Analyzing Network Security using Malefactor Action Graphs

Igor Kotenko[†], *and Mikhail Stepashkin*[††],

St. Petersburg Institute for Informatics and Automation, 39, 14 Liniya, St.-Petersburg, 199178, Russia

Summary
The approach to network security analysis is suggested. It is based on simulation of malefactor's behavior, generating attack graph and calculating different security metrics. The graph represents all possible attack scenarios taking into account network configuration, security policy, malefactor's location, knowledge level and strategy. The security metrics describe computer network security at different levels of detail and take into account various aspects of security. The generalized architecture of security analysis system is presented. Attack scenarios model, common attack graph building procedures, used security metrics, and general security level evaluation are defined. The implemented version of security analysis system is described, and examples of express-evaluations of security level are considered.

Key words:
Vulnerability Assessment, Risk Assessment, Security Metrics, Network attacks, Management of Computing Security.

Introduction

There are many different reasons of security violations in computer networks: security policy errors, vulnerabilities, incorrect configuration, etc. Malefactors can use different vulnerabilities and bottlenecks of network configuration and security policy and perform different penetration strategies. These strategies are directed to different network resources and include various assault actions chains. Malefactors can step-by-step compromise network hosts and realize different security threats.

Therefore, during computer network design and maintenance, designer or administrator should *check whether network configuration parameters and security procedures provide necessary security level*. Moreover, at exploitation stage, the configuration of computer networks can be changed, and it is also necessary to perform network monitoring, analyze available vulnerabilities and evaluate security level. At design stage the specifications of network configuration and security policies are the main input for security analysis. At exploitation stage the main input are the actual parameters of network configuration and security policy. The complexity of computer network security management causes the necessity to develop powerful *automated security analysis systems*. These systems should allow finding and correcting errors in network configuration, reveal possible assault actions for different security threats, determine critical network resources and choose effective security policy appropriate to threats.

At *design stages*, the different approaches to security analysis and evaluation of general security level can be used, for example, based on qualitative and quantitative techniques of risk analysis [1, 2]. We think the perspective directions in evaluating security of large-scaled networks are simulating possible malefactor's actions, building the representation of these actions as attack graphs, the subsequent checking of various properties of these graphs, and determining security metrics which can explain possible ways to increase security level. At *exploitation stages*, passive and active methods of vulnerability assessment are used. The passive methods do not allow estimating the possible routes of malefactor's penetration. The active methods can not be applied in all situations, as lead to operability violation of network services or the system as a whole. The combination of passive obtaining appropriate data about network configuration and security policy, attack modeling and simulation, attack graph construction, and automatic reasoning can be satisfactory approach to solve these problems.

There are a lot of papers which consider different approaches to security analysis. For example, the different methods for representing attack scenarios are used: attack trees [26], formal grammars [9], cause-effect model of attacks [4], structured tree-based description [7], object-oriented discrete event simulation [3], knowledge based models [27], etc. [1, 2] describe the possible risk analysis techniques for estimating security level. [23] proposes model checking technique for network vulnerability analysis. [11, 12] suggest the technique of attack graph evaluation based on model checking, Bayesian and probabilistic analysis. [25] presents algorithms for generating scenario graphs based on symbolic and explicit-state model checking. These algorithms ensure producing counterexamples for determining safety and liveness properties. [24] proposes an approach for analyzing different attack scenarios. The approach is based on high-level specification language, a translation from this language to constructs of model checker SPIN, applying optimization techniques and model checking for

Manuscript received June 5, 2006.
Manuscript revised June 25, 2006.

automated attack scenario analysis. In [13] the game theory based method of evaluating security is suggested. The authors view the interactions between an attacker and the administrator as a two-player stochastic game and construct the game model. The approach offered in [28] is intended for performing penetration testing of formal models of networked systems for estimating security metrics. [29] proposes an approach for construction of attack graph. Using graph methods they identify the attack paths with the highest probability of success. [10] describes global metrics which can be used to analyze and proactively manage the effects of complex network faults and attacks, and recover accordingly. [22] offers a methodology and a tool for vulnerability analysis which can automatically compute possible attack paths and verify some security properties. [6] proposes approach to estimate the risk level of critical network resources using behavior based attack graphs and Bayesian technique. [20] suggests the logic programming approach to automatically fulfill network vulnerability analysis. In [15] the common approach, attack graph visualization techniques and the tool for topological network analysis are considered.

In the paper we try to develop *a new approach to security evaluation based on comprehensive simulation of malefactor's actions, construction of attack graphs and computation of different security metrics. The main difference of offered approach* from examined ones consists in the way of simulating attacks (we use a multi-level model of attack scenarios) and applying constructed attack graphs (for different locations of malefactors) for determining a family of security metrics and comprehensive evaluation of security properties. Security analysis system based on the offered approach is intended for usage at different stages of computer network life cycle (fig. 1). The results of security analysis are as follows: (1) vulnerabilities detected; (2) routes (graphs) of possible attacks; (3) bottlenecks in network; (4) different security metrics, which can be used for general security level evaluation of computer network (system) and its components. Obtained results allow producing the valid recommendations for elimination of detected bottlenecks and strengthening common security level. If it is necessary, the user repeats the process of security evaluation.

The work is organized as follows. *Section 2* describes the generalized architecture of security analysis system based on offered approach. *Section 3* defines the model of attack scenarios used for attack simulation and considers the common attack graph generated. *Section 4* specifies security metrics. *Section 5* describes the procedure for evaluating a common security level. *Section 6* presents the security analysis system implemented. Conclusion surveys main work results and future research.

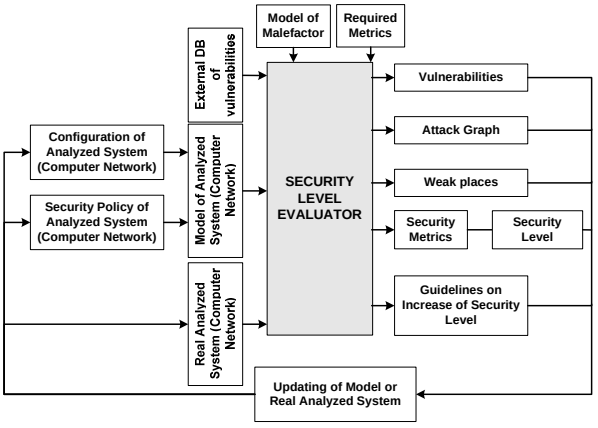


Fig. 1. Generalized architecture of security analysis system.

2. Security Analysis System Architecture

The generalized architecture of Security Analysis System (SAS) is depicted in fig.2 and contains the following components: (1) user interface; (2) network interface; (3) module of generating the internal representation of the model of analyzed system and security policy; (4) module of data control; (5) data repository; (6) module of data repository update; (7) module of generating the general attack graph; (8) module of malefactor's model realization; (9) reports generation module.

At the design stage, SAS operates with the model of analyzed computer network (system). This model is based on design specifications of computer network configuration and security policy.

Module of user interface provides the user with ability to control all components of SAS, set the input data, inspect reports, etc. *Network interface* provides interaction with external environment (sending requests to external vulnerabilities databases for updates and communicating with data sources).

Module of generating the internal representation of the model of analyzed system and security policy converts the information about network configuration and security policy received from collector (at exploitation stage) or from user (at design stage, this information is specified on System Description Language (SDL) and Security Policy Language (SPL)) into internal representation. Used specifications of analyzed network (system) and security policy should describe network components with the necessary degree of detail — the used software (in the form of names and versions of software) should be set. If required data is absent (for example, the user has not defined the version of used network service), the system should suggest to the user to enter the necessary information on the basis of *database of software*. So the

module of data control is used for detection of the incorrect or undefined data which are necessary for the security level evaluation. For example, the user can make a mistake in the name of network service or specify that port 21 is opened on a given server, but not specify what application serves requests directed to this port. For elimination of errors arising at input of specifications the module of data control provides to the user a choice of necessary data, using database of software names.

Data repository consists of the following groups of databases (DB): (1) the group of DB about network configuration and security policy; (2) the group of DB of actions; (3) the group of additional databases.

The group of databases about network and security policy consists of the following bases: (1) DB of network configuration; (2) DB of used security policy; (3) DB of malefactor view on network configuration; (4) DB of malefactor view on used security policy. Structurally given DB (bases about network configuration and about used security policy) mutually coincide and contain information about network architecture and its parameters (for example, types and versions of used operating systems, list of opened ports, etc.) and about rules which describe the network operation. DB about network configuration is an internal representation of analyzed network specification which is used for generating the result of attack action when the general attack graph is under construction. DB of malefactor view on network configuration is an internal representation of analyzed network (how it is imagined by malefactor). This representation is a result of attack actions sequence. DB about used security policy includes the common rules of network functioning, for example "the local user of host *H* can not start the application *A*". It is possible to plan the sequence of malefactor's actions on basis of DB of malefactor view on used security policy (for example, according to security policy only local administrators can read file *F*, therefore the malefactor must gain the administrator privileges to read this file, i.e. he must realize certain actions).

The group of action databases consists of the following bases: (1) DB of actions which use vulnerabilities; (2) DB of reconnaissance actions; (3) DB of common actions. *DB of actions which use vulnerabilities* (unlike other bases of the given group) is constructed on basis of external vulnerabilities database. Attack actions of this DB are divided on the following groups: (1) actions which are directed to gaining privileges of local user; (2) actions which are directed to gaining privileges of administrator; (3) actions which are directed to confidentiality violation, (4) integrity violation and (5) availability violation. Examples of actions of this database are: "ServU-local-priv-esc", "Utilman", etc. (these actions are used in the case study). *DB of reconnaissance actions* contains actions which are directed to remote information gathering about

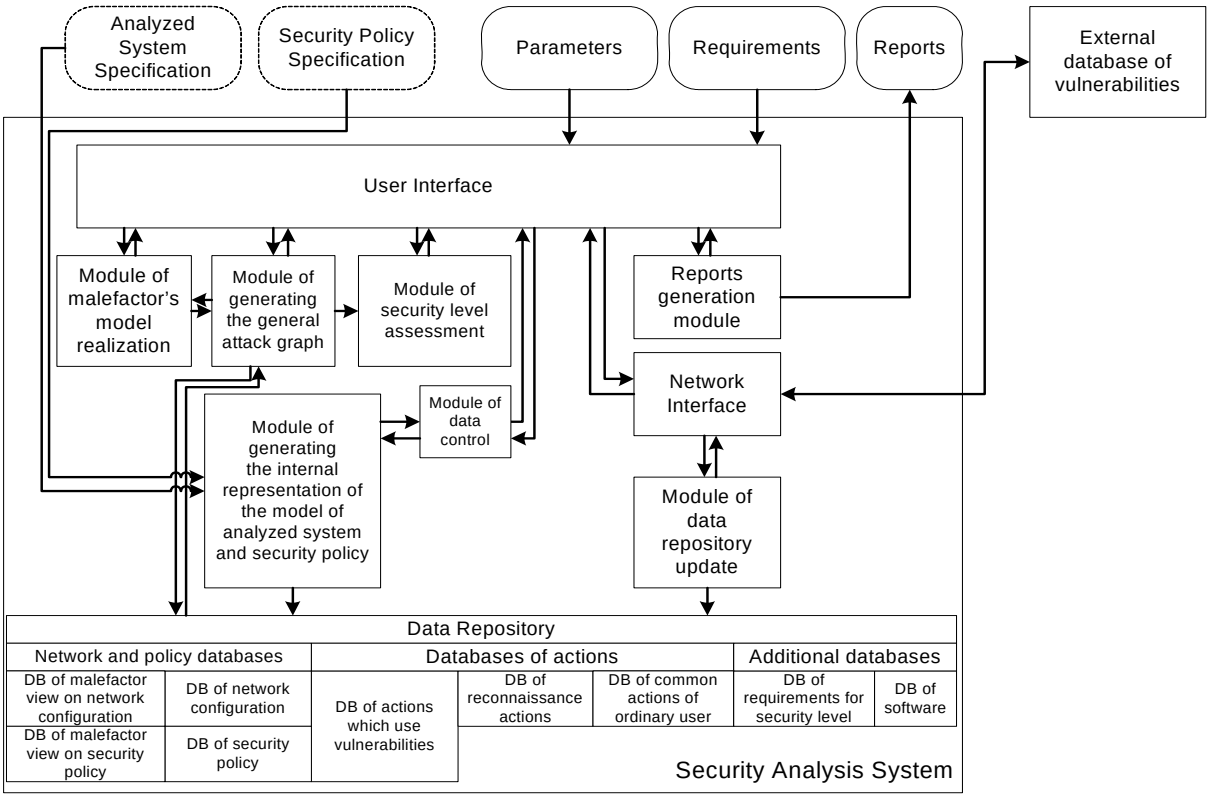


Fig. 2. Generalized architecture of security analysis system.

host or network. Information about methods and tools needed for realization of reconnaissance actions can be obtained by expert methods. Examples of actions of this database are: "Nmap-OS", "Ping", etc. (these actions are used in the case study). *DB of common actions* contains information about possible actions which execute according to user's privileges. For example, preparatory actions can concern to this type of actions, and also such actions, as "reading file", "copying file", "file deletion", "deleting of the directory", etc. Given actions can be used for realization of different threats. The condition of successful realization of the action (for example, version of the software) and result of its impact on the attacked object (for example, service crashing) are stored in DB for each action.

The group of additional databases consists of the following bases: (1) DB of requirements and (2) DB of software. *DB of requirements* contains predefined sets of security metrics values (set by experts). Each set corresponds to the certain security class regulated by international standards or other normative documents. The *database of software* is used by module of data control for detection of errors in the used specifications of computer network and for generating recommendations on using software tools.

The *module of data repository update* downloads the open vulnerability databases (for example, OSVDB — open source vulnerability database [19]) and translates them into database of attack actions. The *module of generating general attack graph* builds attack graph by modeling malefactor's attack actions in the analyzed computer network using information about available attack actions of different types (attack actions, reconnaissance actions and common actions of ordinary users), about network configuration and used security policy. This module sets up security metrics of elementary objects in the vertexes of attack graph. On basis of these metrics the module of security level assessment calculates metrics of combined objects. The *module of malefactor's model realization* determines a malefactor's skill level, his initial position and knowledge about analyzed network. Malefactor's skill level determines the set of actions used by malefactor and the attack strategy. The *module of security level assessment* generates combined objects of attack graph (routes, threats), calculates security metrics, evaluates common security level, compares obtained results with requirements, finds "weak" places, generates recommendations on strengthening security level. *Reports generation module* shows vulnerabilities detected by SAS, represents "weak" places, recommendations on strengthening security level.

At the maintenance stage the subsystem of data collection is used for construction of the model of analyzed computer network. This subsystem is depicted in fig. 3 and consists of the following components: (1) various data source (host's software agents, security blocks, various components of the proactive security monitor); (2) information collector.

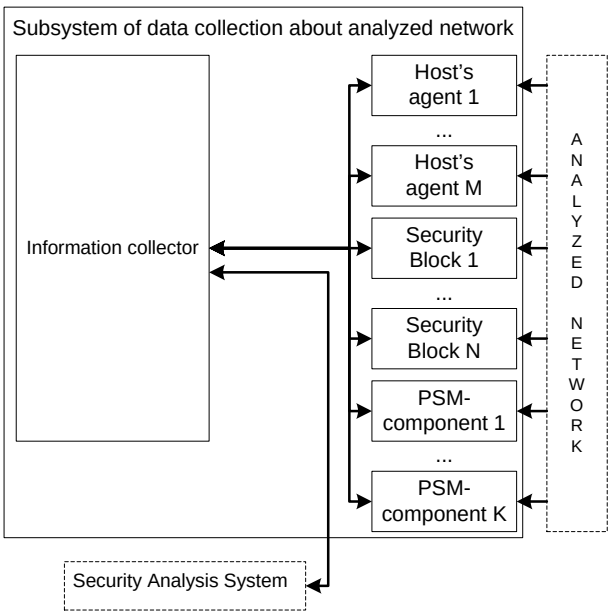


Fig. 3. Architecture of the subsystem of data collecting data.

3. Common Attack Graph

Our model of attack scenarios is hierarchical and contains three levels: integrated, script and actions.

Integrated level determines high-level purposes (threats) of the security analysis and attack objects. Integrated level allows coordinating several scenarios performed by one or several malefactors. *Script level* takes into account initial malefactor's qualification and knowledge about computer network, defines attack object and attack purpose (for example, "OS determining", "denial of service", etc.). Script level sets script stages used, including reconnaissance, penetration (initial access to the host), privileges escalation, threat realization, traces hiding and backdoors creation. The lower scenario elements serve for detailing of these phases. The *low level* of attack model describes low-level malefactor's actions and exploits.

The algorithm of generating the common attack graph is based on the attack scenarios model developed. It is intended for creation of attack graph which describes all possible routes of attack actions in view of malefactor's initial position, skill level, network configuration and used security policy. The algorithm of generating common attack graph is based on realization of the following action

sequence: (1) actions which are intended for malefactor's movement from one host onto another; (2) reconnaissance actions for detection of "live" hosts; (3) reconnaissance scenarios for detected hosts; (4) attack actions based on vulnerabilities and actions of ordinary users.

All *objects of attack graph* are divided into two groups: base objects and combined objects. *Base objects* define the graph vertexes. They are linked to each other by edges for forming different sequences of malefactor's actions. *Combined objects* are built on the basis of linking the elementary objects by arcs. Objects of types "host" and "attack action" are base (elementary) objects. Set of objects "*hosts*" includes all hosts discovered and attacked by malefactor. Set of objects "*attack action*" contains all distinguishable actions of malefactor.

All *attack actions* are divided into the following classes: Reconnaissance actions; Preparatory actions (within the limits of malefactor's privileges). These actions are used for creation of conditions needed for realization of other attack actions; Actions to gain the privileges of local user and of administrator; Confidentiality, Integrity and Availability violation.

Objects of types "route", "threat" and "graph" are *combined objects*. *Route* is a collection of linked vertexes of general attack graph (hosts and attack actions), first of which represents a host (initial malefactor's position) and last has no outgoing arcs. *Threat* is a set of various attack routes having identical initial and final vertexes. We classify *threats* as follows: (1) *Primary threats* – threats of confidentiality, integrity and availability violation; (2) *Additional threats* – threats of gaining information about host or network, gaining privileges of local user and administrator. *Graph* is integration of all threats.

4. Security Metrics

About 150 different metrics were constructed on basis of common attack graph. *According to division of attack graph objects* the set of security metrics can be divided into metrics of base objects (hosts and attack actions) and metrics of combined objects (routes; threats; attack graph). *According to the order of calculation* we differentiate primary and secondary metrics. Primary metrics are defined from attack graph; secondary ones are calculated on the basis of primary. All security metrics are divided into basic and auxiliary according to whether they are used for evaluating a common security level. *Basic security metrics* are directly used for the evaluation. *Auxiliary security metrics* serve for building detailed picture of network security and are required, for example, for detecting bottlenecks and generating recommendations on security level strengthening. Table 1 contains examples of security metrics used.

Table 1: Examples of security metrics

Metrics based on network configuration
Quantity of hosts, firewalls, Linux hosts, Microsoft Windows hosts, hosts with antivirus software installed, hosts with personal firewalls, hosts with host-based intrusion detection systems, etc.
Metrics of hosts
Criticality level, etc.
Metrics of attack actions
Criticality level; Damage level caused by attack action which takes into account the criticality level of host; Access complexity; Base Score; Confidentiality Impact; Availability Impact; Access Complexity, etc.
Metrics of attack routes
Route length expressed in vulnerable hosts; Quantity of different hosts of route where malefactor gains the privileges of administrator; Route average Base Score; Maximum Access Complexity; Damage level of route; Maximum damage level of route, etc.
Metrics of threats
Minimum and maximum quantity of different vulnerable hosts used for threat realization; Quantity of different routes which lead to threat realization; Damage level of threat; Maximum damage level of threat; Access Complexity of threat; Admissibility of threat realization; Risk level of threat, etc.
Metrics of common attack graph
by hosts
Quantity of different vulnerable hosts of graph;, Quantity of different hosts where malefactor gains the privileges of administrator, etc.
by attack actions
Quantity and set of different attack actions, Average Base Score of all different attack actions, etc.
by routes
Quantity of routes in graph; Quantity of routes leading to confidentiality, integrity, availability violations, etc.
by threats
Quantity of treats in graph, Quantity of treats leading to confidentiality, integrity, availability violations, etc.
combined (integral)
Array of quantity of routes which are passing through each host; Array of quantity of different vulnerabilities which are discovered on each host; Integral security metric "Security level", etc.

We define the following metrics as basic ones: Criticality level of host h (*Criticality(h)*); criticality level of attack action a (*Severity(a)*); Damage level caused by attack action and taking into account the criticality level of host (*Mortality(a,h)*); damage level of route S and threat T (*Mortality(S)* and *Mortality(T)*); "Access complexity" of attack action a, route S and threat T (*AccessComplexity(a), AccessComplexity(S), AccessComplexity(T)*); Admissibility of threat realization (*Realization(T)*); Risk

level of threat *T* (*RiskLevel(T)*); General security level of computer network (*SecurityLevel*).

Some security metrics are calculated on basis of standard *Common Vulnerability Scoring System* [5]. CVSS metrics are divided into base, temporal and environmental. *Base indexes* define *criticality* of vulnerability. *Temporal indexes* determine *urgency* of vulnerability at the given time. *Environmental indexes* should be used by organizations for priorities arrangement at time of generating plans of vulnerabilities elimination. CVSS metrics of attack actions can be obtained from external databases of vulnerabilities, for example, NVD [17].

5. Common Security Level

The offered technique for qualitative express-assessment of security level contains the following stages: (1) Calculation of security metrics of basic and combined objects; (2) Estimation of qualitative assessments of risk level for all threats; (3) Security level evaluation on basis of risk levels of all threats.

The damage level caused by attack action depends on criticality levels of host and attack action. We define this value as *Mortality(a,h)*.

Criticality level of host (*Criticality(h)*) is determined by designer (or administrator) using three level scale (High, Medium, Low), taking into account the purpose of given host and its functions. The administrator (or designer) determines a criticality level of hosts (for example, administrator may set a high criticality level for all hosts of network). For example, Table 2 may be used for evaluating criticality level, if accessibility is the main factor. In this case the maximum criticality level is defined for those hosts, incorrect functioning of which leads to impossibility of using network resources by legitimate users.

Table 2: Evaluation of *Criticality(h)*

Criticality(h)	Host Type
High	DNS servers, corporate routers, domain controllers; Servers and workstations with critical information
Medium	web-, mail- and ftp-servers, firewalls
Low	Personal workstations

According Table 2, the maximum criticality level is defined for those hosts, incorrect functioning of which leads to impossibility of using network resources by legitimate users. For example, the failure of corporate router leads to the following disadvantages: external users can not use network resources (public servers) located in the demilitarized zone; internal users can not use the Internet resources. Next, according to criticality level decreasing, work servers follow. Functioning of each server is an important part of successful work of organization as a whole. The minimum criticality level is defined for personal workstations. As a rule, impact of short-term violations of these workstations on an organization is not important.

Criticality level of attack action Severity(a) is calculated with usage of CVSS index "BaseScore" (*BaseScore(a)*) as follows [18]:

$$Severity(a)=\begin{cases} Low, & BaseScore(a)\in[0.0,3.9] \\ Medium, & BaseScore(a)\in[4.0,6.9] \\ High, & BaseScore(a)\in[7.0,10.0] \end{cases}.$$

Damage level Mortality(a,h) caused by attack action and taking into account the criticality level of host is calculated according to Table 3.

Table 3: Evaluation of *Mortality(a,h)*

Criticality(h)	*Severity(a)*		
	High	Medium	Low
High	High	High	Medium
Medium	High	Medium	Low
Low	Medium	Low	Low

Damage level of threat is defined by the latest attack action of one of the threat routes (and by host *h* corresponding to this action):

$$Mortality(T)=Mortality(a_T,h_T),$$

where a_T – the latest attack action of the given threat, h_T – host attacked by a_T. The values of *Mortality(T)* are as follows: *High* – stopping of critical subdivisions of organization which leads to essential financial losses; *Medium* – short-term stopping of critical processes or systems which leads to limited financial losses in one subdivision of organization; *Low* – the damage do not cause the essential financial losses.

However, it is possible that a malefactor at threat realization causes the greater damage than damage calculated by latest attack action of the threat. Then it is necessary to define the following metrics: maximum damage level of route *S* and threat *T*. These metrics can be calculated as follows:

$$Mortality^{max}(S)=\max_i(Mortality(a_i,h_i)), i\in[1,N_S],\ a_i\in S,$$

$$Mortality^{max}(T)=\max i(Mortality(S_i)),\ i\in[1,N_T],\ S_i\in T,$$

where N_S – length of route (quantity of attack actions in the route); N_T – quantity of routes in the threat *T*.

For calculation of *threat risk level* it is necessary to evaluate the threat realization admissibility (*Realization(T)*) and to use the FRAP technique (with

usage of threat damage level *Mortality(T)*, obtained earlier).

"Access Complexity" index of CVSS (*AccessComplexity(a)*, where *a* – attack action) is used for computation of threat realization admissibility. This index belongs to a group of base indexes and is specified for each attack action. The possible values of this index are: *High* – there are specific conditions for vulnerability usage (attack action realization), for example a specific time or network service configuration, interaction with human, etc; *Low* – there are no specific conditions, i.e. the vulnerability is always exploitable. Then, *"Access Complexity" index of route S* can be calculated as follows:

$$AccessComplexity(S) = \begin{cases} High, \exists k \in [1, N]: \\ AccessComplexity(a_k) = High \\ Low, \forall k \in [1, N]: \\ AccessComplexity(a_k) = Low \end{cases},$$

where $S = \{a_i\}_{i=1}^{N}$ – attack scenario (route); N – length of route (quantity of attack actions).

The following formula can be used for computation of *"Access Complexity" index of threat T* (represented as a set of various attack routes having the same initial and final vertexes):

$$AccessComplexity(T) = \begin{cases} Low, \ \exists k \in [1, N_S]: \\ AccessComplexity(S_k) = Low \\ High, \ \forall k \in [1, N_S] \\ AccessComplexity(S_k) = High \end{cases},$$

where $T = \{S_k\}_{k=1}^{N_S}$ – threat; N_S – quantity of different routes of threat T; $S_k = \{a_i\}_{i=1}^{N_k}$ – attack scenario (route); N_k – quantity of attack actions in the route.

Then *admissibility of threat T realization* can be computed in the following way:

$$Realization(T) = \begin{cases} High, \ AccessComplexity(T) = Low \\ Low, \ AccessComplexity(T) = High \end{cases}.$$

Threat risk level (*RiskLevel(T)*) can be evaluated according to the matrix of risks depicted in Table 4.

Table 4: Evaluation of threat risk level

Realization(T)	*Severity(T)*		
	High	Medium	Low
High	A	B	C
Low	B	C	D

Threat risk level can be interpreted as follows: **Level A** – the actions concerned with risk (for example, using a new security software or eliminating detected vulnerabilities) should be fulfilled immediately; **Level B** – the actions concerned with risk should be undertaken; **Level C** – monitoring a situation is required (possibly, it is not required to undertake additional actions); **Level D** – additional actions are not required.

Network security level (*SecurityLevel*) is defined (according to obtained values of risk level for all threats) in the following way:

$$SecurityLevel = \begin{cases} Green, \ \forall i \in [1, N_T] \, RiskLevel(T_i) = D \\ Yellow, \ \forall i \in [1, N_T] \, RiskLevel(T_i) \le C \\ Orange, \ \forall i \in [1, N_T] \, RiskLevel(T_i) \le B \\ Red, \ \exists i \in [1, N_T] \ RiskLevel(T_i) = A \end{cases},$$

where D<C<B<A, N_T – the quantity of all threats.

6. Implementation

Based on the suggested approach the *security analysis system* has been implemented. The network model which SAS uses at the design stage is based on the XML specifications of network configuration expressed in *System Description Language (SDL)* and security policy in *Security Policy Language (SPL)*. At the exploitation stage the network model is created on basis of data collected from network. The main SAS components are *DataRepository*, *ModelInitializator*, *DataControl*, *GraphBuilder*, *GraphAnalyzer*, *ReportGenerator*, *InformCollector* and *DataUpdater*.

The principal components of *Data Repository* are the database (DB) of network configuration and security policy (*NetworkModel*), including DB of malefactor's conception on configuration and security policy, and the DB of actions (*Attacks*). The component *NetworkModel* contains information about network architecture and its parameters (for example, types and versions of used operating systems, applications, list of opened ports, etc.) and rules which describe security policy. The component *Attacks* consists of DB of actions which use vulnerabilities and DB of usual user actions.

ModelInitializator converts the information about network configuration and security policy into internal representation. *DataControl* is used to detect the incorrect or undefined data which are necessary for evaluating security level. For example, the user can make a mistake in the name of a service or specify that port 21 is opened, but do not specify what application serves the requests on this port.

GraphBuilder builds attack graph by simulation of malefactor's actions using information about available

attack actions, network configuration and used security policy. This module sets up security metrics of elementary objects in attack graph vertexes. On basis of these metrics, *GraphAnalyzer* calculates metrics of combined objects. *ReportGenerator* shows vulnerabilities detected by SAS, represents bottlenecks and recommendations on strengthening security level. *InformCollector* is used for collection of information derived from host software agents, representation of the derived information using SDL and SPL and transfer of this data to the *ModelInitializator*. *DataUpdater* downloads the XML specifications of vulnerabilities from open vulnerability database OSVDB [19] and translates them into database of attack actions.

Figure 4 shows SAS user interface. It is divided into four basic parts: (1) The SAS *main menu*; (2) The *top area of the main window* with the following tabs: "Analyzed Network Model" – representation of network model; "Malefactor's Network Model" – malefactor's conception on configuration and security policy at the given attack stage; "General Attack Graph" – general attack graph representation; (3) The *bottom area of the main window* which displays the SAS log, vulnerable hosts and revealed vulnerabilities, security metrics, requirements and reports; (4) Management buttons area.

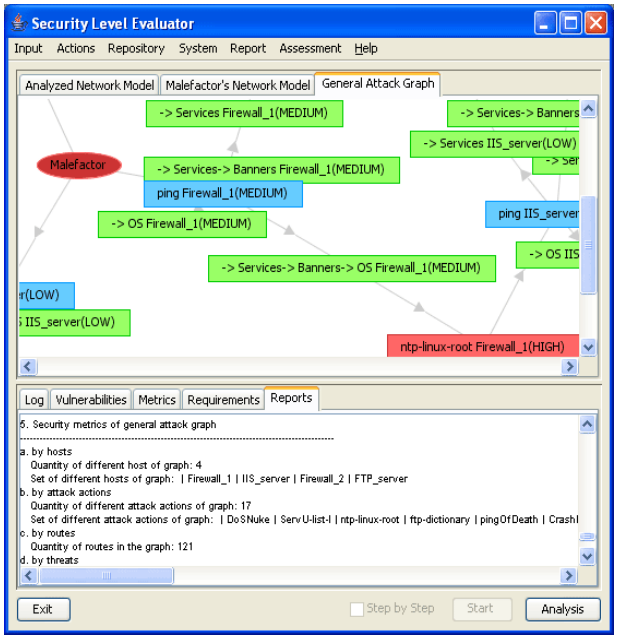


Fig. 4. SAS user interface.

Let us consider SAS operation for the simple test network depicted in Figure 5. During construction of common attack graph there are the following main changes in malefactor's conception about network (depicted in step-by-step mode at Malefactor's Network Model tab – see Figure 6):

- After executing the "ping" attack (according to rules of traffic routing), malefactor gets to know about host "Server" (Figure 6,a);
- Malefactor executes attack action which uses the vulnerability in ftp service and allows to gain administrator privileges (host "Server" is highlighted by red) (Figure 6,b);
- Malefactor uses gained privileges to get all information about "Server". This information allows him to understand that "Server" is connected to another switch (the port forwarding is used in the network). Hence, it is advantageous for malefactor to change location since he gets the access to other network segment. Figure 6,c shows a breach between hosts which appears because malefactor does not know through what host (router) the network packets move from his host ("Malefactor") to host "Server" (Figure 6,c);
- Malefactor moves to the host "Server", executes the "ping" action and gets to know about a set of hosts which he tries to attack sequentially (Figure 6,d).

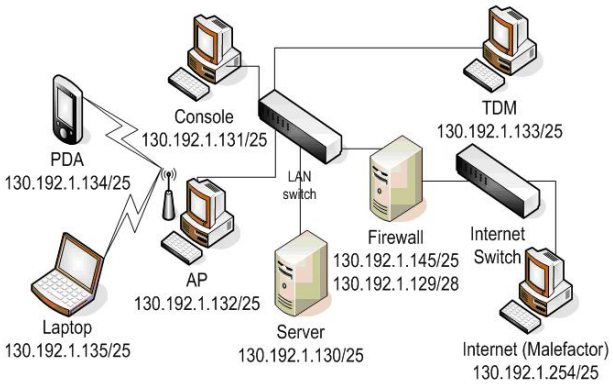
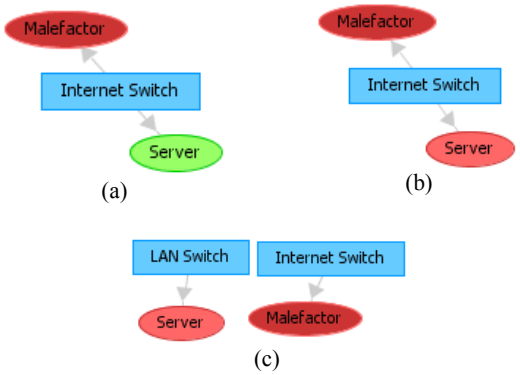
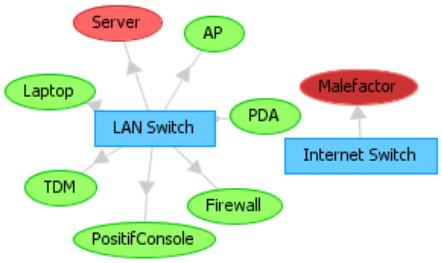


Fig. 5. Structure of test network.





(d)

Fig. 6. Malefactor's representations about the network.

The common attack graph for the test network is depicted in Figure 7. The following denotations are used: (1) *Red oval* describes initial position of malefactor; (2) *Blue rectangle* represents "ping" attack action. In the rectangle the name of attacked host and damage level caused by attack action are displayed; (3) *Green rectangle* is used for representation of reconnaissance scenarios; (4) *Pink rectangle* describes "denial of service" actions; (5) *Gray rectangle* represents attack actions which are directed to gain the privileges of local user; (6) *Red rectangle* describes attack actions directed to gain the privileges of administrator.

The main results of security analysis for this example are as follows: (1) detected vulnerabilities (for example, "ServU-MDTM" in Figure 4); (2) values of security metrics (for example, quantity of attack routes which are passing through "Server"); (3) reports on the status of basic security aspects including recommendations on security level increase. Security analysis showed that the network security level is red. Next actions of user should become: (1) elimination of detected vulnerabilities and bottlenecks by updating network configuration and security policy; (2) repeated security analysis.

7. Conclusion

The paper offered the approach and software tool for security analysis of computer networks. The suggested approach possesses the following *peculiarities*: (1) Usage of integrated family of different models based on expert knowledge, including malefactor's models, multilevel models of attack scenarios, building attack graph, security metrics computation and security level evaluation; (2) Taking into account diversity of malefactor's positions, intentions and experience levels; (3) Usage (during construction of common attack graph) not only of the parameters of computer network configuration, but the rules of security policy used; possibility of estimating the influence of different configuration and policy data on the security level value; (4) Taking into account not only attack actions (which use vulnerabilities), but the common actions of legitimate users and reconnaissance actions; (5) Possibility of investigating various threats for different network resources; (6) Possibility of detection of bottlenecks (hosts and applications responsible for the most serious attack actions, routes and threats); (7) Possibility of querying the system in the "what-if" way, for example, how the general security level will change if the certain parameter of network configuration or security policy is changed or information about new vulnerability is added; (8) Usage of updated vulnerabilities databases (for example, Open Source Vulnerability Database (OSVDB) [19]); (9) Usage of widespread CVSS approach

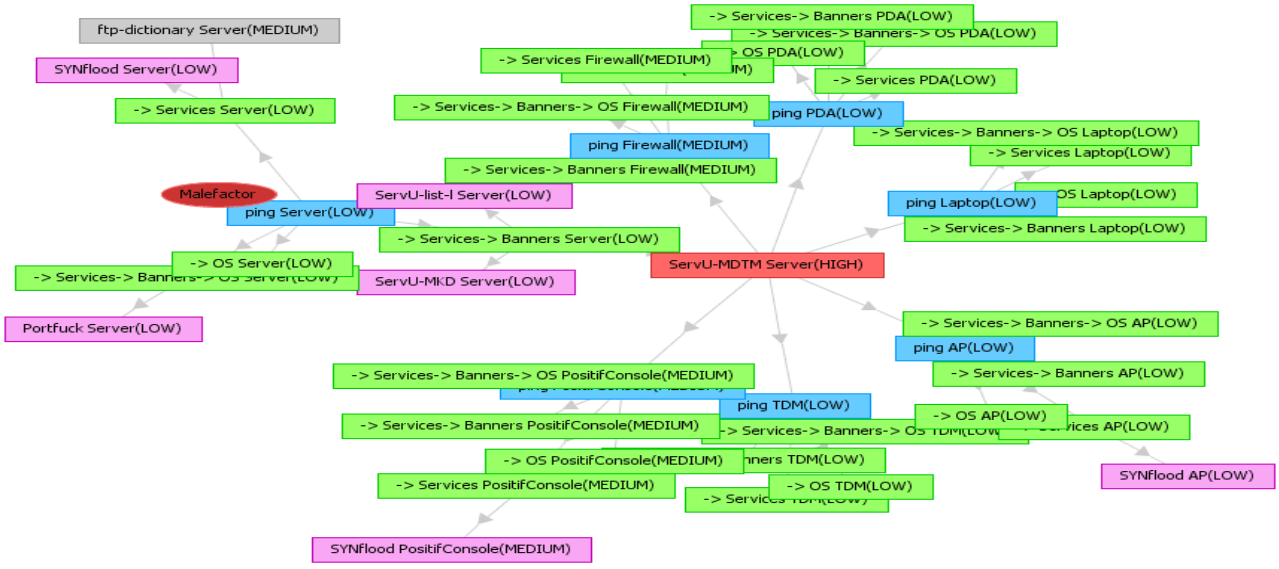


Fig. 7. Common attack graph of the tested network (for external malefactor).

[5]; (10) Usage of qualitative techniques of risk analysis (in particular, modified techniques of evaluating attack criticality of SANS/GIAC and FRAP [8]).

The future research will be devoted to comprehensive experimental assessment of offered approach and improving the models of computer attacks and security level evaluation.

Acknowledgments

This research is being supported by grant of Russian Foundation of Basic Research (№ 04-01-00167), grant of the Department for Informational Technologies and Computation Systems of the Russian Academy of Sciences (contract №3.2/03) and partly funded by the EC as part of the POSITIF project (contract IST-2002-002314).

References

[1] Alberts, C., Dorofee, A., 2002. *Managing Information Security Risks: The OCTAVE Approach.* Addison Wesley.
[2] Chapman, C., Ward, S., 2003. *Project Risk Management: processes, techniques and insights.* Chichester, John Wiley.
[3] Chi, S.-D., Park, J.S., Jung K.-C., Lee, J.-S., 2001. Network Security Modeling and Cyber Attack Simulation Methodology. *LNCS, Vol.2119*.
[4] Cohen, F., 1999. Simulating Cyber Attacks, Defenses, and Consequences. In *IEEE Symposium on Security and Privacy*, Berkeley, CA.
[5] CVSS, 2006. *Common Vulnerability Scoring System.* Retrieved April 14, 2006, from http://www.first.org/cvss/
[6] Dantu, R., Loper, K., Kolan P., 2004. Risk Management using Behavior based Attack Graphs. In *International Conference on Information Technology: Coding and Computing*.
[7] Dawkins, J., Campbell, C., Hale, J., 2002. Modeling network attacks: Extending the attack tree paradigm. In *Workshop on Statistical and Machine Learning Techniques in Computer Intrusion Detection*, Johns Hopkins University.
[8] FRAP, 2006. *Facilitated Risk Analysis Process.* Retrieved April 14, 2006, from http://www.peltierassociates.com/
[9] Gorodetski, V., Kotenko, I., 2002. Attacks against Computer Network: Formal Grammar-based Framework and Simulation Tool. *LNCS, V.2516.*
[10] Hariri, S., Qu, G., Dharmagadda, T., Ramkishore, M., Raghavendra, C. S., 2003. Impact Analysis of Faults and Attacks in Large-Scale Networks. In *IEEE Security&Privacy, September/October*.
[11] Jha, S., Linger, R., Longstaff, T., Wing, J., 2000. Survivability Analysis of Network Specifications. In *Intern. Conference on Dependable Systems and Networks*. IEEE CS Press.
[12] Jha, S., Sheyner, O., Wing, J., 2002. Minimization and reliability analysis of attack graphs. *Technical Report CMU-CS-02-109*, Carnegie Mellon University.
[13] Lye, K., Wing, J., 2005. Game Strategies in Network Security. *International Journal of Information Security, February*.
[14] McNab, C., 2004. *Network Security Assessment.* O'Reilly Media, Inc.
[15] Noel, S., Jajodia, S., 2005. Understanding complex network attack graphs through clustered adjacency matrices. In *Proc. 21st Annual Computer Security Conference (ACSAC).*
[16] Netfilter, 2006. *Netfilter/iptables documentation*. Retrieved April 14, 2006, http://www.netfilter.org/documentation/
[17] NVD, 2006. *National Vulnerability Database*. Retrieved April 14, 2006, http://nvd.nist.gov/
[18] NVD-Severity, 2006. *National Vulnerability Database Severity Ranking*. Retrieved April 14, 2006, http://nvd.nist.gov/cvss.cfm
[19] OSVDB, 2006. *The Open Source Vulnerability Database*. Retrieved April 14, 2006, http://www.osvdb.org/
[20] Ou, X., Govindavajhala, S., Appel, A.W., 2005. MulVAL: A Logic-based Network Security Analyzer. In *14th Usenix Security Symposium*.
[21] Peltier, T.R., Peltier, J., Blackley, J.A., 2003. *Managing a Network Vulnerability Assessment*. Auerbach Publications.
[22] Rieke, R., 2004. Tool based formal Modelling, Analysis and Visualisation of Enterprise Network Vulnerabilities utilising Attack Graph Exploration. In *Proceedings EICAR*.
[23] Ritchey, R. W., Ammann, P., 2000. Using model checking to analyze network vulnerabilities. In *IEEE Symposium on Security and Privacy*.
[24] Rothmaier, G., Krumm, H., 2005. A Framework Based Approach for Formal Modeling and Analysis of Multi-level Attacks in Computer Networks. *LNCS, Vol.3731*.
[25] Sheyner, O., Haines, J., Jha, S., etc., 2002. Automated generation and analysis of attack graphs. In *IEEE Symposium on Security and Privacy*.
[26] Schneier, B., 1999. Attack Trees. *Dr. Dobb's Journal, Vol.12*.
[27] Shepard, B., Matuszek, C., Fraser, C.B., etc., 2005. A Knowledge-based approach to network security: applying Cyc in the domain of network risk assessment. In *The Seventeenth Innovative Applications of AI Conference.*
[28] Singh, S., Lyons, J., Nicol, D.M., 2004. Fast Model-based Penetration Testing. In *Proceedings of the 2004 Winter Simulation Conference*.
[29] Swiler, L., Phillips, C., Ellis, D., Chakerian, S., 2001. Computer-attack graph generation tool. In *Proceedings DISCEX '01*.

Igor Kotenko graduated with honors from St.Petersburg Academy of Space Engineering and St.Petersburg Signal Academy. He obtained the Ph.D. degree (1990) and the National degree of Doctor of Engineering Science (1999). He is Professor of computer science and a head of computer security research group in St.Petersburg Institute for Informatics and Automation. Author of more than 300 scientific works.

Mikhail Stepashkin graduated with honors from St.Petersburg State University (2002). He is research fellow of computer security research group in St.Petersburg Institute for Informatics and Automation. Author of more than 20 scientific works.