MANAGING EVOLVING SECURITY SITUATIONS

Jyri Kivimaa
Cooperative Cyber Defence
Centre of Excellence
Tallinn, Estonia

Andres Ojamaa
Institute of Cybernetics at
Tallinn University of Technology
Tallinn, Estonia

Enn Tyugu
Cooperative Cyber Defence
Centre of Excellence
Tallinn, Estonia

ABSTRACT

A method is described that takes into account the investments done in the security and/or achieved security confidence in planning new security measures. The method uses new integral security metrics and the well-known graded security model. A precondition for the application of this method is the availability of expert knowledge or statistical data for the model in use that describes a class of situations where the analyzed security situation belongs to. For a number of situations at present, this information has been extracted from standards of graded security. For specific military communications applications the data must be collected from a log analysis of characteristic attacks and security reports, as well as by the traditional knowledge acquisition means.

1. INTRODUCTION

The security situation in cyber space is changing rapidly. This requires continuous analysis of security situations and continuous security management: selection of security measures, planning of investments for security measures groups. Our goal is to provide a method for planning security measures not only for a fixed time point, but to do this for a longer time period, possibly, investing into the security gradually. This paper presents a method that is an extension of the Pareto-optimal security situation analysis implemented in an expert system [4]. It takes into account the legacy systems and security levels achieved by means of former investments. This enables one to plan the usage of resources considering evolving security situations over a longer time period.

Comprehensive security planning is a complex task. This can be seen from the complexity of standards and requirements like Common Criteria [7] or ISKE [1]. Standards prescribe minimal required measures, and usually do not include economic parameters—the costs of implementing the security measures. A detailed cost-benefit analysis of cyber security [2] may require months. An alternative approach is to manage security on the basis of security requirements. It is efficient, if reasonably good expert knowledge of security requirements and goals is available. We have taken this approach.

A well-known graded security methodology [6, 8] is based on a comprehensive but coarse grained model, and provides a way of planning security and calculating costs. In our paper [4] we have shown how to use the graded security model for finding optimal solutions depending on the given security situation. However, a description of a situation there reflects neither the investments already done into security nor the levels of security already achieved. Based on the application of a discrete dynamic programming method described in [5], one can solve rather complex security optimization problems on ordinary PCs and laptops. This enabled us to extend the optimization method for longer time intervals, solving the optimization problem stepwise.

This paper is organized as follows. In the next section we present briefly the graded security method that provides the functional dependencies needed for calculations. A separate section (Section 3) is devoted to the discussion of the integral security metrics needed for comparing the solutions. These metrics were introduced for the first time in [4]. The following Section 4 includes a brief description of the software used for making calculations. Section 5 includes a discussion of the influence of the legacy security on new security solutions. It presents formulas needed for planning evolving security measures. Section 6 includes descriptions of solvable legacy security problems and some solutions.

2. GRADED SECURITY MODEL

Here we briefly introduce variables and functions used in the graded security model. The overall security of a system is described by a *security class*. It shows how the security goals (confidentiality, integrity, availability, ...) are satisfied. It is determined by assigning *security levels* to *security goals*, and is denoted by a respective tuple of pairs, e.g., `C2I1A1M2` for the system that has the second level of confidentiality *C*, the first level of integrity *I* etc.

To achieve the security goals, proper security measures have to be taken. There may be a large number (hundreds) of measures. It is reasonable to group them into security measures groups $g_1, g_2, \ldots, g_n$. The grouping should be done in such a way that measures of one and the same group will always be used for achieving one and the same level of security. One uses a function f that produces a set of required security measures $f(l, g)$ for a given security measures group g and a security level l of the group. A security class determines the required security level for each group of security measures. Let us denote by s a respective function that produces a security level $s(K, g)$ for a group g when the security class is K. An *abstract security profile* is an assignment of security levels $(0, 1, 2,$ or $3)$ to each group of security measures. This can be expressed by the tuple $p = (s(K, g_1), s(K, g_2), \ldots, s(K, g_n))$, where p denotes the abstract security profile and the elements of the tuple p are indexed and appear in the tuple in the same order as the groups of security measures $g_1, g_2, \ldots, g_n$ have been indexed. Knowing the cost function $h(l, g)$ that gives the costs r required for implementing security measures of a group g for a level l, one can calculate the costs of implementing a given abstract security profile:

$$costs(p) = \sum_{i=1}^{n} h(l_i, g_i) \,,$$

where $p = (l_1, l_2, \ldots, l_n)$.

The goal is to keep the value $costs(p)$ as low as possible, guaranteeing a required security. It is assumed that by applying security measures, one achieves security goals with some confidence. The security confidence c of a group g that satisfies the security level l is given by a function $e(l, g)$ and it is a numeric value between 0 and 100 for each group of security measures.

3. INTEGRAL SECURITY METRICS

The graded security model uses coarse-grained metrics differentiating three or four security levels for each security goal. To compare security situations in general, one needs a more precise metric that expresses the quality of a security situation by one numeric value. It is reasonable to take into account influences of all security measures on the overall security of the system. The simplest choice would be to calculate the mean security confidence of all groups. However, the influence of groups on the overall security is different. Therefore, the best solution would be to use partial derivatives of the security measure depending on the security confidences of the groups. These derivatives could be used as coefficients of the security confidences when calculating their mean value. Unfortunately, these derivatives are hard to determine. Instead of the derivatives, one can use empirically found weights of the security confidences.

We have introduced a security metric in [4] that evaluates a security situation on the basis of security confidences provided by the security measures groups. We describe the overall security of a system by means of an integrated security metric S that is a *weighted mean security confidence*, called also *integral security confidence*:

$$S = \sum_{i=1}^{n} a_i c_i \,,$$

where c_i is security confidence of the i-th security measures group, a_i is the weight of the i-th group, and

$$\sum_{i=1}^{n} a_i = 1 \,.$$

Using a linear combination of security confidences of measures groups is reasonable as long as a security situation does not change too rapidly. (The gradient of the integral security confidence in the space of confidences of security measures groups can be estimated in such a case and its components used as the required coefficients.)

4. VISUALIZING A SECURITY SITUATION

In this section we very briefly present a tool for making calculations on graded security models. This is a software package with a visual language for specifying security situations and problems. The package has been developed on the basis of the visual software development environment CoCoViLa [3], and it has been described in more detail in [4] and [5]. The package includes expert knowledge for a particular class of security situations. This expert knowledge is usable only for demonstrating the method—it has been taken mainly from [9].

Fig. 1 shows a specification of a security planning problem. The toolbar has buttons for defining components that will constitute a specification. It includes two buttons for defining security measures groups: one for groups with standard values of parameters, and another for groups with parameters defined as inputs. It includes also buttons for defining a security class, for selecting an optimization method and for defining a graphical output. All these components are also visible on the scheme in Fig. 1. This scheme is a specification of a problem for finding a Pareto-optimal solution for a security class `C2I1A1M2` and specific parameters given for two security measures groups: *User training* and *Encryption*. Each security measures group has a pop-up window. This window is shown for the *Encryption* group in Fig. 1.

We use this package for all calculations on the graded security model. The package is extended with new components for solving the legacy security problems described in the following sections, see Sections 5 and 6.

5. LEGACY SECURITY INFLUENCE

The widely used graded security model is based on the assumption that former investments into the security and already existing security situation do not influence the outcome of the investments planned. The former investments are sometimes included in the total amount of investments calculated. These investments may be included with a factor less than one, but this is still a rough approximation. We propose here an approach that more precisely takes into account the already achieved security.

Let us fix a security measures group and consider only one group of security measures here. Then we can use a simplified form of the functions h and e for calculating costs r and security confidence c—without showing explicitly the security measures group:

$$\begin{aligned} r &= h(l)\,, \\ c &= e(l)\,. \end{aligned}$$

We use also a function for calculating security level l for invested costs, which is an inverse function of h:

$$l = h^{-1}(r)\,.$$

We need data for already existing security:

l' – existing level of security,
c' – existing security confidence.

To continue analysis of security investments, we need a function H that calculates the needed additional investments r depending on the existing security level l' and the required security level l:

$$r = H(l, l')\,.$$

It may seem that instead of the function H one can use a function h^* that calculates the required resources for increasing security level by Δl, where $\Delta l = l - l'$:

$$r = h^*(\Delta l)\,.$$

It is easy to see that in the case when no investments in the security have been done before, i.e. when $l' = 0$, the function h^* coincides with the already known function h. However, in the case of $\Delta l = 0$ and $l' > 0$ we have to consider the degradation of security as well—the security level will decrease with time. This shows that the usage of h^* instead of H would be quite a rough approximation.

This analysis is valid for all security measures groups. But in the general model, we have to introduce an argument g (group number) in each function considered here. This gives us the functions:

$$\begin{aligned} r &= H(l, l', g)\,, \\ r &= h^*(\Delta l, g)\,. \end{aligned}$$

These functions should be obtained from expert knowledge.

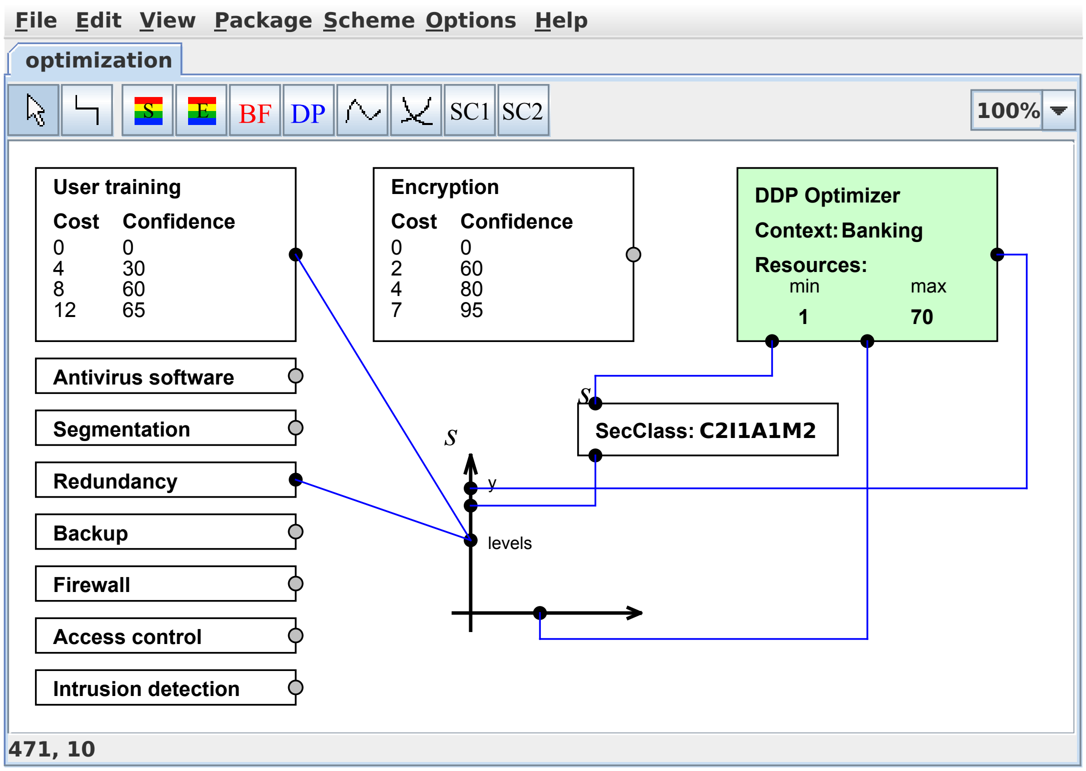
Another approach would be to use security confidence c instead of security level. These variables are bound by the function e in the graded security model:

$$c = e(l)\,.$$

The relation between costs and security confidence is expressed by the formulas:

$$\begin{aligned} r &= h(e^{-1}(c))\,, \text{ and} \\ c &= e(h^{-1}(r))\,. \end{aligned}$$

Knowing the already achieved security confidence, one can ask to calculate additional investments for achieving the new security confidence (or keeping the required confidence level). This requires the knowledge of a new function E that gives the costs r for achieving required



User training

Cost	Confidence
0	0
4	30
8	60
12	65

Encryption

Cost	Confidence
0	0
2	60
4	80
7	95

DDP Optimizer
Context: Banking
Resources:

min	max
1	70

SecClass: C2I1A1M2

Figure 1. Visual specification of a security situation

security confidence c by upgrading the given security confidence c':

$$r = E(c, c')\,.$$

As discussed above, one can sometimes assume that the costs depend only on the difference Δc of security confidences:

$$\Delta c = c - c'\,,$$

and use the function e^* that calculates the costs:

$$r = e^*(\Delta c)\,.$$

Again, in the general model we have to introduce an argument g (group number) in each function considered here. This gives us the functions for calculating costs in the general case:

$$\begin{aligned} r &= E(c, c', g)\,, \\ r &= e^*(\Delta c, g)\,. \end{aligned}$$

Concluding the analysis here we can say that, for taking into account the legacy security measures in calculating resources required for achieving a given security confidence, we need one of the functions H, h^*, E or e^*. It is preferable to use H or E, because these describe the security situation more precisely. In practice, these functions are represented in a tabular form as expert knowledge. One would like to solve an inverse problem—calculate achievable security confidence for given resources. This is done by using one of the inverse functions H^{-1} or E^{-1} representable by the same tables as H and E:

$$\begin{aligned} l &= H^{-1}(r, l', g)\,, \\ c &= E^{-1}(r, c', g)\,. \end{aligned}$$

Let us call the functions H, h^*, E, e^*, H^{-1} and E^{-1} legacy functions.

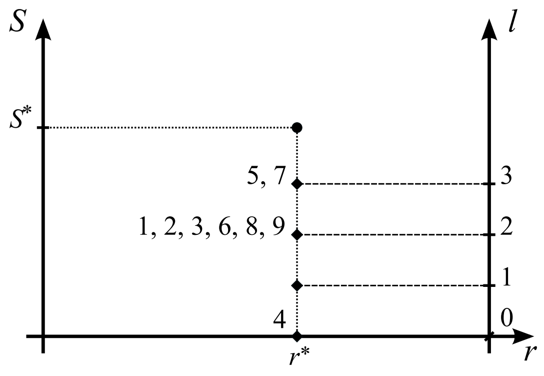
The legacy values of l and r are bound by the functions h and h^{-1} as follows:

$$\begin{aligned} r' &= h(l')\,, \text{ and} \\ l' &= h^{-1}(r')\,. \end{aligned}$$

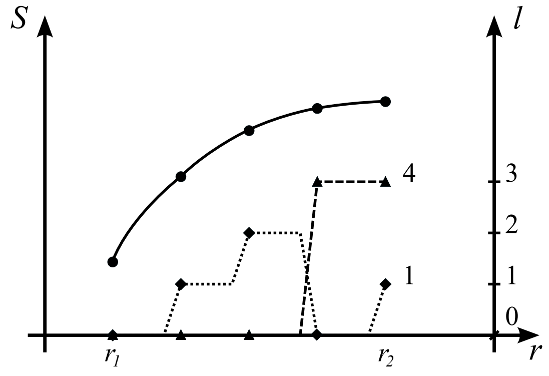
Therefore we can use legacy resources r' instead of l' as inputs of the calculations. We use this in an example in Section 6.

6. OPTIMIZING EVOLVING SECURITY

Security planning can be performed in two different ways. The traditional way is to decide somehow which



(a) Optimal resource assignment for r^*



(b) Pareto-optimal solutions

Figure 2. Solutions of the optimization problem of finding the best assignments of resources to different security measures groups

security levels are required, and to calculate the required resources, using a function H or E. This is an application of the well-known *graded security method* [6]. The security levels are usually prescribed by some standards in this case.

Another way is to solve the *inverse problem*: for given resources find the best assignment of the resources to different security measures groups. This is an optimization problem that can be solved by means of discrete dynamic programming as shown in [5]. The quality of a solution is evaluated by the integral security metric S introduced in [4] and described in Section 3. Fig. 2a shows a solution of the inverse problem: the value of S for given resources r, and also selected security levels of security measures groups. The levels for the groups numbered from 1 to 9 are shown on the right side scale.

Besides the value of S, one may have to consider constraints put on the solution by the security class K, if it is given—all security goals prescribed by K must be satisfied. If priorities are assigned to the security goals, then it is possible to solve a more general problem: find the best possible security solution that satisfies the goal with the highest priority and, if possible, then satisfies also a goal with the next higher priority etc.

Our experiments have shown that the dynamic programming method is fast enough for solving even a more general problem: finding a Pareto-optimal set of security solutions for a given range of resources. Simply speaking, this means that the problem above must be solved for many values of resource r and the result must be plotted as a curve as shown in Fig. 2b.

Fig. 3 shows such a curve for resources from 1 to 70 units. It is obtained by using the expert system described in [4] for the problem specified in Fig. 1. We can see that the security class is `C2I1A1M2` and that two security measures groups (*User training* and *Encryption*) get specific input values for the functions h and e. Other measures groups use the values from the built-in expert system.

In Fig. 3, the lower graphs indicate (on the scale shown on the right) the optimal levels of two measures groups (*Redundancy* and *User training*) corresponding to the given amount of resources. These graphs are not monotonic as can be seen in this example at the resource values 35 and 36. For a more detailed explanation see [5].

Let us consider now the inverse problem considering also the legacy security: given a security class K, resources r, existing security levels l' and a legacy function H^{-1}, find the security solution with the highest value of mean weighted security confidence S that satisfies all security goals of K. This problem may or may not have a solution. Even if it does not have a solution, the problem without the constraint K (without the requirements on security goals) will have a solution. It is interesting to notice that, in the case when the problem has a solution, this solution may be different from the solution obtained without the constraint K.

Fig. 4 shows a solution for both cases: the red curve presents a solution for the problem with a constraint $K =$ `C3I1A1M2`, and the green curve presents a solution for the unrestricted problem. We can see the cases where prescribing K gives worse values of S.

For solving the legacy problems we have extended the expert system by adding the legacy information to the

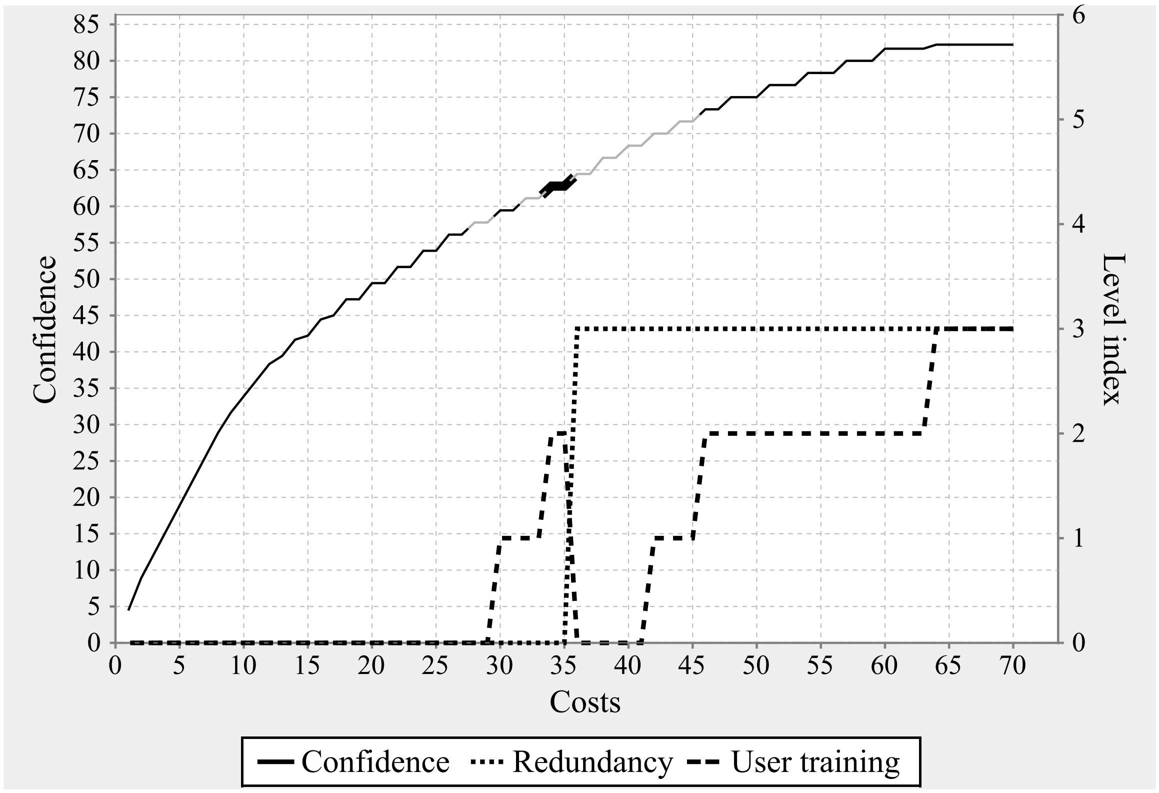


Figure 3. Solution of the problem

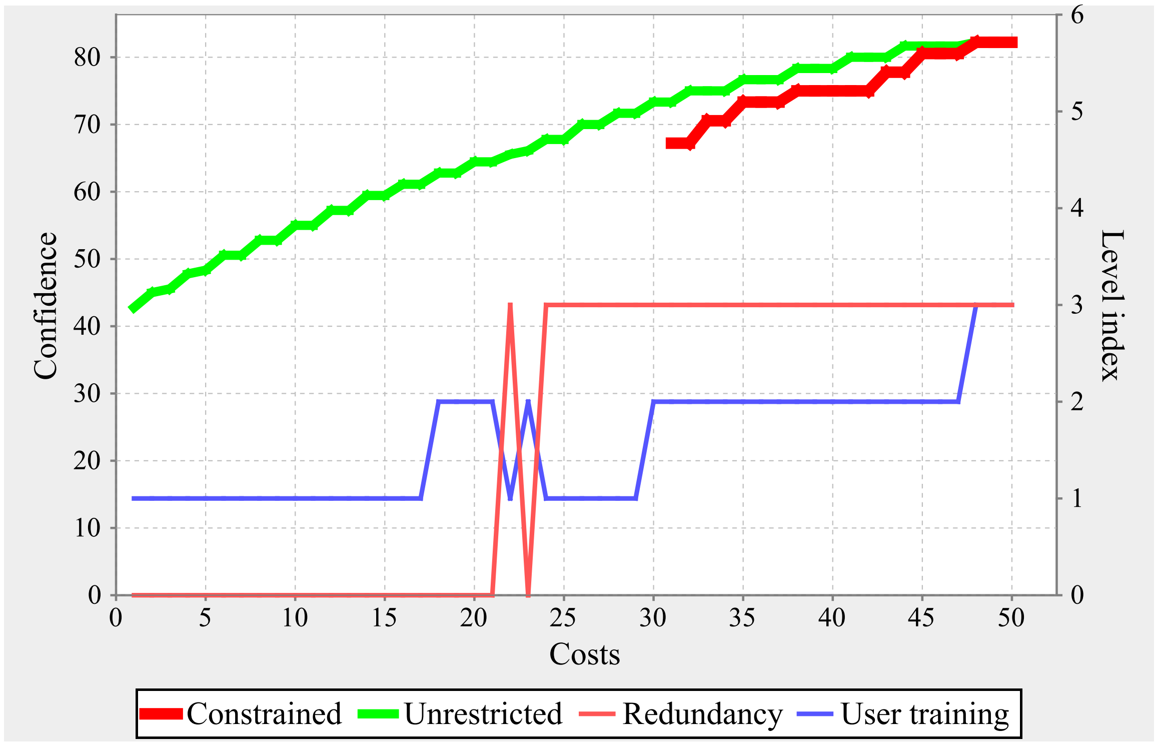


Figure 4. Solutions with and without a constraint

Table 1. Values of legacy resource and decay

g	r'	q
User training	4	0.3
Antivirus software	4	0.6
Segmentation	4	0.2
Redundancy	0	0.3
Backup	4	1.0
Firewall	4	0.5
Access control	4	0.2
Intrusion detection	4	0.5
Encryption	4	0.2

components representing security measures groups, and adding the calculation of the legacy function

$$c = e(h^{-1}(r_0))\,,$$

where $r_0 = r + (1-q)r'$ is an effective resource that takes into account both current resource r and decayed value of the legacy resource r'; q is a decay of a resource, $q < 1$. We have used the values of legacy resource r' and decay q given in Table 1.

Knowing the legacy function, we can plan optimal security measures for a number of time intervals (years) in advance. The values l' of existing security levels must be given as initial data. The values of l' for each following year must be taken equal to the values of l of the previous year. The Pareto-optimal set is a surface in a multidimensional space with coordinates $r, y, l_1, \ldots, l_n$ and S, where y is the year number in this case.

Even if we consider Pareto-optimal solutions only for one year, visualization of the Pareto-optimal set is possible only in a special case when all security levels of all security measures groups are equal. In this case, the Pareto-optimal set is a surface in the three-dimensional space r, S, l, where l is the confidence level of all measures groups, and this can be visualized.

7. CONCLUDING REMARKS

The software developed in the present work for analyzing security situations is easy to use for security experts. The developed experimental tool has a simple graphical interface and a visualization component that supports its usage by security managers of all levels. The experiments have also shown that stability of optimal solutions found by the presented method is good. However, the practical applicability of the software will depend on the availability of good expert data representing the legacy function as well as functional dependencies of the graded security model. The developed software has been designed as an expert system. It supports easy inclusion of new expert knowledge, but expert knowledge acquisition is always a complicated task. For specific military communications applications the data must be collected from a log analysis of characteristic attacks and security reports, as well as by the traditional knowledge acquisition means.

Finally, the contemporary security landscape is dynamic and rapidly changing. This is the main reason for developing agile methods of security situation management. The presented method of managing evolving security situations is one of these.

ACKNOWLEDGMENTS

We thank the Cooperative Cyber Defence Centre of Excellence, the Estonian Defence Forces Training and Development Centre of Communication and Information Systems, and the Estonian Ministry of Defence for the support of this work.

REFERENCES

[1] Estonian Informatics Centre. *Estonian Information Systems Three-Level Security Baseline System – ISKE version 4.01.* http://www.ria.ee/27220 (10 Apr 2009).

[2] L. A. Gordon, M. P. Loeb. *Managing Cybersecurity Resources: A Cost-Benefit Analysis.* McGraw-Hill, 2006.

[3] P. Grigorenko, A. Saabas, E. Tyugu. *Visual tool for generative programming*. ACM SIGSOFT Software Engineering Notes, 2005, 30, 5, 249–252.

[4] J. Kivimaa, A. Ojamaa, E. Tyugu. *Graded security expert system.* CRITIS 2008: Third International Workshop on Critical Information Infrastructure Security, Rome, October 13–15 2008. Springer, LNCS, 2009.

[5] A. Ojamaa, E. Tyugu, J. Kivimaa. *Pareto-optimal situation analysis for selection of security measures.* MILCOM 08: Assuring Mission Success: Unclassified Proceedings, San Diego, November 17–19 2008, 7p.

[6] C. E. Pasterczyk. *A graded approach to ISO 9000 implementation for records managers.* Association of Records Managers and Administrators international annual conference, Toronto (Canada), 25–29 September 1994.

[7] *The Common Criteria.* http://www.commoncriteriaportal.org/ (10 Apr 2009)

[8] U. S. Department of Defense. *National Industrial Security Program Operating Manual (NISPOM).* 2006.

[9] U. S. Department of Defense, Defense Information Systems Agency. *CyberProtect, version 1.1.* July 1999. http://iase.disa.mil/eta/product_description.pdf (10 Apr 2009)