Attack-Defence Frameworks: Argumentation-based Semantics for Attack-Defence Trees

Dov M. Gabbay[1], Ross Horne[1], Sjouke Mauw[1,2], and Leendert van der Torre[1,2]

[1] Department of Computer Science, University of Luxembourg, Luxembourg
[2] SnT, University of Luxembourg, Luxembourg

Abstract. This position paper connects the areas and communities of abstract argumentation and attack-defence trees in the area of security. Both areas deal with attacks, defence and support and both areas rely on applications dealing with human aggressive activities. The unifying idea we use in this paper is to regard arguments as AND-OR attack trees as proposed by Schneier in the security domain. The core model, which is acceptable for both communities, is a pair $(\mathbf{S}, \twoheadrightarrow)$, where $\mathbf{S}$ is a set of attack trees (the "arguments") and $\twoheadrightarrow$ is a binary relation on attack trees (the "attack" relation). This leads us to the notion of an *attack-defence framework*, which provides an argumentation-based semantics for attack-defence trees and more general attack-defence graphs.

1 Introduction

Argumentation is an interdisciplinary research area concerning the study of conflicts that arise due to competing objectives and views across a range of disciplines. Security is an obvious example of such a discipline where there are human actors with competing interests. The interests and objectives of an attacker seeking to obtain secrets, disrupt services, track users, etc., conflict with those of a defender such as system administrators, software engineers, security guards and others professionals that protect our society both online and offline.

It should be of no surprise that there are immediate parallels between argumentation and methods developed for modelling the relationships between the actions of attackers and defenders in security, notably attack-defence trees [1] and defence trees [2]. In this work we show that it is possible to provide directly a semantics for attack-defence trees by building on models of abstract argumentation. However, on the surface, there are a few differences in modelling styles in argumentation compared to attack-defence trees. Notably, in argumentation, various types of relations can be reduced to a single attack-relation tree formed of *attack* relations, whereas established semantics for attack-defence trees based on multisets collapse such trees of layers of attacks, defences, counter-attacks, etc., to a two-layer structure where there is only one layer of attacks, some of which are countered by a layer of defences. We develop *bipolar* argumentation frameworks [3] that incorporate a notion of support [4] and hence are capable of modelling in styles favoured by both the argumentation and security communities. This enables us to translate added value in both directions.

From security to argumentation. Traditionally, arguments are modelled in a fairly binary fashion: if an argument is attacked by another argument that is not attacked then it is out, hence cannot be an acceptable argument. The source of potential confusions arises in argumentation when there are loops, for example loops may be created in legal arguments where witnesses attack each other. In security, the sources of uncertainty are quite different. They come from the fact that many attacks take resources such as security guards, networking equipment, or botnets, which have associate costs, capacities and likelihoods of success. There may be other factors such as the risk of exposing the identity of the attackers leaving them open to prosecution (the feeling of impunity), balanced against the motives of a profile of attacker. For such reasons, semantics proposed for attack-defence trees typically take into account quantities and qualities in various *attribute domains* that indicate the capability of attacker and defenders to fulfil their actions. This quantitative aspect we translate from the attack-defence trees to argumentation frameworks by making explicit a notion of abstract "weapons" that represent the actions and resources that an attacker or defender can use to perpetrate attacks or hold out against them.

From argumentation to security. As mentioned above, much of the attention in the argumentation community revolves around resolving disputes when there are cycles in arguments. Thus the graph structures considered in argumentation are more flexible than the trees stratified into layers of attacks and defences, that form attack-defence trees. While it may be useful for security to incorporate loops, in this work, we take a clearer and simpler first step in that direction. We allow not only trees, but also directed acyclic graphs to appear. Such an extension of attack-defence trees is useful for making explicitly when multiple instances of nodes representing actions of an attacker are in fact the same attack, hence we need not kill all instances to counter that attack, but only the one instance of that action, which of course impacts the resource sensitive analysis [5]. A more adventurous aspect of the modest liberalisation of attack-defence trees that we propose is to forget about the distinction between attacks and defences. We simply have arguments that attack each other, and need not explicitly indicate that the argument is an attack tree associated with an attacker or defender. This allows the modelling of scenarios where two actions of an attacker may be in conflict, for example, enabling a DDoS attack may blow the cover for a stealthy attacker gathering private information from inside the system. Furthermore, a defensive action, such as installing a hypervisor, for separating processes sharing the same underlying hardware may mitigate attacks exploiting vulnerabilities in inter-process communication in software, but may support cache timing side channels at the hardware level. Going further, some nodes may not even be attackers or defenders, they may be engineering requirements such as protocol standards or legal requirements such as clauses of the GDPR regulation that are impacted by a successful attack or by adopting a particular defensive strategy.

Table 1 provides an overview comparing the security and argumentation domains from which this paper draws. Considering the above observations, since these domains were already close we believe that a relatively small step is required to build a general framework accommodating the needs of both communities — in one way we move from trees to more general graphs and in the other direction we bring in resource considerations. For example, it is reasonable that the legal domain may have some resource consideration, e.g., whether an argument stands may take into account the number of witnesses and their credibility. In the security domain, it is reasonable to lift some constraints on patterns of attacks and defences.

Table 1: Comparison between argumentation and security domains

Argumentation frameworks	**Attack-defence trees**
Argumentation is a well-developed area with a community formed over 50 years.	Strong security community using methods inspired by fault trees which have been in use for over 50 years.
Have a range of semantics.	May benefit from improved semantics.
Mainly concerned with loops.	May benefit from handling loops, or at least more general acyclic graphs.
Semantics focus on evidence for claims, i.e., proof certificates.	Could benefit from more proof theory.
Trees are a well-behaved case for this area.	Mainly concerned with trees with a stratified structure, formed by alternating layers of attacks and defences.
Emphasises attack relations, allowing arbitrary alternations between moves of attackers and defenders in their underlying games.	Reduces counter-attacks to a single layer of attacks countered by defences, by using support relations.

We develop these ideas as follows. In Section 2, we provide background on the traditional notion of an argumentation framework and make explicit obvious parallels and differences compared to attack-defence trees. In Section 3, we close the gap between the models by introducing the notion of attack-defence framework, firstly by defining what it means for one attack tree to attack another attack tree, and, secondly, by providing an algorithm accommodating the notion of support. In Section 4, we discuss the argumentation-based model introduced in juxtaposition with key examples of attack-defence trees, and highlight extension and directions enabled.

2 Preliminaries drawing from Argumentation

We briefly summarise mathematical tools of argumentation on which we build. An *argumentation framework* is a pair consisting of a set of arguments S and a relation $\twoheadrightarrow \subseteq S \times S$ called the attack relation. Argumentation traditionally defines set theoretically or algorithmically two subsets for an argumentation framework $(S, \twoheadrightarrow)$.

- The **in set** $E^+ \subseteq S$, which is a maximal (with respect to subset inclusion) conflict-free set such that: if z is such that $\forall y.(y \twoheadrightarrow z \Rightarrow \exists y' \in E \; s.t. \; y' \twoheadrightarrow y)$ then $z \in E$. I.e., any argument attacking an element of E is attacked by another element of E. By *conflict-free*, we mean that no two $x, y \in E$ are such that $x \twoheadrightarrow y$.
- The **out set** $E^- = \{ y \mid \exists x \in E^+ \; s.t. \; x \twoheadrightarrow y \}$.

If we restrict to acyclic graphs these sets partition the set of arguments, i.e., we have $E^+ \cap E^- = \varnothing$ and $E^+ \cup E^- = S$.

In the acyclic setting, the above sets can be generated algorithmically from $(S, \twoheadrightarrow)$ by calculating $E^+ = \bigcup_i E_i^+$ and $E^- = \bigcup_i E_i^-$, where E_i^+ and E_i^- are defined inductively as follows. We say x is not attacked in S_i if $\neg \exists y \in S_i \; s.t. \; y \twoheadrightarrow x$.

1. **Base case:** Let $S_0 = S$, $E_0^+ = \varnothing$ and $E_0^- = \varnothing$.

2. **Inductive case:** Let $S_{n+1} = S_n \backslash (E_n^+ \cup E_n^-)$.
   Let $E_{n+1}^+ = \{x \in S_{n+1} \mid x \text{ is not attacked in } S_{n+1}\}$.
   Let $E_{n+1}^- = \left\{x \in S_{n+1} \mid \exists y \in E_{n+1}^+ s.t.\ y \twoheadrightarrow x\right\}$.

Consider the example argumentation framework in Figure 1. The argumentation framework depicted is also an attack-defence tree [1], where, in attack-defence tree terminology, the attack relations are *countermeasures*, where an action of an attacker is defeated by an action of a defender, or an action of defender is defeated by a counter-attack of an attacker. In the figure, attack relations are represented by dotted double-headed arrows in order to align with the dotted line notation of attack-defence trees. This notation, at the same time, makes explicit the direction of the attack, as attack relation $\twoheadrightarrow$ indicates. The colours are not necessary for argumentation frameworks; they simply allow ease of reading when there is a clear alternation between two actors the proponent and opponent, i.e., the actions of the attacker and defender.

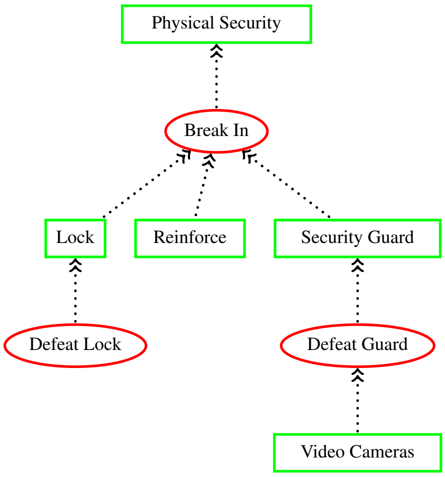


Fig. 1: An argumentation framework which is also an attack-defence tree.

For the example in Figure 1, the **in set** and **out set** are as follows.

$$E^+ = \{Video\ Camera, Defeat\ Lock, Reinforce, Security\ Guard, Physical\ Security\}$$
$$E^- = \{Defeat\ Guard, Break\ in, Lock\}$$

Thus we say *Physical Security* is an *acceptable argument* with respect to E^+, since any argument that attacks it (i.e., *Break In*) is defeated by some element of E^+, (e.g., *Security Guard*). We note that E^+ is a *maximal admissible* set, which, in argumentation terminology is called the *preferred extension*. Thus the algorithm used to generate E^+ emphasises that the preferred extension is easy to compute in the acyclic setting.

In addition to the notion of attack, we require also a notion of support in order to provide an argumentation-based semantics for attack-defence trees. In order to accommodate support — e.g., the act of supporting a security goal of a system with a range of network and physical security measures, as is possible using attack-defence trees — we take a step towards a more general model. We would like to define acyclic *bipolar argumentation frameworks*, that is a pair of relations on a set of arguments S:

$$(S, \twoheadrightarrow, \rightarrow)\text{, where } \twoheadrightarrow \subseteq S \times S, \rightarrow \subseteq S \times S \text{ and } \twoheadrightarrow \cup \rightarrow \text{ is acyclic}$$

The first relation $x \twoheadrightarrow y$ indicates that x **attacks** y. The second relation $x \rightarrow y$ represents that x **supports** y. These bipolar argumentation frameworks accommodate conventions from both argumentation and security.

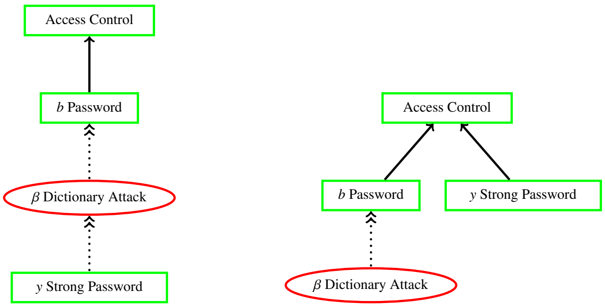


Fig. 2: Modelling counter-defence “Strong Password” as an attack or as a support.

Argumentation convention: Most argumentation approaches reduce support to attacks (i.e., eliminate support). This is achieved by reducing $y \to b$ to $y \twoheadrightarrow \beta \twoheadrightarrow b$ by making use of auxiliary node β. Thus y supports b by defending against an attacker. See for example, the bipolar argumentation framework (which happens to be also an attack-defence tree) to the left of Figure 2. In that example, the Strong Password y, attacks the Dictionary Attack β, in order to support the Password b.

Security convention: One might argue that the above approach drawing directly from argumentation is not quite the right viewpoint, since the Strong Password does not actively attack the Dictionary Attack. What really happens is that the Strong Password strengthens the password to make it more resistant to the Dictionary Attack. This idea is reflected in the existing multiset semantics for attack-defence trees [1] that eliminates counter-attacks by reducing them to supports. Under such semantics for attack-defence trees, an argumentation framework with relations as depicted to the left of Figure 2 might more accurately be modelled, as depicted in the example on the right of Figure 2. In that diagram, instead of employing Strong Password as a counter-attack for Dictionary Attack we employ it as a support for access control.

The use of the support relation from the bipolar argumentation frameworks allows the fact that the Strong Password really is supporting the Access Control mechanism rather than attacking the Dictionary Attack to be made explicit. It is a modelling choice which presentation better respects the situation, a semantics based on argumentation that accommodates support (to be developed in the next section) would likely distinguish these scenarios, i.e., the diagrams in Figure 2 may be distinguished by their “in sets” (which should be a suitable generalisation of preferred extensions). To get a feeling of the intuition behind why this should be the case, observe that in the diagram on the right of Figure 2 nobody attacks the Dictionary Attack so it should be declared “in” by default, that is $\beta \in E^+$; whereas in the diagram on the left the Dictionary Attack is out by default, since it is attacked by a Strong Password that is not attacked by anyone, hence is in by default. Thus in an extended algorithm accommodating support we expect $y \in E^+$ and $\beta \in E^-$ for the attack-defence tree on the right of Figure 2.

In contrast, instead of “Strong Password”, consider employing an anti-bruteforcing defensive mechanism, such as a CAPTCHA, against a Dictionary Attack. This could be considered to be more accurately modelled as an attack on the Dictionary Attack denoted by β rather than in terms of supporting the Access Control goal. This is a modelling choice for the security expert.

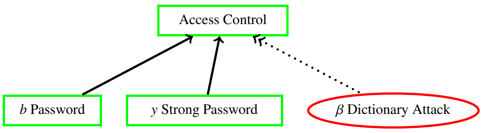


Fig. 3: The dictionary attack here attacks both the password and strong password.

Going further, the diagram in Figure 3 is an attack-defence tree. This is different from the support relation to the right of Figure 2, since by attacking directly the access control argument we suggest that both the Password and Strong Password are killed by the dictionary attack. In order to interpret such scenarios, we require richer structure than provided by traditional argumentation frameworks à la Dung [6]. In order to formally present such a semantics, further machinery is defined in the next section.

3 Attack-defence frameworks: trees attacking trees

The semantics in this section are built out of those in Section 2 and finite sets of multisets of weapons, where weapons are "actions" in attack-defence tree terminology. We start with defining enhanced argumentation frameworks *with joint attacks* (S, R), where S is viewed as a set of weapons (the atomic actions that appear at the leaves of attack trees), and R is more general than just a binary relation over S: we allow the source of the attack to be a multiset of elements of S. Thus R is a relation between finite multisets built from S, say Multiset(S), and elements of S, i.e., $R \subseteq \text{Multiset}(S) \times S$. We use the notation $a * b * c$ to represent the multiset with three elements, the weapons a, b and c.

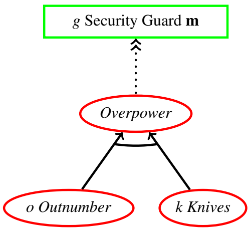


Fig. 4: A joint attack relation as an attack-defence tree.

Allowing multisets of weapons to attack weapons, allows us to model scenarios such as $o * k \; R \; g$, as depicted in Figure 4, using an auxiliary node, labeled *Overpower*. Note that while such scenarios are not expressible using traditional argumentation frames, they do appear in several richer models of argumentation, where such attack relation are called *joint attacks* [7, 8]. In the attack-defence tree notion in Figure 4, the fact that multiple actions/weapon/resources must be used together is depicted using an arc between the arrows, which is a *conjunctive refinement* in attack tree terminology [9].

We will use argumentation frameworks with joint attacks to define a semantics for another argumentation framework with more structure, which we call an *attack-defence framework*, since it will generalise attack-defence trees to more general graphical structures, in the spirit of argumentation frameworks.

We denote attack-defence frameworks as a quintuple:

$$(\mathbf{S}, \twoheadrightarrow, \rightarrow, S, :=) \quad \text{where } \twoheadrightarrow \subseteq \mathbf{S} \times \mathbf{S}, \rightarrow \subseteq \mathbf{S} \times \mathbf{S}, \text{ and } := \subseteq \mathbf{S} \times \text{Set}(\text{Multiset}(S)).$$

Its arguments $\mathbf{m} \in \mathbf{S}$, denoted in bold, are mapped by functional relation := to sets of multisets of weapons drawn from the set S (the set of weapons of the argumentation frame with joint attack above). Think of the resources assigned to arguments as basic AND-OR attack trees in the original sense of Schneier [9]. I.e., each node is part of an attack-defence tree consisting of only the actions of the attacker or those of the defender (the connected green or red components only in the example figures). Attack trees allow for actions to be:

- *conjunctively refined*, requiring several actions to be performed to realise the action refined, as denoted using multisets of actions,
- or *disjunctively refined*, where one of the possible actions in the disjunction suffices to realise the action refined, as denoted using the sets of multisets.

Thus we take the viewpoint that elements of our attack-defence frameworks represent attack trees, more precisely sets of multisets of weapons regarded as AND-OR attack trees flattened after applying the standard mapping to multisets [10] that reduces the attack tree to a disjunctive normal form. From an argumentation perspective we are essentially assuming that arguments are attack trees. These attack trees represent agents carrying each a variety of weapons, where each of these weapons are elements of S and the sets of multisets represent a choice between a number of combination of weapons that may be employed. Note this viewpoint does not distinguish between agents that are attackers or defenders, agents playing any role maybe be equipped with weapons in this manner.

3.1 Interpreting the attack relation of attack-defence frameworks

We first explain how to interpret the attack relation only, for attack-defence frameworks. Consider the following example of an attack tree denoted as a set of multisets: $\mathbf{m} := \{(a_1 * a_2), b_1, b_2\}$, where $a_1, a_2, b_1, b_2 \in S$. The above example may be regarded as the attack tree in Figure 5, where a node denoted with an arc represents conjunctive refinement, and a node without an arc represents disjunctive refinement.

We now explain the meaning of $\mathbf{m}$. We are relying in underlying argumentation frameworks with joint attacks of the form (S, R) in order to provide a semantics for the attack relation.

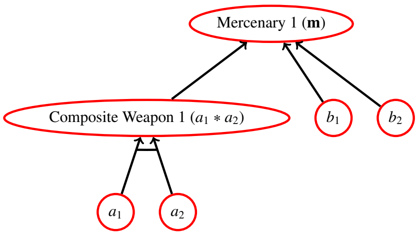


Fig. 5: An attack tree denoted by $\{(a_1 * a_2), b_1, b_2\}$. The resources accumulated at each node are indicated in brackets.

*1) The meaning of $\mathbf{m}$ is a collection of three weapons. The first weapon is a composite weapon built up of two component weapons a_1 and a_2 denoted by $a_1 * a_2$. Note that $*$ is used to denote a multiset consisting of two elements a_1 and a_2. The second weapon is b_1 and the third is b_2.

*2) So, if we want to attack $\mathbf{m}$ we need to attack all three components' weapons and leave $\mathbf{m}$ without weapons. This complements that perspective that, if $\mathbf{m}$ were to be used to attack another attack tree, there are three options for executing the attack and hence, if it is not the case that all three attack options are defeated, then the attack may be perpetrated.

Expressions like $\{(a_1 * a_2), b_1, b_2\}$ are understood as resource weapons, which can be used for attack or for defence. $(a_1 * a_2)$ is a composite weapon which has two components. So to neutralise the composite weapon $(a_1 * a_2)$ we need to kill at least one of its components, and to attack **m** we must attack each of its weapons.

So, if $\mathbf{n} := \{(\alpha_1 * \alpha_2 * \alpha_3), \delta_1, \delta_2\}$ is the set of weapons of another argument, say Mercenary 2, keen to attack **m**, say Mercenary 1, then for **n** to attack **m** it must attack each of **n**'s weapons. This scenario may be represented using the attack-defence tree in Figure 6.

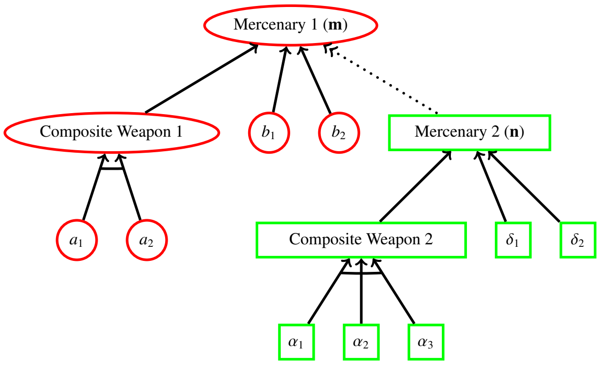


Fig. 6: An attack tree attacking another attack tree.

*3) So, we have:

$$\begin{aligned} \mathbf{n} \twoheadrightarrow \mathbf{m} \ \text{ iff } \ & \mathbf{n} \twoheadrightarrow (a_1 * a_2) \\ & \text{and } \mathbf{n} \twoheadrightarrow b_1 \\ & \text{and } \mathbf{n} \twoheadrightarrow b_2 \end{aligned}$$

So for **n** to attack any single weapon x (such as one of the weapons in **m**), we need a weapon in **n** to attack x. So we follow rule *4):

*4) We interpret disjunctive attacks [11] and attacks on multisets as follows.

$$\begin{aligned} \{z, y\} \twoheadrightarrow x & \text{ iff def. } \ zRx \vee yRx \\ u \twoheadrightarrow z * y & \ \text{ iff def. } \ uRz \vee uRy \end{aligned}$$

where $z * y$ is a weapon with two components. So, for example

$$u \twoheadrightarrow \{(z * y), w\} \text{ iff } (u \twoheadrightarrow (z * y) \text{ and } uRw) \text{ iff } ((uRz \vee uRy) \wedge uRw).$$

Therefore we have

*5) $\{(\alpha_1 * \alpha_2 * \alpha_3), \delta_1, \delta_2\} \twoheadrightarrow x$ iff $[\delta_1 Rx$ or $\delta_2 Rx$ or $(\alpha_1 * \alpha_2 * \alpha_3)Rx]$, and

*6) $\mathbf{n} \twoheadrightarrow \{(a_1 * a_2), b_1, b_2\}$ iff $\mathbf{n} \twoheadrightarrow (a_1 * a_2)$ and $\mathbf{n} \twoheadrightarrow b_1$ and $\mathbf{n} \twoheadrightarrow b_2$.

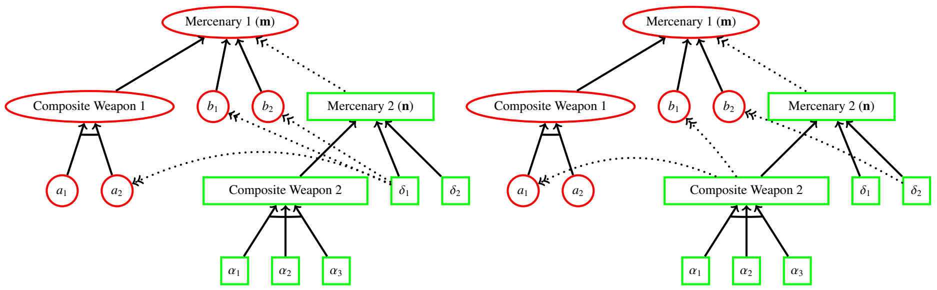


Fig. 7: Two possible joint attack relations realising the attack in Fig. 6.

This gives a full meaning to $\mathbf{n} \twoheadrightarrow \mathbf{m}$ in terms of the underlying argumentation frame *with joint attacks* that can realise the attack relations indicated, where the attack relation is restricted to a multiset of weapons attacking individual weapons.

Thus for the above example in Figure 6, one such underlying argumentation framework generated from $\mathbf{n} \twoheadrightarrow \mathbf{m}$ is the following relation R_1.

$$\delta_1\ R_1\ a_2 \qquad \delta_1\ R_1\ b_1 \qquad \delta_1\ R_1\ b_2$$

Another example that would also realise the attack $\mathbf{n} \twoheadrightarrow \mathbf{m}$ would be relation R_2 defined as follows.

$$\alpha_1 * \alpha_2 * \alpha_3\ R_2\ b_1 \qquad \alpha_1 * \alpha_2 * \alpha_3\ R_2\ a_1 \qquad \delta_2\ R_2\ b_2$$

These two possible joint attack relations realising the attack in Figure 6 are depicted by the respective diagrams in Figure 7, as indicated by the overlaid attack relations from multisets of weapons in attack tree $\mathbf{n}$ to weapons in $\mathbf{m}$. Obviously, this is not an exhaustive list of joint attack relations; indeed there are 54 such joint attack relations realising the attack between the trees in this example. It is sufficient for one of those joint attack relations to be realisable in practice, in order for the attack $\mathbf{n} \twoheadrightarrow \mathbf{m}$ to be realisable in practice.

Following the method illustrated above, it is clear that we can give a semantics for the attack relation on attack-defence frameworks, where trees may attack trees in terms of a set of argumentation frameworks with joint attacks.

For a further example consider Figure 8. Here we have, according to our weapon interpretation the following.

$$\mathbf{b} := \{b_1, b_2\} \quad \beta := \{\alpha_1, \alpha_2\} \qquad \mathbf{x} := \{x_1, x_2\}.$$

We get $\{b_1, b_2\} \twoheadrightarrow \{\alpha_1, \alpha_2\} \twoheadrightarrow \{x_1, x_2\}$.

α_1, α_2 are used as weapons to kill $\{x_1, x_2\}$. So the meaning of $\{\alpha_1, \alpha_2\} \twoheadrightarrow \{x_1, x_2\}$ is $(\alpha_1 R x_1 \wedge \alpha_1 R x_2) \vee (\alpha_1 R x_1 \wedge \alpha_2 R x_2) \vee (\alpha_2 R x_1 \wedge \alpha_2 R x_2) \vee (\alpha_2 R x_1 \wedge \alpha_1 R x_2)$.

The meaning of $\{b_1, b_2\} \twoheadrightarrow \{\alpha_1, \alpha_2\}$ is similar, namely $(b_1 R \alpha_1 \wedge b_1 R \alpha_2) \vee (b_1 R \alpha_1 \wedge b_2 R \alpha_2) \vee (b_2 R \alpha_1 \wedge b_2 R \alpha_2) \vee (b_2 R \alpha_1 \wedge b_1 R \alpha_2)$.

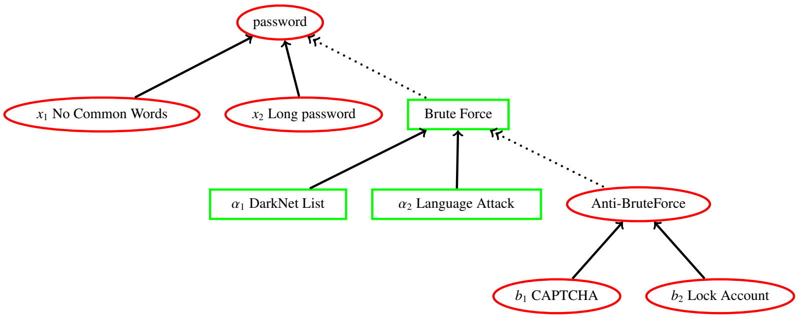


Fig. 8: A variation on Figure 3, where each node has a choice of weapon to employ.

Putting the above together, an argumentation framework that realises the above constraints is depicted in Figure 9. That is, we have an attack relation R such that:

$$b_1 \; R \; \alpha_1 \quad b_1 \; R \; \alpha_2 \quad \alpha_1 \; R \; x_1 \quad \alpha_2 \; R \; x_2$$

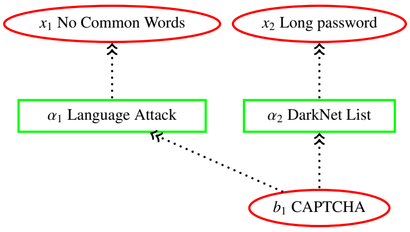


Fig. 9: An example of an argumentation framework realising the attack relations in Figure 8. Notice that the argumentation framework generated need not be a tree.

For the argumentation framework defined by R it is clear that we can ask traditional argumentation questions such as: what is the preferred extension, i.e., the in set E^+, for the realisation of the attack-defence framework in Figure 8, as given in Figure 9. The preferred extension is of course the following set.

$$E^+ = \{b_1, x_1, x_2\}$$

Notice that, since $\mathbf{x} := \{x_1, x_2\}$, for $\mathbf{x}$ to be acceptable it is sufficient that x_1 or x_2 is an acceptable argument. Thus since x_1 and x_2 both happen to be acceptable with respect to E^+, we can claim that $\mathbf{x}$ is acceptable with respect to E^+. Similarly, since $\mathbf{b} := \{b_1, b_2\}$ and b_1 is acceptable with respect to E^+ we have $\mathbf{b}$ is acceptable with respect to E^+ (recall sets represent a disjunctive refinement in attack trees, i.e., a choice of possible attacks, so it is sufficient for one multiset in the set of multisets to be acceptable). In contrast, since $\beta := \{\alpha_1, \alpha_2\}$ and neither α_1 nor α_2 is acceptable with respect to E^+ (equivalently they are both in E^-), β is not an acceptable argument with respect to E^+.

In summary, in this example $\mathbf{b}$ and $\mathbf{x}$ are "in" and β is "out". This is exactly as expected for the traditional argumentation frame, in the sense described in Section 2, where we take $\mathbf{b}$, β and $\mathbf{x}$ to be atomic arguments and define the attack relation as $\mathbf{b} \twoheadrightarrow \beta \twoheadrightarrow \mathbf{x}$. The reason, or evidence for the admissibility of $\mathbf{x}$ is however now more fine grained, reflecting the more fine grained nature of the arguments. Since for each underlying argumentation framework E^+ is unique in this acyclic setting, it is sufficient to say "**x** is an acceptable argument with respect to the argumentation framework defined by R", where one such R is depicted in Figure 9.

3.2 Interpreting the support relation

But what about support? Here we explain support and provide a semantics in terms of an algorithm rewriting the attack-defence frameworks introduced in this work, inspired by the traditional algorithm for argumentation frameworks at the top of Section 2. We make use of the attack tree in Figure 10 to guide the development of an algorithm suitable for both the security and argumentation communities. The example features Cloudbursting, which is the practice of scaling a service temporarily to the Cloud, so as to cope with spikes of demand and to sit out distributed denial of service attacks (DDoS) [12].

In our algorithm, we make use of the concept of a belt. A *belt (a maximal anti-chain)* for a bipolar argumentation frame is a set $B \subseteq S$ such that:

- For no $x, y \in B$ does $x \twoheadrightarrow y$ or $x \rightarrow y$ hold. Hence B is *conflict free*.
- every $z \in S$ is either below or above or in B.

To understand the terminology "maximal anti-chain" used above observe the following. A *maximal chain* in $(S, \twoheadrightarrow)$ is a maximal sequence $x_1, x_2, \dots x_n$ such that for all $i = 1, \dots n-1$, $x_i \twoheadrightarrow x_{i+1}$. Thus every maximal chain containing an argument z intersects a belt B in exactly one point.

The idea is that we start with the belt consisting of all arguments that are not attacked or supported, i.e., the sources of the graph. We then move forwards across the attack-defence framework to another belt reachable by realising the attack and support relation of a node in that belt.

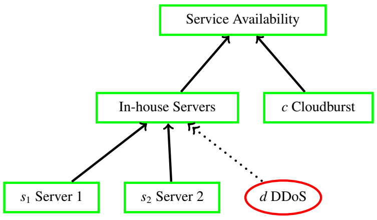


Fig. 10: A bipolar argumentation framework involving support.

We firstly illustrate what our algorithm should do on the attack-defence framework in Figure 10. To be precise, we specify this attack-defence framework as the following quadruple defined in the bullet points below.

- The abstract arguments, i.e., the nodes of the graph:

$$\{\text{Service Availability}, \text{In-house Servers}, \text{Server 1}, \text{Server 2}, \text{Cloudburst}, \text{DDoS}\}$$

- The attack relation:

$$\text{DDoS} \twoheadrightarrow \text{In-house Servers}$$

- The support relation:

$$\text{In-house Servers} \rightarrow \text{Service Availability} \quad \text{Cloudburst} \rightarrow \text{Service Availability}$$

$$\text{Server 1} \rightarrow \text{In-house Servers} \quad \text{Server 2} \rightarrow \text{In-house Servers}$$

- The (initial) resource assignment:

$$\begin{aligned} \text{Service Availability} &\coloneqq \varnothing \\ \text{In-house Servers} &\coloneqq \varnothing \\ \text{Server 1} &\coloneqq \{s_1\} \\ \text{Server 2} &\coloneqq \{s_2\} \\ \text{Cloudburst} &\coloneqq \{c\} \\ \text{DDoS} &\coloneqq \{d\} \end{aligned}$$

Remark 1. Observe that Service Availability and In-house Servers are initially assigned no resources, as indicated by the empty set. This is because these arguments have no resources inherently, by themselves, instead they inherit their resources via support relations from the arguments Server 1, Server 2, and Cloudburst. Thus if we consider the argumentation frame without the DDoS node, we could represent this scenario using the attack tree consisting of the single node Service Availability, assigned the resources $\{s_1, s_2, c\}$. The advantage of using explicit support relations rather than a single node is that we can employ more fine-grained precision indicating that the in-house servers are affected by the DDoS attack, but Cloudbursting is not affected by a DDoS attack.

Observe also that the sub-framework consisting of “In-house Servers”, “Server 1” and “Server 2” could alternatively be modelled by a single node “In-house Servers” with resource assignment $\{s_1, s_2\}$. That modelling decision would not changing the meaning of the tree, since the DDoS attack takes out both servers indiscriminately.

We propose an algorithm that executes as follows on Figure 10.

Initialisation: The initial belt (those nodes that are not attacked or supported by any other node) is defined as follows:

$$\{\text{Cloudburst}, \text{DDoS}, \text{Server 1}, \text{Server 2}\}$$

Note all of these arguments should correspond to leaves of some attack-defence tree and hence should have resources assigned to them, which is indeed the case for this example.

Step 1: We consider some belt reachable from the initial belt by taking at most one step away from the initial belt, with respect to the attacks and supports. For this example, there is only one choice: the following belt, which is reachable by the attack and support relations in one step:

$$\{\text{Cloudburst}, \text{In-house Servers}\}$$

Notice that argument Cloudburst does not advance, if it were to advance we would not have a belt. Firstly, we update the attack-defence framework to reflect the support relations resulting in the attack defence framework where the resources assigned to “Server 1” and “Server 2” are sent to “In-house Servers” — resulting in the new annotation $\{s_1, s_2\}$ for that node.

Secondly, we apply the construction from the previous section to generate an argumentation framework (with joint attacks) based on the weapons given by the resource assignment. I.e., we interpret $d \twoheadrightarrow \{s_1, s_2\}$, thereby generating the relation R_1 consisting of $d\ R_1\ s_1$ and $d\ R_1\ s_2$.

Step 2: As with Step 1 above, we progress to the next belt:

$$\{\text{Service Availability}\}$$

This belt is reachable by two supports from the nodes of the previous belt Cloudburst and In-house Servers, hence we update the resources, assigned to “Service Availability,” by sending the resources from “In-house Servers” and “Cloudburst,” resulting in the annotation $\{s_1, s_2, c\}$ for the node “Service Availability.”

Since there are no attacks for this iteration of the algorithm $R_2 = R_1$.

Output of Algorithm: The result of running the algorithm is an updated *resource assignment* as follows:

$$\begin{aligned}\text{Service Availability} &:= \{s_1, s_2, c\}\\ \text{In-house Servers} &:= \{s_1, s_2\}\\ \text{Server 1} &:= \{s_1\}\\ \text{Server 2} &:= \{s_2\}\\ \text{Cloudburst} &:= \{c\}\\ \text{DDoS} &:= \{d\}\end{aligned}$$

This is accompanied by the argumentation framework on weapons, with relation R defined as.

$$d\ R\ s_1 \qquad d\ R\ s_2$$

Note that, in general there could be a set of argumentation frameworks with joint attacks generated; but, in this case, there is only a single choice of argumentation framework.

Analysis: Consider the output of the algorithm and observe that, since nothing attacks d or c in R they are both elements of preferred extension E^+. Hence c is acceptable with respect to E^+ in the conventional sense of the argumentation framework defined by R_2. Going further, since the argument Service Availability of the updated attack-defence framework has resource annotation $\{s_1, s_2, c\}$ and c is acceptable with respect to E^+, we can say that “*Service Availability* is an acceptable argument with respect E^+.”

We reinterpret the above from the perspective of security. The preferred extension says that if a DDoS attack is active and the option to Cloudbursting is available then we have Service Availability.

3.3 An algorithm for attack-defence frameworks, in its general form

We now distil the general algorithm from the above worked examples.

The input: An attack-defence framework $(\mathbf{S}, \twoheadrightarrow, \rightarrow, S, :=)$. I.e., a bipolar argumentation framework $(\mathbf{S}, \twoheadrightarrow, \rightarrow)$ with a resource assignment := mapping arguments to sets of multisets of weapons built from the atoms in S.

Remark 2. The attack-defence framework may be generated from an attack-defence tree, by assigning a singleton atomic weapon to each action and the empty set of resources to each node in the attack-defence tree. However, more general acyclic graphs of relations and more detailed resource assignments are also permitted.

The initialisation: We define the initial mapping $:=_0$, belt B_0 and set of joint attack relation $\mathcal{R}_0$ as follows.

- $:=_0 = :=$
- $B_0 = \{\mathbf{x} \colon \mathbf{x} \in \mathbf{S} \text{ and there is no } \mathbf{y} \in \mathbf{S} \text{ such that } \mathbf{y} \twoheadrightarrow \mathbf{x} \text{ or } \mathbf{y} \to \mathbf{x}\}$
- $\mathcal{R}_0 = \{\emptyset\}$

The inductive step: Let B_{n+1} be a belt (not necessarily uniquely defined) such that $B_{n+1} \neq B_n$ and for all $\mathbf{y} \in B_{n+1}$ there exists $\mathbf{x} \in B_n$ such that $\mathbf{x} = \mathbf{y}$ or $\mathbf{x} \twoheadrightarrow \mathbf{y}$ or $\mathbf{x} \to \mathbf{y}$, i.e., every element of B_{n+1} is either in B_n or reachable from B_n. Notice there must be some progress forwards, since at lest one element of B_{n+1} must not be in B_n.

The assignment of sets of multisets of weapons to arguments is updated as follows.

$$\mathbf{y} :=_{n+1} S_n \cup \bigcup \{T \colon \exists \mathbf{x} \in B_n \; s.t. \; \mathbf{x} \to \mathbf{y} \wedge \mathbf{x} :=_n T\} \quad \text{where } \mathbf{y} :=_n S_n.$$

The set of relation $\left\{R^i_{n+1}\right\}$ is then updated by using the set of joint attack relations on weapons generated by each $\mathbf{x} \twoheadrightarrow \mathbf{y}$ where $\mathbf{x} \in B_n$, $\mathbf{x} \in B_{n+1}$, Recall, that joint attack relations map mutisets of weapons to single weapons. More precisely, we have $\mathcal{R}_{n+1}$ is defined as follows, where ⊠ is point-wise union of sets of relations:

$$\begin{aligned}\mathcal{R}_{n+1} = \mathcal{R}_n \boxtimes \{\, R \colon & \forall \mathbf{x} \in B_n, \forall \mathbf{y} \in B_{n+1} \; s.t. \; \mathbf{x} \twoheadrightarrow \mathbf{y} \wedge \mathbf{x} :=_{n+1} S \; \wedge \mathbf{y} :=_{n+1} T \wedge \\ & \forall m \in T, \exists w \in m \; s.t. \; \exists n \in S \;\; s.t. \; n \, R \, w \,\}\end{aligned}$$

The above defines more formally the joint attack relations generated as described in Section 3.1.

The output: Assuming the attack-defence framework is finite and acyclic, the algorithm eventually terminates, returning the assignment and set of joint attack relations at that iteration of the algorithm.

Remark 3. This is just one possible algorithm. Note, some security assessments may require more annotations and different algorithms for advancing from one belt to the next, for interpreting the attack relation, and for interpreting the joint attack relation. We return to this point in our discussion of this model, which occupies the remaining sections of the paper.

4 Reorientation from the perspective of attack-defence trees

Let us motivate and explain what we are doing in this paper in a Socratic fashion starting bottom up with the security requirements driven by examples of attack-defence trees. This approach enables us to compare existing treatments of attack-defence trees in the security area, with existing treatments of such frameworks in the argumentation area. This enables us to export ideas and technical tools from the argumentation area into the security area.

We take as a starting point an attack-defence tree in Figure 1 of reference [1], reproduced in the Figure 11. We study this figure and compare it, bit by bit with argumentation frameworks, and try to see how to understand it in a new improved more detailed point of view. Viewed as a bipolar argumentation framework (i.e., a graph formed from attack and support relations) Figure 11 has the following characteristics.

1. The graph has no cycles. (The handling of cycles is still an open problem in the attack-defence tree context, while it is more central in the argumentation context.)
2. The graph has a single top node (let us call it the goal g) to be defended and it is layered as a tree with layer 1 defending/protecting g and each layer $n+1$ attacking the previous layer n and or defending layer $n-1$.
3. The graph uses joint attacks and joint supports.
4. The nodes have internal meaningful contents. They are not atomic letter nodes. This should be taken into account when offering semantics for the tree.

There are several ways of looking at Figure 11.

1. As a traditional formal argumentation framework. This works only for limited examples, such as Figure 1.
2. As a graph for a game between two players (the defender/protector of g and the attacker of g) the levels/layers are moves and countermoves of the players. This view is better but still not exactly right. We shall also discuss this. The graph can be flattened to a mini-max matrix. All defences can put forward in layer 1 — consisting of all possible best strategic defensive moves — and the attacker can attack all possible defensive strategies and the net result is the solution. The problem with this view is that we need to address more features of the application, for example the temporal evolution of moves, the availability and cost of resources and the local reasoning and aim of each player and, prospectively, the treatment of cycles.
3. As action counter action temporal sequence between two agents, the one protecting g and the other in principle attacking g. This is a much better view but it needs to be fine-tuned to various applications.

We now ask how do we proceed, and where do we find the connection and use of argumentation in the attack-defence trees context? We start with examples from both areas and step by step, using a Socratic method, add components that converge towards our target theory.

Let us now look at formal argumentation frameworks and find a framework to the formal argumentation community, (Figure 12) which may be, on the face of it, similar to what Figure 11 seems to be. We then continue our analysis of Figure 11 . Consider Figure 12. In this figure we use a single arrow for support "→" and a double arrow for attack "↠". To start our comparison, the nodes in Figure 12 are explained and exemplified by nodes in Figure 11 in parentheses below.

Explanation of the nodes of Figure 12:

- g is the goal to protect (Data Confidentiality)
- a, b are supports (Physical Security, Network Security, etc.)
- α, β, γ attack the support (Break In, Dictionary Attack, Corruption)

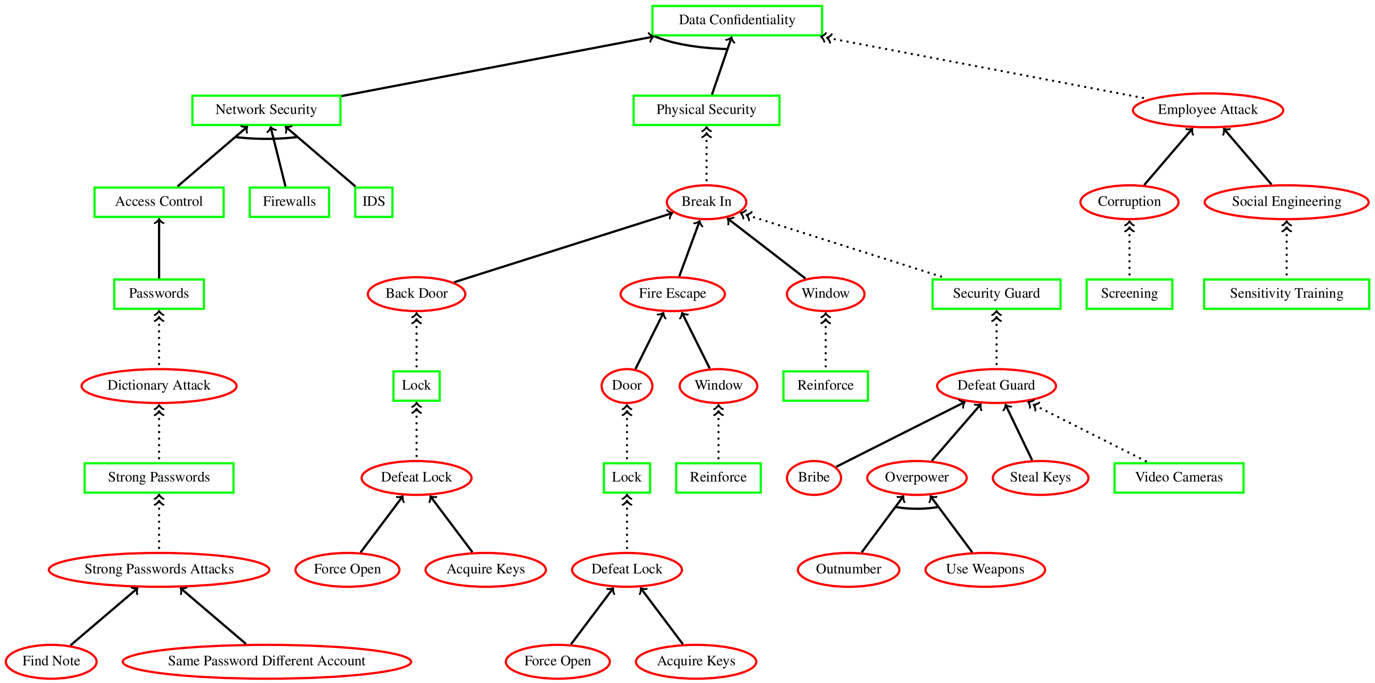


Fig. 11: An ADTree for protecting data confidentiality from reference [1]
]

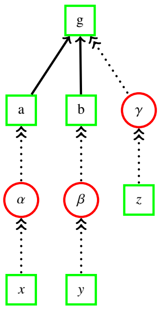


Fig. 12: A scenario with goals, attacks and defences, in terms of attack relations and as an attack defence tree.

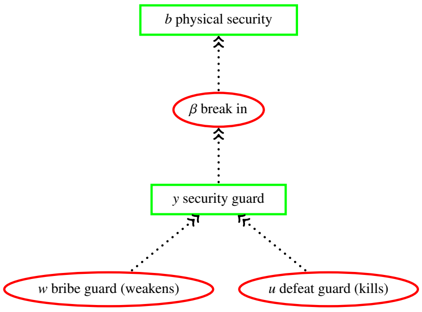


Fig. 13: A subtree of Figure 11, where some attacks defeat and others weaken.

- x, y, z support a, b, c by attacking the attacks (Security Guard, Strong Password, etc.).

Comparing Figure 11 and Figure 12, let us make some observations.

Observation 1. In Figure 11, consider the subpart of the figure represented by Figure 2. In this figure the node y does not attack β in the sense of "killing" β but makes b stronger so that it can withstand the attack of β. In other words, the part of the figure (namely the formal attack and defence sub-figure to the left of Figure 2) can be transformed to the bipolar argumentation framework to the right of Figure 2.

Figure 2 represents a bipolar argumentation framework, that is a framework with attack and defence (in argumentation terminology). One of the interpretations of such frameworks, from the argumentation point of view, is that to attack and kill a node b, we need also to kill all of its supporters (i.e., we need to attack y as well). Adopting established terminology [13], the set $\{b, y\}$ forms a *support group*. Indeed this is also the security view of the attack and defence in Figure 11, in that the attacks must continue on node y = Strong Password. Indeed in Figure 11, y = Strong Password is attacked by "Strong Password Attacks" (i.e. Find Note, Same Password Different Accounts).

Observation 2. On the other hand, the part of Figure 11 depicted in Figure 13 consisting of b, β, y with the additional options, u and w, is different. It has the additional feature that it the security guard is attacked in two possible ways: bribing, which weakens the guard and may be ineffective, and killing, which removes the guard.This observation departs from mainstream argumentation. In argumentation, if a node x attacks a node y (i.e. $x \twoheadrightarrow y$), then if x is alive then the attack on y is always successful and x kills y and y is dead. There is no intermediate result such as weakening y, which might be accommodated in a more resource sensitive model.

From the perspective of security, a limitation of lifting directly from argumentation without reworking the semantics is that resource considerations remain limited — all arguments are either "in" or "out" with respect to some joint attack relation. For attack-defence trees, when determining whether an argument such as "data confidentiality" is maintained we consider the resources assigned to an attacker profile. Resources may be specialised equipment or expertise, a budget or time; while profiles of attackers may include cybercriminals, rogue states, script kiddies or cyberterrorists. Only by combining such viewpoints can we estimates the vulnerabilities a system is exposed to and priorities mitigating those attacks with limited security resources.

Instead of calculating whether nodes are in or out we may wish to calculate quantities that remain after being attacked. For instance in Figure 10, for some attacker and defender profiles, there may not be sufficient budget for the defender to use Cloud-bursting, but there is not sufficient motive for the attacker to perpetrate the DDoS attack any way. Bringing in such resource considerations from security would be a contribution to the area of argumentation.

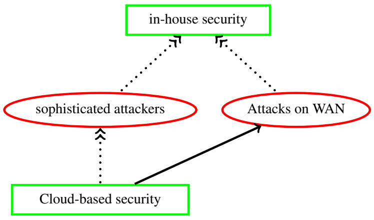


Fig. 14: Scenario where a defensive action supports a new attack.

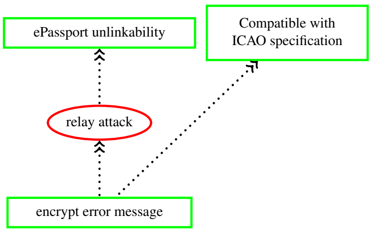


Fig. 15: Scenario with multiple goals, and goals that are not necessarily security related.

Observation 3. Consider the framework in Figure 14. This is an acyclic graph rather than a tree. There is no attack-defence node distinction: a “green” node can support a “red” node (colours are meaningless in this model, they simplify making connection with established attack-defence tree notations).

Scenario for Figure 14: A company with a limited cyber security budget may not have the resources to defend against sophisticated attackers using in house security solutions. Their solution to defeat these sophisticated cyber attacks is to outsource part of their infrastructure to a secure Cloud environment. The dedicated expertise and tools behind the Cloud-based security solution does reduce the risk of the company becoming exposed to certain sophisticated attacks on their in-house infrastructure; however, this move does leave open the organisation to new attacks. Thus, a side-effect of employing Cloud-based security is that new attacks that exploit the fact that certain operations are occurring over a WAN are enabled. Thus the use of certain defences may support new attacks.

Notice that, while we do not have side effects in Figure 11, it is possible to add examples of side effects. In Figure 13, killing the guard may activate a Murder Investigation as a side effect and we might not want that.

Observation 4. The scenario in Figure 15 presents multiple goals, which would not be permitted if we restrict to trees. The privacy goal is to ensure ePassport holders cannot be linked from one session to the next, which is called *unlinkability*. There are attacks on unlinkability, involving relaying messages to remote readers [14]. Note furthermore, that such attacks do not completely compromise unlinkability, e.g., ePassport holders cannot be tracked forever, only in a limited time window, so there are resource considerations here.

The effectiveness of these relay attacks on ePassport unlinkability could be reduced by encrypting error messages that leak information. The added dimension is that the defensive action of encrypting an error message attacks a second goal, which is to satisfy the ICAO specification for ePassports so that the ePassport is compatible with ePassport readers internationally. Thus there may be multiple goals, and not all goals need be security related.

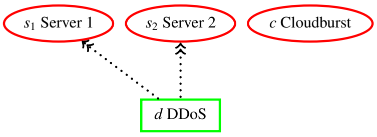


Fig. 16: Diagrammatic representation of the argumentation framework generated algorithmically in Section 3.2, which forms a disconnected acyclic graph with no single root node.

An additional reason for permitting multiple goals and even disconnected acyclic graphs is illustrated in Figure 9. That figure depicts a graph with multiple sinks which is an attack relation realising that realises another attack relation that formed a tree. Thus by permitting general acyclic graphs we can use graphical notation to depict both attack-defence trees, where arguments may be attack trees, and its semantics given by a set of joint attack relations where the target of each attack is an atomic action or weapon. To see why such acyclic graphs need not be connected, observe that the joint attack relation generated by the running example in Section 3.2 can be depicted as in Figure 16. Recall that nothing attacked the Cloudburst argument whose resources were denoted by the weapon c.

Observation 5. In formal argumentation frameworks, a node x attacking several targets attacks all of them in the same way. There is no option for different attacks for different targets. This is not the case in Figure 11, “defeat lock” attacking the back door is most likely not the same as the one attacking the front door. The attacks are directional.

Observation 6. In Security, there is a stress on resources, hence the use of linear logic in semantics for attack trees [15, 16]. Formal argumentation is based on classical logic.

Observation 7. The structure of an attack-defence framework could be taken further, to provide a still finer semantics for attack-defence trees, by introducing an explicit *conjunctive support relation*. For example, consider the first attack-defence tree in Figure 17.

Existing semantics for attack-defence tree in the literature, and also the semantics in this paper, are not sensitive to the fact that the reason that Overpower is countered is that people were searched upon entry to the building. Indeed, the existing semantics would assign the same meaning to the first tree in Figure 17 and the two other scenarios.

In terms of the semantics provided, Overpower will be assigned the attack tree denoted by $\{o * k\}$ and a set of two attack relations, say $\{R_1, R_2\}$, will be generated,

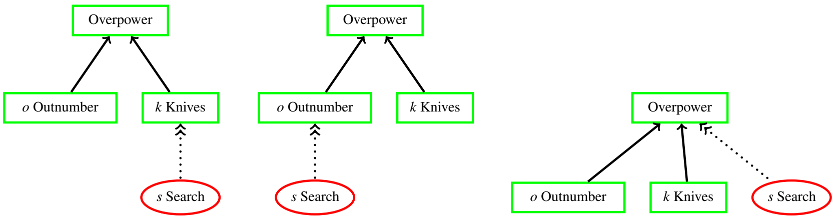


Fig. 17: A case for conjunctive support with two variations.

where $s\ R_1\ k$ and $s\ R_2\ o$. Thus the semantics are currently indiscriminate about which weapons or actions are countered by the argument Search — it is not necessarily the Knives, as the first attack-defence tree in Figure 17 might intuitively suggest. As, explained above, established multiset semantics in the literature [1] would also not make it explicit that only the argument Knives is attacked.

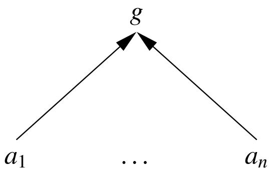
The above limitation of the semantics could be resolved by an explicit conjunctive support, which is interpreted in the algorithm by extending the weapons in the nodes supported, using multiset union, by using the resources available to the source node. This would enable the three scenarios in Figure 17 to be distinguished, since the generated attack relations would be $\{R_1\}$, $\{R_2\}$ and $\{R_1, R_2\}$ respectively.

A further advantage of breaking down all nodes in an attack-defence tree into arguments is that we can refer explicitly to sub-goals of attackers, not just the roots of trees. That is, we can ask questions, such as whether a sub-goal is an acceptable argument with respect to some preferred extension. Recent work on attack trees, has argued for the value of giving sub-goals an explicit status [17].

Remark 4 (Summary of discussion in Section 4). We summarise the points learnt from our discussion in this section. To give good argumentation like semantics for Figure 11 describing a security scenario, we need to enrich argumentation with the following features:

1. And/or attacks and defence (this we have already in argumentation).
2. Allow converting attack to support and support to attacks (this has been done previously [18], but for only the numerical case).
3. Allow for weakening attacks (as well as attacks which fail) in a directional way. (This means that for the same live x and different targets, say for example, $x \twoheadrightarrow y_1, x \twoheadrightarrow y_2$, and $x \twoheadrightarrow y_3$, it is possible that the attack of x on y_1 will succeed, the attack on y_2 will fail and the attack on y_3 will only weaken y_3. Compare this with numerical attacks which change the strength of the target by a numerical factor.)
4. Deal with side effects in the formal argumentation level, because in practice for example when you hack into a server you may cause side effects.
5. We need one more principle: Consider below, where we have nodes $a_1, \ldots, a_n$ supporting g. To make sure we successfully kill g we need to kill *all* of $a_1, \ldots, a_n$.

This is for the case where all the a_i are independent supports.



This is not like how it goes in logical and legal argumentation. If we have $a_1 \vdash g, \dots, a_n \vdash g$, then attacking or falsifying all a_i does not mean that g is false. There may be some new $x \vdash g$.

In the model introduced in this paper, we embody this principal by assigning arguments representing intermediate nodes in an attack tree no resources initially. Since such nodes inherit all their resources from their supports, killing all their supports kills the intermediate argument.

6. Running Global Side effects. Each node costs money. Guards need to be paid, Keys need to be acquired, etc.. We have a global budget node which needs to be treated as a special weapon node.
7. Local support. This principle has to do with supporting local nodes in the middle of the tree. We note that in Figure 11 all the support nodes actually support the security of the data. There is a sequence of nodes:

   acquire keys $\twoheadrightarrow$ lock door $\twoheadrightarrow$ break in through door.

   So let us add support to acquire key the support we add is "increase budget to buy keys". This support is not for server security, it supports locally the attack of acquire keys.

5 Conclusion

This position paper proposes *attack-defence frameworks*, defined in Section 3, which build on concepts in argumentation so that we may assess the acceptability of arguments in security scenarios described by attack-defence trees. Attack-defence frameworks borrow from some more recent developments in argumentation, namely:

- bipolar argumentation frames that incorporate support as well as attack,
- joint attacks for describing scenarios where multiple resources must be used together to execute an attack,
- and disjunctive attacks allowing multiple possible ways of realising an attack.

In addition, attack-defence frameworks take into account resource considerations, by annotating arguments with sets of multisets of weapons or actions, which are essentially attack trees. This semantics generates multiple possible ways of realising attacks, which can, in turn, be used to explain *why* arguments such as Data Confidentiality or intermediate goals such as Physical Security, or Lock Doors are acceptable arguments.

The development of attack-defence frameworks has been guided by examples from the security domain. However, this model has been developed with other fields in mind such as legal argumentation (think lawyers attacking each other), ecology (think of species competing with and supporting each other) and medical sciences (think of the side effect of taking medicine along the lines of Figure 14), hence may be broadly applied. For security specifically, a key added value of this work is the notion of evidence for an argument, as given by *preferred extensions* for example, which is a central notion in the various semantics investigated in the argumentation domain. The model admits general graphical structures to be described thus we are not restricted to trees, nor are we restricted to asking question about a goal represented by a root node.

References

1. Kordy, B., Mauw, S., Radomirović, S., Schweitzer, P.: Attack–defense trees. Journal of Logic and Computation **24** (2012) 55–87
2. Bistarelli, S., Fioravanti, F., Peretti, P.: Defense trees for economic evaluation of security investments. In: First International Conference on Availability, Reliability and Security (ARES'06). (2006) 8 pp.–423
3. Cayrol, C., Lagasquie-Schiex, M.: On the acceptability of arguments in bipolar argumentation frameworks. In Godo, L., ed.: Symbolic and Quantitative Approaches to Reasoning with Uncertainty, 8th European Conference, ECSQARU 2005, Barcelona, Spain, July 6-8, 2005, Proc. Volume 3571 of Lecture Notes in Computer Science., Springer (2005) 378–389
4. Boella, G., Gabbay, D.M., van der Torre, L.W.N., Villata, S.: Support in abstract argumentation. In Baroni, P., Cerutti, F., Giacomin, M., Simari, G.R., eds.: Computational Models of Argument: Proc. COMMA 2010, Desenzano del Garda, Italy, September 8-10, 2010. Volume 216 of Frontiers in Artificial Intelligence and Applications., IOS Press (2010) 111–122
5. Wideł, W.: Formal modeling and quantitative analysis of security using attack-defense trees. PhD thesis (2019)
6. Dung, P.M.: On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. Artificial Intelligence **77** (1995) 321–357
7. Gabbay, D.M.: Semantics for higher level attacks in extended argumentation frames part 1: Overview. Studia Logica **93** (2009) 355–379
8. Nielsen, S.H., Parsons, S.: A generalization of Dung's abstract framework for argumentation: Arguing with sets of attacking arguments. In Maudet, N., Parsons, S., Rahwan, I., eds.: Argumentation in Multi-Agent Systems, Berlin, Heidelberg, Springer Berlin Heidelberg (2007) 54–73
9. Schneier, B.: Attack trees. Dr. Dobb's journal **24** (1999) 21–29
10. Mauw, S., Oostdijk, M.: Foundations of attack trees. In Won, D.H., Kim, S., eds.: Information Security and Cryptology - ICISC 2005, Springer Berlin Heidelberg (2006) 186–198
11. Gabbay, D., Gabbay, M.: Theory of disjunctive attacks, part i. Logic Journal of the IGPL **24** (2016) 186–218
12. Armbrust, M., Fox, A., Griffith, R., Joseph, A.D., Katz, R.H., Konwinski, A., Lee, G., Patterson, D.A., Rabkin, A., Stoica, I., Zaharia, M.: A view of cloud computing. Commun. ACM **53** (2010) 50–58
13. Gabbay, D.: Logical foundations for bipolar argumentation networks. J. Logic Computation **26** (2016) 247–292
14. Filimonov, I., Horne, R., Mauw, S., Smith, Z.: Breaking unlinkability of the ICAO 9303 standard for e-passports using bisimilarity. In Sako, K., Schneider, S., Ryan, P.Y.A., eds.: Computer Security - ESORICS 2019 - 24th European Symposium on Research in Computer Security, Luxembourg, September 23-27, 2019, Proceedings, Part I. Volume 11735 of Lecture Notes in Computer Science., Springer (2019) 577–594
15. Horne, R., Mauw, S., Tiu, A.: Semantics for specialising attack trees based on linear logic. Fundam. Inform. **153** (2017) 57–86

16. Eades III, H., Jiang, J., Bryant, A.: On linear logic, functional programming, and attack trees. In Cybenko, G., Pym, D.J., Fila, B., eds.: 5th International Workshop on Graphical Models for Security GraMSec@FLoC 2018, Oxford, UK, July 8, 2018, Revised Selected Papers. Volume 11086 of Lecture Notes in Computer Science., Springer (2018) 71–89
17. Mantel, H., Probst, C.W.: On the meaning and purpose of attack trees. In: 2019 IEEE 32nd Computer Security Foundations Symposium (CSF). (2019) 184–199
18. Barringer, H., Gabbay, D.M., Woods, J.: Temporal, numerical and meta-level dynamics in argumentation networks. Argument & Computation **3** (2012) 143–202