Assessing the Risk of an Information Infrastructure through Security Dependencies

F.Baiardi+, S.Suin+, C.Telmon+, M.Pioli*

(+)Dipartimento di Informatica, Università di Pisa
(*) Enel Distribuzione, ENEL
{f.baiardi, stefano}@.unipi.it

Abstract. We outline a framework for the risk assessment of information infrastructures that generalizes the notion of dependency with respect to attributes such as confidentiality, integrity or availability. Dependencies are used to model an infrastructure at distinct abstraction levels and to discover attack strategies as well as risk mitigation plans. A plan is formulated in terms of set of countermeasures because single countermeasures may be ineffective due to alternative threat attack strategies. We do not detail the assessment steps but show how the proposed framework support their integration to define risk mitigation plans. Lastly, we consider programming tools to support the assessment.

Keywords: risk assessment, mitigation plan, countermeasure, vulnerability, ranking

1 Introduction

A risk mitigation plan is the output of a risk assessment of an ICT, or information, infrastructure [1, 3, 5, 6] that defines the countermeasures to be applied to reduce the risk at an acceptable level for the owner. Risk is formally defined as the product of the probability that a threat implements a successful attack and the corresponding impact that is the damage due to the attack. If the vulnerabilities are known, then each vulnerability V may be paired with the risk it introduces because of the attacks it enables. The return of the investment to remove V [8, 16] is the difference between this risk and the cost to remove V. The problem posed by this approach is that the involved probabilities can be determined only if historical data about the infrastructure is available. For most information infrastructures this is seldom the case. Furthermore, to mitigate risk, several countermeasures have to be applied simultaneously because of alternative attack strategies that compose simple attacks into more complex ones [2, 10, 14, 15]. Hence, the return of removing a single vulnerability cannot be easily estimated and approximated strategies are adopted that return a ranking of vulnerabilities that defines the order to remove them, i.e. to apply attack countermeasures [**?**].

We present an approximated risk assessment strategy for information infrastructures that ranks countermeasures to define risk mitigation plans. To define cost effective plans, the strategy ranks sets of countermeasures rather than single ones. The framework describes an infrastructure as a set of interdependent components, each defining a set of operations that read and update an internal state and that may be

invoked by a user subset. Three attributes for each component are introduced, namely confidentiality, integrity and availability. Each attribute is controlled by invoking some of the component operations. The relations among components are described through dependencies [3, 9], each involving a set of source components, a destination one and a security attribute for each component. The meaning is that the control of the attributes of the source components implies that of the attribute of the destination one. The infrastructure is represented as a labeled hypergraph with a node for each component and a hyperarc for each dependency.

Sect.2 of the paper defines the representations of the infrastructure, of attacks and threats. Sect.3 considers countermeasures and it introduces minimal set of countermeasures. Sect.4 defines the ranking of countermeasures and mitigation plans. Sect.5 introduces the notion of risk and Sect.6 discusses the development of programming tool to assist the assessment. Important analyses such as the vulnerability or the impact ones will not be described, as they require methodologies fully orthogonal to our framework that only aims to integrate their results into a risk mitigation plan.

2 Modelling Infrastructures, Attacks and Threats

This section introduces the description of an infrastructure, of the attacks and the threats that implement these attacks.

2.1 Component Dependencies and Infrastructure Hypergraph

The framework describes the infrastructure as a set of interdependent components, each consisting of some internal state and operations it implements. For each component, three security attributes are considered:

1. confidentiality. Its control implies the ability of reading the component state ;
2. integrity. Its control implies the ability of updating the state;
3. availability. Its control implies the ability of managing the component, i.e. of determining the users that can invoke its operations.

The framework does not describe the state or the operations and represents user rights as a set of pairs $\langle component, attribute \rangle$, where attribute $\in \{c, i, a\}$. A user controls an attribute because of either the component operations it can invoke or dependencies from other components. Each dependency is characterized by the source components from where it originates, by a destination one and by an attribute for each component, Distinct dependencies correspond to alternative ways of controlling an attribute. Dependencies are inspired to cascade failures and domino effect models [3, 12].

A first dependency is the one between a password and the resources it controls. Anyone that controls the password confidentiality, i.e. can read it, can access the resources that the password protects. If distinct operations requires distinct passwords, then each one enables a limited control on the component, i.e. to control all the components attributes, the confidentiality of several passwords has to be controlled. If, instead, just one password is introduced, the control of its confidentiality implies that of any component attribute. A second example concerns a web server connected to

a network through components that route and deliver requests to the server. Anyone that controls the availability of these components can deny the access to any user and fully control the server availability. Consider now a firewall that protects a computer network. The control of the availability of the communication infrastructure between the network and the outside world depends upon the integrity of the firewall rules.

The infrastructure hypergraph IH is a labeled directed hypergraph that represents components and dependencies. IH includes a node N(C) for each component C and an hyperarc from {n(C1),..., n(Ck)} to n(C) for a dependency from {C1,..., Ck} to C. As shown in Fig. 1, there is one label for each tail of the hyperarc, to denote the attribute to be controlled, and one for the head, to denote the controlled attribute.

Starting from the set S of the rights of a user U and the dependencies in IH, we determine all the attributes U control by taking into account paths of IH and by computing TC(S,IH), the transitive closure of S. At first, we consider an element $\langle C, w\rangle$ of S and mark any tail leaving from N(C) and labeled by w. After examining all the elements in S, we consider an hyperarc h such that all its tails have been marked. Assume that the head of h is labelled by g and that the destination node is d. We mark d and any tail leaving from d that is labeled by g. Furthermore, if $g = a$, U manages the component C represented by d. Hence, we also mark any tail leaving d because U can assign to itself any rights on C and so it can control any attribute of C. The procedure is iterated till no node can be marked. The components corresponding to the marked nodes, together with the labels of the hyperarcs used to reach each node, define the transitive closure of S. The reasons why we assume a user grants rights only to itself are detailed in the following. As an example, in the hypergraph in Fig.1, a user that controls the confidentiality of c7 and the integrity of c8 controls the availability of c5, the confidentiality of c6 and the integrity of c4 as well.

2.2 Modeling Threats and Attacks

This section describes the modeling of threats and of elementary attacks. We will consider only intelligent threat that try to achieve some of their goals. To handle uniformly insiders and external threats, we model each of such threats as a further user that owns some rights on some component.

In the following, a user U is anyone interested in attacking the infrastructure and it is paired with information about its goals, the resources it can accesses and its initial rights. SGoal(U) is the set of the goals of U, each goal g in SGoal(U) is a set of rights. U achieves g when it owns any right in g. R(U) describes the resources U can exploit in its attacks. We assume that elements of R(U) are tuples of elements where each element describes a distinct resource kind such as computational resources, know how, knowledge about the infrastructure and so on. We do not detail alternative definitions of R(U) because we are only interested in a partial order for each kind of resources. This orders supports the definition of a partial order $\lhd$ among the tuples of R(U) based upon a pair wise comparison of the tuples. The last information paired with U is Init(U), the initial rights of U. This set models the capabilities of U before any attack and initially it is equal to the legal set of rights of U.

Consider, as an example, a web server that receives its data from a database system. This infrastructure may be modeled as two components, the web server and the

database. The integrity of web server data depends upon the integrity of those in the database. Here, three classes of users can be introduced that initially have, respectively, some rights on the web server, on the database or no rights at all. The first two classes model users that can access or update information in the components, the last one the users that cannot access any information. The goal of a user that aims to deface the web server is to control web server integrity. Instead, that of a user interested in manipulating the data is to control the database integrity. Because of dependencies, any user that controls the database integrity also controls that of the web server.

We describe now the modeling of elementary attacks that are composed into attack strategies against the infrastructure [11, 15, 18–21]. Even if the definition of elementary attack depends upon the component abstraction level, we define an attack A to be elementary if it consists of a sequence of predefined actions against one infrastructure component denoted by T(A). These actions may be successful if each vulnerability in a set of component vulnerabilities V(A) exists. Distinct sets of vulnerabilities corresponds to distinct attacks that may exploit the same mechanism, i.e. a buffer or a stack overflow. The **precondition** pre(A) of A is the set of rights a user U needs to execute A. If A is successful, U achieves a set of rights post(A), the postcondition or the **effect** of A, disjoint from pre(A). Owning the rights in pre(A) is a necessary but not sufficient condition to execute A, because this also requires that U can access R(A), the resources to implement A. This is possible if R(A)$\triangleleft$ R(U). If A is successful, U owns a set of rights equal to TC(U(BA)$\cup$ post(A), IH), the transitive closure of the union of the rights U(BA),that U owns before A, with those acquired through A. IH is the infrastructure hypergraph.

The last information paired with A is comp(A), a relative evaluation of the complexity of A. The lowest complexity corresponds to an attack that can be implemented by a programming tool, because it can be executed by any user. The opposite extreme of the range is an attack that cannot be automatized and that require a deep technical know how and detailed knowledge on the infrastructure. The number of discrete intervals within the range depends upon the accuracy of the assessment.

2.3 Infrastructure Evolutions

We consider now how a user U can achieve a goal in SGoal(U) by composing elementary attacks into strategies. Since a user continues to execute elementary attacks till achieving one of its goals, we can characterize the state of the infrastructure in terms of the users that are considered and by the transitive closure of the rights of each user. Transitions from a state to another one are fired by successful elementary attacks. Hence, we model an infrastructure state S as a tuple. $\langle \langle U1, S1 \rangle, ..., \langle Un, Sn \rangle \rangle$ where U1, .., Un are the users and Si is the set of rights of Ui. ER(Ui,S) denotes the rights of Ui in the state S. Since ER(Ui,S) is computed through a transitive closure, in any state ER(Ui,S)= TC(ER(Ui,S),IH) holds for $1 \leq i \leq n$. For each infrastructure state S, an attack A is feasible for Ui if pre(A) $\subseteq$ ER(Ui, S) and R(A)$\triangleleft$ R(Ui).

At first, we consider attack strategies of a single user and then generalize to a set of users. An **evolution due to Ui** is a non empty sequence of infrastructure states St1, ..., Stfin where:

1. St1 is the initial infrastructure state,
2. in any intermediate state Stj the attack Aj is feasible for Ui,
3. after the execution of Aj in Stj, U owns the transitive closure of rights in the union of post(A) with ER(Ui,Stj),
4. in the final state, and only in this state, Ui achieves g, a goal in SGoal(Ui),
5. by executing Aj, Ui achieves at least one rights *er* such that:
   (a) Ui does not own *er* in the state Stj before executing Aj,
   (b) *er* belongs either to g or to the preconditions of an attack executed after Aj.

The last condition implies that each evolution attack allows Ui to owns at least one right that belongs either to g or to the precondition of one of the following attacks. Empty sequences are neglected because each user has to execute at least one attack to achieve any goal. The number of attacks bounds the length of evolutions due to a single user because a user executes each attack at most once.

Each evolution due to Ui describes a strategy of Ui to compose elementary attacks to achieve the corresponding goal. Hence, if there is not an evolution where Ui achieves g, one of its goals, no sequence of attacks enables Ui to achieve g. If all the evolutions due to Ui are known, we can represent the corresponding infrastructure states and state transitions as a finite state automaton. The automaton has a single initial state, the initial infrastructure state, and a final state for each goal in SGoal(Ui) that Ui can achieve. The attack graph of Ui represents states and transitions of the automaton as graph nodes and arcs. An attack path starts in the initial node of the graph, corresponding to the initial state of the automaton, and ends in a success node, a state where Ui achieves one of its goals. Distinct goals corresponds to distinct states. Each arc is paired with an attack [18, 1, 8]. Consider, as an example an user Ui that can achieve one of its goals by executing an attack A, but this requires some privileged account on a computer node. Hence, at first Ui should control the confidentiality of a password of an account and then increase its level of privilege through a second attack before executing A. Besides the initial states and the final one, the automaton has at least two further states where Ui controls, respectively, the confidentiality of a password of a non privileged account and a privileged account,

Two distinct evolutions are ***equivalent*** if they enable a user to achieve the same goal. Such evolutions correspond to distinct paths of the attack graph leading to the same final node. Two *disjoint equivalent* evolutions exploit distinct elementary attacks. Consider, as an example, a processing node N and the set S of nodes of the infrastructure that N trusts. A user that controls any node in S, control N as well. If the goal of U is the control of N, all the evolutions that result in the control distinct nodes in S are equivalent because they enable U to achieve its goal. They are disjoint because their elementary attacks have distinct targets. Consider now the evolutions resulting in the control of the same node in S. If they compose in a different way the same elementary attacks, then they are equivalent but not disjoint.

When considering evolutions due to a set of users, three cases are of interest:

- **concurrent** evolution: each user autonomously achieves one of its goals because any user will grant only to itself a right on a component it manages. Since this evolution results from the interleaving of one evolution for each user, the assessment can consider any user in isolation,

- **collusion** evolution: two users, U1 and U2, cooperate because U1 grants to U2 at least one right on a component U1 manages,
- **competition**: U1 revokes at least one right of U2 to stop an attack. This models either a denial of service attack or a defense against an attacker.

Both concurrent and collusion evolutions are monotonic because a user never loses a right. A worst case for collusion evolutions can be deduced by introducing virtual users, each owning the rights of the users that cooperate. Even if the framework can describe any evolution, this work is focused on concurrent evolutions. Since concurrent evolutions can be described through automata as well, we can build both an automaton and a graph that describe any sequence of attacks of any user against the infrastructure.

The notion of evolution is an important difference between our framework and those focused on reliability because it is strongly related to both and-or attack trees [7] and goal oriented planning [17] as it defines user attack plan against the infrastructure. In planning terminology, an infrastructure state corresponds to the current state of the world, while attacks are the operators that update this state till a goal is achieved. Hence, the computation of evolution can exploit most planning algorithms and the corresponding heuristics. Monotonic evolutions simplify the planning because a user never loses a right. However, while planning algorithms are usually focused on one optimum or optimal plan, we are interested in discovering all evolutions.

3 Countermeasures

After defining enabling set of an evolution and attack countermeasures, we introduce minimal sets of countermeasures, the basic building blocks of risk mitigation plans. En(ev),the **enabling set** of the evolution ev, includes all the elementary attacks executed in the steps of ev. This notion generalises that of evolution because it neglects the details of an attack strategy and focuses on the attacks. Distinct ways of composing the same attacks are neglected, because stopping just one attack stops all the corresponding evolutions. This implies that if E enables at least one evolutions, we are not interested in enabling sets that includes E, because all the corresponding evolutions are stopped if we stop those enabled by E. An enabling set E is **minimal** if no proper subset of E is an enabling set as well, we can stop all the evolutions if we know all minimal enabling sets. An enabling set is not minimal anytime an intermediate state of the corresponding evolutions is a final state for a distinct evolution. A countermeasure C(A) for an attack A exploits any combination of the followings:

- remove one of the vulnerabilities in A(V),
- update dependencies among components to prevent users that execute A from achieving all the rights in post(A),
- update the rights of some users,
- increase the resources that A requires so that some user cannot implement it.

The application of C(A) stops A because either A fails or the user that executes A cannot own the rights in post(A). We say that a countermeasure stops an evolution anytime it stops at least one of the evolution attacks. In terms of attack graphs, C(A)

cuts, i.e. removes, the arcs associated with A. **Static** countermeasures are applied before an attack occurs, **dynamic** ones are applied as the attack goes on to remove some user rights. Hence, they are strongly related to competition evolutions and will not be considered in the following. We assume that for each attack there is at least one static countermeasure. There is no loss of generality here because we can always update the infrastructure to remove some component vulnerabilities. Notice that the same countermeasure can stop several attacks.

A **complete** set S of countermeasures stops any concurrent evolutions or, equivalently, it stops at least one attacks in each enabling set. This implies that some users can acquire some rights because some elementary attacks may be successful, but no user will achieve any of its goals. From another point of view, only intermediate states of the attack automaton can be reached but no final one. In term of the attack graph, a complete set defines a cut set that partitions the graph so that no subgraph includes both an initial node and a success one. A complete set is **minimal** if none of its proper subsets is complete. A minimal set defines a smallest set of countermeasures because it stops at least one attack for each minimal enabling set. The computation of a minimal set is an NP-hard problem and several Montecarlo or approximated strategies may be applied. These strategies can be updated to compute any minimal set. As discussed in the next section, distinct minimal sets correspond to alternative risk mitigation plans.

Assume now that, for each attack A, V(A) includes just one vulnerability and that any countermeasure removes just one vulnerability so that countermeasures can be mapped into vulnerabilities and the other way around. In this case, we can evaluate the role of V in the evolutions through an index Cr(V) defined as the percentage of minimal sets that remove V, i.e. that stop the attack A where V(A)=V. As Cr(V) approaches one, it becomes more and more important to remove V to stop the evolutions. Cr(V) may be useful if V is a newly discovered vulnerability or if its introduction is planned because of cost efficiency reason.

4 Countermeasure Ranking

To rank countermeasure, first of all we notice that cost effective risk mitigation plans should consider minimal sets only and that distinct minimal sets result in alternative plans. Our approach define a risk mitigation plan through two steps:

1. single ranking to define a partial order with reference to one minimal set,
2. global ranking that merges single rankings.

A partial order is adopted to rank sets of countermeasures rather than single ones. In this way, we consider disjoint evolutions or, from another perspective, distinct success paths of an attack graph that lead to the same final node. If several disjoint evolutions enable some users to achieve the same goal, stopping just a subset of these evolutions does not prevent users from achieving the goal. This shows that countermeasure and vulnerabilities are correlated so that applying a countermeasure is useless if other vulnerabilities enable the user to achieve the same goal through distinct attacks. This implies that two countermeasures (vulnerabilities) are correlated if they stops attacks

(are exploited by attacks) in disjoint evolutions. Hence, ranking correlated countermeasures may be inconsistent if the ranking is used to plan their adoption. Notice that countermeasures for attacks in equivalent but not disjoint evolutions are not correlated because the evolutions may be stopped by a countermeasure for an attack they share. The next two subsections show how we take correlation into account.

4.1 Single Ranking

First of all we introduce the notion of non redundant subset. RM is a non redundant subset of a minimal set of countermeasures M if it stops the evolutions in Se(RM) while none of its proper subsets stops all these evolutions. Each non redundant subset of M is the smallest subset of M that has to be applied to stop all the evolutions in Se(M), those that are stopped by M. Non redundant sets play a critical role to define a risk mitigation plan because if we apply a redundant set of countermeasures, some of its countermeasures are useless and a smaller, and hence less expensive, set achieves the same result. In other words, the application of a redundant set S has the same utility of the largest non redundant sets included in S.

We consider the partially order set, poset, that orders all the non redundant sets of M according to set inclusion. The bottom of the poset is ϕ and the top is M. Any maximal chain from ϕ to M of length n defines a n-1 steps risk mitigation plan where the i-th step applies the countermeasures in the difference set between the i-th set and the (i+1)-th one. Consider the minimal set $\{C1, C2, C3, C4\}$ and assume that each of C1 and C4 stops all the evolutions that result, respectively, in the goals g1 and g4. Instead, C2 and C3 stop two equivalent and disjoint evolutions resulting in g23. Hence only by applying both countermeasures simultaneously g23 cannot be achieved. The non redundant sets are $\{C1\}$, $\{C4\}$, $\{C2, C3\}$, $\{C1, C2, C3\}$, $\{C2, C3, C4\}$, $\{C1, C4\}$, $\{C1, C2, C3, C4\}$. As shown in the poset in Fig.2 a), the chain $\{C4\}$, $\{C2, C3, C4\}$, $\{C1, C2, C3, C4\}$ defines the plan that applies at first C4, then both C2 and C3 and, at last, C1.

4.2 Global Ranking

To define a global ranking we merge all the posets into a global one. The bottom of the global poset is ϕ, while each minimal set is a maximum. Maximal chains in the poset define all alternative risk mitigation plans to stop all the evolutions. Any non redundant set S belongs to a number of maximal chains that depends upon the number of minimal sets that include S.

The choice of the most appropriate plan depends upon not only the maximal chains but also financial parameters such as the amount of the resource to be invested in countermeasures, the distribution in time of these resources and the return of delaying an investment. The framework does not define a strategy to choose one maximal chain and, consequently, one plan because these financial parameters fully determine the optimal one. Hence, only the space of possible plans is defined. Suppose, as an example, that the resources currently available do not support the implementation of all the countermeasures in a minimal set and little information is available on future investment. Here, we can privilege those chains resulting in some degrees of freedom. This corresponds

to the adoption of a least commitment plan corresponding to those chains that cross all the subsets from where any minimal set can be reached. In this case, the choice of a chain that leads to one minimal set only may be inappropriate, because it freezes the set of countermeasures to be implemented even if little information is available on future investments. Any update to this plan corresponds to the choice of a disjoint chain but, in turns, this implies that some of the countermeasures previously applied are redundant, i.e. useless, in the new plan. Instead, if accurate information about future investments is available, we can choose the optimal chain for the considered time distribution of the investment and neglect all the other ones.

Consider again the minimal set $\{C1, C2, C3, C4\}$ and assume that $\{C1, C2, C5, C6\}$ is another minimal set where C5 and C6 are countermeasures that stop the same evolutions of C3 and C4. The sets $\{C1\}$, $\{C3, C4\}$ and $\{C5, C6\}$ are some of the non redundant sets that appear in the global poset shown in Fig.2b). Any strategy that applies at first either $\{C5, C6\}$ or $\{C3, C4\}$ commits itself to one minimal set, instead those that at first apply either $\{C1\}$ or $\{C2\}$ can freely choose any of the two sets.

5 Taking Risk into Account

Till now we have neglected two important risk related issues, namely the impact and the probability of a successful attack. While the former can be estimated in a fairly accurate way, approximating the latter is one of the goals of the framework. Some preliminary considerations about risk have already been introduced when considering the resources an attack requires. In fact, by exploiting this information, we can consider not any attack strategy as in unconditionally security but only those that can be implemented by the considered users. The remaing part of this sections show other ways to introduce risk into the framework by constraining the evolutions that are considered.

At first, we define further attributes of evolutions to consider the risk they pose. The first one is the impact of the evolutions, i.e. the loss of the infrastructure owner if a user achieves a goal. We assume that there is an impact, i.e. a loss, if and only if a user achieves any of its goals. This is fully general because the owner can pair an impact with any subset of rights owned by a user. The existence of disjoint evolutions implies that a user may exploit distinct strategies to achieve the same goal, i.e. the same impact. The benefit of a set of countermeasures is defined as the sum of the impacts the set avoids. If several users achieve the same goal, we assume that the corresponding impact is the maximum of those of each user. If a set of countermeasures does not stops all the evolutions resulting in the same goal, then the benefit cannot consider the corresponding impact.

A further attribute evaluates the evolution complexity, a non decreasing function of n, the numbers of evolution attacks, and of the complexity of each attack. Alternative definitions are the sum of the complexities, the average complexity multiplied by some function of n and even the largest complexity of evolution attacks. The framework does not freeze this definition because distinct ones are appropriate in distinct contexts. Furthermore, the definition can consider any historical data about attacks.

To take into account the probability that an evolution occurs, we prune evolutions with a large complexity or a low impact. Hence, we prune any evolution that has

1. an impact lower than ImpT,
2. a complexity larger than OvCompT,
3. an elementary attack with a a complexity larger than CompT,
4. a number of elementary attacks larger than MaxAtT.

The definition of the threshold values may exploit any statistics or historical data on attacks. After the pruning, we update both minimal sets of countermeasures and the benefit of countermeasures by neglecting pruned evolutions. The corresponding minimal sets are denoted as a reduced minimal set, r-minimal set. The ranking of vulnerabilities and countermeasures does not change because it is independent of the definition of minimal set. An alternative definition of r-minimal sets considers the cost of countermeasures so that a set is r-minimal if its countermeasures have the lowest cost or the best cost-benefit ratio among all minimal sets.

6 Programming Tools

Automatic tools are important not only to reduce the time to implement the assessment, but also to guarantee that no evolution has been neglected. This is fundamental for infrastructure with a large number of components.

The two most complex steps of an assessment are the computations of all evolutions and of minimal, or r-minimal, sets of countermeasures. We notice that both computations may exploit a backtracking mechanism to compute all the evolutions due to a single user. Instead, evolutions due to distinct users may be computed in parallel. Another important operator is the transitive closure of rights because it is applied for each evolution attack.

We believe that the programming framework more appropriate to take all these features into account is the logic programming one that offers backtracking as a native feature and can handle graph data structure in a fairly simple way. In this framework, an evolution corresponds to a deduction in a theory that describes the infrastructure, the attacks, the initial set of rights and the goals of each user. Also the transitive closure of a user rights is the deduction, from the set of axioms describing the user rights, of all the theorems of the theory represented by the hypergraph. This computation can be implemented in two different ways. One represents each hyperarc as a dstinct clause of the logic program and applies the program to the initial set of rights. Instead, in the other version, the program consists of a set of hypergraph independent clauses applied to both the hypergraph and the user rights. Here, the program implements an inference engine independent from the infrastructure that is a program input. The main advantage of the first solution is a better execution time but the program has to be updated anytime the hypergraph changes. Instead, the second version can be directly applied to the hypergraph without requiring an intermediate translation step. Now the clauses are more complex, but an hypergraph update has no impact on the program. However, both solutions can exploit the built in backtracking to explore all the paths in the hypergraph and deduce all the rights of a user. The computation of evolutions heavily exploits backtracking as well because it has to consider any combination of elementary attacks in each infrastructure state. An optimal computation works backward from a goal to the initial infrastructure state.

7 Conclusion

We have presented a framework to define risk mitigation plan based upon a ranking of set of countermeasures and that considers alternative attack strategies of a threat. Any countermeasure that stops just one strategy and neglects the equivalent ones is not cost effective because it cannot avoid all the impacts. Instead, risk mitigation plans should be defined as a sequence of sets where each set stops all equivalent strategies.

Future developments concern an extensive experimentation with reference to real infrastructures and a detailed analysis of collusion and competition evolutions as well as dependencies related to time or to state values of a component. A further problem is the application of the framework to interdependent infrastructures.

References

1. C. Alberts, A.Dorofee, *Managing Information Security Risks*. Addison-Wesley, 2002.
2. P.Ammann, et al. *Scalable, Graph-based Network Vulnerability Analysis*, 9th ACM Conf. on Computer and Communications security, Nov. 2002, Washington, DC, USA
3. R.J. Anderson *Security Engineering A Guide to Building Dependable Distributed Systems*. John Wiley Sons, 2001.
4. F.Baiardi, et al.*Constrained Automata: a Formal Tool for ICT Risk Assessment*, NATO Advanced Research Workshop on Information Security and Assurance, Marocco, June 2005
5. B.Barber, J. Davey, *The use of the CCTA risk analysis and management methodology CRAMM*. Proc. MEDINFO92, North Holland, 1589 pp.1593, 1992.
6. CORAS: A platform for risk analysis of security critical systems. IST-2000-25031, 2000.
7. J. Dawkins, C. Campbell, J. Hale, *Modeling Network Attacks: Extending the Attack Tree Paradigm*, Statistical and Machine Learning in Computer Intrusion Detection, June 2002.
8. L. Gordon,M. Loeb. The economics of information security investment. *ACM Trans. on Information and System Security* 5(4) 2002. pp.438-457.
9. IEC 1025: 1990 Fault tree analysis (FTA).
10. S. Jha, O. Sheyner , J. Wing, *Two Formal Analysis of Attack Graphs*, 15th IEEE Computer Security Foundations Workshop , p.49, June 2002.
11. P. Moore, R. J. Ellison, R. C. Linger,*Attack modelling for information security and survivability*, CMU/SEI- 2001-TN001.
12. National Infrastructure Advisory Council, The Common Vulnerability Scoring System, Final Report and Reccomandations, Oct. 2004
13. P. Ning, et al., *Constructing attack scenarios through correlation of intrusion alerts*, 9th ACM Conf. on Computer and Communications Security, Nov. 2002, Washington, DC, USA.
14. C. Phillips, L. Painton Swiler, *A graph-based system for network-vulnerability analysis*,Workshop on New Security Paradigms, p.71-79, Sept.1998.
15. R. Ritchey, et al., *Representing TCP/IP Connectivity For Topological Analysis of Network Security*, 18th Annual Computer Security Applications Conf, p.25, Dec. 2002.
16. S.Schechter, M. Smith. 2003. *How much security is enough to stop a thief?*. Proc. of the Financial Cryptography Conf, Guadeloupe, Jan. 2003.
17. S.Russell, P.Norving, *Artificial Intelligence: a Modern Approach*, Prentice Hall, 2003
18. O. Sheyner, et al., *Automated Generation and Analysis of Attack Graphs*, Proc. of the 2002 IEEE Symposium on Security and Privacy, May 12-15, 2002.
19. O. M. Sheyner, *Scenario Graphs and Attack Graphs*, CMU-CS-04-122,2004.
20. L.P. Swiler, C. Phillips, D. Ellis, S. Chakerian, *Computer-Attack Graph Generation Tool*, Proc. of the DARPA Information Survivability Conf, June 2001.

21. V.Swarup, S.Jajodia, J.Pamula, *Rule-Based Topological Vulnerability Analysis*, 3rd Int. Wor. on Math. Methods, Models and Arc. for Network Security, S.Petersburg Sept. 2005.

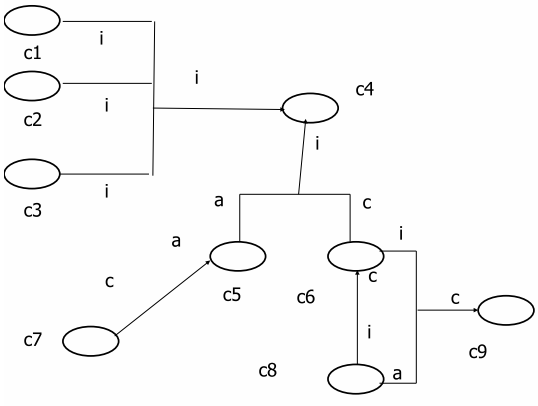

Fig. 1. An Infrastructure Hypergraph

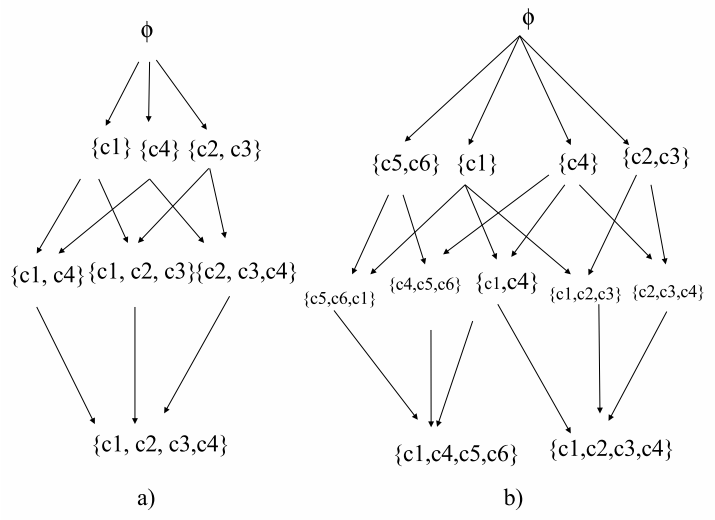

Fig. 2. Local and Global Ranking