The Workshops of the Thirtieth AAAI Conference on Artificial Intelligence
Artificial Intelligence for Cyber Security: Technical Report WS-16-03

Toward Argumentation-Based Cyber Attribution

Eric Nunes and **Paulo Shakarian**
Arizona State University
Tempe, AZ 85281, USA
Email: {enunes1, shak}@asu.edu

Gerardo I. Simari
Inst. for Computer Science and Engineering
(Univ. Nac. del Sur–CONICET), Bahia Blanca, Argentina
E-mail: gis@cs.uns.edu.ar

Abstract

A major challenge in cyber-threat analysis is combining information from different sources to find the person or the group responsible for the cyber-attack. It is one of the most important technical and policy challenges in cyber-security. The lack of ground truth for an individual responsible for an attack has limited previous studies. In this paper, we overcome this limitation by building a dataset from the capture-the-flag event held at DEFCON, and propose an argumentation model based on a formal reasoning framework called DeLP (Defeasible Logic Programming) designed to aid an analyst in attributing a cyber-attack to an attacker. We build argumentation-based models from latent variables computed from the dataset to reduce the search space of culprits (attackers) that an analyst can use to identify the attacker. We show that reducing the search space in this manner significantly improves the performance of classification-based approaches to cyber-attribution.

Introduction

A major challenge in cyber-threat analysis is to find the person or the group responsible for a cyber-attack. This is known as cyber-attribution (Shakarian, Shakarian, and Ruef 2013) and it is one of the central technical and policy challenges in cyber-security. Oftentimes, the evidence collected from multiple sources provides a contradictory viewpoint. This gets worse in cases of deception where either an attacker plants false evidence or the evidence points to multiple actors, leading to uncertainty. In our text on cyber-warfare (Shakarian, Shakarian, and Ruef 2013) we discuss the difficulties that an intelligence analyst faces in attributing an attack to a perpetrator given that deception might have occurred, and how the analyst needs to explore deception hypotheses under the given attack scenario.

However, one of the major drawbacks of the study and evaluation of cyber-attribution models is the lack of datasets with the ground truth available regarding the individual party responsible for the attack—this has limited previous studies. To overcome this, we built and leveraged a dataset from the capture-the-flag event held at DEFCON (cf. the dataset section for a detailed discussion). In previous work, we used this dataset to study cyber-attribution, and framed it as a multi-label classification problem and applied several machine learning and pruning techniques; the results of this work are discussed in (Nunes et al. 2015). Based on our observations, machine learning approaches fail in situations of deception, where similar attributes point towards multiple attackers. Standard machine learning approaches treat this problem as "labeling noise" leading to a random selection of attacker. We propose to address this issue using a formal logical framework. Specific contributions of this paper include,

- a model for cyber-attribution created in the DeLP argumentation framework;
- experiments demonstrating that using argumentation-based tools can significantly reduce the number of potential culprits in a cyber-attack; and
- experiments showing that the reduced set of culprits, used in conjunction with classification, leads to improved cyber-attribution decisions.

Related Work

Currently, cyber-attribution is limited to identifying machines (Boebert 2010) as opposed to the hacker or their affiliation to a group or a state organization. An example of such a technical attribution approach is WOMBAT (Dacier, Pham, and Thonnard 2009), where a clustering technique is used to group attacks to common IP sources. A method that combines information from different sources was proposed in (Walls 2014), where forensic information from diverse sources is considered but inconsistency or uncertainty due to deception is not considered. A less rigorous mathematical model, known as the Q model (Rid and Buchanan 2015), has been proposed recently; there, the model answers queries from an analyst, both technical (tools used) and non-technical (environment related), and by combining these query answers the analyst tries to attribute an attack to a party. Unfortunately, there are no experimental results evaluating the effectiveness of this model.

Concurrently, we have devised a formal logical framework for reasoning about cyber-attribution (Shakarian et al. 2015a; 2015b). This reasoning model explores multiple competing hypotheses based on the evidence for and against a particular attacker before it attributes the attack to a specific party. With this approach we get a clear map regarding

the arguments that led to the decision. This paper is the first to provide experimental results using a formal logical framework to build a reasoning model.

The rest of the paper is organized as follows. We present the argumentative model based on Default Logic Programming (DeLP) (García and Simari 2004). This is followed by a description of our DEFCON capture-the-flag dataset and an analysis on the use of deception within this data. We then give a summary of our results from (Nunes et al. 2015) and discuss how we built our argumentation model for cyber-attribution with DeLP. Experimental results are discussed in the subsequent section.

DEFCON CTF Dataset

The DEFCON security conference sponsors and hosts a capture-the-flag (CTF) competition every year, held on site with the conference in Las Vegas, Nevada. The CTF competition can be viewed as a microcosm of the global Internet and the careful game of cat and mouse between hacking groups and security firms. Teams are free to use different technical means to discover vulnerabilities: they may reverse engineer programs, monitor the network data sent to their services, and dynamically study the effects that network data has on unpatched services. If a team discovers a vulnerability and uses it against another team, the first team may discover that their exploit is re-purposed and used against them within minutes.

The organizers of DEFCON CTF capture all of the network traffic sent and received by each team, and publish this traffic at the end of the competition (DEFCON 2013). This includes IP addresses for source and destination, as well as the full data sent and received and the time the data was sent or received. This data is not available to contestants; depending on the organizers' choice from year to year, the contestants either have a real time feed but with the IP address obscured, or a full feed delivered on a time delay ranging from minutes to hours.

In addition to the traffic captures, copies of the vulnerable services are distributed by the organizers, who usually do not disclose the vulnerabilities they engineered into each service; however, competitors frequently disclose this information publicly after the game is finished by preparing technical write-ups.

The full interaction of all teams in the game environment are captured by this data. We cannot build a total picture of the game at any point in time, since there is state information from the servers that is not captured, but any exploit attempt would have to travel over the network and that would be observed in the data set.

Analysis

We use the data from the CTF tournament held at DEFCON 21 in 2013; the dataset is very large, about 170 GB in compressed format. We used multiple systems with distributed and coordinated processing to analyze the entire corpus—fortunately, analyzing individual streams is easy to parallelize. To analyze this data, we identified the TCP ports associated with each vulnerable service. From this information, we used the open source tool *tcpflow*[1] to process the network captures into a set of files, with each file representing data sent or received on a particular connection.

This produced a corpus of data that could be searched and processed with standard UNIX tools, like *grep*. Further analysis of the game environment provided an indicator of when a data file contained an exploit. The game stored keys for services in a standard, hard-coded location on each competitor's server. By searching for the text of this location in the data, we identified data files that contained exploits for services.

With these data files identified, we analyzed some of them by hand using the Interactive Disassembler (IDA) to determine if the data contained shell-code, which in fact was the case. We used an automated tool to produce a summary of each data file as a JSON encoded element. Included in this summary was a hash of the contents of the file and a histogram of the processor instructions contained in the file. These JSON files were the final output of the low level analysis, transforming hundreds of gigabytes of network traffic into a manageable set of facts about exploit traffic in the data. Each JSON file is a list of tuples (time-stamp, hash, byte-histogram, instruction-histogram). The individual fields of the tuple are listed in Table 1.

Table 1: Fields in an instance of network attack

Field	**Intuition**
byte_hist	Histogram of byte sequences in the payload
inst_hist	Histogram of instructions used in the payload
from_team	The team where the payload originates (attacking team)
to_team	The team being attacked by the exploit
svc	The service that the payload is running
payload_hash	The md5 of the payload used
time	Date and time of the attack

Table 2: Example event from the dataset

Field	**Value**
byte_hist	0×43:245, 0×69:8, $0\times3a$:9, $0\times5d$:1, ...
inst_hist	cmp:12, svcmi:2, subs:8, movtmi:60 .....
from_team	Robot Mafia
to_team	Blue Lotus
svc	02345
payload_hash	2cc03b4e0053cde24400bbd80890446c
time	2013-08-03T23:45:17

[1] https://github.com/simsong/tcpflow

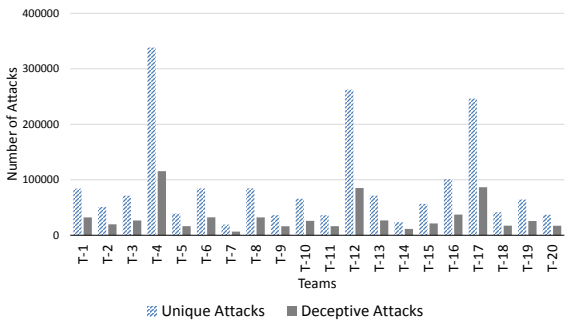

Figure 1: Number of unique and deceptive attacks directed towards each team.

After this pre-processing of the network data packets, we obtained around 10 million network attacks. There are 20 teams in the CTF competition; in order to attribute an attack to a particular team, apart from analyzing the payloads used by the team, we also need to analyze the behavior of the attacking team towards their adversary. For this purpose, we separate the network attacks according to the team being targeted. Thus, we have 20 such subsets, which we represent as T-i, where $i \in \{1, 2, 3, ..., 20\}$. An example of an event in the dataset is shown in Table 2.

We now discuss two important observations from the dataset, which make the task of attributing an observed network attack to a team difficult.

Deception: In the context of this paper we define an attack to be *deceptive* when multiple adversaries get mapped to a single attack pattern. In the current setting we define deception as the scenario in which the same exploit is used by multiple teams to target the same team. Figure 1 shows the distribution of unique deception attacks with respect to the total unique attacks in the dataset based on the target team. These unique deceptive attacks amount to just under 35% of the total unique attacks.

Duplicate attacks: A duplicate attack occurs when the same team uses the same payload to attack a team at different points in time. Duplicate attacks can be attributed to two reasons. First, when a team is trying to compromise another's system, it does not just launch a single attack but a wave of attacks with very little time difference between consecutive attacks. Second, once a successful payload is created that can penetrate the defense of other systems, it is used more by the original attacker as well as the deceptive one as compared to other payloads. We group duplicates as either being *non-deceptive* or *deceptive*. Non-deceptive duplicates are the copies of the attacks launched by the team that first initiated the use of a particular payload; on the other hand, deceptive duplicates are all the attacks from the teams that did not initiate the use.

Argumentation Model

Our approach relies on a model of the world where we can analyze competing hypotheses in a cyber-operation scenario. Such a model must allow for contradictory information, as it must have the capability to reason about inconsistency in cases of deception.

Before describing the argumentation model in detail, we introduce some necessary notation. Variables and constant symbols represent items such as the exploits/payloads used for the attack, and the actors conducting the cyber-attack (in this case, the teams in the CTF competition). We denote the set of all variable symbols with $\mathbf{V}$ and the set of all constants with $\mathbf{C}$. For our model we require two subsets of $\mathbf{C}$: $\mathbf{C}_{act}$, denoting the actors capable of conducting the cyber-operation, and $\mathbf{C}_{exp}$, denoting the set of unique exploits used. We use symbols in all capital letters to denote variables. The running example is based on a subset of our DEFCON CTF dataset:

Example 1. *Consider the following actors and cyber-operations from the CTF data:*

$$\begin{aligned} \boldsymbol{C}_{act} &= \{\textit{bluelotus, robotmafia, apt8}\} \\ \boldsymbol{C}_{exp} &= \{\textit{exploit}_1, \textit{exploit}_2, ..., \textit{exploit}_n\} \end{aligned}$$

■

The language also contains a set of predicate symbols that have constants or variables as arguments, and denote events that can be either *true* or *false*. We denote the set of predicates with $\mathbf{P}$; examples of predicates are shown in Table 3. For instance, $\textit{culprit}(\textit{exploit}_1, \textit{bluelotus})$ will either be true or false, and denotes the event where *bluelotus* used $\textit{exploit}_1$ to conduct a cyber-operation.

A ground atom is composed by a predicate symbol and a tuple of constants, one for each of the predicate's arguments. The set of all ground atoms is denoted as $\mathbf{G}$. A literal L is a ground atom or a negated ground atom; hence, literals have no variables. Examples of ground atoms formed using predicates in Table 3 are shown in the following example. We denote a subset of $\mathbf{G}$ with $\mathbf{G}'$.

Example 2. *The following are examples of ground atoms over the predicates given in Table 3.*

$$\begin{aligned} \boldsymbol{G}' : \quad & \textit{attack}(\textit{exploit}_1, \textit{bluelotus}), \\ & \textit{deception}(\textit{exploit}_1, \textit{apt8}), \\ & \textit{culprit}(\textit{exploit}_1, \textit{apt8}) \end{aligned}$$

Table 3: Example Predicates and explanation

Predicate	**Explanation**
attack($\textit{exploit}_1$, *bluelotus*)	$\textit{exploit}_1$ was targeted towards the team Blue Lotus.
replay_attack($\mathcal{E}$, *Y*)	Exploit $\mathcal{E}$ was replayed by team *Y*.
deception($\textit{exploit}_1$, *apt8*)	Team *apt8* used $\textit{exploit}_1$ for deception.
time_diff(I, *Y*)	Team *Y* was deceptive within the given time interval I.
culprit($\textit{exploit}_1$, *apt8*)	Team *apt8* is the likely culprit for the attack (using $\textit{exploit}_1$ on the target team).

■

We choose a structured argumentation framework (Rahwan, Simari, and van Benthem 2009) to build our argumentation model, which allows for competing ideas to deal with contradictory information. Due to such characteristics, argumentation-based models are favorable for cyber-attack scenarios. Our approach works by creating arguments (in the form of a set of rules and facts) that compete with each other to attribute an attack to a given perpetrator. In this case, arguments are defeated based on contradicting information in other arguments. This procedure is known as a *dialectical process*, where the arguments that are undefeated prevail. An important result is the set of all the arguments that are *warranted* (not defeated) by any other argument, which give a clear map of not only the set of attackers responsible for the cyber-operation but also the arguments supporting the conclusion.

The transparency of the model lets a security analyst not only add new arguments based on new evidence discovered in the system, but also get rid of incorrect information and fine-tune the model for better performance. Since the argumentation model can deal with inconsistent information, it draws a natural analogy to the way humans settle disputes when there is contradictory information available (García and Simari 2004). The model provides a clear explanation as to why one argument is chosen over others, which is a desirable characteristic for both the analyst and for organizations to make decisions and policy changes. We now discuss some preliminaries for the argumentation model.

Defeasible Logic Programming

DeLP is a formalism that combines logic programming with defeasible argumentation; full details are discussed in (García and Simari 2004). The formalism is made up of several constructs, namely facts, strict rules, and defeasible rules. Facts represent statements obtained from evidence, and are always true; similarly, strict rules are logical combinations of facts that always hold. On the contrary, defeasible rules can be thought of as strict rules that may be true in some situations, but could be false if contradictory evidence is present. These three constructs are used to build arguments. DeLP programs are sets of facts, strict rules and defeasible rules. We use the usual notation to denote DeLP programs: denoting the knowledge base with $\Pi = (\Theta, \Omega, \Delta)$, where Θ is the set of facts, Ω is the set of strict rules, and Δ is the set of defeasible rules. Examples of the three constructs are provided with respect to the dataset in Figure 2 and in the *Latent variables* section. We now describe the constructs in detail.

Facts (Θ) are ground literals that represent atomic information or its negation ($\neg$). The facts are always true and cannot be contradicted.

Strict Rules (Ω) represent cause and effect information that is always true. They are built from using a combination of ground literals and are of the form $L_0 \leftarrow L_1, ...L_n$, where L_0 is a ground literal and $\{L_i\}_{i>0}$ is a set of ground literals.

Defeasible Rules (Δ) comprise knowledge that can be true if no contradictory information is provided. These rules are the defeasible counterparts of strict rules; they are of the form $L_0 \prec L_1, ..., L_n$, where L_0, is the ground literal and $\{L_i\}_{i>0}$ is a set of ground literals. Strong negation is allowed for both strict and defeasible rules to represent contradictory information.

Θ :
- $\theta_1 =$ attack($exploit_1$, *bluelotus*)
- $\theta_2 =$ first_attack($exploit_1$, *robotmafia*)
- $\theta_3 =$ last_attack($exploit_1$, *apt8*))
- $\theta_4 =$ time_diff(*interval*, *robotmafia*)
- $\theta_5 =$ most_frequent($exploit_1$, *pwnies*)

Ω :
- $\omega_1 =$ culprit($exploit_1$, *pwnies*) $\leftarrow$ most_frequent($exploit_1$, *pwnies*), replay_attack($exploit_1$)
- $\omega_2 =$ $\neg$ culprit($exploit_1$, *robotMafia*) $\leftarrow$ last_attack($exploit_1$, *apt8*), replay_attack($exploit_1$)

Δ :
- $\delta_1 =$ replay_attack($exploit_1$) $\prec$ attack($exploit_1$, *bluelotus*), last_attack($exploit_1$, *apt8*)
- $\delta_2 =$ deception($exploit_1$, *apt8*) $\prec$ replay_attack($exploit_1$), first_attack($exploit_1$, *robotmafia*)
- $\delta_3 =$ culprit($exploit_1$, *apt8*) $\prec$ deception($exploit_1$, *apt8*), replay_attack($exploit_1$)
- $\delta_4 =$ $\neg$culprit($exploit_1$, *apt8*) $\prec$ time_diff(*interval*, *robotmafia*)

Figure 2: A ground argumentation framework.

When a cyber-attack occurs, the model derives arguments as to who could have conducted the attack. Derivation follows the same mechanism as logic programming (Lloyd 2012). DeLP incorporates defeasible argumentation, which decides which arguments are warranted and it blocks arguments that are in conflict.

Figure 2 shows a ground argumentation framework demonstrating constructs derived from the CTF data. For instance, θ_1 indicates the fact that $exploit_1$ was used to target the team *Blue Lotus*, and θ_5 indicates that team *pwnies* is the most frequent user of $exploit_1$. For the strict rules, ω_1 says that for a given $exploit_1$ the attacker is *pwnies* if it was the most frequent attacker and the attack $exploit_1$ was replayed. Defeasible rules can be read similarly; δ_2 indicates that $exploit_1$ was used in a deceptive attack by APT8 if it was replayed and the first attacker was not APT8. By replacing the constants with variables in the predicates we can derive a non-ground argumentation framework.

Definition 1. ***(Argument)*** *An argument for a literal L is a pair $\langle \mathcal{A}, L \rangle$, where $\mathcal{A} \subseteq \Pi$ provides a minimal proof for L meeting the requirements: (1) L is defeasibly derived from $\mathcal{A}$, (2) $\Theta \cup \Omega \cup \Delta$ is not contradictory, and (3) $\mathcal{A}$ is a minimal subset of Δ satisfying 1 and 2, denoted $\langle \mathcal{A}, L \rangle$.*

$\langle \mathcal{A}_1, \textsf{replay\_attack}(\textit{exploit}_1) \rangle \quad \mathcal{A}_1 = \{\delta_1, \theta_1, \theta_3\}$
$\langle \mathcal{A}_2, \textsf{deception}(\textit{exploit}_1, \textit{apt8}) \rangle \quad \mathcal{A}_2 = \{\delta_1, \delta_2, \theta_2\}$
$\langle \mathcal{A}_3, \textsf{culprit}(\textit{exploit}_1, \textit{apt8}) \rangle \quad \mathcal{A}_3 = \{\delta_1, \delta_2, \delta_3\}$
$\langle \mathcal{A}_4, \neg\textsf{culprit}(\textit{exploit}_1, \textit{apt8}) \rangle \quad \mathcal{A}_4 = \{\delta_1, \delta_4, \theta_3\}$

Figure 3: Example ground arguments from Figure 2.

Literal L is called the conclusion *supported by the argument, and $\mathcal{A}$ is the* support. *An argument $\langle \mathcal{B}, L \rangle$ is a* subargument *of $\langle \mathcal{A}, L' \rangle$ iff $\mathcal{B} \subseteq \mathcal{A}$.*

The following shows arguments for our running example scenario.

Example 3. *Figure 3 shows example arguments based on the knowledge base from Figure 2. Note that the following relationship exists:*

$\langle \mathcal{A}_1,$ *replay_attack*$(\textit{exploit}_1) \rangle$ *is a subargument of* $\langle \mathcal{A}_2,$ *deception*$(\textit{exploit}_1, \textit{apt8}) \rangle$ *and* $\langle \mathcal{A}_3,$ *culprit*$(\textit{exploit}_1, \textit{apt8}) \rangle$.

■

For a given argument there may be counter-arguments that contradict it. For instance, referring to Figure 3, we can see that $\mathcal{A}_4$ attacks $\mathcal{A}_3$. A *proper defeater* of an argument $\langle A, L \rangle$ is a counter-argument 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. The default criterion used in DeLP for argument comparison is *generalized specificity* (Stolzenburg et al. 2003).

A sequence of arguments is referred to as an *argumentation line*. There can be more than one defeater argument, which leads to a tree structure that is built from the set of all argumentation lines rooted in the initial argument. In a dialectical tree, every child can defeat its parent (except for the root), and the leaves represent the undefeated arguments. The tree thus creates a map of all possible argumentation lines that decide if an argument is defeated or not.

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:

- 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, if the root of $\mathcal{T}^*(\langle \mathcal{A}, L \rangle)$ is marked "U", then $\mathcal{T}^*(\langle \mathcal{A}, h \rangle)$ *warrants* L and that L is *warranted* from Π. (Warranted arguments correspond to those in the grounded extension of a Dung argumentation system (Dung 1995).)

In practice, an implementation of DeLP accepts as input sets of facts, strict rules, and defeasible rules. Note that

Θ : $\theta_1 =$ attack ($\mathcal{E}$, *X*)
$\theta_2 =$ first_attack ($\mathcal{E}$, *Y*)
$\theta_3 =$ last_attack ($\mathcal{E}$, *Y*)

Figure 4: Facts defined for each test sample.

while the set of facts and strict rules is consistent (non-contradictory), the set of defeasible rules can be inconsistent. We engineer our cyber-attribution framework as a series of defeasible and strict rules whose structure we have created, but are dependent on values learned from a historical corpus. Then, for a given incident, we instantiate a set of facts for that situation. This information is then provided as input into a DeLP implementation that uses heuristics to generate all arguments for and against every possible culprit for the cyber attack. Then, the DeLP implementation creates dialectical trees based on these arguments and decides which culprits are *warranted*. This results in a reduced set of potential culprits, which we then use as input into a classifier to obtain an attribution decision.

Latent Variables

In (Nunes et al. 2015) we leveraged machine learning techniques on the CTF data to identify the attacker. We will now provide a summary of the results obtained.

The experiment was performed as follows. We divide the dataset according to the target team, building 20 subsets, and all the attacks are then sorted according to time. We reserve the first 90% of the attacks for training and the remaining 10% for testing. The byte and instruction histograms are used as features to train and test the model. Models constructed using a random forest classifier performed the best, with an average accuracy of 0.37. Most of the misclassified samples tend to be deceptive attacks and their duplicates. When using machine learning approaches it is difficult to map the reasons why a particular attacker was predicted, especially in cases of deception where multiple attackers were associated with the same attack. Knowing the arguments that supported a particular decision would greatly aid the analyst in making better decisions dealing with uncertainty. To address this issue we now describe how we can form arguments/rules based on the latent variables computed from the training data, given an attack for attribution.

We use the following notation: let $\mathcal{E}$ be the test attack under consideration aimed at target team *X*, *Y* represent all the possible attacking teams, and $\mathcal{D}$ be the set of all deceptive teams (those using the same payload to target the same team) if the given attack is deceptive in the training set. For non-deceptive attacks, $\mathcal{D}$ will be empty. We note that facts cannot have variables, only constants (however, to compress the program for presentation purposes, we use *meta-variables* in facts). To begin, we define the facts described in Figure 4; fact θ_1 states that attack $\mathcal{E}$ was used to target team *X*, θ_2 states that team *Y* was the first team to use the attack $\mathcal{E}$ in the training data, and similarly θ_3 states that team *Y* was the last team to use the attack $\mathcal{E}$ in the training data. The first and last attacking team may or may not be the same. We study

$$
\begin{array}{lll}
\Omega: & \omega_1 = & \text{culprit}(\mathcal{E}, Y) \leftarrow \text{last\_attack}(\mathcal{E}, Y), \\
 & & \quad \text{replay\_attack}(\mathcal{E}). \\
\hline
\Delta: & \delta_1 = & \text{replay\_attack}(\mathcal{E}) \prec \text{attack}(\mathcal{E}, X), \\
 & & \quad \text{last\_attack}(\mathcal{E}, Y).
\end{array}
$$

Figure 5: Defeasible and strict rule for non-deceptive attack.

$$
\begin{array}{lll}
\Theta: & \theta_1 = & \text{deception}\ (\mathcal{E}, X) \\
 & \theta_2 = & \text{frequent}\ (\mathcal{E}, \mathcal{D}_f) \\
\hline
\Omega: & \omega_1 = & \text{culprit}(\mathcal{E}, \mathcal{D}_f) \leftarrow \text{frequent}(\mathcal{E}, \mathcal{D}_f), \\
 & & \quad \text{deception}\ (\mathcal{E}, \mathcal{D}_i) \\
 & \omega_2 = & \neg\text{culprit}(\mathcal{E}, Y) \leftarrow \text{first\_attack}(\mathcal{E}, Y), \\
 & & \quad \text{deception}(\mathcal{E}, X) \\
\hline
\Delta: & \delta_1 = & \text{replay\_attack}(\mathcal{E}) \prec \text{attack}(\mathcal{E}, X), \\
 & & \quad \text{last\_attack}(\mathcal{E}, Y) \\
 & \delta_2 = & \text{deception}(\mathcal{E}, \mathcal{D}_i) \prec \text{replay\_attack}(\mathcal{E}), \\
 & & \quad \text{first\_attack}(\mathcal{E}, Y) \\
 & \delta_3 = & \text{culprit}(\mathcal{E}, \mathcal{D}_i) \prec \text{deception}(\mathcal{E}, \mathcal{D}_i), \\
 & & \quad \text{first\_attack}(\mathcal{E}, Y)
\end{array}
$$

Figure 6: Facts and rules for deceptive attacks.

the following three cases:

Case 1: Non-deceptive attacks. In non-deceptive attacks, only one team uses the payload to target other teams in the training data. It is easy to predict the attacker for these cases, since the search space only has one team. To model this situation, we define the set of defeasible and strict rules shown in Figure 5. Defeasible rule δ_1 checks whether the attack was replayed in the training data—since it is a non-deceptive attack, it can only be replayed by the same team. Strict rule ω_1 then puts forth an argument for the attacker (culprit) if the defeasible rule holds and there is no contradiction for it.

Case 2: Deceptive attacks. These attacks form the majority of the misclassified samples in (Nunes et al. 2015). The set $\mathcal{D}$ is not empty for this case; let $\mathcal{D}_i$ denote the deceptive teams in $\mathcal{D}$. We also compute the most frequent deceptive attacker from $\mathcal{D}$. Let the most frequent attacker be denoted as $\mathcal{D}_f$. Figure 6 shows some of the DeLP components that model this case. Fact θ_1 indicates if the attack $\mathcal{E}$ was deceptive towards the team X and θ_2 indicates the most frequent attacker team $\mathcal{D}_f$ from the deceptive team set $\mathcal{D}$. The strict rules ω_1 states that the attacker should be $\mathcal{D}_f$ if the attack is deceptive and ω_2 indicates that in case of deception the first attack team Y is not the attacker. For the defeasible rules, δ_1 deals with the case in which the attack $\mathcal{E}$ was replayed, δ_2 deals with the case of deceptive teams from the set $\mathcal{D}$ and δ_3 indicates that all the deceptive teams are likely to be the attackers in the absence of any contradictory information.

Case 3: Unseen Attacks. The most difficult attacks to attribute in the dataset are the unseen ones, i.e., attacks encountered in the test set that did not occur in the training set. To build constructs for this kind of attack we first compute

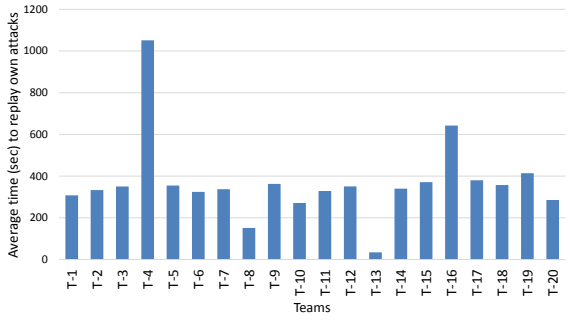

Figure 7: Average time for duplicate attacks for each team.

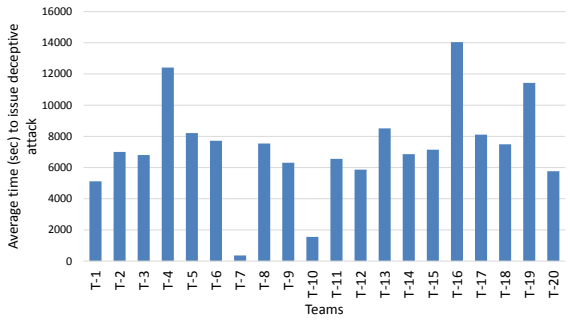

Figure 8: Average time for deceptive attacks for each team.

the K nearest neighbors from the training set according to a simple Euclidean distance between the byte and instruction histograms of the two attacks. In this case we choose $K = 3$. For each of the matching attack from the training data we check if the attack is deceptive or non-deceptive. If non-deceptive, we follow the procedure for Case 1, otherwise we follow the procedure for Case 2. Since we replace one unseen attack with three seen attacks, the search space for the attacker increases for unseen attacks.

Attacker Time Analysis

The CTF data provides us with time stamps for the attacks in the competition. We can use this information to come up with rules for/against an argument for a team being the attacker. We compute the average time for a team to replay its own attack given that it was the first one to initiate the attack (see Figure 7). It can be observed that teams like *more smoked leet chicken* (T-13) and *Wowhacker-bios* (T-8) are very quick to replay their own attacks as compared to other teams.

Figure 8 on the other hand shows the average time for a team to replay an attack initiated by some other team. This refers to the time the team takes to commit a deceptive attack. Teams like *The European* (T-7) and *Blue lotus* (T-10) are quick to commit deception, while others take more time.

We use this time information to narrow down our search space for possible attackers. In particular, for a deceptive test sample, we compute the time difference between the test sample and the training sample that last used the same payload. We denote this time difference as $\triangle t$, and include it

as a fact θ_1. We then divide the deceptive times from Figure 8 into appropriate intervals; each team is assigned to one of those time intervals. We then check which time interval $\triangle t$ belongs to and define a defeasible rule δ_1 that makes a case for all teams not belonging to the interval to not be the culprits, as shown in Figure 9.

$$\Theta: \quad \theta_1 = \quad \text{timedifference}\ (\mathcal{E}, X)$$

$$\Delta: \quad \delta_1 = \quad \begin{array}{l}\text{For } Y \notin \text{interval:} \\ \neg\text{culprit}(\mathcal{E}, Y) \prec \text{timedifference}\ (\mathcal{E}, X).\end{array}$$

Figure 9: Time facts and rules. Interval indicates a small portion of the entire deceptive time (for instance $<$ 2000 sec., $>$ 8000 sec., and so on).

Experiments

We use the same experimental setup as in (Nunes et al. 2015). We sort the dataset by time for each target team and then use the first 90% of the data for training and the rest 10% for testing. We compute the constructs for all test samples based on the cases discussed in the previous section, and then input these arguments to the DeLP implementation. For each test sample we query the DeLP system to find all possible attackers (culprits) based on the arguments provided. If there is no way to decide between contradicting arguments, these are blocked and thus return no answers. Initially, the search space for each test sample is 19 teams.

Figure 10 shows the average search space after running the queries to return set of possible culprits. There is a significant reduction in search space across all target teams. The average search space is reduced from 19 teams to 5.8 teams (standard deviation of 1.92). To evaluate how much the reduced search space can aid an analyst in predicting the actual culprit, we compute one more metric—we check if the reduced search space has the ground truth (actual culprit) present in it. Figure 11 shows the fraction of test samples with ground truth present in it. For the majority of the target teams, we have ground truth present in more than 70% of the samples with reduced search space. For some teams like *more smoked leet chicken* (T-13) and *raon_ASRT (whois)* (T-17) the average reduced search space is as low as 1.82 and 2.9 teams, with high ground truth fraction of 0.69 and 0.63 respectively. The reason why there is a large search space for some teams is the presence of a larger number of unseen attacks. As discussed earlier, each unseen attack is replaced by the three most similar attacks from the training set, and hence the search space becomes larger.

We next perform predictive analysis on the reduced search space. The experimental setup is similar to the one described earlier; the only difference is that now instead of having a 19-team search space as in the previous case, the machine learning approach is allowed to make a prediction from the reduced search space only. We use random forest as the machine learning approach, which has shown to have the best performance for CTF data (Nunes et al. 2015). Table 4 gives a summary of the results.

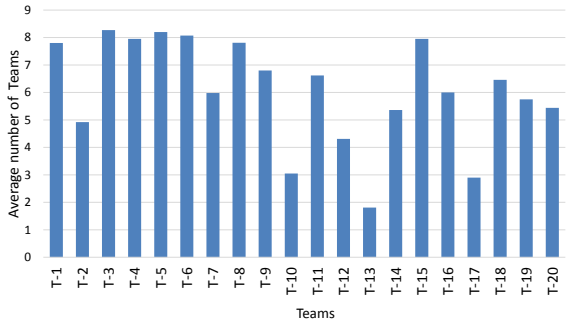

Figure 10: Average Search space after running the query to compute warranted arguments.

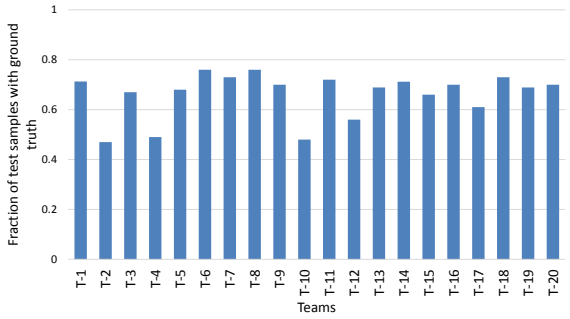

Figure 11: Fraction of test samples with the ground truth in the reduced search space.

We report three accuracies for the 20 teams. "Prev. Acc." represents the accuracies achieved after running Random forest without applying the argumentation based techniques, as reported in (Nunes et al. 2015). "Acc. (ground truth)" considers only the test samples where ground truth is present in the reduced search space. With this setting we are able to correctly predict on average more than 70% of the test samples, with T-4, T-13 and T-14 having accuracies over 80%. On the other hand, "Acc. (all)" also takes into account test samples where ground truth was not present in the reduced search space. It is clear that all those test samples will be misclassified since the ground truth is not present in the search space at all. Even accounting for those samples, the average accuracy is 0.5, which is significantly better than the average accuracy of 0.37 in (Nunes et al. 2015). The results are statistically significant ($t(20) = 6.83$, $p < 0.01$).

Conclusion

In this paper we showed how an argumentation-based framework (DeLP) can be leveraged to improve cyber-attribution decisions. We demonstrated this concept using the DeLP framework applied to CTF data, which reduced the set of potential culprits allowing for greater accuracy when using a classifier for cyber-attribution.

We are currently looking at implementing our probabilistic variant of DeLP (Shakarian et al. 2015b), which assigns probabilities to the arguments, helping in this way to decide which arguments prevail when contradictory information is

Table 4: Summary of Results

Team	Acc. (ground truth)	Acc. (all)	Prev. Acc.
T-1	0.69	**0.51**	0.45
T-2	0.72	**0.45**	0.22
T-3	0.55	**0.40**	0.30
T-4	0.81	**0.44**	0.26
T-5	0.68	**0.45**	0.26
T-6	0.62	0.49	**0.5**
T-7	0.70	**0.53**	0.45
T-8	0.75	**0.61**	0.42
T-9	0.68	**0.50**	0.41
T-10	0.77	**0.42**	0.30
T-11	0.62	**0.44**	0.37
T-12	0.76	**0.43**	0.24
T-13	0.85	**0.63**	0.35
T-14	0.71	**0.52**	0.42
T-15	0.57	**0.38**	0.30
T-16	0.68	**0.48**	0.43
T-17	0.80	**0.58**	0.42
T-18	0.68	**0.50**	0.48
T-19	0.70	**0.51**	0.41
T-20	0.74	**0.51**	0.48

present. We are also designing our own CTF event in order to obtain more realistic data.

Acknowledgments

Authors of this work were supported by the U.S. Department of the Navy, Office of Naval Research, grant N00014-15-1-2742 as well as the Arizona State University Global Security Initiative (GSI) and funds provided by CONICET and Universidad Nacional del Sur, Argentina. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Office of Naval Research.

References

Boebert, W. E. 2010. A survey of challenges in attribution. In *Proceedings of a workshop on Deterring CyberAttacks*, 41–54.

Dacier, M.; Pham, V.-H.; and Thonnard, O. 2009. The wombat attack attribution method: some results. In *Information Systems Security*. Springer. 19–37.

DEFCON. 2013. Defcon: Capture the flag.

Dung, P. M. 1995. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artificial intelligence* 77(2):321–357.

García, A. J., and Simari, G. R. 2004. Defeasible logic programming: An argumentative approach. *Theory and practice of logic programming* 4(1+ 2):95–138.

Lloyd, J. W. 2012. *Foundations of logic programming*. Springer Science & Business Media.

Nunes, E.; Kulkarni, N.; Shakarian, P.; Ruef, A.; and Little, J. 2015. Cyber-deception and attribution in capture-the-flag exercises. In *Proceedings of the 2015 International Symposium on Foundation of Open Source Intelligence and Security Informatics (FOSINT-SI)*.

Rahwan, I.; Simari, G. R.; and van Benthem, J. 2009. *Argumentation in artificial intelligence*, volume 47. Springer.

Rid, T., and Buchanan, B. 2015. Attributing cyber attacks. *Journal of Strategic Studies* 38(1-2):4–37.

Shakarian, P.; Simari, G. I.; Moores, G.; and Parsons, S. 2015a. Cyber attribution: An argumentation-based approach. In *Cyber Warfare*. Springer. 151–171.

Shakarian, P.; Simari, G. I.; Moores, G.; Paulo, D.; Parsons, S.; Falappa, M.; and Aleali, A. 2015b. Belief revision in structured probabilistic argumentation. *Annals of Mathematics and Artificial Intelligence* 1–43.

Shakarian, P.; Shakarian, J.; and Ruef, A. 2013. *Introduction to cyber-warfare: A multidisciplinary approach*. Newnes.

Stolzenburg, F.; García, A. J.; Chesnevar, C. I.; and Simari, G. R. 2003. Computing generalized specificity. *Journal of Applied Non-Classical Logics* 13(1):87–113.

Walls, R. J. 2014. *Inference-based Forensics for Extracting Information from Diverse Sources*. Ph.D. Dissertation, University of Massachusetts Amherst.