An Optimization Approach using Soft Constraints for the Cascade Vulnerability Problem

Christian Servin, Martine Ceberio, Eric Freudenthal
Department of Computer Science
University of Texas at El Paso
Computer Science Building Room 234
500 W. University Drive
El Paso, TX 79968-0518, USA
{christians, mceberio, efreudenthal}@utep.edu

Stefano Bistarelli
Dipartimento di Scienze
Università "G. d'Annunzio", pescara, Italy
bista@sci.unich.it
Istituto di Informatica e Telematica, CNR, Pisa, Italy
stefano.bistarelli@iit.cnr.it

***Abstract*—In the discipline of computer security, the field of Trust Management Design is dedicated to the design of trusted systems, in particular trusted networks. One common trusted mechanism used these days is the *Multi-Level Security* (MLS) mechanism, that allows simultaneous access to systems by users with different levels of security clearance in an interconnected network. Vulnerability arises when an intruder takes advantage of the network connectivity and creates an inappropriate flow of information across the network, leading to the so-called Cascade Vulnerability Problem (CVP).**

In this article, we extend an existent approach to this problem proposed by Bistarelli et al. [1] that models, detects and properly eliminates the CVP in a network. This particular approach expresses a solution of the problem using Constraint Programming. We incorporate real-world criteria to consider into this approach, such as the bandwidth, electricity, cost of connections. Considering such features in CVP results in generating a constraint optimization problem.

I. Introduction

A. Motivation

In inter-connected systems, where several computers share information with each other, problems may arise when inappropriate information starts to flow through. For example, let us consider a simple scenario of a university composed of three departments: payroll, financial aid, and academic services. We know that the payroll department deals with sensitive information, such as social security numbers, dates of birth, amounts of wages, etc. The financial aid department may use information that payroll owns. Similarly, the academic department communicates with the financial aid department. An intruder can take advantage of this network connectivity and create an inappropriate flow of information across the network, leading to the so-called Cascade Vulnerability Problem (CVP) [14].

Several approaches have been proposed to solve this problem. Among them, the approach of Bistarelli et al. [1] is of particular interest to us: it expresses a solution of the problem using Constraint Programming, and more specifically soft constraints [3], [4]. This approach not only enables to detect the vulnerable paths in a network, but also to eliminate appropriate elements of the network that provoke the security leakage.

Bistarelli et al. provide an efficient algorithm to determine the minimal number of network cuts required to limit information leakage below some threshold level. However, our key observation is that the operational value of providing connectivity is not uniform. For example, it may be far more valuable for an online store to sever links between its web server and multiple internal personnel databases, than to sever its single link to the internet. The former might cause inconvenience, however the latter "minimum number of cuts" solution would disconnect the store from its customers.

Considering such features in CVP results in generating a constraint optimization problem. In this paper, we explore an extension of the work of Bistarelli et al., in which we take into account the above-mentioned criteria.

Our contribution to solving the CVP consists in assigning and taking into account values associated to the systems and to the connections between systems. We call these values weights, they are assigned to connections, and these weights will determine the cost of the communication.

B. Outline of the Article

In Section II, we present preliminary notions, important to understand our contribution: in computer security, in particular the Cascade Vulnerability Problem, the current approaches proposed; and background on Constraint Programming, more specifically, soft-constraints. In Section III, we describe the problem we are solving, in a high level manner, including our proposed approach along with the algorithm. In Section IV, we trace our approach on an example. Finally, we conclude and draw directions for future work in Section V.

II. Background

A. Trusted Systems

In the discipline of computer security, the field of Trust Management Design is concerned about designing trusted systems. A trusted system (operating system, network, software, etc.) is said to be trusted if we have confidence that it provides memory protection, file protection, general object access control, and user authentication [12].

B. MLS

A *Multilevel Security* (MLS) mechanism allows simultaneous access to systems by users with different levels of security clearance. As a result, these security clearances prevent users from obtaining access to information for which they do not own authorization.

A MLS system enforces a lattice-based security policy ℓ of security levels, that has ordering relation $\leq$. Given $x, y \in \ell$, $x \leq y$ means that information may flow from level x to y; e.g., C $\leq$ S $\leq$ T.

C. Cascade Vulnerability Problem (CVP)

The Cascade Vulnerability Problem (CVP) is a problem that arises in interconnected approved networks on trusted network interpretation. An approved network is a network that every computer system that belongs to it, agrees to own a security assurance level. The CVP arises when an intruder takes advantage of the network connectivity to compromise information across a range of sensitivity levels, and the span of accessed levels exceeds the accreditation range of any of the computers.

D. Assurance Levels

The security criteria define a lattice, A, of assurance levels with ordering $\leq$. Given $x, y \in A$, then $x \leq y$ means that a system evaluated at y is no less secure than a system evaluated at x, or alternatively, that an intruder that can compromise a system evaluated at y can compromise a system evaluated at x. Let S define a set of all possible systems. We define $accred : S \rightarrow A$ where *accred(s)* gives the assurance level of system $s \in S$, and is taken to represent the minimum effort required by an intruder to compromise system s.

E. Current Approaches

Several approaches were proposed that model and detect the CVP [9], [10], and solve this problem [11]. Among them, the approach of [1]. is of particular interest to us, as it expresses a solution of the problem using Constraint Programming, and more specifically soft constraints.

This approach models a network that is affected by a CVP with soft constraints. In addition to detecting the vulnerable paths in the network, this approach also eliminates appropriate elements in the network that cause the security leakage. This approach, although efficient, is limited because it overlooks important real world criteria such as cost, delay, bandwidth, money, among others.

F. Constraint Programming and Soft Constraints

Constraint programming is a powerful paradigm for solving large scale problems such as combinatorial search, optimization, scheduling bioinformatics among many others. [13].

The advantage of this paradigm consists in modeling real-world problems like the cascade vulnerability in terms of their constraints, and finding an assignment to all the variables that satisfies the constraints. As soon as the problem is modeled in these terms, the rest of the work is in charge of constraint solvers; obtaining a set of all possible solutions of the problem.

1) Semiring-based Soft Constraints: Semiring-based constraints constitute an extended framework for constraint solving. Although the framework of classical Constraint Satisfaction Problems is very expressive and offers a natural formalism for representing problems, it has evident limitations, mainly due to the fact that it does not offer enough flexiblility to represent real-life scenarios where the knowledge is neither completely available nor crisp [4].

On the other hand, soft constraint programming is particularly useful to model and solve what we called over-constrained and preference-based problems, such as the Cascade Vulnerability Problem.

The semiring-based CSP framework is based on a semiring structure where the carrier set of the semiring specifies the values to be associated with each tuple of values of the variable domain, and the two semiring operations (+ and $\times$) model constraint projection and combination respectively.

2) Modeling and Detecting CVP: If an intruder takes advantage of the network connectivity, resulting that the flow of information leaks from system to system, we call this a cascade leakage.

If the effort that the intruder has to put to break into the system(s) is known to be lower than the downgrading of information that the intruder obtains, then there is a cascading effect.

In order to determine the existence of such a cascading effect, we need to compare the effort required to compromise the network against the risk of compromising the system as a whole. We use the risk constraints defined in [1]:

$$R = \{r_{(p_1^s, p_i^d)} i \in \{2, \ldots, n\}\}$$

The weight of each instance of $r_{(p_1^s, p_i^d)}$ represents the risk associated with the path from P_1^s to P_i^d.

A cascading path can be identified as any path η where the risk associated with the path exceeds the effort to compromise it. In other words, it means that the following constraint is satisfied:

$$\otimes R\eta > \otimes \varepsilon\eta$$

By incorporating this constraint to the constraint model, the existence of a solution to the model indicates that there exists a cascading path.

G. Minimal Weighted Set Cover

Our approach to solve the Cascade Vulnerability Problem is based on the theory of *minimal weighted set cover* (MWSC). The MWSC is solved as an optimization problem that models resource-selection problems [6].

1) Formal definition: The MWSC takes as an input a *set system* $(U, S), and c$ with $\bigcup_{s \in S} S = U$, weights $c : S \rightarrow \Re_+$. A set system is a pair (U, F) where U is a non-empty finite set, and F a family of subsets of U [7].

The objective is to find a *minimum weight set cover* of (U, S), i.e., a subfamily $R \subseteq S$ where R is the solution of the following constrained optimization problem:

$$\begin{cases} \min \sum_{r \in R} c_i \\ \text{s. t. } \bigcup_{r \in R} R = U \\ \text{and } \bigcup_{r \in R} S_r = U \end{cases}$$

III. Our Approach.

The approach that is of particular interest to us is the work proposed by Bistarelli et al. [1]. We are extending this approach, by associating weights to all connections between computer systems. These weights are real world values such as bandwidth, delay, cost, etc. We use the Minimum Weighted Set Cover theory as our theoretical basis to determine the least expensive cuts that should be performed in the network, in order to ensure the security of our network.

A. Brief Explanation

Our approach is illustrated in Figure 1:
Given a network composed by computers (A), the network in Figure 1 is modeled by the approach of Bistarelli et al. [1]. In case a CVP is detected, we move to the next step (which is modeled as a graph). In case no CVP is detected, the network is considered to be *CVP free*.

Our contribution to solve this problem starts at (B), where the computers that share the same security connectivity level are arranged into sets of computers. Given these sets of computers with their respective adjacent edges, this interpretation leads us to represent the problem as a MWSC problem (C), and to solve it using the MWSC greedy algorithm [15]. This MWSC solver performs the algorithm mentioned in Section III-F. The output (D) is a trusted network where the connections that were initially causing the CVP were cut, and the cost of the cuts is the cheapest among the possible cuts to solve the CVP problem.

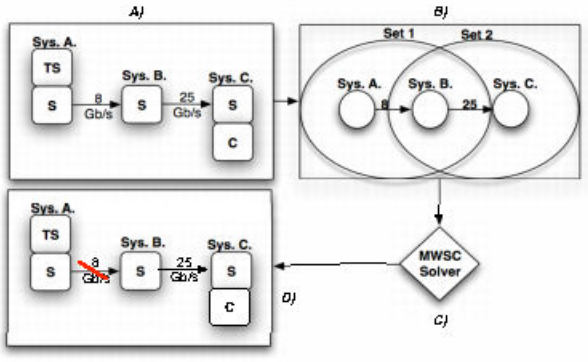

Fig. 1. Our approach.

B. Input Data (informal description).

The input to our algorithm is a network with a possible security leakage, known as the *cascade vulnerability problem*. The network is composed of computer systems with some connections between each other. The input also contains additional information, such as, for every connection, bandwidth, delay, monthly cost, etc.

Taking into account these additional features (such as bandwidth) constitutes our improvement to the approach of Bistarelli et al. [1].

C. Desired output (informal description).

The expected result is a network, constituted of as many computer systems as the original one, but where appropriate cuts in the connections and new connections have been performed. By appropriate, we mean that the chosen transformations (cuts) are the least significant ones and that these transformations make the network secure.

D. Input Data (formal description): graph representation.

Let us consider a network N, composed of a finite number n of computers systems $\{s_1, s_2, \ldots, s_n\}$. Each computer system may have connections to other computer systems.

This problem can be naturally expressed as a directed weighted graph. A directed weighted graph is denoted by $G = (V, E)$, where V is a set of vertices (systems), E is a set of ordered pairs of vertices called edges, and each edge is associated with some value that we call weights. These weights are given by a weight function $w : E \to [0, \infty[$. We indicate the weight of an edge (u, v) by w_{uv}.

The direction of the edges denotes the permitted flow between systems in a Multi-Level Security (MLS) mechanism e.g., it is not permitted that a top-secret information flow down to a classified system. The values on the edges are called weights, that denote the criteria mentioned in previous section.

In Figure 2, the *Top Secret* (TS) information in *System A* is allowed to flow to the TS level in *System B*, and from the *Secret* (S) level in *System B* to the S level in *System C*. The criterion to be considered is *bandwidth*. There is an allowed flow of information from *System A* to *System B* of 8 Gb/s. There is another flow of information from *System B* to *System C* of 25 Gb/s.

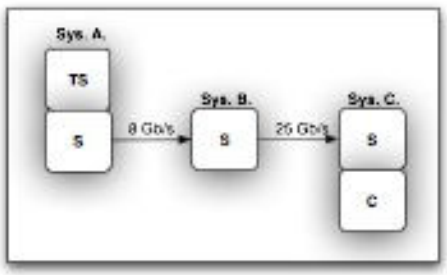

Fig. 2. A network composed of 3 systems. Here, weights represent the bandwidth.

E. The Minimum Weighted Set Cover relation with CVP

In order to get rid of a Cascade Vulnerability Problem in a network, it is necessary to cut the connections that are causing the cascading leakage. However, in cutting these connections, the services provided by the network are affected. Therefore, in order to maintain a secure network when eliminating the CVP, it is preferable to cut the least expensive connections (i.e., with least significant weights) as possible.

It is possible to do so by solving a corresponding Minimum Weighted Set Cover (MWSC) problem. Indeed, we translate our CVP problem into a Minimum Weighted Set Cover (MWSC) problem, in order to determine which connections are the most appropriate to cut (i.e., the cheapest ones) among those generating the information leakage.

The algorithm that we follow to solve the CVP problem, where criteria on the connections are taken into account, is described in the next subsection.

F. Algorithm

1) Modeling:

(i) We model each system in N according to its secure information flow. The flow reflects the accesses that are permitted by the system's MLS mechanism.

(ii) Once the systems are modeled, we model the network connectivity to determine if the network flow is *permitted* or *invalid*. This depends on each system security level synchronization, e.g., A TS level in a system is connected to another system TS level. We use risk and effort constraints (as described earlier in this paper) to model the network.

2) Detect the existence of a CVP: At this step, we determine whether the network has a security leakage or not.

(i) Compare the *effort* required to compromise the network $\otimes R\eta$ against the *risk* of compromising the $\otimes\epsilon\eta$ system as a whole. This is done to determine the existence of cascade vulnerabilities. These constraints are explained in Section II.

(ii) In case the risk constraint exceeds the value of the effort constraint ($\otimes R\eta > \otimes\epsilon\eta$), we say that a solution(s) was found.

IF a solution(s) was found **THEN**
cvp_found $\leftarrow$ *TRUE*
solution_set$\{\}$ $\leftarrow$ *get_solutions()*
ELSE
cvp_found$\leftarrow$ *FALSE* %A cascade-free network
solution_set$\{\}$ $\leftarrow$ $\{\}$

3) Model the network as a graph: If there is a Cascade Vulnerability Problem, the network is then represented as a graph. Once the network is translated as a graph, we determine the sets of computers. For every computer that share the same security level are considered to be in the same set of computers.

(i) Systems $s_i \in G$ are connected to other systems in the network through an edge $e(s_i, s_j)$. Every system that shares the same security level synchronization edge with another computer, is candidate to be in the same *set system* S.

(ii) Every set S_i is composed of a set of computers connected to each other $\{s_1, s_2, \ldots, s_i\}$, with the corresponding associated values on the edges.

IF *is_adjacent*(s_i, s_j) == TRUE **THEN**
$S_i := (s_i, s_j, w_{ij})$

4) Apply MWSC Operational Procedure: Once the graph is modeled we apply the MWSC procedure.

- As an input, we have:

$$\begin{array}{ll} \text{Universe} & U = \{s_1, s_2, \ldots, s_n\}, \\ \text{Subsets} & S = \{S_1, \ldots, S_k\}, \\ \text{Costs} & c = \{c_1, c_2, \ldots, c_k\} \end{array}$$

- The goal is:

To find a set $I \subset \{1, 2, \ldots, n\}$

$$\begin{cases} I \text{ minimizes } \sum_{i \in I} c_i \\ \text{and } \bigcup_{i \in I} S_i = U \end{cases}$$

We apply the *greedy set cover algorithm* described in [15]. This algoritm selects an approximation of a minimal set of removed links, and keeps track of the weights of each set.

Set $C := \emptyset$ **(Is a set of elements covered so far)**
Set $S := \emptyset$ **(The average cost per newly covered element)**

WHILE $C \neq U$ **DO**:
Find the set whose average cost is smaller, say *S*.
Let $\alpha = \frac{c(S)}{|S \backslash C|}$, i.e., is minimum avg. cost of *S*.
Pick *S*, and for each $e \in S \backslash C$, set a weight(*e*) = α.
$C \leftarrow C \bigcup S$.
Output the picked sets.

IV. Example

Consider the example given in the motivation. A common university is composed of several departments. In this particular example, we have the *payroll, financial aid records, academic services, and student services* departments. These departments are connected in the same order as shown in Figure 3. Each department owns TS, S, or C information, or a combination of these assurance levels.

Given this network, we need to figure out whether there is a potential cascade vulnerability leakage. If there is a leakage, we need to remove the least significant computer connections

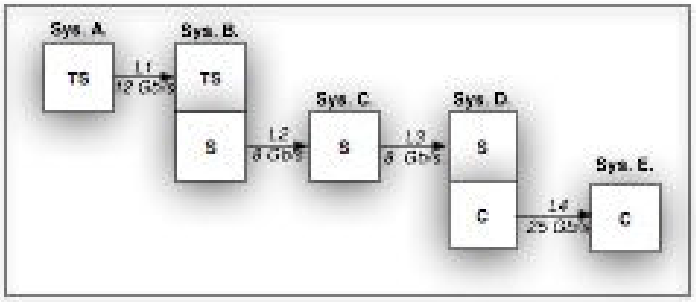

Fig. 3. A Network composed of 5 computers systems. A: payroll, B: financial aid, C: records, D: academic services, and E: student services

that are causing this leakage, in order for the resulting network to be cascade free.

- First, we use the approach of [1] to model the problem using the ($\otimes R\eta > \otimes \epsilon\eta$) constraints. Figure 4 illustrates the detection of the CVP using these constraints.

- The next step consists in representing the network as a graph in order to obtain system sets. Consider the network as a graph, the computer systems as vertexes, and their connections as edges. Adjacent systems belong to the same system set.

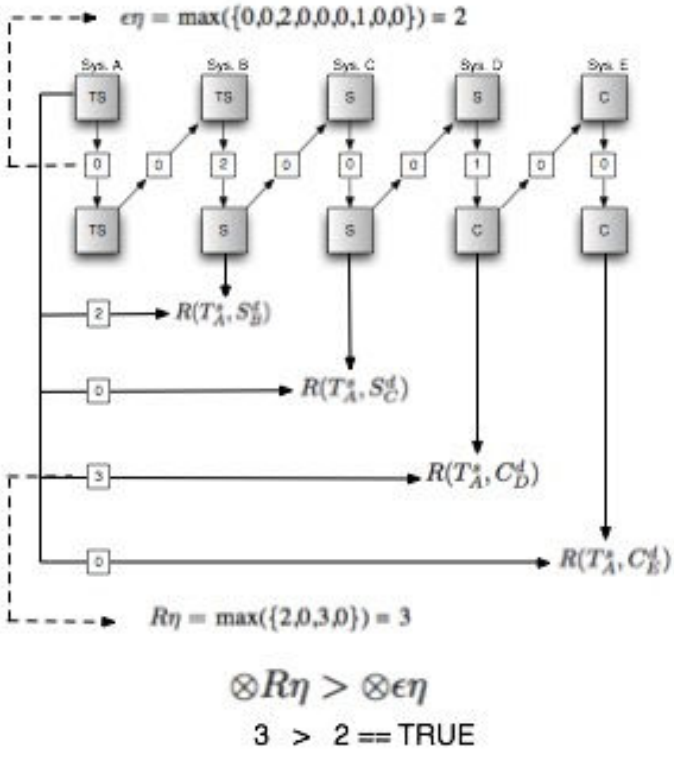

Fig. 4. CVP Detection using Constraints Risk and Effort

In our example, *payroll* and *financial aid* are connected because of their security level synchronization *(TS)*, resulting in having these two computers in the same system set. In the same way, there is a connection between financial aid, records, and academic services that synchronizes *S* information, and so on.

- According to the security level connectivity in the network, we obtain the following sets:

$$S_1 = \{system_A, system_B\}$$
$$S_2 = \{system_B, system_C, system_D\}$$
$$S_3 = \{system_D, system_E\}$$

- For every system generated, we assign a value (which are the costs) according to the connectivity weights that each edge in the set has. In this way, we obtain a set of costs: c = $\{c(S_1), c(S_2), c(S_3)\}$, where $c(S_1) = 12$, $c(S_2) = 16$, $c(S_3) = 33$.

- Once we have the system sets $\{S_1, S_2, S_3\}$, and their associated costs, we invoke the MWSC procedure, that will perform the *Greedy algorithm for Set Cover*. For every iteration, we keep track of the cost of each set and the value of each element in the sets. After running this algorithm, we obtain the following table. following results:

Step	*Set Selection*	*C*	*W*	*Systems*
Step 1	S_1	6	12	{A,B}
	S_2	8	8	{C,D}
	S_3	33	25	{E}
Step 2	S_2	5.3	8	{B,C,D}
	S_1	12	12	{A}
	S_3	33	25	{E}
Step 3	S_3	16.5	25	{D,E}
	S_2	8	8	{B,C}
	S_1	12	12	{A}

After running the greedy algorithm for the set cover three times, we obtained the results mentioned in Table 1. The logical solution using other approaches would consist in eliminating the elements with lower assurance levels. For instance, in this case, systems *D* and *E* would be deleted.

On the contrary, when we run the algorithm proposed in this article, we can observe that, for each iteration, these elements, *D* and *E*, obtained a high cost value, which happens to be the most expensive in the network.

The element in the network that, on average, has the smallest connection value, is system B, which belongs to S_2. Therefore the candidate connection to be eliminated is the one that connects system *A* with system *B*, because it is the least expensive connection in the network, despite the fact that it owns the TS information.

V. Conclusion

In this article, we considered a problem of security leakage in a network of computer systems. The particular problem we addressed is well-known as the Cascade Vulnerability problem, in a *Multi-Level Security* (MLS) environment.

Our approach consisted in extending the previous work of Bistarelli et al. [1], by taking into account values on the connections of the network: some connections are indeed more valuable than others, and we needed to represent such information.

Our contribution was the following: We kept the approach of Bistarelli et al., using soft constraints to model, detect and solve the Cascade Vulnerability problem (CVP), but we extended it by using a Minimum Weight Set Cover approach to deal with the connections' values. This way, we were able not only to detect the CVP (if there was any), but also to make the least expensive cuts in the network's connections. The system, after the cuts provided by our approach, was leakage free.

As future work, we will implement our algorithm and test it on real MLS mechanism networks. In addition, we plan to apply this methodology in trusted models different from MLS, to see if the results make sense in another kind of environment. We also plan to consider not only values on the connections of the network, but also weights of each computer system.

Acknowledgments

The authors would like to acknowledge the fact that this work was partially supported by the NSF program: *LSAMP Bridge to the Doctorate*, grant No. HRD-0217691.

References

[1] S. Bistarelli, S. N. Foley and B. O'Sullivan Journal of Computer Security, "A soft constraint-based approach to the cascade vulnerability problem", Volume 13, N. 5, pp. 699 - 720; Special Issue: Security Track at ACM Symposium on Applied Computing 2004.

[2] S. Bistarelli, S.N. Foley, and B. O'Sullivan, "Detecting and Eliminating the Cascade Vulnerability Problem from Multi-level Security Networks using Soft Constraints", in: Proceedings of Innovative Applications of Artificial Intelligence Conference (IAAI-04), July 25-29, 2004.

[3] S. Bistarelli, U. Montanari, and F. Rossi, "Constraint Solving over Semirings", in: Proceedings of IJCAI95, Morgan Kaufmann, 1995.

[4] S. Bistarelli, U. Montanari and F. Rossi, "Semiring-based Constraint Satisfaction and Optimization", in Journal of ACM, vol.44, n.2, pp. 201-236, March 1997.

[5] S. Bistarelli, A. eds. 2004 *Semirings for Soft Constraint Solving and Programming*, Springer, Lecture Notes in Computer Science, Vol. 2962, ISBN: 3-540-21181-0

[6] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. A. eds. 2001. *Introduction to Algorithms*, Second Edition. MIT Press and McGraw-Hill.

[7] B. Korte, J. Vygen. A. eds. 2006/ *Combinatorial Optimization*, Third Edition, Springer.

[8] U. Feige. "A Threshold of lnn for Approximating Set Cover". Journal of the ACM (JACM), v.45 n.4, p.634 - 652, July 1998.

[9] S.N. Foley. "Conduit cascades and secure synchronization". In: ACM New Security Paradigms Workshop, 2000.

[10] L. Gong, X. Qian. "The complexity and composability of secure interoperation". In: Proceedings of the Symposium on Security and Privacy. Pg190200. IEEE Computer Society Press.

[11] S. Gritzalis and D. Spinellis, "The cascade vulnerability problem: The detection problem and a simulated annealing approach to its correction", Microprocessors and Microsystems 21(10) (1998), 621-628.

[12] C. P. Pfleeger, S. L. Pfleeger, A. eds. 2003. *Security in Computing*, Third Edition. Prentice Hall.

[13] F. Rossi, P. Van Beek, T. Walsh, A. eds. 2006. *Handbook of Constraint Programming*, Elsevier.

[14] TNI.Trusted computer system evaluation criteria: Trusted network interpretation. Technical Report NCSC-TG-005, National Computer Security Center, 1987. Red Book.

[15] V. V. Vazirani, A. eds. 2003. *Approximation Algorithms*, Springer.