Attack Graph based Evaluation of Network Security

Igor Kotenko and Mikhail Stepashkin

SPIIRAS, 39, 14 Liniya, St.-Petersburg, 199178, Russia

{ivkote, stepashkin}@comsec.spb.ru

Abstract. The perspective directions in evaluating network security are simulating possible malefactor's actions, building the representation of these actions as attack graphs (trees, nets), the subsequent checking of various properties of these graphs, and determining security metrics which can explain possible ways to increase security level. The paper suggests 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 approach is intended for using both at design and exploitation stages of computer networks. The implemented software system is described, and the examples of experiments for analysis of network security level are considered.

Keywords: Network security, Vulnerability Assessment, Risk Assessment, Security Metrics, Network attacks

1 Introduction

The increase of networks and security mechanisms complexity, vulnerabilities and potential operation errors as well as malefactors' possibilities causes the necessity to develop and use powerful *automated security analysis techniques*. These techniques should allow revealing possible assault actions, determining vulnerabilities, critical network resources and security bottlenecks, and finding out and correcting errors in network configurations and security policies.

At *design stages*, the different approaches to security analysis can be used, for example, based on qualitative and quantitative risk analysis. Approaches based on building the representation of malefactor's actions in the form of attack trees or attack graphs, the subsequent checking of various properties of these trees or graphs on the basis of usage of different methods (for example, model checking), and determining various security metrics are the perspective directions in evaluating security level of large-scaled networks. 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 methods (for obtaining appropriate data about network configuration and security policy), procedures of attack graph construction, and automatic reasoning allows solving partially these two problems.

The paper is devoted to creating the architecture, models and prototypes of security analysis system (SAS) based on construction of attack graphs and computation of different security metrics on the basis of combination of qualitative risk analysis techniques. SAS is based on the following functions: simulating malefactor's activity; building possible assault actions graph; analyzing malefactors' actions from different network points and directed to implementing various security threats; revealing vulnerabilities and security "weak places" (the most critical computer network components); calculating different security metrics and evaluating general security level; comparison of retrieved metrics and user requirements and elaboration of recommendations on security increase. The work is organized in the following way. *Section 2* is an overview of relevant works and the suggested approach peculiarities. *Section 3* represents the model of attack scenarios and the common attack graph generated. *Section 4* specifies security metrics and main phases of evaluating a general security level. *Section 5* emphasizes the approach complexity problems and solutions. *Section 6* considers the generalized architecture of SAS. *Section 7* describes the examples of experiments fulfilled with SAS. *Conclusion* surveys the results and further research.

2 Related Work and the Approach Peculiarities

There are a lot of works which consider *various approaches to security analysis*.

Alberts and Dorofee [1] as well as Chapman and Ward [2] described different risk analysis techniques for estimating security level. Ritchey and Ammann [14] proposed model checking technique for network vulnerability analysis. Jha et al. [6] suggested the technique of attack graph evaluation based on model checking and probabilistic analysis. Sheyner et al. [16] presented algorithms for generating scenario graphs based on symbolic and explicit-state model checking. These algorithms ensure producing counterexamples for determining safety and liveness properties. Rothmaier and Krumm [15] suggested an approach for analyzing different attack scenarios based on a high-level specification language, a translation from this language to constructs of model checker, applying optimization techniques and model checking for automated attack scenario analysis.

Lye and Wing [7] suggested the security evaluation method based on game theory. 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 by Singh et al. in [17] is intended for performing penetration testing of formal models of networked systems for estimating security metrics. Swiler et al. [18] proposed an approach for construction of attack graph which can identify the attack paths with the highest probability of success. Hariri [5] described global metrics which can be used to analyze and proactively manage the effects of complex network faults and attacks, and recover accordingly. Rieke [13] offered a methodology and a tool for vulnerability analysis which can automatically compute attack paths and verify some security properties. Dantu et al. [4] proposed an approach to estimate the risk level of critical network resources using behavior based attack graphs and Bayesian technique. Ou et al. [12] suggested a logic programming approach to automatically fulfill network vul-

nerability analysis. Noel and Jajodia [8] considered the common approach, attack graph visualization techniques and the tool for topological vulnerability analysis. Ning et al. [10] suggested different techniques to construct high-level attack scenarios.

The paper suggests *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 differences of offered approach from examined ones consist in *the way of modeling assault actions* (we use a multi-level model of attack scenarios) and applying constructed attack graphs (for different locations of malefactors) *to determine a family of security metrics and to evaluate different security properties*. While the first feature has been taken into account partly in previous works, the second one mainly has not been considered. The third peculiarity of offered approach is that it can be used at different stages of computer network life cycle, including design and exploitation stages. At design stage, the Security Analysis System (SAS) founded on this approach should use the given specifications of computer network and security policy. At exploitation stage, it interacts with a real computer network getting the necessary configuration and policy data in passive mode. The results of security analysis are vulnerabilities detected, attacks routes (graphs), network bottlenecks, security metrics, which can be used for general security level evaluation of network and its components. Obtained results allow producing the valid recommendations for eliminating detected vulnerabilities and bottlenecks, as well as strengthening the network security level.

3 Attack Scenarios and Generalized Attack Graph

Generalized attack scenario model is hierarchical and contains three levels: integrated level, script level and action level. The *integrated level* determines high-level purposes of the security analysis directed to main security threats realization and analyzed objects (particular hosts, network fragments or the whole network). Integrated level allows coordinating of several scenarios. These scenarios may be performed by both one malefactor and malefactors group. The *script level* takes into account malefactor's skill and initial knowledge about network, defines attack object and purpose (for example, "host OS determining", "denial of service", etc.). Script level contains a set of script stages and substages. The main stages are reconnaissance, penetration (initial access to the host), privileges escalation, threat realization, traces hiding, backdoors creation. The *action level* describes low-level malefactor's actions and exploits.

The algorithm of generating the common attack graph is intended for building the 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 is based on the action sequence set in the attack scenarios model: actions which are intended for malefactor's movement from one host onto another; reconnaissance actions for detection of "live" hosts; reconnaissance actions for detected hosts; attack actions based on vulnerabilities and actions of ordinary users.

All *objects of general attack graph* are divided into two groups: base (elementary) objects and combined objects. *Base objects* define the graph vertexes. They are linked by edges for forming the 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. Objects of the types "route", "threat" and "graph" are *combined objects*. *Route* of attack 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. Classification of attack actions allows differentiating threats as *primary threats* (confidentiality, integrity and availability violation) and *additional threats* (gaining information about host or network, gaining privileges of local user or administrator).

4 Security Level Evaluation

Determining each security metric and the general security level of analyzed network can be realized in different ways. We use two approaches for security level evaluation: Qualitative express assessment on basis of qualitative methodologies of risk analysis; Quantitative computation of network security level (on basis of Bayesian networks, possibility theory and fuzzy sets). This paper presents the first approach.

The set of security metrics was constructed on basis of general attack graph. Security metrics describe security of both base objects and complex objects of general attack graph. *Examples of security metrics* are as follows: (1) *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.); (2) *Metrics of hosts* (Criticality level, etc.); (3) *Metrics of attack actions* (Criticality level; Damage level; Access complexity; Base Score; Confidentiality Impact; Availability Impact; Access Complexity, etc.); (4) *Metrics of attack routes* (Route length expressed in vulnerable hosts; Route average Base Score; Maximum Access Complexity; Damage level of route; Maximum damage level of route, etc.); (5) *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.); (6) *Metrics of common attack graph* (Quantity of different vulnerable hosts of graph; Quantity and set of different attack actions, Average Base Score of all different attack actions, Quantity of routes leading to confidentiality, integrity, availability violations, Quantity of treats leading to confidentiality, integrity, availability violations, Integral security metric "Security level", etc.).

Some security metrics are calculated on basis of standard *Common Vulnerability Scoring System* [3]. CVSS metrics are divided into three main groups: *Base indexes* define *criticality* of vulnerability (attack action realizing given vulnerability); *Temporal indexes – urgency* of the vulnerability at the given point of time; *Environmental indexes* should be used by organizations for priorities arrangement at time of generating plans of vulnerabilities elimination.

The offered approach of qualitative express assessment of network security level contains the following stages: (1) Calculating the criticality level of hosts (*Criticality(h)*, $\forall h \in [1,N_H]$, N_H − hosts amount) using three-level scale (*High, Medium, Low*); (2) Estimating the criticality level of attack actions (*Severity(a)*, $\forall a \in [1,N_A]$, N_A − actions amount) using the CVSS algorithm of action criticality assessment; (3) Calculating the damage level of attack actions (*Mortality(a,h)*, $\forall h \in [1,N_H]$, $\forall a \in [1,N_A]$) taking into account criticality levels of actions and hosts; (4) Determining the damage level of all threats (*Mortality(T)=Mortality(a_T,h_T)*, $\forall T \in [1,N_T]$, where N_T − threats amount, a_T − latest attack action directed on the host h_T for threat *T*); (5) Calculating the metrics of "Access complexity" for all attack actions (*AccessComplexity(a)*, $\forall a \in [1,N_A]$), all routes (*AccessComplexity(S)*, $\forall S \in [1,N_S]$, N_S − routes amount), and all threats (*AccessComplexity(T)*, $\forall T \in [1,N_T]$); (6) Estimating the admissibility of threats realization (*Realization(T)*, $\forall T \in [1,N_T]$) using the metrics of "Access complexity"; (7) Network security level (*SecurityLevel*) evaluation using the estimations of threats admissibility and damage level caused by threats realization. Four security levels are used: Green, Yellow, Orange and Red.

5 Complexity Problems and Solutions

The complexity of generating the attack graph is determined by the quantity of malefactor's actions. The given quantity depends mainly on the quantity of hosts in analyzed network (N_H) and the quantity of used vulnerabilities (exploits) from the internal database of vulnerabilities (N_V).

Let us consider the test network which includes *n* hosts. Each of these hosts has vulnerabilities allowing malefactor to gain a root privileges on the host and to move to the compromised host to attack others. During scanning process the malefactor can reveal all *n* hosts and realize all attack actions and movements to the captured hosts. Therefore the following formula can be used to approximately compute the complexity of generating the attack graph:

$$F\left(N_H,N_V\right) = N_H N_V F\left(N_H - 1, N_V\right) = N_H N_V \left(N_H - 1\right) N_V F\left(N_H - 2, N_V\right) = N_V{}^{N_H} N_H!$$

The complexity of attack graph analysis is determined by the complexity of attack graph depth-first traversal and is equal to $O(V + E)$, where V − graph vertexes; E − graph edges.

This discussion shows that the given approach faces a combinatorial explosion in complexity. So, it can be applied with success to small networks, but cannot be used without corresponding modification for large scaled networks.

The following *approaches for reducing the complexity* of generating the attack graph are suggested:

1. Splitting the network into fragments, parallel computing for each fragment with subsequent combination of results.

2. Aggregation and abstraction of representations of attack actions and (or) network objects:

- ▪ *Using attack actions*. The type of aggregation or abstraction is selected according to offered generalized attack scenario model. The examples of attack actions aggregation objects are as follows: main stages (reconnaissance, penetration, privileges escalation, etc.), attack class (buffer overflow, DoS, etc.), attack subclass (for example, specific type of buffer overflow). Thus, at attack graph construction it is possible to merge a set of vertexes representing actions of one type in one vertex (for example, actions "ping", "get banners", "get services" can be merged into a set of actions named "reconnaissance").
- ▪ *Using network objects*. The combination of several hosts of network segment, network segment, the combination of network segments can be aggregated network objects. Thus, the analyzed network can be splitted into aggregated objects which are represented as one object. Such object can be characterized by a set of parameters inherent to all its components (hosts, routers, network switches). The list of operating systems types, the list of network services, etc. can be used as elements of such set of parameters. The reduction of attack graph complexity in such case is achieved because instead of a big set of vertexes (where the hosts are targets of attack actions), the significantly smaller set of vertexes (where the aggregated objects are targets of attack actions) are displayed on the graph.
- ▪ Combining various approaches to aggregation and abstraction.

3. Combining parallel computing, aggregation and abstracting.

In further work the development of various algorithms for generating the attack graph is supposed. These algorithms will differ by accuracy and complexity.

6   Security Analysis System Architecture

The architecture of Security Analysis System (SAS) is depicted in fig.1.

User interface provides the user with ability to control all components of SAS, set the input data, inspect reports, etc. *Generator of network and security policy internal model* converts the information about network configuration and security policy into internal representation. The input information is received from Information collector (at exploitation stage) or from specifications expressed in System Description Language (SDL) and Security Policy Language (SPL) (at design stage). These specifications should describe network components and security with the necessary degree of detail – the used software (in the form of names and versions) should be set. *Data controller* is used for detection of incorrect or undefined data.

The network configuration and security policy database contains information on network configuration and security policy rules (this part is used for generating attack action results) as well as malefactor's view of network and security policy (it is generated as the results of attack actions). It is possible to plan the sequence of malefactor's actions on basis of this database (for example, if malefactor has user privileges and needs to read a file F, and according to security policy only local administrators can read this file, then malefactor must do actions to gain the administrator privileges).
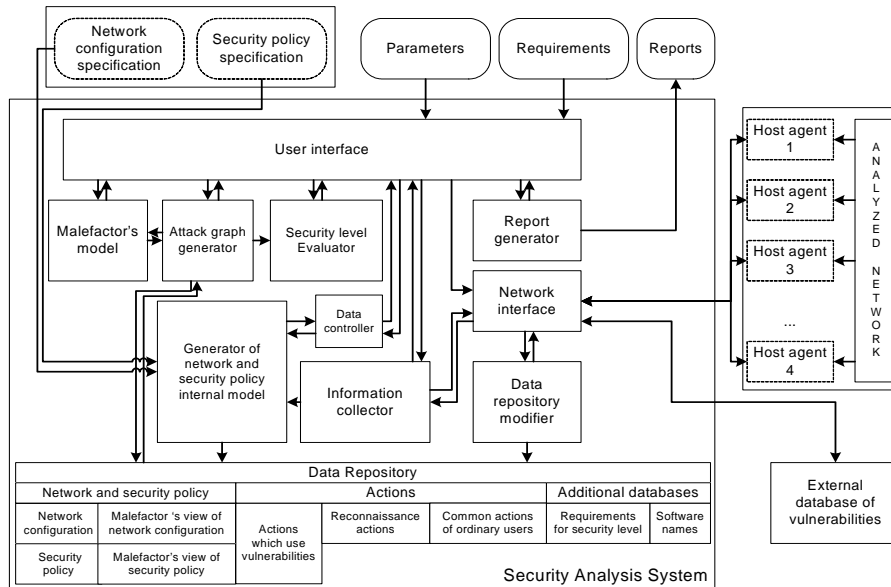
Fig. 1. Architecture of security analysis system

Actions database includes the rules of "IF-THEN" type determining different malefactor's operations. IF-part of each rule contains action goal and (or) conditions. The condition is compared with the data from network configuration and security policy database. THEN-part contains the name of action which can be applied and (or) the link on exploit and post-condition which determines a change of network state (impact on an attacked object). *Actions which use vulnerabilities* (unlike other bases of the given group) are constructed automatically on basis of external vulnerabilities database OSVDB [11]. *Common actions* contain actions which are executed according to user's privileges (for example, "file read", "file copy", "file delete", etc.). Databases of reconnaissance and common actions are created by experts.

DB of requirements contains the 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 names* is used by *Data controller* for detection of errors in the specifications of computer network (e.g. when user writes "Orakle" instead of "Oracle") and for generating recommendations on using software tools. In case of detecting discrepancy the conflict is resolved by choosing the correct software name from the list suggested.

Data repository modifier downloads the open vulnerability databases (we use OSVDB [11]) and translates them into actions database. *Attack graph generator* builds attack graph by modeling malefactor's actions using information about network configuration, security policy and available actions from data repository. This module sets up security metrics of elementary objects. On basis of these metrics *Security level evaluator* calculates the metrics of combined objects, evaluates security level, compares obtained results with requirements, finds bottlenecks, and generates recommendations on strengthening security level. *Malefactor's model* determines

malefactor's skill level, his initial position and knowledge on network. Malefactor's skill level determines the attack strategy and the set of actions used by malefactor.

Hosts agents serve for passive data gathering. On the basis of these data the network and security policy internal model is formed at exploitation stage. For example, the agents can make the analysis of configuration files of operating system and other software components. *Network interface* provides interaction with external environment (sending requests to external vulnerabilities databases for updates and communicating with agents). *Information collector* interacts with *host agents* and receives from them information about network devices and settings of software components.

7   Experiments

Fig. 2 shows the structure of test computer network used in experiments.
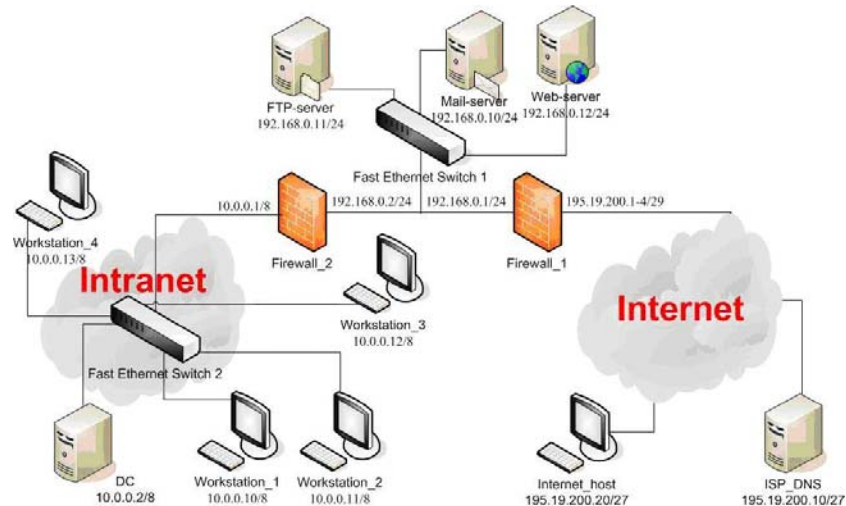

Fig. 2. Structure of test network

When user is working with SAS, he needs to perform the following operations: (1) Loading network configuration and security policy; (2) Setting security requirements; (3) Choosing the high-level purpose of security analysis process; (4) Setting the values of parameters of security analysis; (5) Security analysis process; (6) Modification of network configuration and security policy (if needed).

Network specification in specialized language (SDL) allows defining network topology, information about operating systems of the network hosts, TCP/IP protocol stack settings, services, etc. Security policy description in the specialized language (PDL) allows specifying the network traffic filtration rules for boundary hosts, confidence relations, authentication and authorization rules, etc. Network traffic filtration rules are specified as table collection [9]. Port forwarding rules are specified by tables

PREROUTING and FORWARD (all incoming connections not described in the table are forbidden). Tab. 1 contains notation for the main elements of attack graph.

Tab. 1. Attack graph elements notation

Malefactor action	Malefactor action (severity and access complexity of action are in brackets)	Malefactor location PRIVILEGES	Malefactor's location and privileges
Final malefactor action	Final malefactor's action	Attacked host (criticality)	Attacked host and its criti-cality level (in brackets)

Let us consider how SAS works at design stage. Let input data for security analysis are as follows: (1) port forwarding rules for host Firewall_1 are in Tab. 2; (3) Firewall_1 and Firewall_2 trust to all DMZ hosts; (4) malefactor is located at external network at host Malefactor, and has administrator privileges; (5) security analysis purpose is to analyze all kinds of threats (integrity, availability, confidentiality violation); (6) security analysis task is to analyze all DMZ and LAN hosts; (7) requirements to analyzed network: the given network should have security level better than Orange. Fig. 3 shows general attack graph for example 1.

Let us consider shortly the process of building the attack graph. At first malefactor is located at the "Malefactor" host and performs 'Ping Hosts" attack. The attack allows him to determine live hosts. Malefactor receives data about four hosts (FTP_server, Web_server, Mail_server and Firewall_1) with IP 195.19.200.1-4 (actually this is only Firewall_1, but malefactor does not know it). Then malefactor analyzes every host separately. Let us consider analysis of the host with IP 195.19.200.2. Four reconnaissance scripts are generated: (1) "Nmap serv" (open port scanning); (2) "Nmap OS" (OS type and version determining); (3) "Nmap serv"+"Banner" (open ports scanning and services identifying); (4) "Nmap serv" +"Banner" +"Nmap OS". After every reconnaissance script realization, malefactor checks if host information satisfies the conditions of actions that use vulnerabilities.

Tab. 2. Port forwarding rules for host Firewall_1

Comment	Destination		Forward to…	
	IP	Port	IP	Port
Web_server	195.19.200.3	80	192.168.0.12	80
FTP_server	195.19.200.2	21	192.168.0.11	21
MAIL_server POP3	195.19.200.4	110	192.168.0.10	110
MAIL_server SMTP	195.19.200.4	25	192.168.0.10	25
MAIL_server RDC	195.19.200.4	3389	192.168.0.10	3389

The result of the "Nmap serv" action for host with IP 195.19.200.2 is open port list for FTP_server host (there is one open port – 21), since in accordance to port forwarding table incoming connections to IP 195.19.200.2:21 (where 21 is destination port) are forwarded into 192.168.0.11:21. Thus malefactor determines availability of one open port and he can attack it with "SYN flood" assault action. After performing second reconnaissance script ("Nmap OS"), malefactor receives information that does not allow to perform any assault action. After performing third reconnaissance script, malefactor can use three assault actions: (1) password searching ("FTP dict"); (2) denial of service attack ("ServU-MKD"); (3) privileges escalating ("ServU-

MDTM"). First two actions are final. Third action allows malefactor to get administrator privileges and all FTP_server host information. Administrator privileges allows malefactor to go into the host and to attack other hosts.
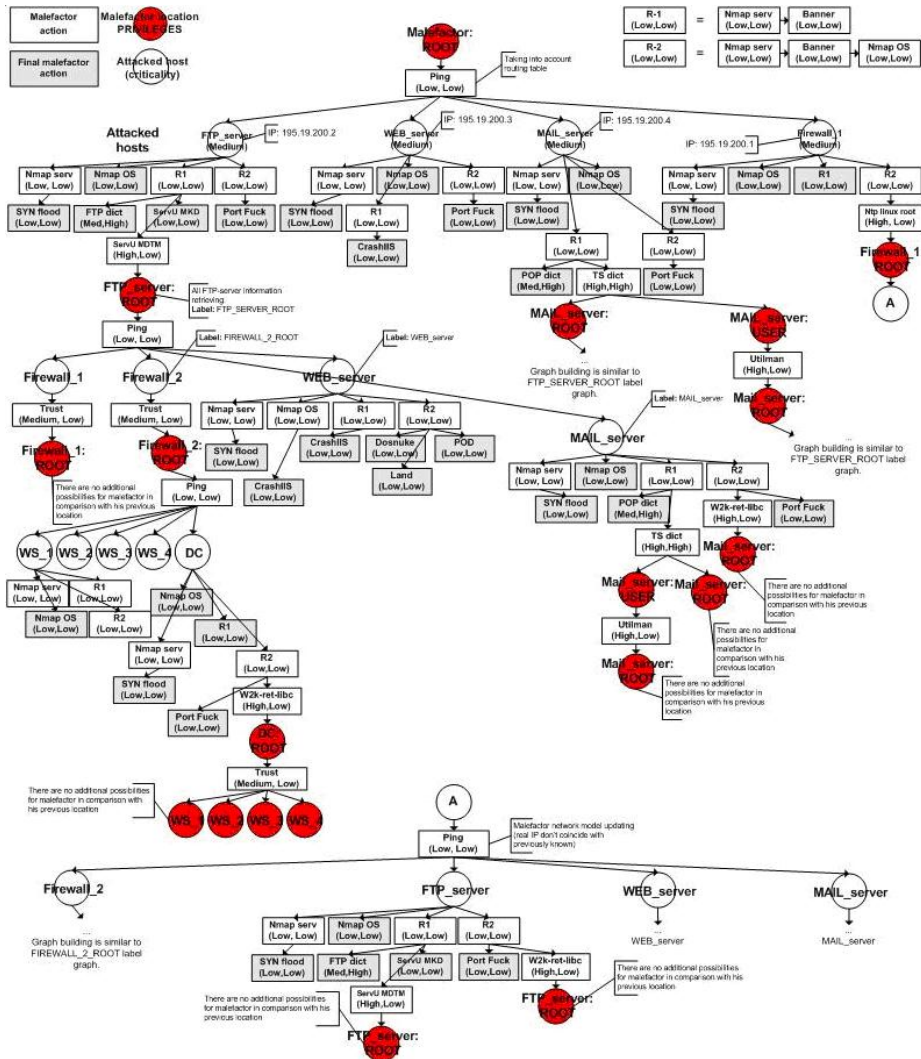

Fig. 3. General attack graph

Malefactor finds out that real FTP_server host IP (192.168.0.11) does not coincide with 195.19.200.2. Therefore, there is port forwarding in the network, and malefactor is at other subnetwork. This fact is critical to malefactor when he decides to change his location to the captured FTP_server host. Malefactor changes location and performs "Ping Hosts" action. He finds out that there are four hosts and consequently

analyzes them with above-mentioned scheme. In addition he can get administrator privileges at hosts Firewall_1 and Firewall_2 because they trust to FTP_server.

Some of the security analysis results are as follows: Network bottlenecks – Firewall_1, FTP_server, … ; Critical vulnerabilities – NTP_LINUX_ROOT, Serv-U MDTM, … ; Graph has routes and threats with high mortality (for example, route Malefactor-Ping-FTP_server(Nobody)-Nmap serv-Banner-ServU MDTM-FTP_server(Root) …); *SecurityLevel=Red*. The computer network security level does not satisfy user's requirements (better than Orange) and requires immediate actions for eliminating of the revealed software vulnerabilities and security policy bottlenecks.

8 Conclusion

The paper offered the approach and software tool for vulnerability analysis and security level assessment of computer networks, intended for implementation at various stages of a life cycle of computer networks. Offered approach is based on construction of attack graphs and computation of different security metrics.

The suggested approach possesses the following *peculiarities*:
- Usage for security level evaluation of integrated family of different models based on expert knowledge, including malefactor's models, multilevel models of attack scenarios, building common attack graph, security metrics computation and security level evaluation;
- Taking into account diversity of malefactor's positions, intentions and experience levels;
- 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;
- Taking into account not only attack actions (which use vulnerabilities), but the common actions of legitimate users and reconnaissance actions which can be realized by malefactor when he gains certain privileges on compromised hosts;
- Possibility of investigating various threats for different network resources;
- Possibility of detection of "weak places" (for example, the hosts responsible for a lot of attack routes and the highest quantity of vulnerabilities);
- 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;
- Usage for attack graph construction of updated vulnerabilities databases (the Open Source Vulnerability Database (OSVDB) [11] is used);
- The "CVSS. Common Vulnerability Scoring System" [3] approach is used for computation of a part of primary security metrics.

The future research will be devoted to improving the models of computer attacks, the algorithms of attack graph generation and security level evaluation differing by accuracy and complexity, and experimental assessment of offered approach.

References

1. Alberts, C., Dorofee, A.: Managing Information Security Risks: The OCTAVE Approach. Addison Wesley (2002)
2. Chapman, C., Ward, S.: Project Risk Management: processes, techniques and insights. Chichester, John Wiley (2003)
3. CVSS. Common Vulnerability Scoring System. URL: http://www.first.org/cvss/
4. Dantu, R., Loper, K., Kolan P.: Risk Management using Behavior based Attack Graphs. International Conference on Information Technology: Coding and Computing (2004)
5. Hariri, S., Qu, G., Dharmagadda, T., Ramkishore, M., Raghavendra, C. S.: Impact Analysis of Faults and Attacks in Large-Scale Networks. IEEE Security&Privacy, September/October (2003)
6. Jha, S., Sheyner, O., Wing, J.: Minimization and reliability analysis of attack graphs. Technical Report CMU-CS-02-109, Carnegie Mellon University (2002)
7. Lye, K., Wing, J.: Game Strategies in Network Security. International Journal of Information Security, February (2005)
8. Noel, S., Jajodia, S.: Understanding complex network attack graphs through clustered adjacency matrices. Proc. 21st Annual Computer Security Conference (ACSAC) (2005)
9. Netfilter/iptables documentation. URL: http://www.netfilter.org/documentation/
10. Ning, P., Cui, Y., Reeves, D, Xu, D.: Tools and Techniques for Analyzing Intrusion Alerts. ACM Transactions on Information and System Security, Vol. 7, No. 2 (2004)
12. OSVDB: The Open Source Vulnerability Database. URL: http://www.osvdb.org/
13. Ou, X., Govindavajhala, S., Appel, A.W. : MulVAL: A Logic-based Network Security Analyzer. 14th Usenix Security Symposium (2005)
14. Rieke, R.: Tool based formal Modelling, Analysis and Visualisation of Enterprise Network Vulnerabilities utilising Attack Graph Exploration. EICAR 2004 (2004)
15. Ritchey, R. W., Ammann, P.: Using model checking to analyze network vulnerabilities. Proceedings of IEEE Computer Society Symposium on Security and Privacy (2000)
16. Rothmaier, G., Krumm, H.: A Framework Based Approach for Formal Modeling and Analysis of Multi-level Attacks in Computer Networks. LNCS, Vol.3731 (2005)
17. Sheyner, O., Haines, J., Jha, S., Lippmann, R., Wing, J.M.: Automated generation and analysis of attack graphs. Proc. of the IEEE Symposium on Security and Privacy (2002)
18. Singh, S., Lyons, J., Nicol, D.M.: Fast Model-based Penetration Testing. Proceedings of the 2004 Winter Simulation Conference (2004)
19. Swiler, L., Phillips, C., Ellis, D., Chakerian, S.: Computer-attack graph generation tool. DISCEX '01 (2001)