Modeling and selecting countermeasures using CP-net and Answer Set Programming

Stefano Bistarelli[1,2], Fabio Fioravanti[1], Pamela Peretti[1], and Irina Trubitsyna[3]

[1] Dipartimento di Scienze, Università "G. d'Annunzio", Pescara, Italy
{bista,fioravanti,peretti}@sci.unich.it

[2] Istituto di Informatica e Telematica, CNR, Pisa, Italy
Stefano.Bistarelli@iit.cnr.it

[3] DEIS Università della Calabria, Rende, Italy
irina@deis.unical.it

Abstract. In this paper, we present CP-defense trees for modelling security scenarios and for expressing qualitative preferences over attacks and countermeasures, and we show how to select the set of preferred countermeasures able to protect a system by translating CP-defense trees to *Answer Set Optimization* programs which contains preferences among attacks and countermeasures. By computing the optimal answer set of the ASO program corresponding to the CP-defense tree we are able to automatically select the set of preferred countermeasure able to mitigate all the vulnerabilities in the modeled security scenario.

1 Introduction

Providing an adequate level of protection to an enterprise's IT assets is becoming increasingly important. As a consequence, security spending constitutes a conspicuous part of an enterprise budget for IT. In order to focus the real and concrete threats that could affect an enterprise's assets, a risk management process is needed in order to identify, describe and analyze the possible vulnerabilities that must be eliminated or reduced. The final goal of the process is to make security managers aware of the possible risks, and to guide them towards the adoption of a set of countermeasures which can bring the overall risk under an acceptable level, while minimizing the total cost of the security investment.

An instrument that can be used to determine the possible attacks that can harm a system, and the necessary countermeasures are *defense trees*. Defense trees, DT [1], an extension of attack trees [12], have been introduced as a methodology for the analysis of attack/defense security scenarios. A DT is an *and/or* tree, where leaf nodes represent the vulnerabilities and the set of countermeasures available for their mitigation, `and`-nodes represent attacks composed of a set of actions that have to be performed as a whole to damage the system, `or`-nodes represent attacks composed of a set of alternative actions that damage the system. Notice that to defeat `and` attacks it is enough to patch one of the vulnerabilities (by selecting a single countermeasure), whilst to stop `or` attacks, one countermeasure for each of the actions composing the attack has to be selected.

The overall goal is to use the defense tree representation of attacks and countermeasures for the selection of the *best* set of countermeasures (w.r.t. specific preference criteria such as cost, efficiency, etc.), that can stop all the attacks to the system.

Several works in literature [8, 11] address this problem by following a quantitative approach: the security manager must provide economic quantitative estimates to attacks and countermeasures which are then used to compute the cost-optimal set of countermeasures. However, the use of quantitative evaluations in the analysis of an IT system is often very difficult and expensive because many factors can influence the attacks and the selection of countermeasures. In most cases such a quantitative evaluation is not possible or reliable, and as a consequence security managers tend to take decisions according to their experience and intuition alone, without following any kind of structured process.

In these situations, it seems reasonable to provide qualitative estimates, which are usually easier to elicit, rather than quantitative ones. In [2] a method has been proposed to model preferences over attacks and countermeasures using CP-nets, but no implementation was provided, making the method unfeasible for big and complex scenarios.

The preference among countermeasures and the dependency between attacks and countermeasures are modeled in [6] by an *answer set optimization* (ASO) program. The `and` and `or` composition of a branch is then obtained by a syntactic composition of the ASO programs, whose semantics completely respects the intended meaning given in [2]. The semantics of the obtained ASO program provides a set of ordered answer sets representing the ordered sets of countermeasure to be adopted. In order to deal with ordered attacks (from the more to the less dangerous), the model is extended by introducing a corresponding rank (meta-preferences) among the preference rules of an ASO program. The introduction of this kind of meta-preferences allows us to prefer the adoption of countermeasures covering the more dangerous of the attacks.

The paper is structured as follows: we introduce CP-defense trees in Sec. 2. In Sec. 3 we propose two different methods to compose preferences: the `and`-composition and the `or`-composition. In Sec. 4 we present a method for translating - under suitable conditions - the CP-defense tree into an Answer Set Optimization (ASO) program.

2 CP-defense trees

The *CP-defense tree* structure allows a system administrator to determine, in a qualitative manner, the attack strategies that an attacker can follow to damage a system, the different actions that compose each attack and the security measures that can be introduced into the system. CP-defense trees are based on two different instruments: the *defence tree* and the *CP-network*.

Defense tree [1] are an instrument that can be used to determine and to graphically represent the possible attacks that can harm the assets of an IT system,

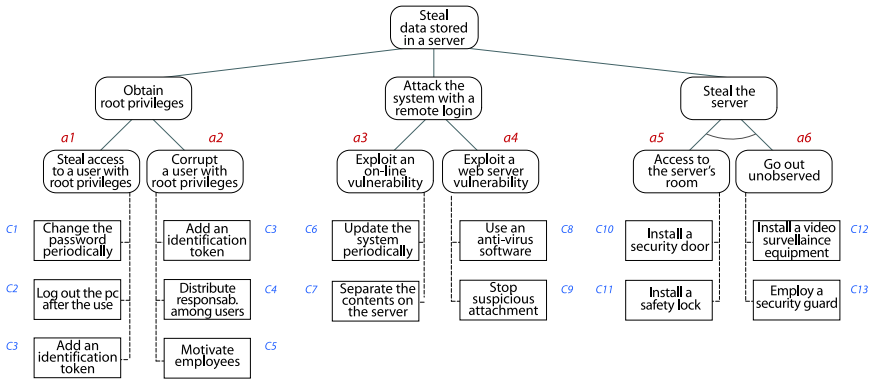


Fig. 1. An example of defense tree.

and the necessary countermeasures. They are an extension of attack trees [12], a formal way of describing how attacks against a system can be performed. An attack tree is built as follows:

- the *root* of the tree is associated with an asset of the IT system under consideration,
- *leaf nodes* represent simple subgoals which lead the attacker to (partially) damage the asset by exploiting a single vulnerability,
- *non-leaf nodes* (including the tree root) represent attack subgoals and can be of two different types: `or`-nodes: are used to represent subgoals that are completed as soon as any of its child nodes is achieved; `and`-nodes: are used to represent subgoals which require all of its child nodes to be completed. In the following we draw an horizontal line across the arcs connecting an `and`-node to its children to distinguish it from an `or`-node.

Each path from leaf nodes to the root ending in an achieved subgoal represents a different attack strategy in the considered scenario.

A defense tree is built by labeling each leaf node of an attack tree with a set of possible countermeasures able to stop that particular attack action. The attack tree is represented by using round nodes and solid edges, then it is enriched by labeling each leaf node with a set of possible countermeasures, represented by square nodes and connected by means of dotted lines.

Fig. 1 shows an example of a defense tree to model an attack/defense scenario for an enterprise's server used to store information about customers: rounded-box nodes denote the attack strategies and the different actions the attacker needs to perform, while square box nodes denote the different countermeasures the system administrator can adopt.

CP-networks [3–5] are a graphical formalism for specifying and representing qualitative conditional preference relations.
CP-nets capture *ceteris paribus* preference statements like, for example, *"I prefer red wine to white wine if vegetable soup is served"*. This preference statement

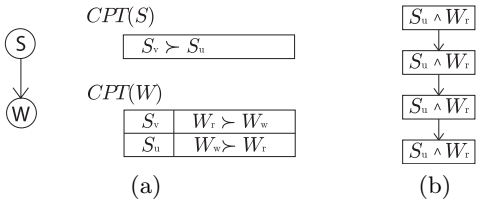


Fig. 2. An example of CP-net and the corresponding induced preference graph.

means that, given two meals both containing a vegetable soup, the meal with red wine is preferred to the meal with white wine, *all else being equal*.

The conditional *ceteris paribus* semantics, adopted in CP-nets, requires that for each variable x in the variable set V, a user specifies the parent variable $Pa(x)$ that can affect his preferences over the values of x. This information is used to create the CP-net graph in which each node x has $Pa(x)$ as its immediate predecessor. For instance, in the example above the variable set V consists of two variables S and W, representing the soup and the wine choice respectively. $Pa(W) = S$ as the assignment of S (the choice of soup) can impact the preference over the values of W (wine choice).

So, given a particular value assignment to $Pa(x)$, the user can determine a preference order over the domain of x, denoted as $\mathcal{D}(x)$, all *other things being equal*. This conditional preference over the values of X is collected in a *conditional preference table* (CPT). For each assignment to $Pa(X)$, $CPT(X)$ specifies a strict partial order over $\mathcal{D}(X)$. Notice that the right side of a conditional preference table is composed of a set of admissible values and by a partial order over these values. When the set is not specified, we assume that it is composed only of the values used in the partial order. Thus in the example above the conditional table associated with variable W contains only one line $S_v : W_r \succ W_w$, representing the preference of red wine (W_r) to white wine (W_w) in the case of the vegetable soup (S_v).

Example 1. Consider a preference specification described by the following sentences: *"I prefer vegetable soup to another type of soup"*, *"I prefer red wine to white wine if vegetable soup is served"* and *"If another type of soup is served the white wine is preferred to the red wine"*. This preference specification extends the above-discussed one and can be modeled by using a CP-net presented in Fig. 2(a). The variable set is $V = \{S, W\}$, $D(S) = \{S_v, S_u\}$, $D(W) = \{W_r, W_w\}$, where S_v and S_u represent the vegetable and the another type of soup, while W_r and W_w state for red and white wine. $Pa(W) = S$ because the type of soup influences the selection of wine. The $CPT(S)$ and $CPT(W)$ reports the preference over soup choice, described by the first sentence, and wine choice, described by the two latter sentences, respectively. △

The CP-net can be used to build an *induced preference graph* [4], that is an acyclic directed graph where the nodes correspond to the complete assignments of the variables of the CP-net, and there is an edge from node o' to node o if and only if the assignments at o' and o differ only in the value of a single variable X and o is preferred to o'. For instance, Fig. 2(b) reports the induced

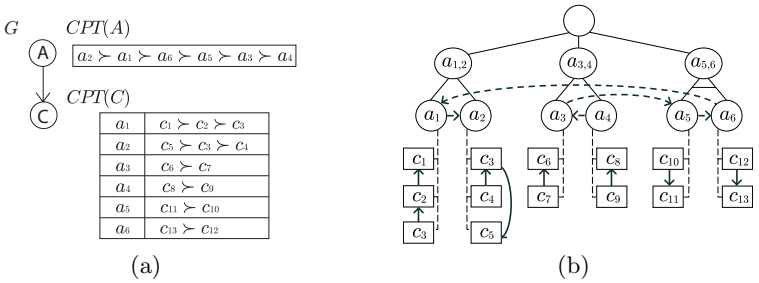


Fig. 3. A CP-defense tree.

preference graph corresponding to the CP-net presented in Ex. 1. It shows that the assignment $S_u \wedge W_r$, corresponding to the root of the induced preference graph, is the less preferred assignment, while the assignment $S_v \wedge W_r$ is the most preferred assignment.

CP-defense trees. The integration of defense trees and CP-nets has been recently proposed in [2] as a methodology to help the system administrator to analyze a security scenario and to give him a model to represent preferences among countermeasures. The resulting structure, called *CP-defense tree*, extends the defense tree representation with preference specification.

The idea is to use the defense tree representation for the selection of the best set of countermeasures (w.r.t. specific preference criteria such as cost, efficiency, etc.) that can stop all the attacks to the system. To guide this selection, a CP-net structure, able to model preferences over attacks and countermeasures can be used.

As an example consider the integration of the defense tree of Fig. 1 and the CP-net of Fig. 3(a). The variables A and C represent the (attack) action and the countermeasure selection respectively. The preference over countermeasures is conditioned by the type of action, so $Pa(C) = A$. $CPT(A)$ describes the preferences over actions $a_i \in D(A)$, ordered according to the impact produced on the system. For instance, the system administrator prefers to protect from the action a_2 rather than from the action a_1 because action a_2 is more dangerous than action a_1. $CPT(C)$ collects the preference among countermeasures $c_k \in D(C)$ for each type of action. The reason for the preference specification can be different: efficiency, price, etc. In this example, for a given attack action a the application of the countermeasure c is preferable to c' when c is less expensive than c'.
The corresponding CP-defense tree is presented in Fig. 3(b).
Observe, that the preference order over actions described in $CPT(A)$ is represented with dotted arrows, while conditional preferences over countermeasures described in $CPT(C)$ are represented by using solid arrows. The arrows are directed from the less preferred to the more preferred outcome.

The final goal of a system administrator is to determine the best defense strategy. Thus, he has to derive the minimal sets of countermeasures able to stop all actions and the partial order among them. In particular, using the preference

tables $CPT(C)$ and $CPT(A)$, he has to combine the preference orders among countermeasures, described in $CPT(C)$, and obtain a new preference order over *sets* of countermeasures able to stop all actions.

3 Composition of preferences

The composition of preference orders can be performed by using two composition operations corresponding to the cases of `and`-attack and `or`-attack.

3.1 `and`-composition

An `and`-attack is an attack composed of a set of actions that an attacker has to successfully achieve to obtain his goal. To protect the system from this type of attack the system administrator can select a countermeasure for any of the actions composing the `and`-attack. In fact, it is enough to stop one of the actions to stop the attack.

In the following we say that a set of countermeasures S *covers an* ***and****-attack* A if there exists a countermeasure $c \in S$ covering at least an attack action $a \in A$.

Given an `and`-attack, denoted $u_1 \wedge ... \wedge u_n$, composed of n actions $u_1, ..., u_n$, with sets of countermeasures $C_{u_1}, ..., C_{u_n}$ respectively, the minimal sets of countermeasures able to cover all actions $(u_1 \wedge ... \wedge u_n)$ have just one countermeasure from $C_{u_1} \cup ... \cup C_{u_n}$. The new partial order $\succ_{u_1 \wedge ... \wedge u_n}$, describing the preference among countermeasures for $u_1 \wedge ... \wedge u_n$, can be obtained by combining the orders $\succ_{u_1}, ..., \succ_{u_n}$ describing the preference orders among countermeasures associated with actions $u_1, ..., u_n$ respectively.
The `and`-composition operator $\wedge$, modelling this situation, is defined as follows:

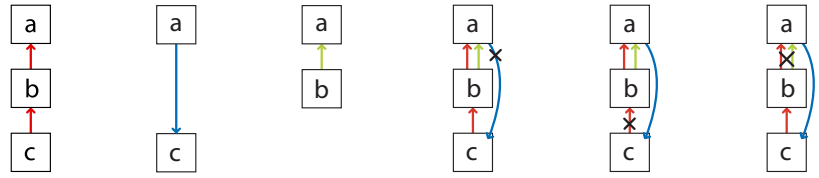
Definition 1. (`and`-composition) *Let $\mathcal{N} = (V, E)$ be a CP-net, $V = \{x_1, ..., x_n\}$ be a set of variables, and $u_1, ...u_k$ be instantiations of the variable $x_j = Pa(x_i)$.*

The ***and****-composition $\wedge(\succ^i_{u_1}, ..., \succ^i_{u_k})$ of the partial orders $\succ^i_{u_1}, ..., \succ^i_{u_k}$, described by $CPT(x_i)$, is a new partial order $\succ^i_{u_1 \wedge ... \wedge u_k}$ among elements in $D(x_i)$, such that $\forall a, b \in D(x_i)$:*

$$a \succ^i_{u_1 \wedge ... \wedge u_k} b \iff \begin{cases} \exists j \in [1..k] \text{ s.t. } a \succ^i_{u_j} b \text{ or} \\ \exists j, l \in [1..k] \text{ s.t. } \forall x, y : a \not\succ^i_{u_j} x \text{ and } y \not\succ^i_{u_l} b \text{ and } u_j \succ u_l. \end{cases}$$

For the sake of simplicity, let consider now the `and`-composition of two partial orders. Suppose $\mathcal{N} = (V, E)$ be a CP-net, with $V = \{A, C\}$ and $Pa(C) = A$, describing the preferences of CP-defense tree. Suppose also that $u_1, u_2 \in D(A)$ are two attack actions. The `and`-composition operator can be applied to the partial orderings $\succ_{u_1}$ and $\succ_{u_2}$ described in $CPT(C)$ in order to obtain a new partial order $\succ_{u_1 \wedge u_2}$ among countermeasures. In particular, given two countermeasures $a, b \in D(C)$, the countermeasure a is preferred to the countermeasure b in $\succ_{u_1 \wedge u_2}$, if

- a is preferred to b in, at least, one of the partial orders $\succ_{u_1}$, $\succ_{u_2}$;



(a) x (b) y (c) z (d) $z \succ x \succ y$ (e) $y \succ z \succ x$ (f)

Fig. 4. An example of `and`-composition.

– a is a minimal (i.e. one of the worst) countermeasures in $\succ_{u_1}$, b is a maximal (i.e. one of the best) countermeasures in $\succ_{u_2}$ and the attack u_1 is more dangerous than u_2, i.e. $u_1 \succ u_2$.

Thus, the `and`-composition preserves the partial orderings among the countermeasures, corresponding to each attack action and introduces the *bridge preference relation*, connecting the corresponding orderings, by considering the preferences over the values of the parent variable $A = Pa(C)$: if $u_1 \succ u_2$ then the countermeasures able to mitigate the risk of u_1 are preferable to the countermeasures able to mitigate the risk of u_2.

As an example consider the `and`-attack $x \wedge y$, in the case than $CPT(C) = \{x : a \succ b \succ c,\ y : b \succ c\}$ and $CPT(A) = \emptyset$. The preference order $\succ_{x \wedge y} = \wedge(\succ_x, \succ_y) = \{a \succ b \succ c\}$. The countermeasure a is the best choice for this situation.

Suppose now that $CPT(C) = \{x : a \succ b,\ y : c \succ d\}$, whereas $CPT(A) = \{y \succ x\}$. Observe that in this case the actions have disjoint sets of countermeasures. By applying the bridge preference relation $d \succ a$ we obtain the preference order $\succ_{x \wedge y}$: $c \succ d \succ a \succ b$, so c is the best countermeasure for $x \wedge y$.

It should be noted that, by applying the `and`-composition operator, we may introduce cycles in the induced preference graph. There are two possible solutions to this problem:

1. if there is some preference order between the parents $Pa(x_i)$ of the variable x_i, then we can use it to delete some edges from the induced preference graph. For example, if we know that an action is more dangerous than another one, we can say that the countermeasures for the first action are preferable;
2. if two actions are not comparable then there is no preference order between the assignment of $Pa(x_i)$. In this case we can use some algorithms like, for example the Floyd's algorithm [9] for remove cycles from the induced preference graph (by randomly removing an edge).

The following example shows how to use both these solutions.

Example 2. Consider a set of (attack) actions $A = \{x, y, z\}$, a set of countermeasures $C = \{a, b, c\}$ and $CPT(C) = \{x : a \succ b \succ c,\ y : c \succ a,\ z : a \succ b\}$, describing the partial orders $\succ_x, \succ_y$ and $\succ_z$ depicted in Fig. 4(a), 4(b) and 4(c) respectively.
The `and`-composition $\wedge(\succ_x, \succ_y, \succ_z)$ returns the preference order among the countermeasures able to stop the `and`-attack $x \wedge y \wedge z$: $\succ_{x \wedge y \wedge z} = \{a \succ b \succ c \succ a\}$, which introduces a cycle in the graph. If a preference order over actions exists, we can use

it to delete some edges from the network. For instance, if we know that z, is more dangerous than x and that x is more dangerous than y ($z \succ x \succ y$), we can remove the less preferred edge (a, c), generating the cycle. Consequently, we obtain the preference order $\succ_1 = \{a \succ b \succ c\}$, presented in Fig. 4(d). Vice-versa, if we know that the action y is more dangerous than z and that z is more dangerous than x ($y \succ z \succ x$), we prefer the countermeasure associated with the actions y. Thus, we can delete the edge (c, b), as shown in Fig. 4(e), and obtain a different order: $\succ_2 = \{c \succ a \succ b\}$. If there is no preference relation between actions, we have to randomly choose and delete an edge for removing the cycle. In this way we could obtain another order of preference like, for instance, $\succ_3 = \{b \succ c \succ a\}$, reported in Fig. 4(f). △

3.2 `or`-composition

An `or`-attack is an attack that can be performed with different and alternative actions: the attacker can complete successfully any of its actions to obtain his goal. To protect the system from this type of attack, the system administrator has to select one countermeasure for each of the actions composing the `or`-attack.

In the following we say that a set of countermeasures S *covers an* `or`*-attack* A if for each action $a \in A$ there exists a countermeasure $c \in S$ covering such attack.

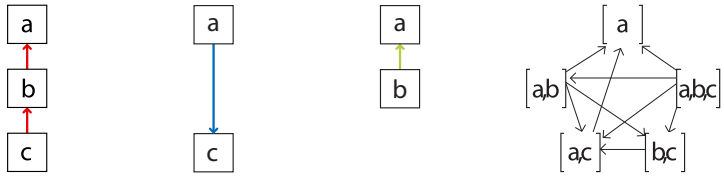
Example 3. Consider an `or`-attacks A composed of three actions x, y and z. Suppose that $CPT(C) = \{x : a \succ b \succ c,\ y : c \succ a$ and $z : a \succ b\}$. The set $\{a\}$ covers the `or`-attack A because it is able to mitigate the risk associated with actions x, y and z. The set $\{b\}$, on the contrary, doesn't cover A because it is not able to protect the system from the attack action y. △

Given an `or`-attack $u_1 \vee ... \vee u_k$, composed of k actions $u_1, ..., u_k$, with sets of countermeasures $C_1, ..., C_k$ respectively, the sets of countermeasures able to cover all actions $u_1, ..., u_k$ can be derived by considering the combinations of countermeasures from $C_1, ..., C_k$. The partial order among these sets can be obtained by taking into account (i) the minimality condition (redundant countermeasures should be avoided) and (ii) the partial orders $\succ_{u_1}, ..., \succ_{u_k}$ describing the preference orders among countermeasures associated with actions $u_1, ..., u_k$ respectively.

Definition 2. (`or`-composition) *Let $\mathcal{N} = (V, E)$ be a CP-net, where $V = \{x_1, ..., x_n\}$ is a set of variables, $u_1, ..., u_k$ be instantiations of the variable $x_j = Pa(x_i)$ and $\succ^i_{u_1}, ..., \succ^i_{u_k}$ be partial orders, described by $CPT(x_i)$, over the sets $C_1, ..., C_k \subseteq D(x_i)$ respectively.*
We say that the `or`*-composition of the partial orders $\succ^i_{u_1}$, ..., $\succ^i_{u_k}$ denotes a partial order $\succ^t$ over tuples $\langle c_1, ..., c_k \rangle$ of $C_1 \times ... \times C_k$ defined as follows: $\forall \bar{c} = \langle c_1, \ldots, c_k \rangle, \bar{c}' = \langle c'_1, \ldots, c'_k \rangle$*

$$\bar{c} \succ^t \bar{c}' \iff \exists j \in [1..k],\ c_j \succ^i_{u_j} c'_j \text{ and } \forall h \in [1..k] s.t. h \neq j, (c'_h \not\succ^i_{u_h} c_h \text{ or } u_j \succ u_h)$$

Let us denote by $Set(\bar{c})$ the set of countermeasures occurring in a tuple $\bar{c}$, and let us denote by $Cm(S)$ the set of tuples of countermeasures $\bar{c}$ s.t. $Set(\bar{c}) = S$.



(a) x (b) y (c) z (d) $x \vee y \vee z$

Fig. 5. An example of `or`-composition.

The `or`*-composition* $\vee(\succ^i_{u_1}, ..., \succ^i_{u_k})$ *of the partial orders* $\succ^i_{u_1}$, ..., $\succ^i_{u_k}$ *denotes a partial order* $\succ^i_{u_1 \vee ... \vee u_k}$ *over sets of countermeasures defined as follows:*

$$S \succ_{u_1 \vee \cdots \vee u_k} S' \iff \begin{cases} S \subset S' \text{ or} \\ \\ \exists \bar{c} \in Cm(S), \bar{c}' \in Cm(S') \quad \bar{c} \succ^t \bar{c}' \\ \text{and } \forall \bar{d} \in Cm(S), \bar{d}' \in Cm(S') \quad \bar{d}' \not\succ^t \bar{d} \end{cases}$$

The following example shows how the `or`-composition operator works.

Example 4. Consider an `or`-attack $A = x \vee y \vee z$ and $CPT(C) = \{x : a \succ b \succ c, y : c \succ a, z : a \succ b\}$, describing the partial orders $\succ_x, \succ_y$ and $\succ_z$ depicted in Fig. 5(a), 5(b) and 5(c) respectively. We want to determine the preference order over sets of countermeasures covering the attack $x \vee y \vee z$. First of all we have to determine the possible tuples of countermeasures $\langle c_x, c_y, c_z \rangle$, those components c_x, c_y and c_z cover the actions x, y and z respectively. Then we derive the sets of countermeasures, by removing the duplicate elements from these tuples:

$$\begin{aligned} \{a\} &\leftarrow \langle a, a, a \rangle \\ \{a, b\} &\leftarrow \langle b, a, a \rangle, \langle a, a, b \rangle, \langle b, a, b \rangle \\ \{a, c\} &\leftarrow \langle a, c, a \rangle, \langle c, c, a \rangle, \langle c, a, a \rangle \\ \{b, c\} &\leftarrow \langle b, c, b \rangle, \langle c, c, b \rangle \\ \{a, b, c\} &\leftarrow \langle b, c, a \rangle, \langle a, c, b \rangle, \langle c, a, b \rangle \end{aligned}$$

Now we have to compare these sets: the set $\{a\}$ is preferable to the sets $\{a, b\}$ and $\{a, c\}$ because $\{a\} \subset \{a, b\}$, $\{a\} \subset \{a, c\}$; the sets $\{a\}$, $\{a, b\}$, $\{a, c\}$ and $\{b, c\}$ are preferable to the set $\{a, b, c\}$ because they are proper subsets of it; the set $\{b, c\}$ is preferable to $\{a, b\}$ because $\langle b, c, a \rangle$ is preferable to $\langle b, a, a \rangle$ and no tuple in $Cm(\{a, b\})$ is preferable to a tuple in $Cm(\{b, c\})$; the set $\{a, c\}$ is preferable to $\{b, c\}$ because $\langle a, c, a \rangle$ is preferable to $\langle b, c, b \rangle$ and no tuple in $Cm(\{b, c\})$ is preferable to a tuple in $Cm(\{a, c\})$; the set $\{a, c\}$ is preferable to $\{a, b\}$ because $\langle a, c, a \rangle$ is preferable to $\langle b, a, a \rangle$ and no tuple in $Cm(\{a, b\})$ is preferable to a tuple in $Cm(\{a, c\})$. Fig. 5(d) shows the corresponding induced preference graph, where the set $\{a\}$ is the preferable solution for the `or`-attack $x \vee y \vee z$. △

4 From CP-defense trees to ASO programs

The use of CP-defense trees and the methodology proposed in [2] aims to find best set of countermeasures able to stop all the attacks to the system by compos-

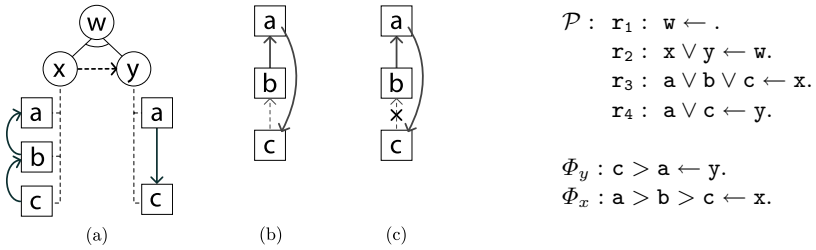


Fig. 6. An example of `and` attacks (with cycles) and the corresponding ASO program.

ing together the preference orderings associated to each branch of the tree. This approach is good and elegant for representing small scenarios, but the directly application of the composition operators becomes difficult for big and complex scenarios. This problem can be avoided by translating the CP-defense tree into ASO program, whose semantics completely respects the intended meaning given in [2], and by solving the obtained program with CHOPPER, an ASO solver described in [7].

4.1 Translation of *and-or* attacks into ASO programs

The ASO framework can profitably be used in order to represent a CP-defense tree structure and find the best set of countermeasures able to stop all the attacks. In this section the translation of `and/or` attacks into ASO programs whose semantics realizes `and/or`-composition of actions, presented in Sections 3.1 and 3.2, will be discussed.

Translation of and-attacks into ASO programs. Let us start with an example of `and`-attack . Consider the `and`-attack w, composed of two actions x and y, where $y \succ x$, presented in Fig. 6(a).
To protect the system from this kind of attack it is enough to stop just one action composing the attack. The order among countermeasures obtained by `and`-composition of orders corresponding to these actions is depicted in Fig. 6(b). We can notice that in this case a cycle is obtained. Since the countermeasure of the most dangerous attack has to be considered as preferred, the cycle can be broken by removing one of the arcs among the countermeasure of the less dangerous attack x. More precisely, the preference relations, described in (D_y, $\succ_y$) have to be considered as more important, and the relation $b \succ_x c$, generating (transitively) the conflict, has to be omitted. Graphically, this corresponds to removing the arc as shown in Fig. 6(c).

Let us now consider how to model this by using ASO programs. The attack action x and the preference order over the corresponding countermeasures a, b, and c generate the following ASO program $\langle \mathcal{P}_x, \Phi_x \rangle$:

$\mathcal{P}_x$ $\quad \mathtt{r}_{\mathtt{x}_1} : \mathtt{x} \leftarrow . \quad \mathtt{r}_{\mathtt{x}_2} : \mathtt{a} \vee \mathtt{b} \vee \mathtt{c} \leftarrow \mathtt{x}.$ $\qquad\qquad$ Φ_x $\quad \varrho_{\mathtt{x}_1} : \mathtt{a} > \mathtt{b} > \mathtt{c} \leftarrow \mathtt{x}.$

where the rules r_{x_1} and r_{x_2} introduce the action and the possible countermeasures respectively, while ϱ_{x_1} represents the preference order among them. The same result is obtained for the attack action y, the corresponding program $\langle \mathcal{P}_y, \Phi_y \rangle$ is the following:

$$\mathcal{P}_y \quad \mathtt{r_{y_1}} : \mathtt{y} \leftarrow . \quad \mathtt{r_{y_2}} : \mathtt{a} \vee \mathtt{c} \leftarrow \mathtt{y}. \qquad\qquad \Phi_y \quad \varrho_{\mathtt{y_1}} : \mathtt{c} > \mathtt{a} \leftarrow \mathtt{y}.$$

In order to model the `and`-node presented in Fig. 6(a), a new program $\langle \mathcal{P}, \Phi_y, \Phi_x \rangle$ (see right side of Fig. 6) is generated combining the rules in $\langle \mathcal{P}_x, \Phi_x \rangle$ and $\langle \mathcal{P}_y, \Phi_y \rangle$. $\mathcal{P}$ introduces two new rules: r_1 represents the root action w, while r_2 combines the action x and y in such way that only one of them must be stopped. The other rules are a simply added without any change.
The answer sets of $\mathcal{P}$ are $M_1 = \{w, x, a\}$, $M_2 = \{w, x, b\}$, $M_3 = \{w, x, c\}$, $M_4 = \{w, y, c\}$ and $M_5 = \{w, y, a\}$. In order to establish the optimal answer set, the ASO semantics firstly constructs the satisfaction vectors reporting the degree of satisfaction for ϱ_{y_1} and ϱ_{x_1} for each model: $V_{M_1} = [I, 1]$, $V_{M_2} = [I, 2]$, $V_{M_3} = [I, 3]$, $V_{M_4} = [1, I]$ and $V_{M_5} = [2, I]$.
Since ϱ_{y_1} is more important than ϱ_{x_1}, the comparison of these vectors under assumption that $I \equiv \infty$ establishes the following order among the answer sets: $M_4 > M_5 > M_1 > M_2 > M_3$. Consequently, M_4 is the optimal answer set and $\{c\}$ is the best set of countermeasures as $c \in M_4$.

The order among answer sets can be also used in order to derive all possible sets of countermeasures and the order among them. Firstly, observe that the possible sets of countermeasures correspond to the extraction of countermeasures from the answer sets and are $\{a\}$, $\{b\}$ and $\{c\}$. The preference order among these sets can be obtained by considering the preference order among the corresponding answer sets. In particular, when a set corresponds to multiple answer sets, the best answer set has to be considered.

In this example both M_1 and M_5 contain countermeasure a, $M_5 > M_1$, thus we have consider M_5; both M_3 and M_4 contain countermeasure c, $M_4 > M_3$, thus we have consider M_4. Consequently, the order among the sets of countermeasures is $\{c\} > \{a\} > \{b\}$ as $M_4 > M_5 > M_2$.

Translation of or-attacks into ASO programs. Let now consider the case of an `or`-attack, i.e. an attack composed of a set of alternative actions one of which has to be successfully achieved to obtain the goal. The protection from this kind of attack consists in the protection from all the actions composing the `or`-attack. Intuitively, the most preferred strategy has to select the best countermeasure for each action.

Consider the `or`-attack w presented in Fig. 7. The corresponding ASO program is generated as follows: $\langle \mathcal{P}_x, \Phi_x \rangle$, $\langle \mathcal{P}_y, \Phi_y \rangle$ and $\langle \mathcal{P}_z, \Phi_z \rangle$ represent, respectively, the programs associated to the actions x, y and z, and the corresponding preferences over the countermeasures a, b and c.

$\mathcal{P}_x$	$\mathtt{x} \leftarrow . \ \mathtt{a} \vee \mathtt{b} \vee \mathtt{c} \leftarrow \mathtt{x}.$	$\mathcal{P}_y$	$\mathtt{y} \leftarrow . \ \mathtt{a} \vee \mathtt{c} \leftarrow \mathtt{y}.$	$\mathcal{P}_z$	$\mathtt{z} \leftarrow . \ \mathtt{a} \vee \mathtt{b} \leftarrow \mathtt{z}.$
Φ_x	$a > b > c \leftarrow x.$	Φ_y	$c > a \leftarrow y.$	Φ_z	$a > b \leftarrow z.$

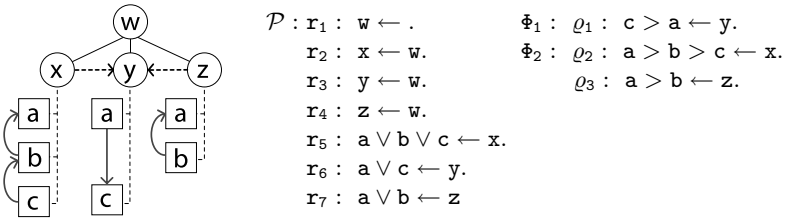


Fig. 7. An example of `or` attacks and the corresponding ASO program.

Then they are combined in the ranked ASO program $\langle \mathcal{P}, \Phi_1, \Phi_2 \rangle$, where $\Phi_1 = \Phi_y$, $\Phi_2 = \Phi_x \cup \Phi_z$ (see the right side of Fig. 7). A new rule r_1, introduced in $\mathcal{P}$, represents the root of the `or`-attack w, while the rules r_2, r_3 and r_4 model the `or`-attack, i.e. that all the three actions must be stopped to stop the w. The answer sets of $\mathcal{P}$ are $M_1 = \{w, x, y, z, a\}$ and $M_2 = \{w, x, y, z, b, c\}$ and describes the application of two alternative sets of countermeasures $\{a\}$, $\{b, c\}$, protecting from the `or`-attack w. In order to establish the optimal answer set, the ASO semantics firstly construct the satisfaction vectors $V_{M_1} = [2, 1, 1]$ and $V_{M_2} = [1, 2, 2]$. Then it compares these vectors, by firstly considering $\varrho_1 \in \Phi_1$, obtaining immediately that $V_{M_2} < V_{M_1}$. Consequently, M_2 is the optimal answer set and $\{b, c\}$ is the preferred set of countermeasures.

Observe that the application of ASO semantics models the or-composition operation in simple and elegant way. For instance, by collecting the minimal answer sets it avoids to select the redundant countermeasures. In fact, in the example above among the sets $M_1 = \{w, x, y, z, a\}$, $M_1' = \{w, x, y, z, a, b\}$, $M_1'' = \{w, x, y, z, a, c\}$, $M_1''' = \{w, x, y, z, a, b, c\}$ only M_1 is considered as the application of the countermeasure a is enough to protect the system.

Observe also that the most preferred strategy suggested by ASO semantics has to prefer the set, containing the best countermeasures for most dangerous actions, to the set containing the smaller number of countermeasures. In fact, in the above example, the set of countermeasures $\{b, c\}$ is preferred to $\{a\}$, having only one countermeasure. A possible extension of the ASO semantics, could be able to take into account the cardinality of the set of countermeasures.

4.2 Translation of CP-defense trees to ASO programs

Given an IT system *root* and the corresponding CP-defense tree $\mathcal{T}$, the selection of the preferred defense strategy can be modeled by means of the corresponding logic program with preferences. In particular, if we assume that (i) attacks are totally ordered, and (ii) the set of countermeasures occurring in every conditional preference rule is totally ordered, a ranked ASO program $L(\mathcal{T}) = \langle \mathcal{P}, \Phi_1, ..., \Phi_k \rangle$ can be constructed, where $\mathcal{P}$ describes the possible defense strategies designed in $\mathcal{T}$, while Φ_1, $..\Phi_k$ model preferences among the attacks, highlighted in $\mathcal{T}$, and establish the preference order among the attacks following the preference orderings among the countermeasures for each single attack. The application of the ASO semantics on $L(\mathcal{T})$ produces the best defense strategies w.r.t $\mathcal{T}$, thus the

optimal solutions of $L(\mathcal{T})$ can be used in order to find the best countermeasure selection.

Given a CP-defense tree with n leaf attack actions a ranked ASO program $L(\mathcal{T}) = \langle \mathcal{P}, \Phi_1, ..., \Phi_k \rangle$, $k \leq n$, can be defined, where

$\mathcal{P}$ is generating program consisting in the following rules r:

1. $\mathtt{root} \leftarrow$, stating that the root of the tree must be protected;
2. $\mathtt{Y_1} \vee \ldots \vee \mathtt{Y_n} \leftarrow \mathtt{X}$, for each `and`-node X having n child nodes $Y_1, \ldots, Y_n$, meaning that at least one attack from $Y_1, \ldots, Y_n$ must be stopped to protect from X;
3. $\mathtt{Y_i} \leftarrow \mathtt{X}$, $i = [1..n]$, for each `or`-node X having n child nodes $Y_1, \ldots, Y_n$, stating that each attack represented by $Y_1, \ldots, Y_n$ must be stopped to protect from X;
4. $\mathtt{C_1} \vee \ldots \vee \mathtt{C_n} \leftarrow \mathtt{X}$, for each leaf node X decorated with n countermeasures $C_1, \ldots, C_n$, stating that at least one countermeasure $C_1, \ldots, C_n$ is able to protect from X.

Each action in the defense tree induces a preference rule in the ASO program. Moreover, if the attack actions are ordered as $A_1 \succ \cdots \succ A_k$, where each A_i is a set of actions $\{a_1 ..., a_{h_i}\}$, a set of preference programs $\Phi_1, \ldots, \Phi_k$, where $\Phi_i = \{\varrho_{a_1}, \ldots, \varrho_{a_{h_i}}\}$, $i = [1, \ldots, k]$, is constructed. For each attack action X the preference rule ϱ_X is defined as follows:

1. $\mathtt{C_1} > \ldots > \mathtt{C_n} \leftarrow \mathtt{X}$, for each leaf node X decorated with n countermeasures $C_1, \ldots, C_n$, where there is a solid arrow from C_i to C_{i-1}, $i = [2, \ldots, n]$. This preference rule states that to protect the attack X the countermeasure C_1 is preferred to C_2, ..., C_{n-1} is preferred to C_n.

The optimal answer set obtained by computing the semantics of the generated program will collect the best countermeasure. The following Example show the result of the above procedure when applied to the CP-defense tree of Fig. 3(b).

Example 5. In this example we present an application of the ASO semantics for analyzing the attack/defense scenario shown in the CP-defense tree of Fig. 3(b). It can be modeled by using the prioritizing program $\langle \mathcal{P}, \Phi \rangle$:

$\mathcal{P}$:
$\mathtt{root} \leftarrow$
$\mathtt{a}_{1,2} \leftarrow \mathtt{root}$
$\mathtt{a}_{3,4} \leftarrow \mathtt{root}$
$\mathtt{a}_{5,6} \leftarrow \mathtt{root}$
$\mathtt{a}_1 \leftarrow \mathtt{a}_{1,2}$
$\mathtt{a}_2 \leftarrow \mathtt{a}_{1,2}$
$\mathtt{a}_3 \leftarrow \mathtt{a}_{3,4}$
$\mathtt{a}_4 \leftarrow \mathtt{a}_{3,4}$
$\mathtt{a}_5 \vee \mathtt{a}_6 \leftarrow \mathtt{a}_{5,6}$
$\mathtt{c}_1 \vee \mathtt{c}_2 \vee \mathtt{c}_3 \leftarrow \mathtt{a}_1$
$\mathtt{c}_3 \vee \mathtt{c}_4 \vee \mathtt{c}_5 \leftarrow \mathtt{a}_2$
$\mathtt{c}_6 \vee \mathtt{c}_7 \leftarrow \mathtt{a}_3$
$\mathtt{c}_8 \vee \mathtt{c}_9 \leftarrow \mathtt{a}_4$
$\mathtt{c}_{10} \vee \mathtt{c}_{11} \leftarrow \mathtt{a}_5$
$\mathtt{c}_{12} \vee \mathtt{c}_{13} \leftarrow \mathtt{a}_6$

Φ :
$\varrho_1 :\ \mathtt{c}_1 > \mathtt{c}_2 > \mathtt{c}_3 \leftarrow \mathtt{a}_1$
$\varrho_2 :\ \mathtt{c}_5 > \mathtt{c}_3 > \mathtt{c}_4 \leftarrow \mathtt{a}_2$
$\varrho_3 :\ \mathtt{c}_6 > \mathtt{c}_7 \leftarrow \mathtt{a}_3$
$\varrho_4 :\ \mathtt{c}_8 > \mathtt{c}_9 \leftarrow \mathtt{a}_4$
$\varrho_5 :\ \mathtt{c}_{10} > \mathtt{c}_{11} \leftarrow \mathtt{a}_5$
$\varrho_6 :\ \mathtt{c}_{12} > \mathtt{c}_{13} \leftarrow \mathtt{a}_6$

The answer sets of $\mathcal{P}$ are ninety, M_1 is an example of them: $M_1 = \{root, a_{1,2}, a_{3,4}, a_{5,6}, a_1, a_2, a_3, a_4, a_5, c_1, c_4, c_6, c_8, c_{10}\}$. Considering the preference order among attacks, as depicted in Fig. 3(b), we can specify the importance of preference rules: ϱ_2 is more important than ϱ_1 for expressing the preferences in the selection of countermeasures

for the attack a_2 that is preferred to a_1, ϱ_1 is more important than ϱ_6 because a_2 is preferred to a_6 and so on. In this way we obtain the following ranked ASO program $\langle \mathcal{P}, \Phi_1, \Phi_2, \Phi_3, \Phi_4, \Phi_5, \Phi_6 \rangle$, where $\Phi_1 = \{\varrho_2\}$, $\Phi_2 = \{\varrho_1\}$, $\Phi_3 = \{\varrho_6\}$, $\Phi_4 = \{\varrho_5\}$, $\Phi_5 = \{\varrho_3\}$ and $\Phi_6 = \{\varrho_4\}$, whereas the generating program is the same. By applying the ASO semantics we obtain that $M = \{root, a_{1,2}, a_{3,4}, a_{5,6}, a_1, a_2, a_3, a_4, a_6, c_1, c_5, c_6, c_8, c_{13}\}$ is the optimal answer set, intuitively, M is the preferred answer set as it contains a_6 which is preferred w.r.t. a_5 and the best options for each preference rule $\varrho_1 \dots \varrho_4$ and ϱ_6. Concluding, $\{c_1, c_5, c_6, c_8, c_{13}\}$ is the preferred set of countermeasures. △

5 Conclusion and future works

In this paper we propose the use of two qualitative instruments for the selection of defense strategies to protect an IT system from the risk of attacks: we use defense trees to model attack/defense scenarios and CP-nets to model qualitative conditional preference over attacks and countermeasures.

Defense trees allow us to represent the possible attack actions that a person can perform to damage an asset of an IT system, and the possible countermeasures able to stop each action.

CP-nets, instead, allow us to model conditional preferences over attacks and countermeasures, and the qualitative preferences of a system administrator in the selection of countermeasures for each attack.

Our idea is to use CP-nets to model the selection of the countermeasures represented in a defense tree. We propose two methods for the composition of these preferences: an `and`-composition to model the preference order in the case of an `and`-attack, and an `or`-composition in the case of an `or`-attack.
The first operation we propose is an `and`-composition, we combine the preference order over the countermeasures represented in the CP-nets associated to attack actions composing the `and`-attack. In this case we also take into consideration the preference between attacks with the purpose of determining the preferred countermeasures. The second operation we propose is an `or`-composition of the CP-nets: we are interested in determining a preference order over sets of countermeasures which are able to cover all actions of an `or`-attack. In particular, we prefer countermeasures which are able to mitigate the risks associated with more than one action.

This paper also discusses the use of ASO programs to represent CP-defense trees, and to reason about them. The ASO approach uses preference rules in order to express the preference relations among the combinations of atoms and introduces the preference order among these rules. The ASO semantics gives simple and an intuitive way for modelling a CP-defense tree and is expressive enough to formulate preferences over countermeasures w.r.t. attacks.

The methodology presented in this paper provides a basis for future work along the following directions.

In a future version of this paper we will give formal proofs about the soundness and the completeness of the translation of CP-defense trees into ASO programs. When presenting the translation method in Section 4 we assumed that (i) attacks are totally ordered, and (ii) the set of countermeasures occurring in every

conditional preference rule is totally ordered. The application of the translation method to the more general case requires further investigation.

We faced the problem of determining defense strategies composed of sets of countermeasures (one for each attack) and only in the case of an `or`-attack we studied how a single countermeasure can cover more than one attack. A possible extension of this work is to investigate how to find sets of countermeasures able to mitigate sets of attacks, also in the case of `or`-attacks.

In this paper we use CP-nets considering only *strict* partial orders, a possible extension can be the use of *non strict partial orders* to model the preference over countermeasures and attacks, and the use of indifference between this variables (for instance $a_1 \succeq a_2$).

We also plan to consider attacks as uncertain variables. In [10] an approach is described to model a real-life problem as a set of variables with finite domains and a set of soft constraints among subsets of the variables. A variable is uncertain if we cannot decide its value. In this case, they associated a possibility degree to each value in its domain, which will tell how plausible it is that the variable will get that value.

References

1. S. Bistarelli, F. Fioravanti, and P. Peretti. Defense tree for economic evaluations of security investment. In *1st Int. Conf. on Availability, Reliability and Security (ARES'06)*, pages 416–423, 2006.
2. S. Bistarelli, F. Fioravanti, and P. Peretti. Using cp-nets as a guide for countermeasure selection. In *ACM Symp. on Applied Computing*, pages 300–304, 2007.
3. C. Boutilier, R. I. Brafman, C. Domshlak, H. Hoos, and D. Poole. Cp-nets: A tool for representing and reasoning with conditional ceteris paribus preference statements. *JAIR*, 21, 2004.
4. C. Boutilier, R. I. Brafman, C. Domshlak, H. Hoos, and D. Poole. Preference-based constrained optimization with cp-nets. *Computational Intelligence, Sp.I. on Preferences*, 2(200):137–157, 2004.
5. C. Boutilier, R. I. Brafman, H. Hoos, and D. Poole. Reasoning with conditional ceteris paribus preference statements. In *15th Conf. on Uncertainty in Artificial Intelligence*, pages 71–80, 1999.
6. G. Brewka, I. Niemela, and M. Truszczynski. Answer set optimization. In *18th Int. Joint Conf. on Artificial Intelligence*, pages 867–872. Morgan Kaufmann, 2003.
7. L. Caroprese, I. Trubitsyna, and E. Zumpano. Implementing prioritized reasoning in logic programming. In *ICEIS (2)*, pages 94–100, 2007.
8. J. Caulkins, E. Hough, N. Mead, and H. Osman. Optimizing investments in security countermeasures: A practical tool for fixed budgets. *IEEE Security and Privacy*, 5(5):57–60, 2007.
9. R. W. Floyd. Non-deterministic algorithms. *J. Assoc. Comp.*, pages 636–644, 1967.
10. M.S. Pini, F. Rossi, and K.B. Venable. Possibility theory for reasoning about uncertain soft constraints. In *ECSQARU*, pages 800–811, 2005.
11. Mehmet Sahinoglu. Security meter: A practical decision-tree model to quantify risk. *IEEE Security and Privacy*, 3(3):18–24, 2005.
12. B. Schneier. Attack trees: Modeling security threats. Dr. Dobb's Journal, 1999.