

Cyber Threat Analysis with Structured Probabilistic Argumentation

Mario A. Leiva¹, Gerardo I. Simari¹,
Guillermo R. Simari¹, and Paulo Shakarian²

¹ Depto. de Cs. e Ing. de la Comp., Universidad Nacional del Sur (UNS)
Inst. de Cs. e Ing. de la Comp. (ICIC UNS-CONICET), Argentina

² Cyber Reconnaissance, Inc., Tempe, Arizona, USA
{mario.leiva,gis,grs}@cs.uns.edu.ar,paulo@cyr3con.ai

Abstract. Capturing the uncertain aspects in cyber threat analyses is an important part of a wide range of efforts, including diagnostics, threat evaluation, and preventing attacks. However, there has been insufficient research and development of modeling approaches that are able to correctly capture and handle such uncertainty. In this work, we present an application example of the DeLP3E framework—a formalism that extends structured argumentation based on logic programming—in the domain of cyber threat analysis; in particular, near real-time analyses such as *incident response* in enterprise networks. The DeLP3E framework provides a unique combination of dialectical reasoning, rule-based inference, and probabilistic modeling to not only offer suggested responses to given situations, but also to explain to the analyst why the system reaches its conclusions.

Keywords: Defeasible Reasoning · Structured Probabilistic Argumentation · Cyber Threat Analysis

1 Introduction

Real-time security analysis is a far more imprecise process than deterministic reasoning. Since we do not know the full range of the attacker’s choices, one source of uncertainty is the attacker’s behavior. As cyberattacks are not always guaranteed to succeed, another source of uncertainty lies in the imperfect nature of exploits. Yet another source of uncertainty is the fact that the defender’s observations on potential attack activities are limited—for example, a well-known problem with analysts in a security operations center (SOC) is “alert fatigue” caused by false positives. Nevertheless, the logical *causality* encoded in a deterministic attack graph is an invaluable aspect of understanding security-related events, and is useful for building practical network defense tools if we can *appropriately* account for the uncertainty inherent in the reasoning process. Cyber threat analysis is a highly-technical intelligence problem where an analyst must consider a variety of sources, each with its associated level of confidence, in order to provide a decision maker insight into who conducted a given operation.

As it is well known that people’s ability to conduct intelligence analysis is limited [5], and due to the highly technical nature of many cyber evidence-gathering techniques, an automated reasoning system would be best suited for the task. Such a system must be able to accomplish several goals, among which we distinguish the following main capabilities [13]:

1. Reason about evidence in a formal, principled manner, i.e., relying on strong mathematical foundations.
2. Consider evidence for cyber intrusion associated with some level of probabilistic uncertainty.
3. Consider logical rules that allow for the system to draw conclusions based on certain pieces of evidence and iteratively apply such rules.
4. Consider pieces of information that may not be compatible with each other, decide which information is most relevant, and express why.
5. Show the actual status of the system based on the above-described features, and provide the analyst with the ability to understand how the system arrives at that conclusion.

In this paper, we present an application example of the DeLP3E framework [14] in the domain of cyber threat analysis—in particular, we focus on near real-time security analyses such as *incident response* to show that it has the aforementioned capabilities. This approach relies on several techniques from the artificial intelligence community, including argumentation, logic programming, and probabilistic reasoning. We first outline the underlying mathematical framework in Section 2; then, in Section 3 we present the application example. Finally, we discuss conclusions and future work in Section 4.

2 Preliminaries on the DeLP3E framework

We now provide a brief introduction to DeLP3E; for full details, we refer the interested reader to [14], which is the source of this summarized presentation.

A DeLP3E model consists of two parts, which correspond to *two separate models of the world*. The first, called the *environmental model* (EM) is used to describe the background knowledge and is probabilistic in nature. The second one, called the *analytical model* (AM) is used to analyze competing hypotheses that can account for a given phenomenon (in our domain, a response to an intrusion). The EM *must be consistent*—this simply means that there must exist a probability distribution over the possible states of the world that satisfies all of the constraints in the model, as well as the axioms of probability theory. On the contrary, the AM will allow for contradictory information as the system must have the capability to reason about competing explanations for a given event. In general, the EM contains knowledge such as evidence, intelligence reporting, or knowledge about actors, software, and systems. The AM, on the other hand, contains ideas the analyst concludes based on the information in the EM. Note that an analyst (or automated system) could assign a probability to statements in the EM, whereas statements in the AM can be true or false depending on

a certain combination (or several possible combinations) of statements from the EM. There are thus two kinds of uncertainty that need to be modeled: probabilistic uncertainty and uncertainty arising from defeasible knowledge. This model allows both to coexist, and also allows for the combination of the two since defeasible rules and presumptions (that is, defeasible facts) can also be annotated with probabilistic events.

We now briefly formally describe these two models as well as a technique for *annotation* of knowledge in the AM with information from the EM—these annotations specify the conditions under which various statements in the AM can potentially be true.

Basic Language. The sets of variable and constant symbols are denoted with \mathbf{V} and \mathbf{C} , respectively. The next component of the language is a set of n -ary predicate symbols; the EM and AM use separate sets of predicate symbols, denoted with \mathbf{P}_{EM} , \mathbf{P}_{AM} , respectively, the two components can, however, share variables and constants. As usual, a *term* is composed of either a variable or constant. Given terms t_1, \dots, t_n and n -ary predicate symbol p , $p(t_1, \dots, t_n)$ is called an *atom*; if t_1, \dots, t_n are constant, then the atom is said to be *ground*. We use \mathbf{G}_{EM} and \mathbf{G}_{AM} to denote the sets of all ground atoms for the EM and AM, respectively.

Given a set of ground atoms, a *world* is any subset of atoms, those that belong to the set are said to be *true* in the world, while those that do not are *false*. Therefore, there are $2^{|\mathbf{G}_{EM}|}$ possible worlds in the EM and $2^{|\mathbf{G}_{AM}|}$ worlds in the AM. These sets are denoted with \mathcal{W}_{EM} and \mathcal{W}_{AM} , respectively. In order to avoid worlds that do not model possible situations given a particular domain, *integrity constraints* of the form *oneOf*(A') can be used, where A' is a subset of ground atoms. Intuitively, such a constraint states that any world where more than one of the atoms from set A' appears is invalid. \mathbf{IC}_{EM} and \mathbf{IC}_{AM} denote the sets of integrity constraints for the EM and AM, respectively, and the sets of worlds that conform to these constraints is denoted with $\mathcal{W}_{EM}(\mathbf{IC}_{EM})$, $\mathcal{W}_{AM}(\mathbf{IC}_{AM})$, respectively. Finally, logical formulas arise from the combination of atoms using the traditional connectives (\wedge , \vee , and \neg). As usual, a world w *satisfies* formula (f), written $w \models f$, iff: (i) If f is an atom, then $w \models f$ iff $f \in w$; (ii) if $f = \neg f'$ then $w \models f$ iff $w \not\models f'$; (iii) if $f = f' \wedge f''$ then $w \models f$ iff $w \models f'$ and $w \models f''$; and (iv) if $f = f' \vee f''$ then $w \models f$ iff $w \models f'$ or $w \models f''$. We use $form_{EM}$ and $form_{AM}$ to denote the set of all possible (ground) formulas in the EM and AM, respectively.

2.1 Environmental Model

In this paper, we generalize the approach taken in [12, 14] for the environmental model by simply assuming that we have a probabilistic model defined over \mathbf{G}_{EM} , which represents a probability distribution over \mathcal{W}_{EM} .

Definition 1 (Probabilistic model). *Given sets \mathbf{P}_{EM} , \mathbf{V} , and \mathbf{C} , and corresponding sets \mathbf{G}_{EM} and \mathcal{W}_{EM} , a probabilistic model Π_{EM} is any function $Pr: \mathcal{W}_{EM} \rightarrow [0, 1]$ such that $\sum_{\lambda \in \mathcal{W}_{EM}} Pr(\lambda) = 1$.*

Examples of probabilistic models that can be used are Bayesian networks (BNs) [10], Markov logic networks (MLNs) [11], extensions of first order logic such as Nilsson’s probabilistic logic [9], or even ad-hoc constructions of *Pr*.

2.2 Analytical Model

Unlike the EM, which describes probabilistic information about the state of the real world, the AM must allow for competing ideas, so *it must be able to represent contradictory information*. As described next, the underlying algorithmic approach allows for the creation of *arguments* based on the AM that may “compete” with each other. In this competition, known as a *dialectical process*, one argument may defeat another based on a *comparison criterion* that determines the prevailing argument. Resulting from this process, the DeLP3E framework will determine arguments that are *warranted* (those that are not *defeated* by other arguments) thereby providing a suitable explanation. The transparency provided by the system can allow analysts to identify potentially incorrect input information and fine-tune the models or, alternatively, collect more information. In short, argumentation-based reasoning has been studied as a natural way to manage a set of inconsistent information, since it is based on the way humans settle disputes. Another desirable characteristic of (structured) argumentation frameworks is that, once a conclusion is reached, we are left with an explanation of how we arrived at it and information about why a given argument is warranted; this is very important information for analysts to have. We now recall some preliminaries of the underlying argumentation framework used, and then introduce the analytical model.

Defeasible Logic Programming with Presumptions (PreDeLP). DeLP with Presumptions (PreDeLP) [7] is a formalism combining Logic Programming with Defeasible Argumentation. We now briefly recall the basics of PreDeLP, and refer the reader to [3, 7] for a full presentation. The formalism contains several different constructs: facts, presumptions, strict rules, and defeasible rules. Facts are statements about the analysis that can always be considered to be true, while presumptions are statements that may or may not be true. Strict rules specify logical consequences of a set of facts or presumptions (similar to an implication, though it does not behave exactly the same) that must always occur, while defeasible rules specify logical consequences that may be assumed to be true when no contradicting information is present. These components are used in the construction of *arguments*, and together comprise PreDeLP programs.

Formally, we use the notation $\Pi_{AM} = (\Theta, \Omega, \Phi, \Delta)$ to denote a PreDeLP program, where:

- Ω is the set of strict rules of the form $L_0 \leftarrow L_1, \dots, L_n$, where L_0 is a ground literal and $\{L_i\}_{i>0}$ is a set of ground literals;
- Θ is the set of facts, written simply as atoms;
- Δ is the set of defeasible rules of the form $L_0 \prec L_1, \dots, L_n$, where L_0 is a ground literal and $\{L_i\}_{i>0}$ is a set of ground literals, and

- Φ is the set of presumptions, which are written as defeasible without a body.

For simplicity, we sometimes refer to Π_{AM} as a set corresponding to the union of its components. Recall that all atoms in the AM must be formed with a predicate from the set P_{AM} , and note that in both strict and defeasible rules, *strong negation* (i.e., classical negation as in first-order logic) is allowed in the head, and thus may be used to represent contradictory knowledge.

For the treatment of contradictory knowledge, PreDeLP incorporates a defeasible argumentation formalism that allows the identification of the pieces of knowledge that are in conflict, and through the previously mentioned *dialectical process* decides which information prevails as warranted. The dialectical process involves the construction and evaluation of arguments that either support or interfere with a given query, building a *dialectical tree* in the process. Formally, we have:

Definition 2 (Argument). *An argument $\langle \mathcal{A}, L \rangle$ for a literal L is a pair of the literal and a (possibly empty) set of the AM ($\mathcal{A} \subseteq \Pi_{AM}$) that provides a minimal proof for L meeting the requirements: (i) L is defeasibly derived from \mathcal{A} , (ii) $\Omega \cup \Theta \cup \mathcal{A}$ is not contradictory, and (iii) \mathcal{A} is a minimal subset of $\Delta \cup \Phi$ satisfying (i) and (ii).*

Literal L is called the conclusion supported by the argument, and \mathcal{A} is the support of the argument. An argument $\langle \mathcal{B}, L \rangle$ is a subargument of $\langle \mathcal{A}, L' \rangle$ iff $\mathcal{B} \subseteq \mathcal{A}$. An argument $\langle \mathcal{A}, L \rangle$ is presumptive iff $\mathcal{A} \cap \Phi$ is not empty. We will also use $\Omega(\mathcal{A}) = \mathcal{A} \cap \Omega$, $\Theta(\mathcal{A}) = \mathcal{A} \cap \Theta$, $\Delta(\mathcal{A}) = \mathcal{A} \cap \Delta$, and $\Phi(\mathcal{A}) = \mathcal{A} \cap \Phi$.

Note that this definition differs slightly from that of [16] where DeLP is introduced; here, strict rules and facts are included as part of the argument.

Given argument $\langle \mathcal{A}_1, L_1 \rangle$, counter-arguments are arguments that contradict it. Argument $\langle \mathcal{A}_2, L_2 \rangle$ *counterargues or attacks* $\langle \mathcal{A}_1, L_1 \rangle$ at literal L'' iff there exists a subargument $\langle \mathcal{A}, L'' \rangle$ of $\langle \mathcal{A}_1, L_1 \rangle$ s.t. set $\Omega(\mathcal{A}_1) \cup \Omega(\mathcal{A}_2) \cup \Theta(\mathcal{A}_1) \cup \Theta(\mathcal{A}_2) \cup \{L_2, L''\}$ is contradictory.

A *proper defeater* of an argument $\langle \mathcal{A}, L \rangle$ is a counterargument that, by some criterion, is considered to be better than $\langle \mathcal{A}, L \rangle$; if the two are incomparable according to this criterion, the counterargument is said to be a *blocking* defeater. An important characteristic of PreDeLP is that the argument comparison criterion is modular, and thus the most appropriate criterion for the domain that is being represented can be selected; the default criterion used in classical defeasible logic programming (from which PreDeLP is derived) is *generalized specificity* [17], though an extension of this criterion is required for arguments using presumptions [7]. We briefly recall this criterion next; the first definition is for generalized specificity, which is subsequently used in the definition of presumption-enabled specificity.

Definition 3. *Let $\Pi_{AM} = (\Theta, \Omega, \Phi, \Delta)$ be a PreDeLP program and let \mathcal{F} be the set of all literals that have a defeasible derivation from Π_{AM} . An argument $\langle \mathcal{A}_1, L_1 \rangle$ is preferred to $\langle \mathcal{A}_2, L_2 \rangle$, denoted with $\mathcal{A}_1 \succ_{PS} \mathcal{A}_2$ iff the two following conditions hold:*

1. For all $H \subseteq \mathcal{F}$, $\Omega(\mathcal{A}_1) \cup \Omega(\mathcal{A}_2) \cup H$ is non-contradictory: if there is a derivation for L_1 from $\Omega(\mathcal{A}_2) \cup \Omega(\mathcal{A}_1) \cup \Delta(\mathcal{A}_1) \cup H$, and there is no derivation for L_1 from $\Omega(\mathcal{A}_1) \cup \Omega(\mathcal{A}_2) \cup H$, then there is a derivation for L_2 from $\Omega(\mathcal{A}_1) \cup \Omega(\mathcal{A}_2) \cup \Delta(\mathcal{A}_2) \cup H$
2. There is at least one set $H' \subseteq \mathcal{F}$, $\Omega(\mathcal{A}_1) \cup \Omega(\mathcal{A}_2) \cup H'$ is non-contradictory, such that there is a derivation for L_2 from $\Omega(\mathcal{A}_1) \cup \Omega(\mathcal{A}_2) \cup H' \cup \Delta(\mathcal{A}_2)$, there is no derivation for L_2 from $\Omega(\mathcal{A}_1) \cup \Omega(\mathcal{A}_2) \cup H'$, and there is no derivation for L_1 from $\Omega(\mathcal{A}_1) \cup \Omega(\mathcal{A}_2) \cup H' \cup \Delta(\mathcal{A}_1)$

Intuitively, the principle of specificity says that, in the presence of two conflicting lines of argument about a proposition, the one that uses more of the available information is more convincing.

Definition 4. Let $\Pi_{AM} = (\Theta, \Omega, \Phi, \Delta)$ be a PreDeLP program. An argument $\langle \mathcal{A}_1, L_1 \rangle$ is preferred to $\langle \mathcal{A}_2, L_2 \rangle$, denoted with $\mathcal{A}_1 \succ \mathcal{A}_2$ iff any of the following conditions hold:

1. $\langle \mathcal{A}_1, L_1 \rangle$ and $\langle \mathcal{A}_2, L_2 \rangle$ are both factual arguments and $\langle \mathcal{A}_1, L_1 \rangle \succ_{PS} \langle \mathcal{A}_2, L_2 \rangle$.
2. $\langle \mathcal{A}_1, L_1 \rangle$ is a factual argument and $\langle \mathcal{A}_2, L_2 \rangle$ is a presumptive argument.
3. $\langle \mathcal{A}_1, L_1 \rangle$ and $\langle \mathcal{A}_2, L_2 \rangle$ are presumptive arguments, and
 - (a) $\neg(\Phi(\mathcal{A}_1) \subseteq \Phi(\mathcal{A}_2))$, or
 - (b) $\Phi(\mathcal{A}_1) = \Phi(\mathcal{A}_2)$ and $\langle \mathcal{A}_1, L_1 \rangle \succ_{PS} \langle \mathcal{A}_2, L_2 \rangle$.

Generally, if \mathcal{A}, \mathcal{B} are arguments with rules X and Y , resp., and $X \subset Y$, then \mathcal{A} is stronger than \mathcal{B} . This also holds when \mathcal{A} and \mathcal{B} use presumptions P_1 and P_2 , resp., and $P_1 \subset P_2$.

A sequence of arguments called an *argumentation line* thus arises from this attack relation, where each argument defeats its predecessor. To avoid undesirable sequences that may represent circular or fallacious argumentation lines, in DeLP an *argumentation line* is *acceptable* if it satisfies certain constraints (see [18]). A literal L is *warranted* if there exists a non-defeated argument \mathcal{A} supporting L .

Clearly, there can be more than one defeater for a particular argument $\langle \mathcal{A}, L \rangle$. Therefore, many acceptable argumentation lines could arise from $\langle \mathcal{A}, L \rangle$, leading to a tree structure. The tree is built from the set of all argumentation lines rooted in the initial argument. In a dialectical tree, every node (except the root) represents a defeater of its parent, and leaves correspond to undefeated arguments. Each path from the root to a leaf corresponds to a different acceptable argumentation line. A dialectical tree provides a structure for considering all the possible acceptable argumentation lines that can be generated for deciding whether an argument is defeated. We call this tree *dialectical* because it represents an exhaustive dialectical¹ analysis for the argument in its root. For argument $\langle \mathcal{A}, L \rangle$, we denote its dialectical tree with $\mathcal{T}(\langle \mathcal{A}, L \rangle)$.

Given a literal L and an argument $\langle \mathcal{A}, L \rangle$, in order to decide whether or not a literal L is warranted, every node in the dialectical tree $\mathcal{T}(\langle \mathcal{A}, L \rangle)$ is recursively marked as “D” (*defeated*) or “U” (*undefeated*), obtaining a marked dialectical tree $\mathcal{T}^*(\langle \mathcal{A}, L \rangle)$ where:

¹ In the sense of providing reasons for and against a position.

- All leaves in $\mathcal{T}^*(\langle \mathcal{A}, L \rangle)$ are marked as “U”s, and
- Let $\langle \mathcal{B}, q \rangle$ be an inner node of $\mathcal{T}^*(\langle \mathcal{A}, L \rangle)$. Then, $\langle \mathcal{B}, q \rangle$ will be marked as “U” iff every child of $\langle \mathcal{B}, q \rangle$ is marked as “D”. Node $\langle \mathcal{B}, q \rangle$ will be marked as “D” iff it has at least a child marked as “U”.

Given argument $\langle \mathcal{A}, l \rangle$ over Π_{AM} , if the root of $\mathcal{T}^*(\langle \mathcal{A}, L \rangle)$ is marked “U”, then $\mathcal{T}^*(\langle \mathcal{A}, L \rangle)$ *warrants* L and that L is *warranted* from Π_{AM} . Note that warranted arguments correspond to those in the grounded extension of a Dung argumentation system [1].

We can extend the idea of dialectical tree to a *dialectical forest*. For a given literal L , a dialectical forest $\mathcal{F}(L)$ consists of the set of dialectical trees for all arguments for L . We denote a marked dialectical forest, the set of all marked dialectical trees for arguments for L , as $\mathcal{F}^*(L)$. Hence, for a literal L , we say it is *warranted* if there is at least one argument for that literal in the dialectical forest $\mathcal{F}^*(L)$ that is labeled “U”, *not warranted* if there is at least one argument for literal $\neg L$ in the forest $\mathcal{F}^*(\neg L)$ that is labeled “U”, and *undecided* otherwise.

2.3 The DeLP3E framework

This framework, originally proposed in [14], is the result of combining the environmental and analytical models presented above (which we denote with Π_{EM} and Π_{AM} , respectively). Intuitively, given Π_{AM} , every element of $\Omega \cup \Theta \cup \Delta \cup \Phi$ only holds in certain worlds in the set \mathcal{W}_{EM} , i.e., these elements are subject to probabilistic events. Each element of $\Omega \cup \Theta \cup \Delta \cup \Phi$ is thus associated with a formula over \mathbf{G}_{EM} (using conjunction, disjunction, and negation, as usual). The notion of *annotation function* associates elements of $\Omega \cup \Theta \cup \Delta \cup \Phi$ with element in $form_{EM}$.

Definition 5 (Annotation function [14]). *An annotation function is any function of the form $af: \Omega \cup \Theta \cup \Delta \cup \Phi \rightarrow form_{EM}$. We use $[af]$ to denote the set of all annotation functions.*

We will sometimes denote annotation functions as sets of pairs $(f, af(f))$ in order to simplify the presentation.

Definition 6 (DeLP3E Program). *Given environmental model Π_{EM} , analytical model Π_{AM} , and annotation function af , a DeLP3E program is of the form $\mathcal{I} = (\Pi_{EM}, \Pi_{AM}, af)$. We use notation $[\mathcal{I}]$ to denote the set of all possible programs.*

In the following, given DeLP3E program $\mathcal{I} = (\Pi_{EM}, \Pi_{AM}, af)$ and $\lambda \in \mathcal{W}_{EM}$, we use notation $\Pi_{AM}(\lambda) = \{f \in \Pi_{AM} \text{ s.t. } \lambda \models af(f)\}$. This gives rise to a *decomposed view* of DeLP3E programs.

The most direct way of considering consequences of DeLP3E programs is thus to consider what happens in each world in \mathcal{W}_{EM} ; that is, the defeat relationship among arguments depends on the current state of the (EM) world.

Definition 7 (Existence of an Argument in a World). *Given DeLP3E program $\mathcal{I} = (\Pi_{EM}, \Pi_{AM}, af)$, argument $\langle \mathcal{A}, L \rangle$ is said to exist in world $\lambda \in \mathcal{W}_{EM}$ if $\forall c \in \mathcal{A}, \lambda \models af(c)$.*

The notion of existence is extended to argumentation lines, dialectical trees, and dialectical forest in the expected way (for instance, an argumentation line exists in λ iff all arguments that comprise that line exist in λ).

The idea of a dialectical tree is also extended w.r.t. worlds; so, for a given world $\lambda \in \mathcal{W}_{EM}$, the dialectical (resp., marked dialectical) tree induced by λ is denoted with $\mathcal{T}_{\lambda(\mathcal{A}, L)}$ (resp., $\mathcal{T}_{\lambda(\mathcal{A}, L)}^*$). We require that all arguments and defeaters in these trees exist in λ . Likewise, we extend the notion of dialectical forest in the same manner (denoted with $\mathcal{F}_{\lambda}(L)$ and $\mathcal{F}_{\lambda}^*(L)$, respectively). Based on these concepts, we introduce the notion of *warranting scenario*.

Definition 8 (Warranting Scenario). *Given program $\mathcal{I} = (\Pi_{EM}, \Pi_{AM}, af)$ and a literal L formed with a ground atom from \mathbf{G}_{AM} , a world $\lambda \in \mathcal{W}_{EM}$ is said to be a warranting scenario for L (denoted $\lambda \vdash_{war} L$) if there is a dialectical forest $\mathcal{F}_{\lambda}^*(L)$ in which L is warranted and $\mathcal{F}_{\lambda}^*(L)$ exists in λ .*

The idea of a warranting scenario is used to formally define DeLP3E entailment. The set of worlds in the EM where a literal L in the AM *must* be true is exactly the set of warranting scenarios—these are the “necessary” worlds: $nec(L) = \{\lambda \in \mathcal{W}_{EM} \mid (\lambda \vdash_{war} L)\}$. Now, the set of worlds in the EM where AM literal L *can* be true is the following—these are the “possible” worlds: $poss(L) = \{\lambda \in \mathcal{W}_{EM} \mid (\lambda \not\vdash_{war} \neg L)\}$.

In the following we use notation $for(\lambda) = \bigwedge_{a \in \lambda} a \wedge \bigwedge_{a \notin \lambda} \neg a$, which denotes the formula that has λ as its only model. We also extend this notion to sets of worlds: $for(W) = \bigvee_{\lambda \in W} for(\lambda)$. Entailment can then be defined as follows:

Definition 9 (DeLP3E Entailment). *Given program $\mathcal{I} = (\Pi_{EM}, \Pi_{AM}, af)$, AM literal L , and probability interval $p \in [l, u]$, we say that \mathcal{I} entails L with probability $p \in [l, u]$ if the probability distribution Pr yielded by Π_{EM} is such that $l \leq \sum_{\lambda \in nec(L)} Pr(\lambda)$ and $\sum_{\lambda \in poss(L)} Pr(\lambda) \leq u$.*

In the next section, we present an example of how DeLP3E can be used in practice, leveraging our recently-introduced graphical tool [6].

3 Application Example

The use case presented here, which is modeled in part using a very simple Bayesian network, is in the domain of cyber threat analysis; in particular, near real-time security analysis such as *incident response* in enterprise networks. Such efforts are aided by the Security Information and Event Management (SIEM) platforms that continuously provide alerts relating to an enterprise’s security posture that are analyzed by SOC analysts. These analysts, in turn, examine the alerts to determine if they correspond to malicious events and, if required, elevate them to more senior analysts for further examination. The desired end

\mathbf{P}_{EM}	<i>netAccess</i>	The attacker obtains network access to the Web server on tcp/80.
	<i>httpdIsVulnerable</i>	The program <code>httpd</code> is a service running on Web server as user <code>apache</code> , listening on tcp/80, and is known to be vulnerable.
	<i>obtainExecutionPrivileges</i>	The attacker obtains code execution privilege on the Web server.
\mathbf{P}_{AM}	<i>errorSystemLogs</i>	Error detected in the system logs.
	<i>vulnerable</i>	The system is in a vulnerable state.
	<i>compromised</i>	The system is compromised.
	<i>underAttack</i>	The system is under attack.
	<i>qualifiedStaff</i>	The staff is trained to handle dangerous situations that may compromise the system.
	<i>webServerVulnerable</i>	The Web server is vulnerable.
	<i>runningOldhttpd</i>	An old version of <code>httpd</code> is running.

Fig. 1. Explanation of the meaning of the predicates used in the running example.

state is to ensure unauthorized activities can be handled effectively by applying appropriate countermeasures to prevent problems from worsening and resolve the incident, removing the threat from the enterprise [15]. As mentioned before, it is important to keep in mind that real-time security analysis is a far more imprecise process than deterministic reasoning [19]—we do not know the attacker’s choices, cyber attacks are not always guaranteed to succeed, and the defender’s observations on potential attack activities are limited. Nevertheless, the logical causality encoded in a deterministic *attack graph* (a graph that illustrates the possible multi-stage attacks in an enterprise network typically by presenting that logical causality relations among multiple privileges and configuration settings [19]), is invaluable to understand security events, and will be useful for building practical network defense tools if we can *appropriately* account for the uncertainty inherent in the reasoning process. Several attempts have been made at using Bayesian networks to model uncertainty in security analysis [8, 2, 4].

A Bayesian network (BN, for short) is a graphical representation of cause-and-effect relationships within a problem domain. More formally, a BN is a Directed Acyclic Graph (DAG) in which the nodes represent variables of interest (propositions), the directed links represent the causal influence among the variables, and the quantification of the influence is represented by conditional probability tables (CPT). Bayesian networks are powerful tools for real-time security analysis *if a BN model can be built that reflects reality* [19]. However, it is important to note that is not trivial to construct a BN from an attack graph [19].

Before presenting the environmental and analytical models of the application example, we first describe the predicates of the sets \mathbf{P}_{EM} and \mathbf{P}_{AM} and their respective meanings—these can be seen in Figure 1. As shown in the table, some of these predicates comprise the analytical model (AM), while others are part of the environmental model (EM). For instance, predicates that describe the behavior of the system under certain circumstances are part of the analytical

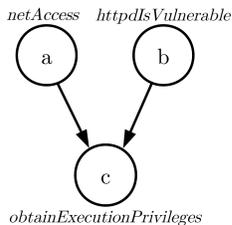


Fig. 2. Simple attack structure.

Worlds	a	b	c	$Pr(\lambda_i)$
λ_1	T	T	T	0.39
λ_2	T	T	F	0.15
λ_3	T	F	T	0.06
λ_4	T	F	F	0.09
λ_5	F	T	T	0.10
λ_6	F	T	F	0.09
λ_7	F	F	T	0.02
λ_8	F	F	F	0.10

Fig. 3. Probability distribution over \mathcal{W}_{EM} .

model. On the other hand, the environmental model contains predicates that are associated with uncertain events.

3.1 Environmental Model

In Figure 2 we show the simple attack structure (representing inherent uncertainty) that we use in our application example, which is taken from [19]. We have the following nodes, which are all Boolean (i.e., can take value *true* or *false*):

- (a) The attacker obtains network access to the Web server running on tcp/80 (*netAccess*).
- (b) The program `httpd` is a service running on the Web server as user `apache`, listening on tcp/80, and is known to be vulnerable (*httpdIsVulnerable*).
- (c) The attacker obtains code execution privileges on the Web server (*obtainExecutionPrivileges*).

The relationship among these nodes is simple: “nodes *a* and *b* together set the stage for *c*”. Hence, we can obtain the basic attack structure, as shown in Figure 2. This logic can be represented by an adequate arrangement of the graph structure and via the conditional probability tables (CPT).

The Bayesian network depicted in Figure 2 describes the relationship between the nodes; due to lack of space, and to improve readability, we omit the detailed specification of the CPTs and instead provide directly the complete probability distribution Pr over all possible worlds \mathcal{W}_{EM} in Figure 3. So, for instance, the probability that an attacker obtains execution privileges (Node *c*), once they have gained access to the network (Node *a*) but there is no known vulnerability in the `httpd` program that is running on the server (Node *b*) is 0.06 (world λ_3). With this, we have defined the environmental model; the next step involves defining the analytical model.

3.2 Analytical Model

Recall that all atoms in the AM must be formed with a predicate from the set \mathbf{P}_{AM} , and that in both strict and defeasible rules, *strong negation* (i.e., classical negation as in first-order logic) is allowed in the head; thus, this machinery may

$$\begin{aligned}
\Theta : \theta_1 &= \text{errorSystemsLogs} \\
\Omega : \omega_1 &= \text{vulnerable} \leftarrow \text{compromised} \\
&\omega_2 = \text{underAttack} \leftarrow \text{compromised}, \text{errorSystemsLogs} \\
\Phi : \phi_1 &= \text{compromised} \prec \\
&\phi_2 = \text{qualifiedPersonnel} \prec \\
\Delta : \delta_1 &= \text{underAttack} \prec \text{compromised} \\
&\delta_2 = \text{webServerVulnerable} \prec \text{runningOldhttpd} \\
&\delta_3 = \sim \text{vulnerable} \prec \text{qualifiedPersonnel}
\end{aligned}$$

Fig. 4. Ground argumentation framework from the running example.

$$\begin{aligned}
af(\theta_1) &= \text{true} \\
af(\omega_1) &= \text{true} \\
af(\omega_2) &= \text{netAccess} \vee \text{obtainExecutionPrivileges} \\
af(\phi_1) &= \text{netAccess} \\
af(\phi_2) &= \text{true} \\
af(\delta_1) &= \text{true} \\
af(\delta_2) &= \text{httpIsVulnerable} \\
af(\delta_3) &= \text{true}
\end{aligned}$$

Fig. 5. An example of an annotation function over the program in Figure 4.

be used to represent contradictory knowledge. In Figure 4 we show the PreDeLP program over the application example.

This program encodes some basic knowledge of the domain; for instance, strict rule ω_2 states that if errors are found in the system logs (*errorSystemsLogs*) and the system is compromised (*compromised*), then the system is under attack (*underAttack*). The defeasible rule δ_2 states that if an old version of the `httpd` service is running on the Web server (*runningOldhttpd*), then we can assume that the web server is vulnerable (*vulnerable*). Following Definition 2, some arguments that can be obtained are:

$$\begin{aligned}
\langle \mathcal{A}_1, \text{vulnerable} \rangle & \quad \mathcal{A}_1 = \{\phi_1, \omega_1\} \\
\langle \mathcal{A}_2, \text{underAttack} \rangle & \quad \mathcal{A}_2 = \{\theta_1, \omega_2, \phi_1\} \\
\langle \mathcal{A}_3, \text{underAttack} \rangle & \quad \mathcal{A}_3 = \{\phi_1, \delta_1\} \\
\langle \mathcal{A}_4, \sim \text{vulnerable} \rangle & \quad \mathcal{A}_4 = \{\phi_2, \omega_3\}
\end{aligned}$$

The next step is to define the annotation function—recall that such functions associate elements of $\Omega \cup \Theta \cup \Delta \cup \Phi$ with elements in $form_{EM}$. Figure 5 shows the annotations for our application example; for instance, the annotation for rule ϕ_1 means that this rule only holds whenever the probabilistic event *netAccess* is true. Elements that are annotated with “true” hold in all possible worlds.

$$\begin{aligned}
\lambda_1 : \left\{ \begin{array}{l} \theta_1, \omega_1, \omega_2, \\ \phi_1, \phi_2, \delta_1, \\ \delta_2, \delta_3 \end{array} \right\} & \lambda_2 : \left\{ \begin{array}{l} \theta_1, \omega_1, \omega_2, \\ \phi_1, \phi_2, \delta_1, \\ \delta_2, \delta_3 \end{array} \right\} & \lambda_3 : \left\{ \begin{array}{l} \theta_1, \omega_1, \omega_2, \\ \phi_1, \phi_2, \delta_1, \\ \delta_3 \end{array} \right\} & \lambda_4 : \left\{ \begin{array}{l} \theta_1, \omega_1, \omega_2, \\ \phi_1, \phi_2, \delta_1, \\ \delta_3 \end{array} \right\} \\
\lambda_5 : \left\{ \begin{array}{l} \theta_1, \omega_1, \omega_2, \\ \phi_2, \delta_1, \delta_2, \\ \delta_3 \end{array} \right\} & \lambda_6 : \left\{ \begin{array}{l} \theta_1, \omega_1, \phi_2, \\ \delta_1, \delta_2, \delta_3 \end{array} \right\} & \lambda_7 : \left\{ \begin{array}{l} \theta_1, \omega_1, \omega_2, \\ \phi_2, \delta_1, \delta_3 \end{array} \right\} & \lambda_8 : \left\{ \begin{array}{l} \theta_1, \omega_1, \phi_2, \\ \delta_1, \delta_3 \end{array} \right\}
\end{aligned}$$

Fig. 6. A depiction of how the DeLP3E program in the application example can be decomposed into one classical PreDeLP program for each possible EM world.

3.3 DeLP3E Program

In summary, for our use case we have built the EM model from Figure 3 (Π_{EM}), the AM model from Figure 4 (Π_{AM}), and the annotation function from Figure 5 (af)—these components give rise to DeLP3E program $\mathcal{I}_{AE} = (\Pi_{EM}, \Pi_{AM}, af)$. Figure 6 shows how \mathcal{I}_{AE} can be decomposed into one classical PreDeLP program $\Pi_{AM}(w)$ for each world $\lambda \in \mathcal{W}_{EM}$.

For instance, $\Pi_{AM}(\lambda_7)$ contains $\theta_1, \omega_1, \phi_2, \delta_1$, and δ_3 because the annotation function associates condition *true* to all of these components; it contains ω_2 because the formula *netAccess* \vee *obtainExecutionPrivileges* is satisfied by λ_7 (since *obtainExecutionPrivileges* is *true*), and it does not contain ϕ_1 and δ_2 because the conditions *netAccess* and *httpIsVulnerable* are *false* in λ_7 .

We now show a set of arguments and the corresponding worlds in which they can exist (as per Definition 7):

$\langle \mathcal{A}_1, \textit{vulnerable} \rangle$	$\mathcal{A}_1 = \{\phi_1, \omega_1\}$	$\{\lambda_1, \lambda_2, \lambda_3, \lambda_4\}$
$\langle \mathcal{A}_2, \textit{underAttack} \rangle$	$\mathcal{A}_2 = \{\theta_1, \omega_2, \phi_1\}$	$\{\lambda_1, \lambda_2, \lambda_3, \lambda_4\}$
$\langle \mathcal{A}_3, \textit{underAttack} \rangle$	$\mathcal{A}_3 = \{\phi_1, \delta_1\}$	$\{\lambda_1, \lambda_2, \lambda_3, \lambda_4\}$
$\langle \mathcal{A}_4, \sim \textit{vulnerable} \rangle$	$\mathcal{A}_4 = \{\phi_2, \omega_3\}$	$\{\lambda_1, \lambda_2, \lambda_3, \lambda_4, \lambda_5, \lambda_6, \lambda_7, \lambda_8\}$
$\langle \mathcal{A}_5, \textit{qualifiedPersonnel} \rangle$	$\mathcal{A}_5 = \{\phi_2\}$	$\{\lambda_1, \lambda_2, \lambda_3, \lambda_4, \lambda_5, \lambda_6, \lambda_7, \lambda_8\}$
$\langle \mathcal{A}_6, \textit{compromised} \rangle$	$\mathcal{A}_6 = \{\phi_1\}$	$\{\lambda_1, \lambda_2, \lambda_3, \lambda_4\}$

We will now focus on how such arguments can be shown to a user as part of human-in-the-loop threat analysis tool.

Towards a Graphical Threat Analysis Tool

The visualization and analysis of these programs, as well as the warranted literals they yield, can be carried out in an extension of the recently introduced DAQAP platform [6]—the main change required for this to be possible is the addition of the capability to incorporate probabilistic models to be used in the EM, and annotation functions for the AM. The DeLP Graph of worlds λ_1 and λ_8 are depicted Figures 7 and 8, respectively.

Based on these graphs, for the DeLP3E program $\mathcal{I}_{AE} = (\Pi_{EM}, \Pi_{AM}, af)$, we can compute an upper and lower bound on the probability of a particular literal

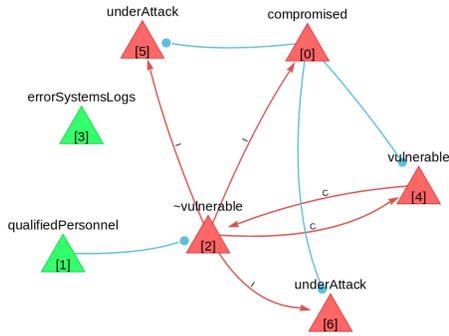


Fig. 7. DeLP Graph of world λ_1 .

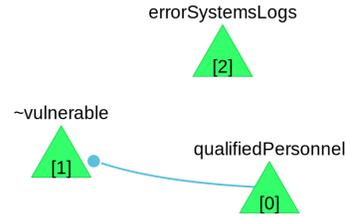


Fig. 8. DeLP Graph of world λ_8 .

(as per Definition 9). The simplest way to do this is to carry out an exhaustive analysis based of the events of the domain and the rules applied according to the existence of such events.

Consider for instance the query for the literal $\sim vulnerable$. The set of worlds that are *warranting scenarios* (cf. Definition 8), i.e., the “necessary” worlds are:

$$nec(\sim vulnerable) = \{\lambda_5, \lambda_6, \lambda_7, \lambda_8\}.$$

On the other hand, the “possible” worlds are:

$$poss(\sim vulnerable) = \{\lambda_1, \lambda_2, \lambda_3, \lambda_4, \lambda_5, \lambda_6, \lambda_7, \lambda_8\} = \mathcal{W}_{EM}.$$

By summing the probabilities of each set of worlds (according to the distribution given in Figure 3), we have that:

$$\begin{aligned} - \sum_{\lambda_i \in nec(\sim vulnerable)} Pr(\lambda_i) &= 0.10 + 0.09 + 0.02 + 0.10 = 0.31 \\ - \sum_{\lambda_i \in poss(\sim vulnerable)} Pr(\lambda_i) &= 1. \end{aligned}$$

Therefore, we can conclude that: $0.31 \leq Pr(\sim vulnerable) \leq 1$; this reflects the fact that the elements in the AM related to having qualified personnel are labeled with *true*. In this way, we can compute the probability intervals associated with each literal and display them in a DeLP Graph, as illustrated in Figure 9. Thus, the analyst can clearly visualize the state of the system and the threats that might be present, perform an evaluation, and execute actions if necessary—for instance, in the example above, staff rotation might trigger changes in the program, causing $Pr(\sim vulnerable)$ to drop.

Though the example presented here is necessarily simplified, it is clear that these capabilities provide human analysts with the means to model complex situations in cyber threat analysis, but saving them from having to carry out the reasoning associated with such situations, which more often than not become too involved to be done manually. Perhaps most importantly, the results are constructive, so the analyst can “trace back” results to better understand how the system arrived at a given conclusion, and make changes in the facts, presumptions, or probabilities if necessary.

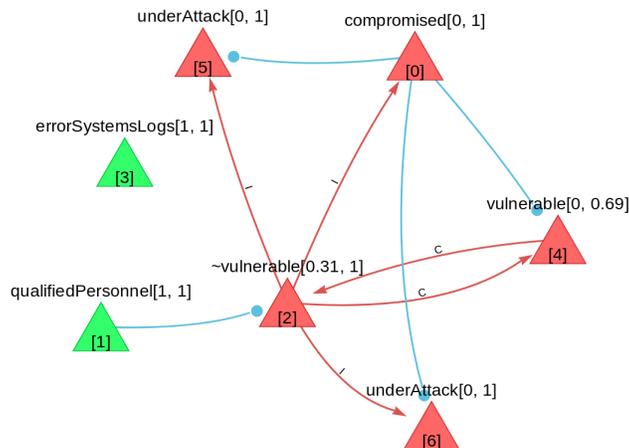


Fig. 9. Example of how results can be visualized as a graph in the DAQAP platform; literals are labeled with the probability intervals derived from the DeLP3E model.

4 Conclusions

In this work we presented an application example to show the capabilities of the DeLP3E framework to handle scenarios in cyber security analysis; in particular, we focused on near real-time security analyses such as *intrusion response* through a combination of probabilistic modeling and argumentative reasoning. We adopted the DeLP3E framework given its unique combination of structured argumentation and annotations that refer to events for which we have underlying probabilistic information—this separation of concerns yields robust models that can handle two essentially different types of uncertainty: probabilistic and that arising from defeasible reasoning.

Ongoing and future work involves the extension of the DAQAP tool to provide security analysts with a platform that allows them, through adequate user interfaces, to carry out complex cyber threat analysis tasks. The following key areas also need to be explored: developing (exact or approximate) algorithms that are more efficient than an exhaustive analysis of possible worlds; this is especially necessary to be able to handle large datasets; semi-automatically learning the EM and AM from data; develop user interfaces that maximize usability and minimize comprehension and response time, which is often crucial in the field; and develop a full version of the DeLP3E framework and deploy it in the DAQAP platform. Future work will be carried out in these directions, focusing on the use of both real and synthetic datasets for empirical evaluation.

Acknowledgments. This work was funded in part by Universidad Nacional del Sur (UNS) under grants PGI 24/N046 and PGI 24/ZN34, and CONICET under grant PIP 11220170100871CO, Argentina, and the EU H2020 Research and

Innovation Programme under the Marie Skłodowska-Curie grant agreement No. 690974 for the project “MIREL”.

References

1. Dung, P.M.: On the acceptability of arguments and its fundamental role in non-monotonic reasoning, logic programming and n-person games. *Artificial intelligence* **77**(2), 321–357 (1995)
2. Frigault, M., Wang, L., Singhal, A., Jajodia, S.: Measuring network security using dynamic bayesian network. In: *Proc. Quality of Protection*. pp. 23–30. ACM (2008)
3. García, A.J., Simari, G.R.: Defeasible logic programming: An argumentative approach. *TPLP* **4**(1-2), 95–138 (2004)
4. Graf, J., Eastwood, S., Yanushkevich, S., Ferber, R.: Risk inference models for security applications. In: *2019 Eighth International Conference on Emerging Security Technologies (EST)*. pp. 1–6. IEEE (2019)
5. Heuer, R.J.: *Psychology of intelligence analysis*. Jeffrey Frank Jones (1999)
6. Leiva, M.A., Simari, G.I., Gottifredi, S., García, A.J., Simari, G.R.: DAQAP: Defeasible argumentation query answering platform. In: *Proc. FQAS 2019*. pp. 126–138 (2019)
7. Martinez, M.V., García, A.J., Simari, G.R.: On the use of presumptions in structured defeasible reasoning (2012)
8. Misuri, A., Khakzad, N., Reniers, G., Cozzani, V.: A bayesian network methodology for optimal security management of critical infrastructures. *Reliability Engineering & System Safety* (2018)
9. Nilsson, N.J.: Probabilistic logic. *Artificial intelligence* **28**(1), 71–87 (1986)
10. Pearl, J.: *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Elsevier (2014)
11. Richardson, M., Domingos, P.: Markov logic networks. *Machine learning* **62**(1-2), 107–136 (2006)
12. Shakarian, P., Simari, G.I., Falappa, M.A.: Belief revision in structured probabilistic argumentation. In: *Proc. FoIKS*. pp. 324–343. Springer (2014)
13. Shakarian, P., Simari, G.I., Moores, G., Parsons, S., Falappa, M.A.: An argumentation-based framework to address the attribution problem in cyberwarfare. In: *Proc. CyberSecurity. ASE* (2014)
14. Shakarian, P., Simari, G.I., Moores, G., Paulo, D., Parsons, S., Falappa, M.A., Aleali, A.: Belief revision in structured probabilistic argumentation. *Annals of Mathematics and Artificial Intelligence* **78**(3-4), 259–301 (2016)
15. Shameli-Sendi, A., Ezzati-Jivan, N., Jabbarifar, M., Dagenais, M.: Intrusion response systems: survey and taxonomy. *Int. J. Comput. Sci. Netw. Secur* **12**(1), 1–14 (2012)
16. Simari, G.R., Loui, R.P.: A mathematical treatment of defeasible reasoning and its implementation. *Artificial intelligence* **53**(2-3), 125–157 (1992)
17. Stolzenburg, F., García, A.J., Chesñevar, C.I., Simari, G.R.: Computing generalized specificity. *Journal of Applied Non-Classical Logics* **13**(1), 87–113 (2003)
18. Stolzenburg, F., García, A.J., Chesñevar, C.I., Simari, G.R.: Computing generalized specificity. *Journal of Applied Non-Classical Logics* **13**(1), 87–113 (2003)
19. Xie, P., Li, J.H., Ou, X., Liu, P., Levy, R.: Using bayesian networks for cyber security analysis. In: *2010 IEEE/IFIP International Conference on Dependable Systems & Networks (DSN)*. pp. 211–220. IEEE (2010)