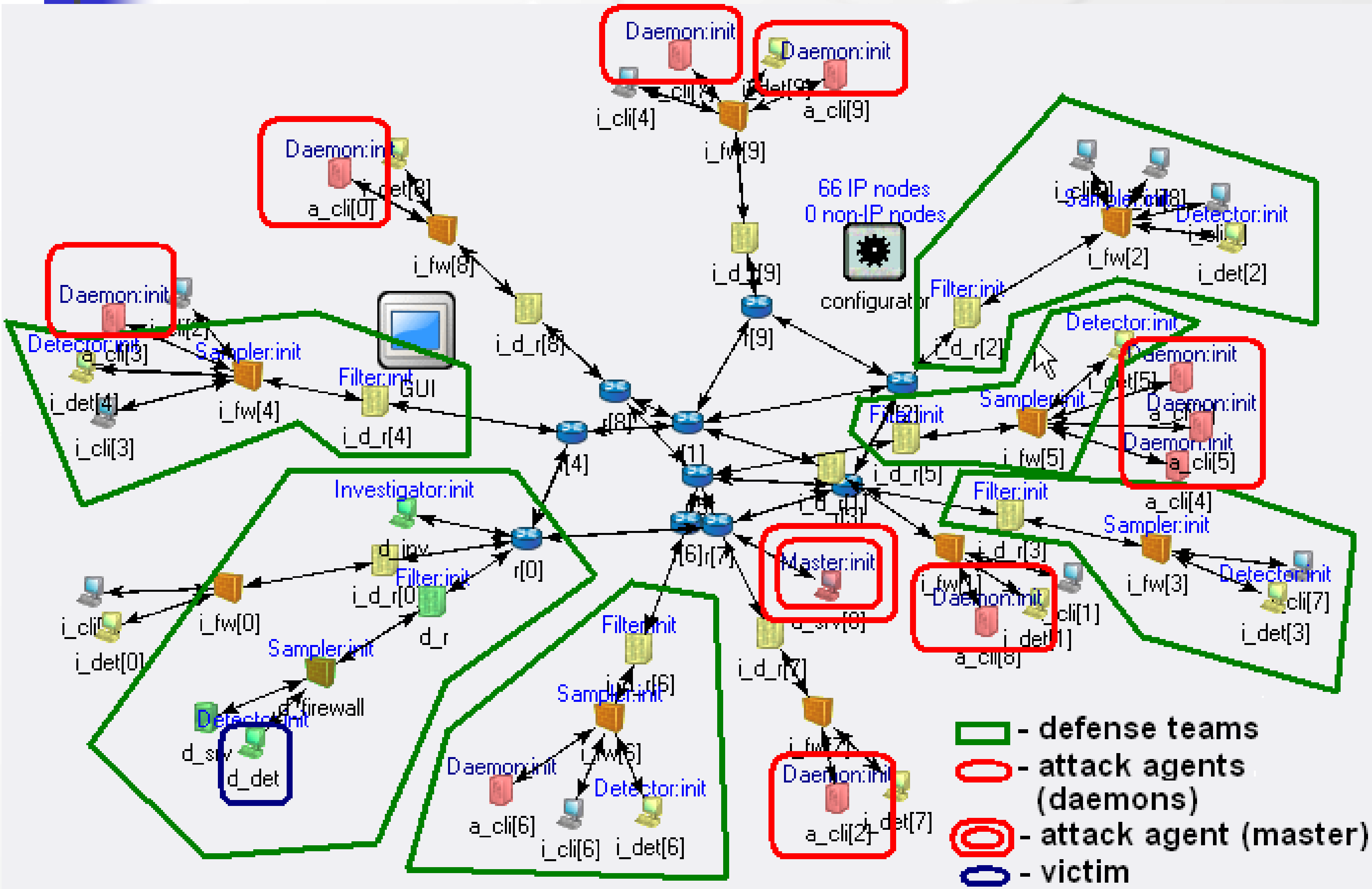
Architecture of Simulation Environment



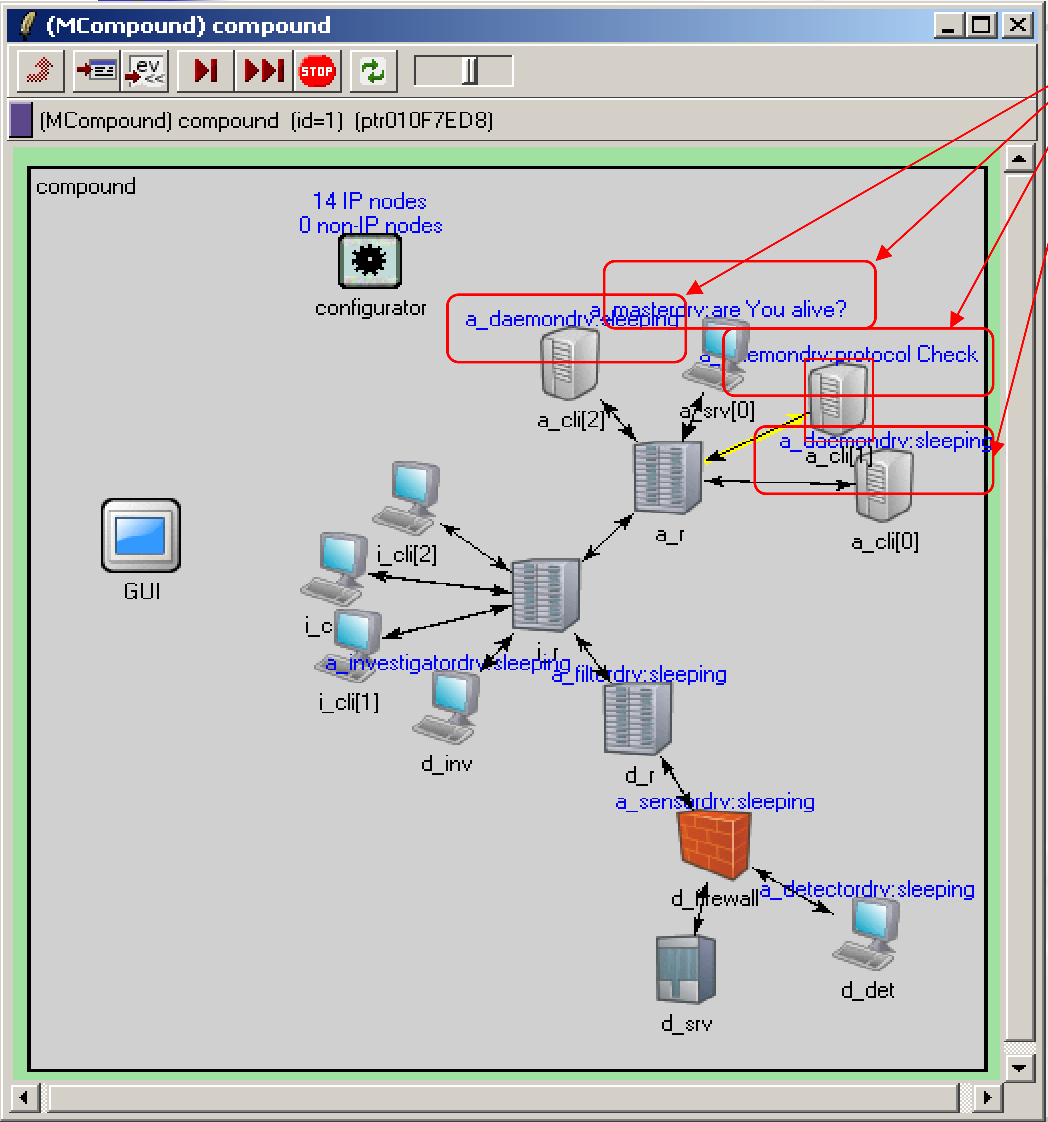
User Interface of Simulation Environment



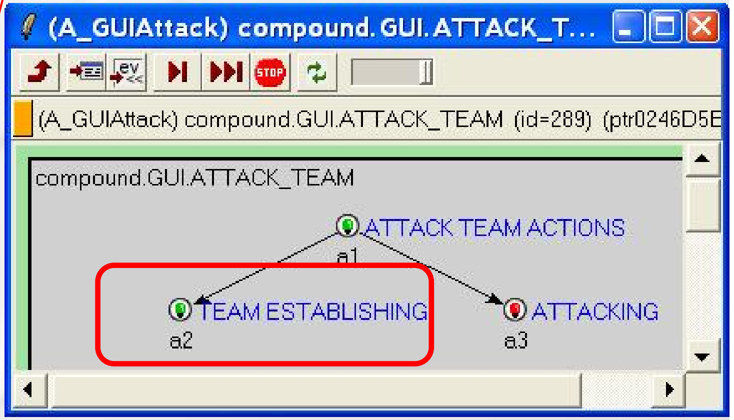
Configuration of the Internet fragment and agent teams



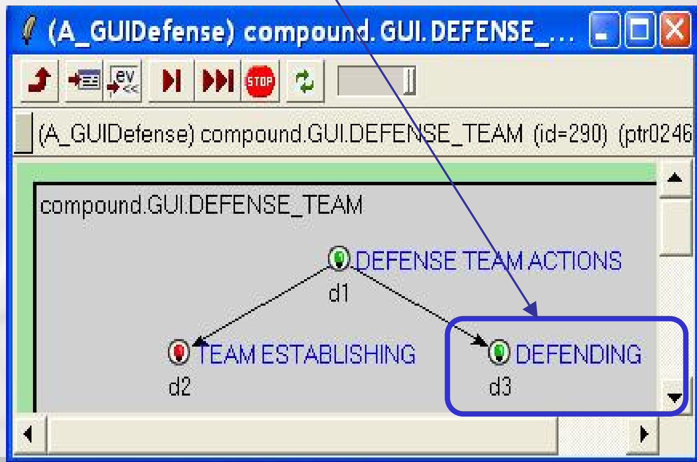
Formation of attack team



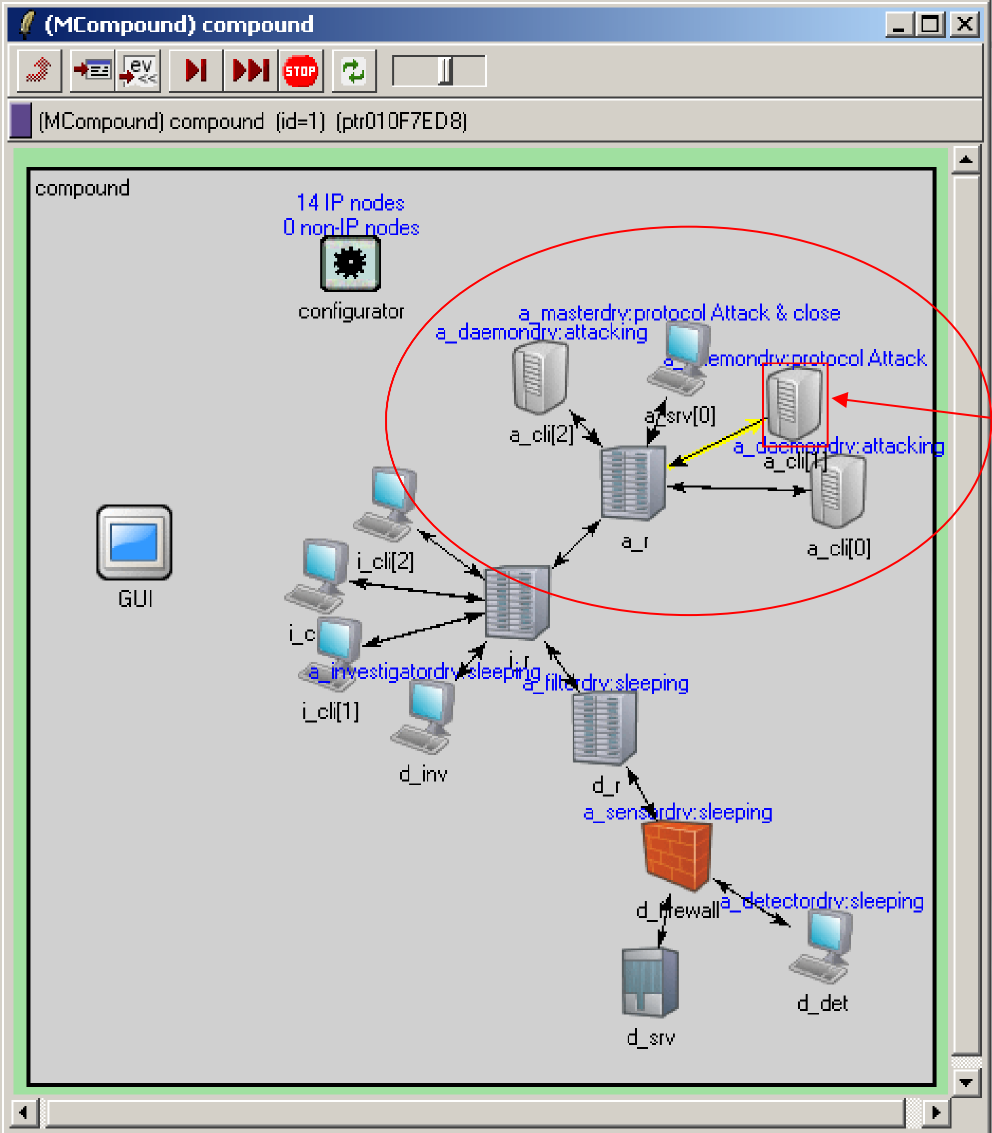
Attack team



Defense team is ready to protection

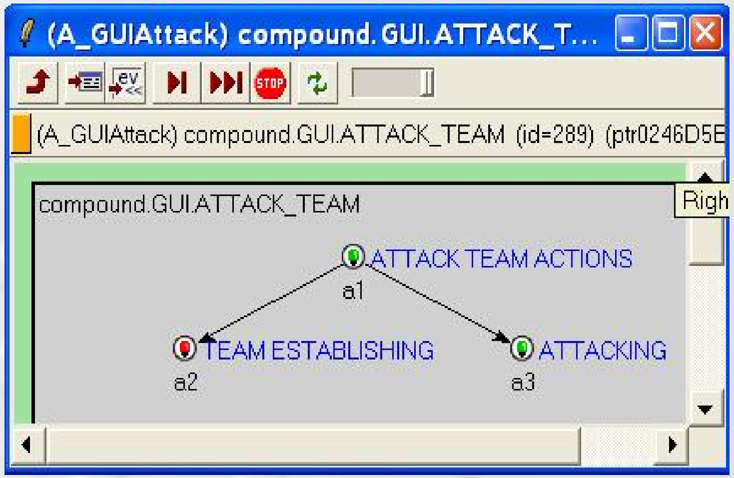


Beginning of the attack

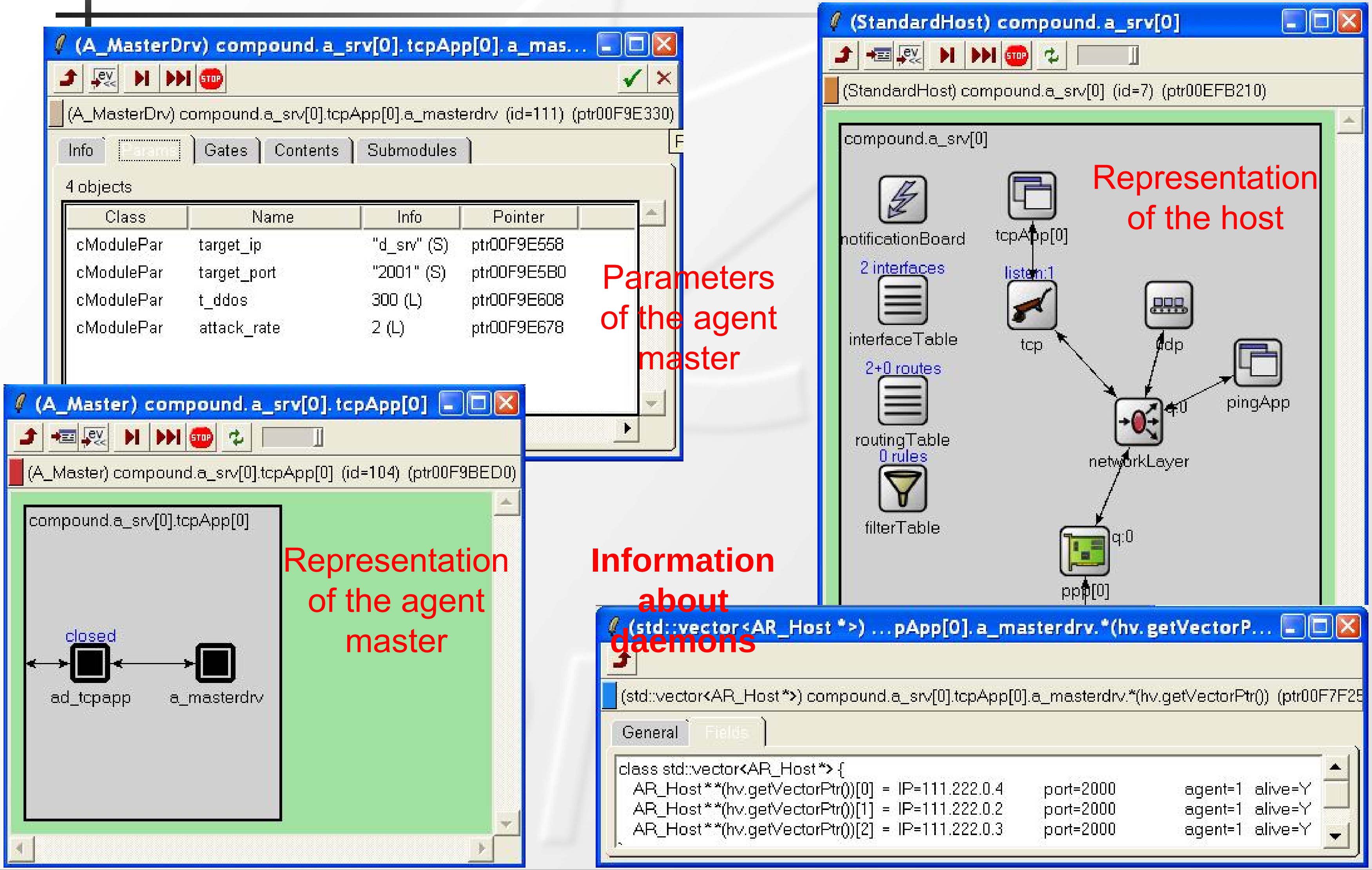


Master is sending to daemons command “initiate the attack”;

Daemons are attacking



Representation of agent "master" and the host where "master" is deployed

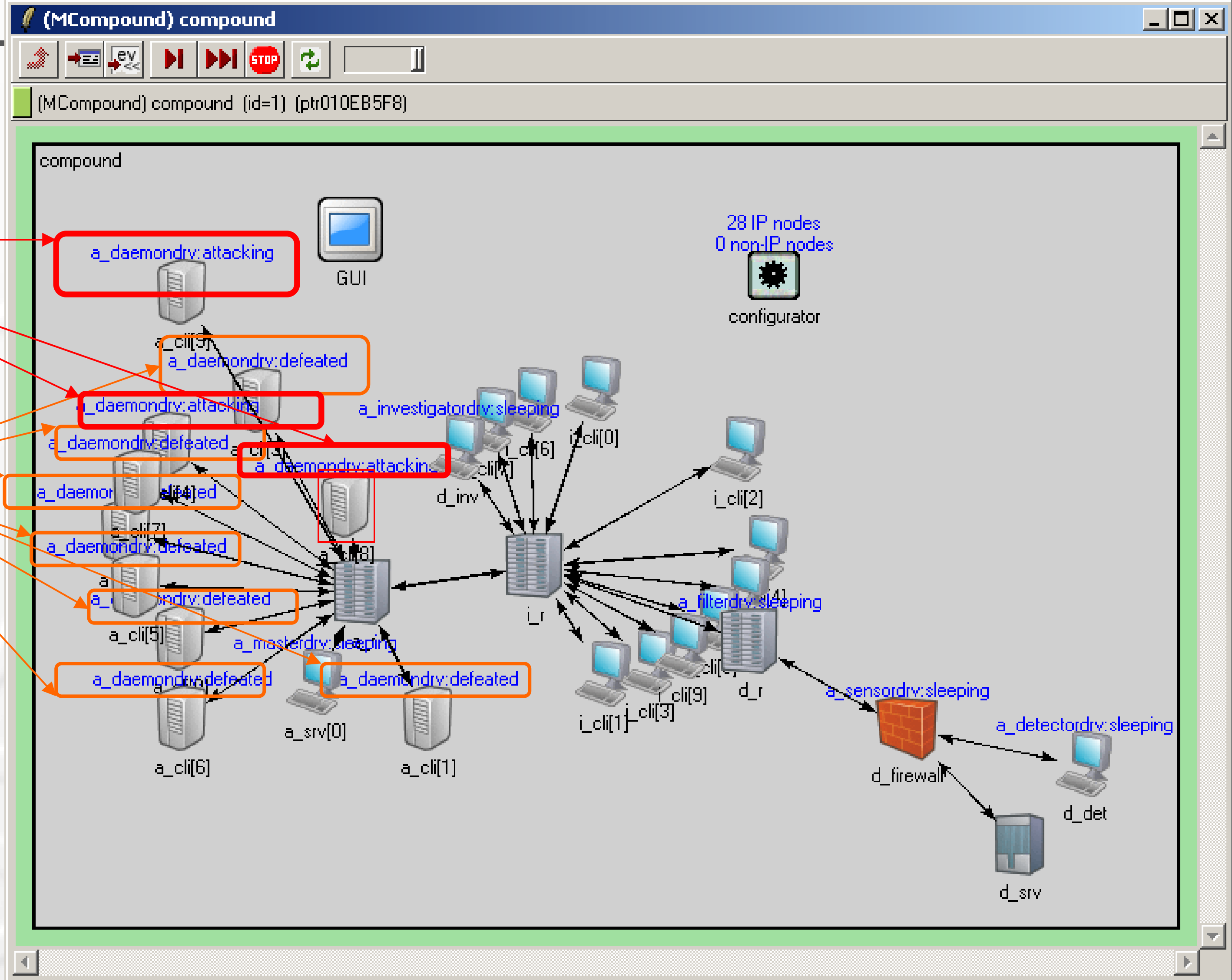


Network state after a set of defense actions

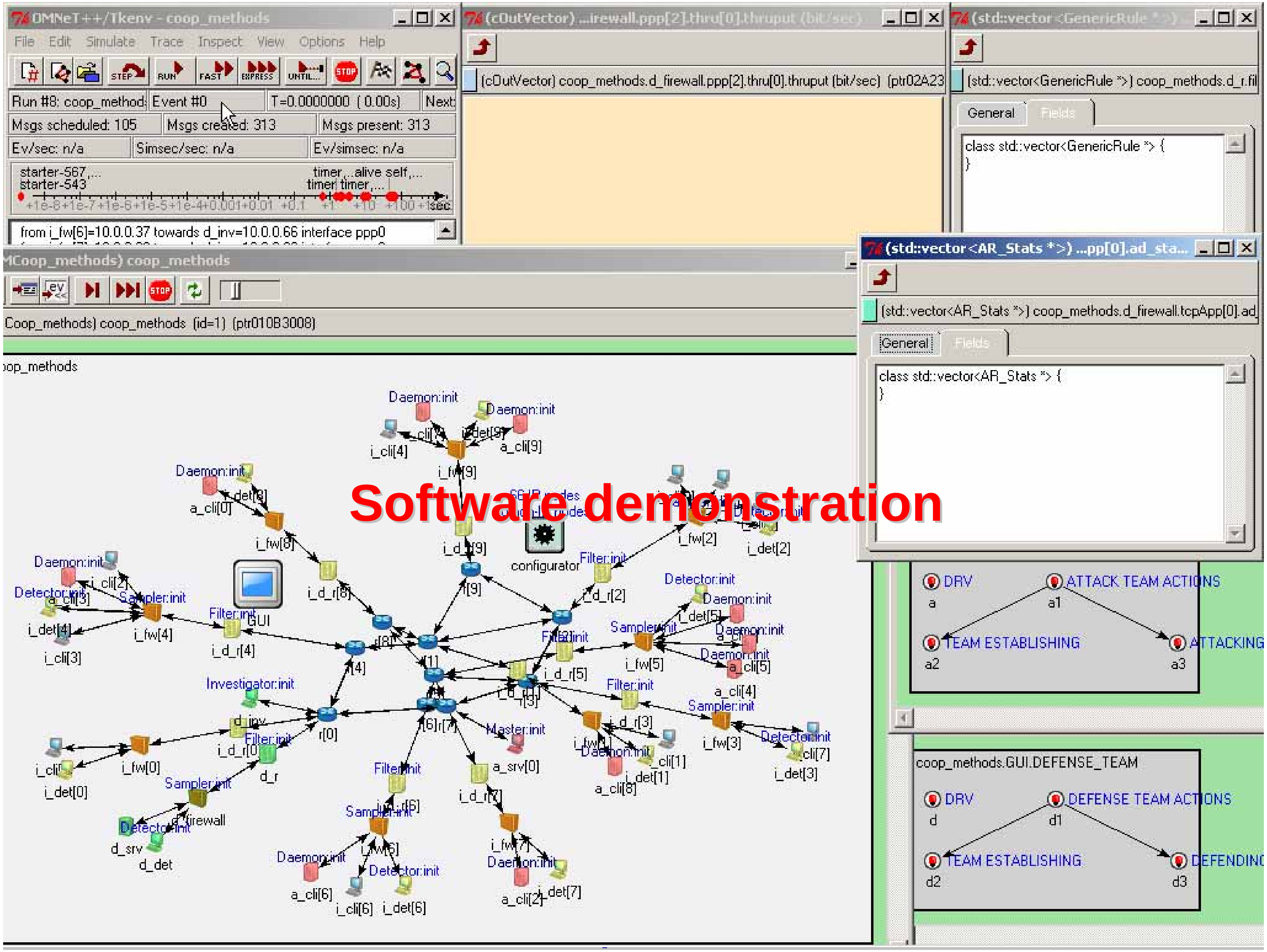
Attacking daemons

Defeated daemons

Example from another experiment



Software demonstration



Decision Making and Acting (1)

- Normal work (interval 0 – 300 seconds)

- Defense team: Formation, start using BPS method

- Attack team: Formation

- Attack team: After 300 seconds - begins the attack actions (intensity of attack for every daemon - 0.5, **no IP spoofing**)

- Defense team: data processing, attack detecting (**using BPS**) and reacting (interval 300 – 350 seconds)

- Defense team: blocking the attack, destroying some attack agents (interval 300 – 600 seconds)

Decision Making and Acting (2)

- Attack team: After 600 seconds **- automatic adaptation** (redistributing the intensity of attack (0.83), changing the method of **IP spoofing (Random)**)

- Defense team: data processing, failing to detect the attack (**using BPS method**) – Detector sees that the input channel throughput has noticeably lowered, but does does not receive any anomaly report from sampler because BPS does not work.

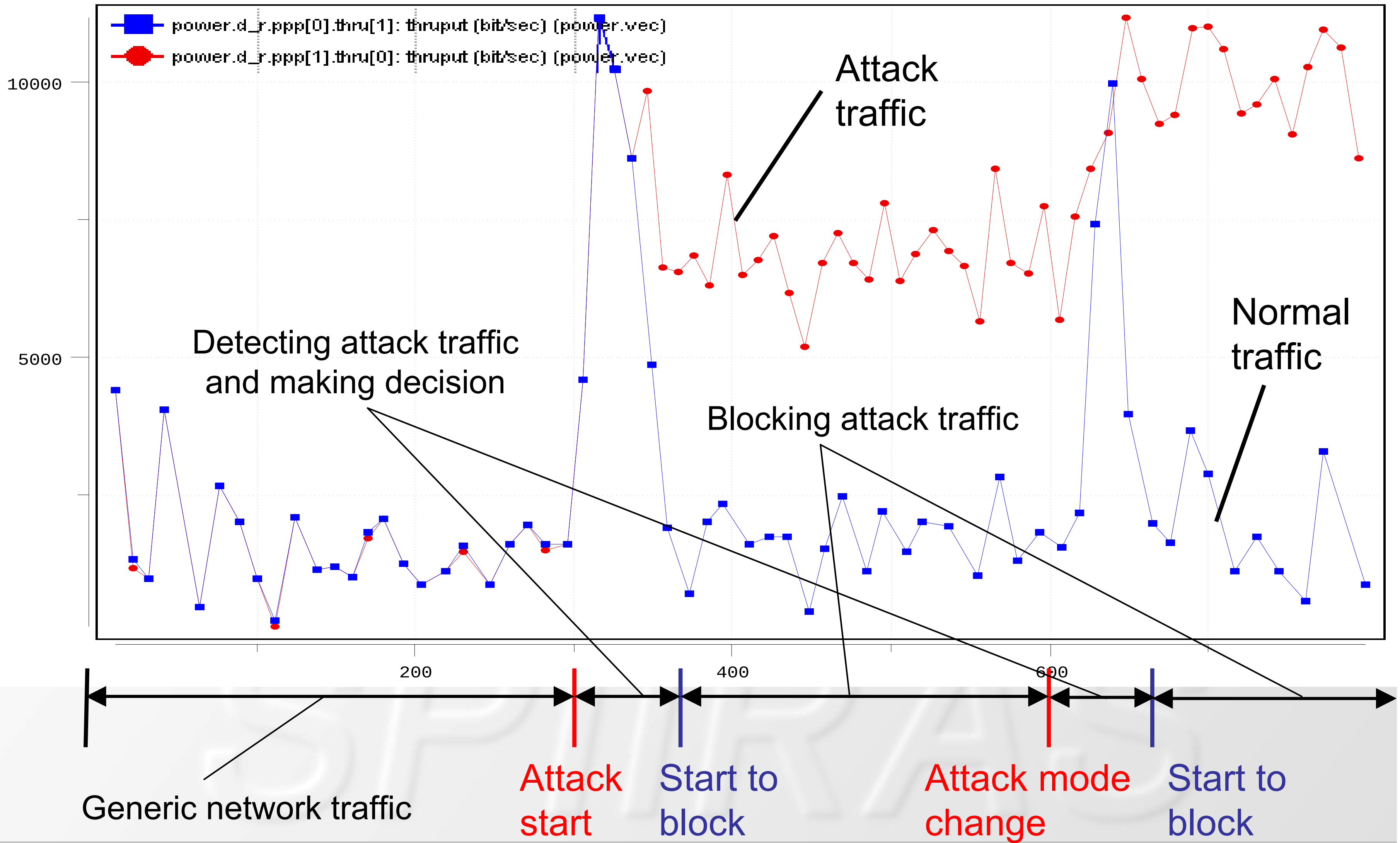
- Defense team: Changing defense method on **SIPM (automatic adaptation)**.

- Defense team: data processing, attack detecting (**using SIPM method**) and reacting – (interval 600 – 700 seconds)

- ……………………………

Scheme of Acting

Graphs of channel throughput



Outline

- Introduction
- Works describing attacks and attack taxonomies
- Works directly coupled with attack modeling and simulation
- Works devoted to descriptions of attack specification languages
- Works on evaluating security systems
- Formal grammar and state machines based approach
- Agent based and packet level simulation approach
- Conclusion

Conclusion: Main Results

- *Different works connected with attack modeling have been considered*.
  - describing attacks and attack taxonomies
  - directly coupled with attack modeling
  - devoted to descriptions of attack specification languages
  - on evaluating security systems
- *Two approaches (formal grammar & state machine based and agent-based & packet level simulation) have been outlined in detail.*
- *Software prototypes* allowing to imitate a wide spectrum of real life attacks. Software code is written in terms of C++, Java 2, MASDK, and OMNeT++.
- *Experiments with the prototypes* including the investigation of attack scenarios for networks with different structures and security policies.