Available online at www.sciencedirect.com
ScienceDirect
Electronic Notes in Theoretical Computer Science
ELSEVIER
Electronic Notes in Theoretical Computer Science 197 (2008) 121–129
www.elsevier.com/locate/entcs

Analyzing Security Scenarios Using Defence Trees and Answer Set Programming

Stefano Bistarelli[a,b,1] Pamela Peretti[a,2] Irina Trubitsyna[c,3]

[a] *Dipartimento di Scienze, Università degli Studi "G. D'Annunzio", Pescara, Italy*

[b] *Istituto di Informatica e Telematica, CNR Pisa, Italy*

[c] *DEIS, Università della Calabria, Rende, Italy*

Abstract

Defence trees are used to represent attack and defence strategies in security scenarios; the aim in such scenarios is to select the *best* set of countermeasures that are able to stop all the vulnerabilities.
In order to represent preferences among possible countermeasures of a given attack, defence trees are enriched with conditional preferences, obtaining a new structure called CP-defence tree. In this paper we transform a CP-defence tree with preferences among attacks and countermeasures in an *Answer Set Optimization* (ASO) program. The ASO program, representing the overall scenario, is a special composition of the programs associated to each branch of a CP-defence tree. We describe an implementation that select the best set of countermeasure able to mitigate all the vulnerabilities by computing the optimal answer set of the corresponding ASO program.

Keywords: Defence tree, Answer Set Programming, CR-Prolog.

1 Introduction

Security has become today a fundamental part of the enterprise investment. In fact, more and more cases are reported showing the importance of assuring an adequate level of protection to the enterprise's assets. *Defence trees*, DT [2], an extension of attack tree [9], have been introduced as a methodology for the analysis of attack/defence security scenarios. A DT is an *and-or* tree, where leaves node represent the vulnerabilities and the set of countermeasures available for their mitigation, `and` nodes represent attacks composed by a set of actions that have to be performed as a whole to damage the system, `or` nodes represent attacks composed by a set of alternative actions that lead to damage the system.

[1] Email: bista@sci.unich.it
[2] Email: peretti@sci.unich.it
[3] Email: irina@deis.unical.it

1571-0661/$ – see front matter
doi:10.1016/j.entcs.2007.12.021

The preference among countermeasures and the dependency between attacks and countermeasures are modelled by an *answer set optimization* (ASO) [5] program. The `and` and `or` composition of the branch is then obtained by a syntactic composition of the ASO programs, whose semantics completely respects the intended meaning given in [3]. 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 use of ranked ASO programs avoids the problem of cycle; in fact, the introduction of meta-preferences gives precedence to the adoption of countermeasures covering the more dangerous of the attacks (and removing in this way the possibility to obtain a cycle).

This paper extends a short version [4], providing an algorithm transforming the CP-defence tree in a corresponding ASO program and describing its implementation.

2 Answer Set Optimization

An answer set optimization program (ASO) is a pair $\langle \mathcal{P}, \Phi \rangle$, where $\mathcal{P}$ is an answer set program [7], while Φ consists of a finite set of preference rules of the form $\varrho : \ C_1 > \cdots > C_k \leftarrow body$, where *body* is a conjunction of literals, i.e. atoms or negation of atoms, and C_is are (combinations) of literals. A preference rule $\varrho \in \Phi$ introduces a preference order between $C_1, ..., C_k$: C_i is preferred to C_j, for $i < j$ and $i, j \in [1..k]$. Intuitively, the answer sets of $\mathcal{P}$ are the (inclusion) minimal sets of atoms satisfying rules in $\mathcal{P}$ and describe the possible solutions of the program, while Φ determines a preference ordering on them.

Let $\Phi = \{\varrho_1, ..., \varrho_n\}$ be a preference program and S be an answer set, then S induces a *satisfaction vector* $V_S = (v_S(\varrho_1), ..., v_S(\varrho_n))$ where:

(i) $v_S(\varrho_j) = I$, if ϱ_j is *Irrelevant* to S, i.e. (a) the body of ϱ_j is not satisfied in S or (b) the body of ϱ_j is satisfied, but none of the C_is is satisfied in S.

(ii) $v_S(\varrho_j) = min\{i : S \models C_i\}$, otherwise. The satisfaction vectors are used to compare the answer sets.

Let S_1 and S_2 be two answer sets, then (i) $S_1 \geq S_2$ if $V_{S_1} \leq V_{S_2}$, i.e. if $v_{S_1}(\varrho_i) \leq v_{S_2}(\varrho_i)$ for every $i \in [1..n]$; (ii) $S_1 > S_2$ if $V_{S_1} < V_{S_2}$, i.e. if $V_{S_1} \leq V_{S_2}$ and for some $i \in [1..n]$ $v_{S_1}(\varrho_i) < v_{S_2}(\varrho_i)$.
A set of literals S is an *optimal answer set* of an ASO program $\langle \mathcal{P}, \Phi \rangle$ if S is an answer set of $\mathcal{P}$ and there is no answer set S' of $\mathcal{P}$ such then $S' > S$.

The ASO strategy is further extended by introducing *meta-preferences* among preference rules: a ranked ASO program is a sequence $\langle \mathcal{P}, \Phi_1, ..., \Phi_n \rangle$ consisting of a logic program $\mathcal{P}$ and a sequence of pairwise disjoint preference programs Φ_i. The rank of a rule $r \in \Phi_1 \cup \cdots \cup \Phi_n$, denoted $rank(r)$, is the unique integer i for which $r \in \Phi_i$. Intuitively, the rank of the preference rule determines its importance: preference rules with lower rank are preferred over preference rules with higher rank. The optimal answer sets can be obtained in the following way: firstly, all answer

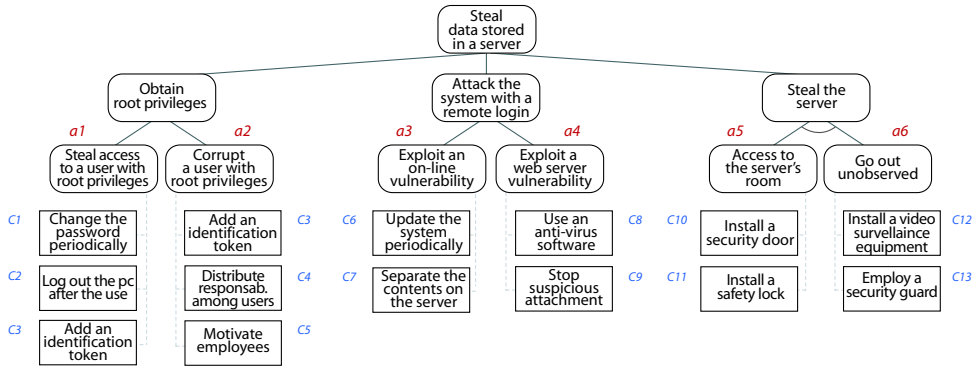


Fig. 1. An example of defence tree.

sets optimal w.r.t. Φ_1 have to be selected; then, have to be selected the ones optimal w.r.t. Φ_2; and so on. More formally, $S_1 \geq_{rank} S_2$ if for every preference rule r' such that $v_{S_1}(r') \leq v_{S_2}(r')$ does not hold, there is a rule r'' such that $rank(r'') < rank(r')$ and $v_{S_1}(r'') < v_{S_2}(r'')$.

3 CP-defence trees

Defence trees [2] are used to represent attack strategies that can be used as mitigation factor. The root of the tree is associated with an asset of the IT system under consideration and represents the attacker's goal. 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) can be of two different types: `or`-nodes and `and`-nodes. Subgoals associated with `or`-nodes are completed as soon as any of its child nodes is achieved, while `and`-nodes represent subgoals which require all of its child nodes to be completed (in the following we draw an horizontal line between the arcs connecting an `and`-node to its children to distinguish it from an `or`-node). Every leaf node has a set of countermeasures. Each countermeasure associated with a leaf node represents a possible way of mitigating risk in an attack scenario where that specific vulnerability is used.

Notice that in order to mitigate the risks deriving from an `or`-attack, the system administrator has to introduce into the system *a countermeasure for each possible action of the branch*. To mitigate the risks associated with an `and`-attack, instead, it *is enough to introduce a countermeasure for one of the attack actions* in the `and`-attack to stop the entire attack.

Example 3.1 An enterprise's server is used to store information about customers. In order to steal these data an attacker can follow different attack strategies: (i) obtain the privileges on the server, by stealing a root user credentials (a_1) or by corrupting a root user (a_2); (ii) attack the system with a remote login, by exploiting an on-line vulnerability (a_3) or by exploiting a web server vulnerability (a_4); (iii) steal the server itself and go out unobserved (a_5 *and* a_6). For each attack strategy the system administrator can use different countermeasures: (i) to reduce the risk of credentials theft he can introduce security policies like a frequent change of password

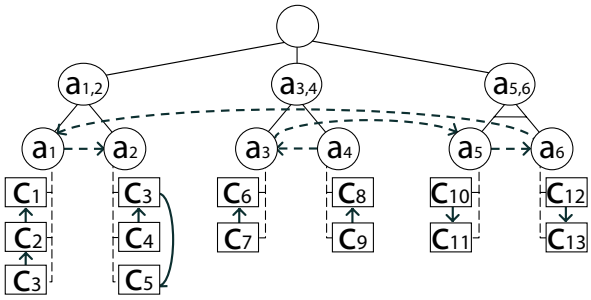


Fig. 2. A CP-defence tree.

(c_1) or the use of an identification token (c_3); (ii) to reduce the risk of a remote attack he can update the system periodically (c_6), use an anti-virus software (c_8), control the email traffic and separate the content on different servers (c_7); (iii) to reduce the risk of the theft of the server itself the system administrator can protect the server's room using a security door or a safety look (c_{10} and c_{11}), install a video surveillance equipment (c_{12}) or employ a security guard (c_{13}).

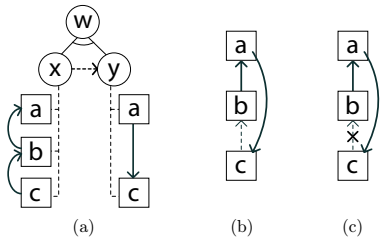
Figure 1 shows the attack/defence scenario: rounded-box nodes denote the attack strategies and the different actions the attacker needs to perform, while square box denote the different countermeasures the system administrator can adopt. △

CP-defence tree [3,4] is an extension of DT able to represent preferences among countermeasures and actions. An example is described in Figure 2. The dotted arrows depicted into the figure describe the preference order of protecting from each attack $a_i \in A$: for instance the system administrator prefers to protect the action a_2 to the action a_1 because action a_2 id more dangerous than action a_2. The solid arrows, instead, represent the preferences among the countermeasures, that protect each action: for example consider the attack action a_1, in this case the countermeasure c_1 is preferred to c_2 that is preferred to c_3 (for instance, because c_1 cost less than c_2 that cost less than c_3). Notice that the arrows are directed from the less preferred to the more preferred outcome.

and/or composition of attacks

The protection from the `and`-attack consists in the protection from one of the actions composing the `and`-attack. Given the `and`-attack, composed by two attack actions u and v, where u is more dangerous than v ($u \succ v$), and given the sets of countermeasures D_u and D_v, protecting from u and v respectively, and two partial order $(D_u, \succ_u)$ and $(D_v, \succ_v)$ describing the preferences among countermeasures covering u and v respectively, the ***and**-composition* of preferences is a new partial order $(D_u \cup D_v, \succ_{v \wedge v})$, where a countermeasure c is preferred to a countermeasure c' if (i) it is preferred in at least one of the partial orders $(D_u, \succ_u)$, $(D_v, \succ_v)$, i.e. $c \succ_u c'$ or $c \succ_v c'$, otherwise (ii) c is the worst countermeasure in $(D_u, \succ_u)$, while c' is the best countermeasure in $(D_v, \succ_v)$, i.e $\forall x \in D_u$, $x \not\succ_u c$ and $\forall y \in D_v$, $c' \not\succ_v y$.

As an example consider the `and`-attack w, composed by two actions x and y, where $y \succ x$, presented in Figure 3(a). The order obtained by `and`-composition is



$\mathcal{P}$: r_1 : w ← .
r_2 : x ∨ y ← w.
r_3 : a ∨ b ∨ c ← x.
r_4 : a ∨ c ← y.

Φ_y : c > a ← y.
Φ_x : a > b > c ← x.

Fig. 3. An example of **and** attacks (with cycles) and the corresponding ASO program.

depicted in Figure 3(b). In this case a cycle is obtained. Since the countermeasure of the worst attacks have to be considered as preferred, the cycle can be broken by removing one of the preference among the countermeasure of the less dangerous attack x. More precisely, the preference relations, described in $(D_y, \succ_y)$ has to be considered as more important, and the relation $b \succ_x c$, generating (transitively) the conflict, has to be omitted. Graphically, this correspond to removing the arc as showed in Figure 3(c). Let us now to consider how to model this by using ASO programs.

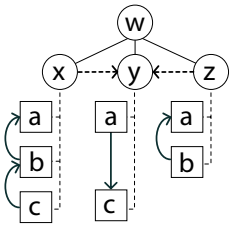
Example 3.2 Consider the **and**-attack depicted in Figure 3(a): the attack action x and the preference order over the corresponding countermeasures a, b, and c generate the ASO program $\langle \mathcal{P}_x, \Phi_x \rangle$, where $\mathcal{P}_x = \{\mathtt{r_{x_1}} : \mathtt{x} \leftarrow . \ \mathtt{r_{x_2}} : \mathtt{a} \vee \mathtt{b} \vee \mathtt{c} \leftarrow \mathtt{x}.\}$, $\Phi_x = \{\varrho_{\mathtt{x_1}} : \mathtt{a} > \mathtt{b} > \mathtt{c} \leftarrow \mathtt{x}.\}$, where the rules r_{x_1} introduces the action and the rule r_{x_2} introduces the possible countermeasures, while ϱ_{x_1} represents the preference order among them. For the attack action y, the ASO program is $\langle \mathcal{P}_y, \Phi_y \rangle$, where $\mathcal{P}_y = \{\mathtt{r_{y_1}} : \mathtt{y} \leftarrow . \ \mathtt{r_{y_2}} : \mathtt{a} \vee \mathtt{c} \leftarrow \mathtt{y}.\}$, $\Phi_y = \{\varrho_{\mathtt{y_1}} : \mathtt{c} > \mathtt{a} \leftarrow \mathtt{y}.\}$. In order to model the **and**-node presented in Figure 3, a new program $\langle \mathcal{P}, \Phi_y, \Phi_x \rangle$ is generated combining the rules in $\langle \mathcal{P}_x, \Phi_x \rangle$ and $\langle \mathcal{P}_y, \Phi_y \rangle$ (see Figure 3). $\mathcal{P}$ 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 should be stopped. The others rules are a simple 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 $V_{M_1} = [\infty, 1]$ [4], $V_{M_2} = [\infty, 2]$, $V_{M_3} = [\infty, 3]$, $V_{M_4} = [1, \infty]$ and $V_{M_5} = [2, \infty]$, reporting the satisfaction degree of each preference rule in the answer sets. Considering both the rules the order among the answer set is $M_4 > M_5 > M_1 > M_2 > M_3$ [5]. Concluding, the new order among the countermeasures is $c > a > b$. △

The protection from **or**-attack consists in the protection from all the actions composing the **or**-attack. Given the **or**-attack X, composed by k actions $u_1, \ldots, u_k$, the sets of countermeasures $D_{u_1}, \ldots, D_{u_k}$ protecting $u_1, \ldots, u_k$ respectively, and the orders among countermeasure $(D_{u_i}, \succ_{u_i})$ for $i = \{1, \ldots, k\}$, then the **or**-composition

[4] In this application the irrelevance corresponds to the worst case.

[5] Notice that both the answer set M_1 and M_5 contain countermeasure a. However, we collect the countermeasure to be applied starting from the best model, so from M_4 we collect c from M_5 we collect a, and from M_2 we collect b.



$\mathcal{P}$: r_1 : w ← .
r_2 : x ← w.
r_3 : y ← w.
r_4 : z ← w.
r_5 : a ∨ b ∨ c ← x.
r_6 : a ∨ c ← y.
r_7 : a ∨ b ← z

Φ_1 : ϱ_1 : c > a ← y.
Φ_2 : ϱ_2 : a > b > c ← x.
ϱ_3 : a > b ← z.

Fig. 4. An example of or attacks and the corresponding ASO program.

is a new order $(D_X, \succ_X)$. The order is defined over the set D_X, whose elements $C_1, ..., C_n$ are the sets of countermeasures covering all the attacks $u_1, ...u_k$, and $\succ_X$ is defined as follows: the set C is preferred to the set C' if there is a permutation π such that for all $i \in [1, \ldots, k]$, c_i is not worst than $c'_{\pi(i)}$, i.e. for $k = k'$, $\exists\pi$ s.t. $c_i \not\prec c'_{\pi(i)}$. Notice also that when the same countermeasure is selected two times (to cover two different attacks), we consider its presence only one time.

Example 3.3 Consider, for instance, the or-attack w presented in Figure 4. 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 program associated to the attack action x, y and z, and the corresponding preferences over the countermeasures a, b and c.

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

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 Figure 4). 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 action must be stopped to stop the w. The answer sets of $\mathcal{P}$ are $M_1 = \{w, x, y, z, a\}$ and $M_2 = \{w, x, y, z, b, c\}$ [6] 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 considering $\varrho_1 \in \Phi_1$, obtaining that $V_{M_2} < V_{M_1}$. Concluding, M_2 is the optimal answer set and $\{b, c\}$ is the preferred set of countermeasures.

4 From CP-defence trees to ASO programs

Given an IT system *root* and the corresponding CP-defence tree $\mathcal{T}$, the selection of the preferred defence strategy can be modelled by means of the corresponding logic program with preferences. In particular, a ranked ASO program $L(\mathcal{T}) = \langle \mathcal{P}, \Phi_1, ..., \Phi_k \rangle$ can be constructed, where $\mathcal{P}$ describes the possible defence strategies designed in $\mathcal{T}$, while Φ_1, ..Φ_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

[6] We reminded that the ASO semantics [7] only collect minimal answer sets, so among the set $M_1 = \{w, x, y, z, a\}$, $M'_1 = \{w, x, y, z, a, b\}$, $M''_1 = \{w, x, y, z, a, c\}$, $M'''_1 = \{w, x, y, z, a, b, c\}$ only M_1 is considered. Moreover, among the sets $M_2 = \{w, x, y, z, b, c\}$, $M'_2 = \{w, x, y, z, b\}$ and $M''_2 = \{w, x, y, z, c\}$ only M_2 is considered because M'_2 and M''_2 do not satisfy respectively rules r_6 and r_7.

the ASO semantics on $L(\mathcal{T})$ produces the best defence 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-defence 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 logic program consisting in the following rules r:

(i) $\mathtt{root} \leftarrow$, stating that the root of the tree must be protected;

(ii) $\mathtt{Y_1} \vee \ldots \vee \mathtt{Y_n} \leftarrow \mathtt{X}$, 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 protect to protect X;

(iii) $\mathtt{Y_i} \leftarrow \mathtt{X}$, $i = [1..n]$, 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 protect to protect X;

(iv) $\mathtt{C_1} \vee \ldots \vee \mathtt{C_n} \leftarrow \mathtt{X}$, 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 X.

Each action in the defence tree induced 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:

(i) $\mathtt{C_1} > \ldots > \mathtt{C_n} \leftarrow \mathtt{X}$, 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-defence tree of Figure 2.

Example 4.1 In this example we present an application of the ASO semantics for analyzing the attack/defence scenario of an enterprise's server used to store information about customers as we described in Example 3.1. Figure 2 shows the CP-defence tree associated to this scenario. It can be modeled by using the prioritizing program $\langle \mathcal{P}, \Phi \rangle$:

$\mathcal{P}$:
$\mathtt{root} \leftarrow$
$\mathtt{a_{1,2}} \leftarrow \mathtt{root}.$
$\mathtt{a_1} \leftarrow \mathtt{a_{1,2}}.$
$\mathtt{a_2} \leftarrow \mathtt{a_{1,2}}.$
$\mathtt{c_1} \vee \mathtt{c_2} \vee \mathtt{c_3} \leftarrow \mathtt{a_1}.$
$\mathtt{c_3} \vee \mathtt{c_4} \vee \mathtt{c_5} \leftarrow \mathtt{a_2}.$
$\mathtt{a_{3,4}} \leftarrow \mathtt{root}.$
$\mathtt{a_3} \leftarrow \mathtt{a_{3,4}}.$
$\mathtt{a_4} \leftarrow \mathtt{a_{3,4}}.$
$\mathtt{c_6} \vee \mathtt{c_7} \leftarrow \mathtt{a_3}.$
$\mathtt{c_8} \vee \mathtt{c_9} \leftarrow \mathtt{a_4}.$
$\mathtt{a_{5,6}} \leftarrow \mathtt{root}.$
$\mathtt{a_5} \vee \mathtt{a_6} \leftarrow \mathtt{a_{5,6}}.$
$\mathtt{c_{10}} \vee \mathtt{c_{11}} \leftarrow \mathtt{a_5}.$
$\mathtt{c_{13}} \vee \mathtt{c_{12}} \leftarrow \mathtt{a_6}.$

Φ :
$\varrho_1 : \mathtt{c_1} > \mathtt{c_2} > \mathtt{c_3} \leftarrow \mathtt{a_1}.$
$\varrho_2 : \mathtt{c_5} > \mathtt{c_3} > \mathtt{c_4} \leftarrow \mathtt{a_2}.$
$\varrho_3 : \mathtt{c_6} > \mathtt{c_7} \leftarrow \mathtt{a_3}.$
$\varrho_4 : \mathtt{c_8} > \mathtt{c_9} \leftarrow \mathtt{a_4}.$
$\varrho_5 : \mathtt{c_{10}} > \mathtt{c_{11}} \leftarrow \mathtt{a_5}.$
$\varrho_6 : \mathtt{c_{12}} > \mathtt{c_{13}} \leftarrow \mathtt{a_6}.$

where $\mathcal{P}$ describes the attack actions that compose the different attack strategies and the countermeasures that compose the defence strategies, while Φ establishes the preference order among them. 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

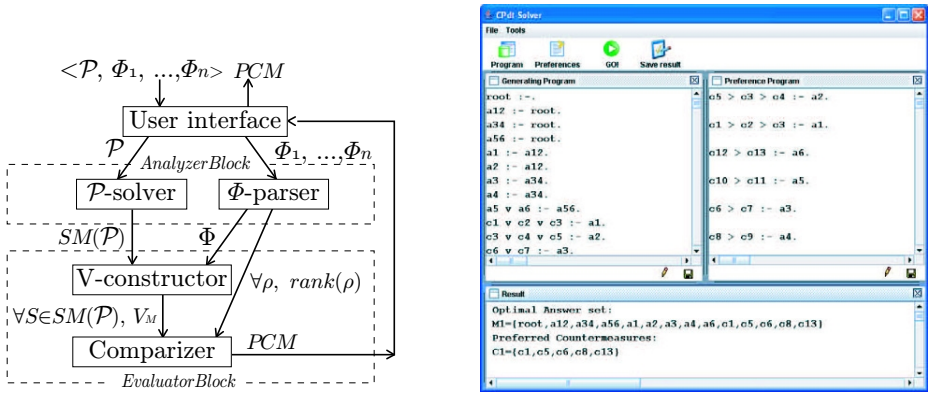


Fig. 5. The CPDT-SOLVER.

the preference order among attacks, as depicted in Figure 2, we can specify the importance of preference rules: ϱ_2 is more important than ϱ_1 because express 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 logic 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 $\varrho_1 \ldots \varrho_4$ and ϱ_6. Concluding, $\{c_1, c_5, c_6, c_8, c_{13}\}$ is the preferred set of countermeasures. △

5 Implementation

CP-defence tree can be solved by coding it in an ASO program (see Section 4) and then our ASO program solver can be used to automatically obtain the set of the best countermeasure. We use for our security scenario analysis the CPDT-SOLVER (see Figure 5(b)), an application-oriented version of CHOPPER[6], realizing ASO semantics over the ranked answer set optimization program, under the assumption that $I \equiv \infty$. The overall architecture of the CPDT-SOLVER prototype is reported in Figure 5(a).

The system can be used by means of a User Interface, $\mathcal{UI}$, which allows to specify the ranked ASO program $\langle \mathcal{P}, \Phi_1, ..., \Phi_n \rangle$, describing the CP-defence tree, and to visualize the obtained set of preferred countermeasures $\mathcal{PCM}$.

The *Analyzer Block* is constituted by two different modules: the $\mathcal{P} - solver$ and the $\Phi - parser$. The Φ-*parser* takes in input the textual file, provided by $\mathcal{UI}$, and extracts the set of preference rules identifying the rank of each rule. In the case of non ranked ASO program $rank(\varrho) = 1$ for each preference rule ϱ. The $\mathcal{P}$-*solver* receives in input the program $\mathcal{P}$ and invokes the DLV prover [8] in order to obtain the answer sets (stable models) of the program $\mathcal{P}$, $\mathcal{SM}(\mathcal{P})$.

The *Evaluator Block* is responsible for optimal answer set discovering, and is constituted by the *V-constructor* and the *Examiner*. The *V-constructor* receives in input $\mathcal{SM}(\mathcal{P})$ and the set of all preference rules, Φ, and constructs the satisfaction vector V_M for each answer set $M \in \mathcal{SM}(\mathcal{P})$. Once the satisfaction vectors have been created, their comparison is performed by means of the *Examiner* module by taken into account the rank of each preference rule. As a result, the set of optimal answer sets is established and the corresponding sets of countermeasures, $\mathcal{PCM}$, are individuated. Finally, the *Examiner* module interacts with $\mathcal{UI}$, which is in charge of memorizing the obtained result in the textual file and/or visualizing it.

6 Conclusion

This paper 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-defence tree and is expressive enough to formulate preferences over countermeasures w.r.t. attacks. Moreover, it resolves cycles owing to the conflicting partial orderings in complex CP-nets, which the earlier approach using CP-nets could not easily handle [3]. The methodology proposed is tested with a small but real example and implemented using the CPDT-SOLVER.

An alternative way consists in the use of CR-Prolog [1], a knowledge representation language based on the answer set semantic enriched with the introduction of consistency-restoring rules (cr-rules). CR-Prolog and ASO implementation are similar, both can be used for our goal. We prefer the latter one because in CR-Prolog we can specify a preference order among cr-rules but not among atoms.

References

[1] M. Balduccini and M. Gelfond. Logic programs with consistency-restoring rules. In *Int. Symp. on Logical Formalization of Commonsense Reasoning*, 2003.

[2] 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.

[3] S. Bistarelli, F. Fioravanti, and P. Peretti. Using cp-nets as a guide for countermeasure selection. In *ACM Symposium on Applied Computing*, pages 300–304, 2007.

[4] S. Bistarelli, P. Peretti, and I. Trubitsyna. Analyzing security scenarios using defence tree and answer sets. In *13th Int. Conf. on Principles and Practice of Constraint Programming*, 2007. Short paper.

[5] G. Brewka, I. Niemela, and M. Truszczynski. Answer set optimization. In *18th Int. Joint Conf. on Artificial Intelligence*, pages 867–872, 2003.

[6] L. Caroprese, I. Trubitsyna, and E. Zumpano. Implementing prioritized reasoning in logic programming. In *Proc. of the ICEIS*, 2007.

[7] M. Gelfond and V. Lifschitz. Classical negation in logic programs and disjunctive databasesg. *New Generation Computing*, 9:365–385, 1991.

[8] N. Leone, G. Pfeifer, W. Faber, F. Calimeri, T. Dell'Armi, T. Eiter, G. Gottlob, G. Ianni, G. Ielpa, K. Koch, S. Perri, and A. Polleres. The dlv system. In *Eur. Conf. JELIA'02*, pages 537–540, 2002.

[9] B. Schneier. Attack trees: Modeling security threats. Dr. Dobb's Journal, 1999.