Cyber Reasoning with Argumentation: Abstracting From Incomplete and Contradictory Evidence

Andy Applebaum[1], Karl Levitt[1], Zimi Li[2], Simon Parsons[3], Jeff Rowe[1] and Elizabeth Sklar[3]
[1]Dept. of Computer Science, University of California Davis, CA 95616
[2]Dept. of Computer Science, City University of New York, NY 10016
[3]Dept. of Computer Science, University of Liverpool, Liverpool, UK
applebau@ucdavis.edu

***Abstract*—Information given to system administrators is often incomplete and contradictory. Even worse, administrators are required to adhere to organizational policies, which frequently contain conflicting goals. While prior work in security has sought to alleviate these concerns, much of it strives to identify attacks and intrusions with approaches that require complete knowledge for analysis. In this paper, we present a framework to addresses the challenges facing administrators by using formal argumentation to generate big-picture conclusions regarding the system. Unlike other schemes, argumentation excels in situations where information is incomplete and knowledge is contradictory. To motivate our approach, we detail a scenario inspired by real-world data taken from the U.C. Davis environment.**

I. Introduction

System administrators are overworked, inundated with alerts, logs and reports from network components. While thorough in some places, the information they are given can still be incomplete and is at times contradictory. Even when the reports they have are clear, administrators can still be at a loss for what to do — organizational policies are rarely consistent themselves and the consequences of a wrong choice can be serious. Responding to information when it is unreliable makes the task of administration challenging.

Automated solutions can ease some administrative tasks, bolstering the stability of a system by automatically responding to component statuses. From a security standpoint, the most common solutions are forms of intrusion detection systems (IDS) — interfacing directly with reports from system components, IDSs are able to identify and in some cases prevent intrusions. Outside of these tasks, however, the function of IDSs is limited, and the possibility of false positives and false negatives further requires the administrator to maintain a watchful eye. Fundamentally, IDSs lack *contextual awareness*, reasoning only about whether an action is an attack or not, never moving towards gaining an understanding of the "big picture" of the network. This disconnect is a significant gap, symptomatic of a hole that a well-trained attacker can exploit. In light of this, this paper describes a novel method that helps a system administrator build an understanding of the big picture.

As an example, consider the situation at U.C. Davis. The campus network features a single ingress/egress point managed by the campus information technology department — this gateway runs the TippingPoint intrusion prevention system (IPS), blocking attacks based on predefined signatures. All attacks found by the IPS are written into a log file, a sample of which can be found in Figure 1. While information rich, this log file is often left unobserved: as told by the U.C. Davis information technology staff, the logs are overly verbose, containing too many alerts to extract any significant meaning. Moreover, because the signatures used by the IPS are limited in scope and false positives are possible, the logs obtained paint an incomplete picture. The work we report here addresses these problems by creating a system that parses the logs and fills in the gaps using general information about the kinds of attacks represented in the logs.

Unlike other work on parsing logs, ours focuses on "big picture" ideas that can be useful to system administrators, guiding future actions as well as identifying fresh perspectives into otherwise sparse conclusions. We achieve this by incorporating *argumentation*, a logical framework designed to handle inconsistent, incomplete and contradictory evidence. Additionally, argumentation is well-suited to this problem due to the variable priorities inherent in the task of securely managing a system — in some situations an administrator may wish to create an air gap when faced with evidence of an attack, while in others the administrator may ignore the evidence due to a need for the system to remain online. Using argumentation, our end product is not only a new set of conclusions about the system, but also an understanding of the *underlying reasoning* behind these conclusions.

The rest of this paper is organized as follows: in Section II we provide a brief background on argumentation, in Section III we outline a high level of our proposed framework, in Section IV we examine a case study using our framework, in Section V we discuss our work towards an implementation of our architecture, in Section VI we discuss related work, and in Section VII we draw conclusions and outline future work.

II. Background: Argumentation

Historically, argumentation is rooted in the fields of philosophy, logic, debate and rhetoric: simply put, the goal of argumentation is to understand and explain the way that humans reason about everyday problems. This specific kind of reasoning is often done in the face of *incomplete* and

978-1-5090-0073-9/15/$31.00 ©2015 IEEE

ID	Time	Source	Target	Type	Count	Severity
328785516	2013-06-01 00:00:37	128.120.179.0	126.205.148.138	0290: Invalid TCP Traffic: Possible Recon Scan (SYN FIN)	1	2
328785520	2013-06-01 00:00:40	90.215.207.246	169.237.149.0	12607: Backdoor: Zero Access Trojan Communication Attempt	1	3
328795922	2013-06-01 00:49:43	67.166.158.11	169.237.6.0	0521: HTTP: webdist.cgi Command Execution	1	4
329975724	2013-06-04 12:14:55	128.120.60.0	67.17.157.22	0290: Invalid TCP Traffic: Possible Recon Scan (SYN FIN)	1	2
330627682	2013-06-06 17:15:33	128.120.106.0	78.88.62.234	12607: Backdoor: Zero Access Trojan Communication Attempt	1	3

Fig. 1: Sample Tipping Point alert entries. Internal IP addresses are anonymized at the last octet.

contradictory evidence, with the human actor making a decision based off of (potentially conflicting) internal priorities. Unlike traditional logics, where conclusions are proven true, argumentation concerns itself with the task of accepting a conclusion *tentatively*, with the possibility of rejecting it later depending on new knowledge.

More recently, argumentation has shifted from a rhetorical tool to a computational model for reasoning with uncertainty and conflicts. This shift is due in no small part to Phan Minh Dung's seminal work [1]: in it, Dung presented the concept of "argumentation frameworks," a simple model capable of representing other systems as well. The essence of Dung's system was to treat arguments as *atomic*, where an argumentation framework is defined based on a set of atomic arguments combined with a set of relations over those arguments signifying "attacks;" notationally, frameworks are written as $F = \langle A, R \rangle$ where A is the set of arguments and R the attack relation, with $i, j \in A$ and $(i, j) \in R$ signifying that argument i attacks the validity of argument j.

Argumentation frameworks inherit complexity through the notion of *acceptability semantics*, which are used to define what it means for an argument to be "reasonable." While many different types of semantics have been proposed, in this work we focus primarily on two:

- *Grounded* semantics, wherein an argument is considered acceptable if it is not attacked by any argument or if each attacker of it is in turn attacked by an acceptable argument.
- *Preferred* semantics, wherein an argument is considered acceptable if there exists a maximal subset of A containing that argument where none of the arguments in that set attack each other and for every argument in that set, if that argument is attacked, its attacker is attacked by at least one argument in the set.

Grounded semantics identify arguments that are *always* reasonable while preferred semantics identify arguments for which there exists some consistent line of reasoning that concludes the argument is acceptable. Figure 2 provides an example framework. Here, we have $A = \{a, b, c, d\}$ with $R = \{(a, b), (c, b), (c, d), (d, c)\}$. Under grounded semantics, the only acceptable argument is a (as it is unattacked), while under preferred there are two acceptable lines of reasoning, each corresponding to a preferred extension: accepting arguments a and c or accepting arguments a and d.

Many extensions of Dung's argumentation frameworks have been proposed; for this project, we chose to work with the ASPIC+ framework [7]. Unlike an abstract framework, ASPIC+ provides *structure* to arguments. Starting with some logical language $\mathcal{L}$, arguments are built from facts — predicates in $\mathcal{L}$ — by chaining rules towards a single conclusion. These

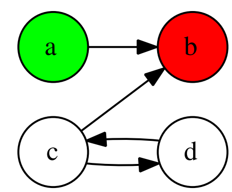

Fig. 2: An example argumentation framework: argument a is green, present in all preferred extensions, argument b is red, absent from all preferred extensions, and arguments c and d are white, each present in exactly one preferred extension.

rules can either be *strict*, as in traditional modus ponens, or *defeasible*, meaning that the consequent *reasonably follows* from the antecedent. Notationally:

$$\begin{aligned} p_1, p_2, ..., p_n &\rightarrow c_1 && \text{strict rules} \\ p_1, p_2, ..., p_n &\Rightarrow c_1 && \text{defeasible rules} \end{aligned}$$

In each case, c_1 denotes the conclusion of a rule and $p_1, ..., p_n$ is the conjunction of premises p_1 through p_n, jointly referred to as the body of the rule. Arguments are constructed through rule chaining, starting with a set of facts (rules that lack premises) and building on those facts with a set of rules to ultimately reach a conclusion. Informally, arguments attack each other when one negates the conclusion of another or when one negates the premise of a rule used by another, with the caveat that strict rules cannot be attacked on their conclusions. An argumentation framework is then instantiated via this construction.

III. Framework

A. Overview

The general workflow of our system is as follows:

1) Obtain reports and status updates from network sensors.
2) Abstract sensor reports into big-picture ideas.
3) Using argumentation, reason about big picture ideas.

Of these steps, our research focus is on the second and third: we assume that network sensors (e.g., system logs, intrusion detection/prevention systems, firewall reports, etc.) are present and trustworthy. Once obtained, reports are fed into a *processor*, which creates predicates to denote important high-level observations about the data. Examples can include:

- A spike in intrusion attempts was seen in the morning.
- Traffic slowly declined over the weekend.
- IP 75.75.75.75 was a frequent target for outgoing alerts.

Logically, we treat the predicates produced by the processor as indisputable facts. From there, the facts are fed into a

Name	Body		Conclusion	Rule	Attacks
A_1		$\rightarrow$	Ω_1		
A_2	Ω_1	$\Rightarrow$	$\textsc{WebServer}(X)$	r_4	A_6, A_4
A_3	$\textsc{WebServer}(X)$	$\Rightarrow$	$\textsc{AllowTraffic}(\texttt{any}, X)$	r_5	
A_4	Ω_1	$\Rightarrow$	$\textsc{CommandServer}(X)$	r_1	A_2, A_3
A_5	$\Omega_1 \wedge$ $\textsc{CommandServer}(X)$	$\rightarrow$	$\textsc{Botnet}(\mathbf{I})$	r_2	
A_6	$\textsc{CommandServer}(X) \wedge$ $\textsc{Botnet}(\mathbf{I})$	$\rightarrow$	$\textsc{BlockTraffic}(\mathbf{I}, X)$	r_3	A_3

Fig. 3: Example arguments built from Ω_1 and rules $r_1, ..., r_5$. Arguments constructed using TOAST [15].

knowledge base consisting of a set of rules encoding security-domain knowledge (what a botnet is, the consequences of a denial of service attack, what a port scan entails, etc.) as well as a set of rules premised on specific observational predicates. After combining the facts into the knowledge base, we construct arguments using the methodology of ASPIC+, ultimately producing an argumentation framework. An example argument is produced below.

- (*Observation:*) External IP x has triggered 30 different alert categories.
- (*Rule:*) Any IP that triggers numerous alerts categories is malicious.
- (*Rule:*) All malicious IPs should be blocked.
- (*Conclusion:*) x should be blocked.

After construction, the argumentation framework is presented to the administrator for review. In this stage, the administrator can view the recommended responses, preferred extensions, pivotal arguments, sources of further information, etc.

B. Reasoning Structure

1) Predicate Construction: Our goal is to provide a general framework that can work with any initial source of information. Nonetheless, as our project is motivated by the TippingPoint alert logs, we focus first on creating an interface between the IPS alerts and the initial fact set. We thus seek to create predicates as *summaries* of the logs, using the following format:

$$Obs(T_0, T_n, C, S, D, R, A)$$

where T_0 and T_n are the start and stop time of the summary, C is a representation of the alert count, S and D representations of source and destination hosts respectively, R the direction (`in` or `out`) and A a representation of the alert class. Put into words, the above predicate — when introduced as a fact — says that between times T_0 and T_n there were roughly C alerts of type(s) A from S to D. C, S, D and A can each take on a specific value (an exact count, a specific IP address, a specific alert, etc.) or a special keyword: `single`, signifying a fixed non-specific value, `few`, signifying a small generic set, `many`, signifying a large generic set, or `any`, signifying any.

As an example, we present the following fact:

$$\rightarrow Obs(\alpha, \beta, \texttt{many}, \texttt{many}, X, \texttt{out}, \textsc{Backdoor}) \qquad (\Omega_1)$$

This states that between times α and β, many outgoing alerts of type $\textsc{Backdoor}$ were triggered from multiple internal hosts to one specific external host X.

2) Rules: Extending to a Knowledge Base: The composition of the premises of a rule places it into one of three categories. In the first category, we have *builder* rules, where each premise is of the form $Obs(\texttt{any})$. As an example:

$$Obs(\alpha, \beta, \texttt{many}, \texttt{many}, X, \texttt{out}, \textsc{Backdoor}) \Rightarrow \textsc{CommandServer}(X) \qquad (r_1)$$

This rule has only one premise — Ω_1 — which defeasibly implies the conclusion that host X is a command server.

For the second category, we have *joiner* rules, where at least one premise is a literal from $\mathcal{L}$ and at least one premise is of the form $Obs(\texttt{any})$. As an example:

$$\textsc{CommandServer}(X) \wedge Obs(\alpha, \beta, \texttt{many}, \texttt{many}, X, \texttt{out}, \textsc{Backdoor}) \rightarrow \textsc{Botnet}(\textbf{local}) \qquad (r_2)$$

This states that if X is a command server and some large set of internal hosts has been triggering $\textsc{Backdoor}$ alerts to X, then we can strictly conclude that there is a local botnet.

The last category of rules is *extrapolatory*. Unlike builder or joiner rules, the premises of extrapolatory rules only consist of literals from $\mathcal{L}$; these rules are meant to extrapolate without using any direct observations. As an example:

$$\textsc{CommandServer}(X) \wedge \textsc{Botnet}(\textbf{local}) \rightarrow \textsc{BlockTraffic}(X) \qquad (r_3)$$

stating that if X is a command server and the local network has a botnet, then traffic from X should be blocked.

Moving towards a concrete argumentation framework, we now consider an alternative explanation to the observation Ω_1:

$$Obs(\alpha, \beta, \texttt{many}, \texttt{many}, X, \texttt{out}, \texttt{any}) \Rightarrow \textsc{WebServer}(X) \qquad (r_4)$$

where we have that Ω_1 satisfies the premise of r_4 (since the latter does not specify an alert type). In words, this rule says that if we see many internal hosts triggering alert warnings when contacting X, then we can conclude that X is some sort of web server, with the assumption being that multiple alerts are a normal part of functioning as a web server. Naturally, this produces a contrariness relation with $\textsc{WebServer}(X)$ and $\textsc{CommandServer}(X)$ mutually exclusive. Building on

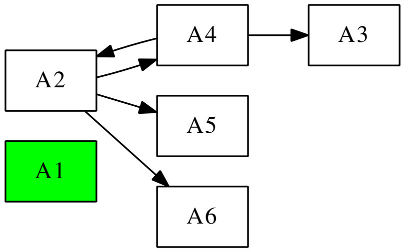

Fig. 4: Arguments from Figure 3 visualized.

this, we can also introduce $\textsc{AllowTraffic}(X)$ — mutually exclusive with $\textsc{BlockTraffic}(X)$ — signifying that traffic from X should be allowed:

$$\textsc{WebServer}(X) \quad \Rightarrow \quad \textsc{AllowTraffic}(X) \qquad (r_5)$$

These rules produce the set of arguments in Figure 3 with the framework itself visualized in Figure 4. Under grounded semantics, the only acceptable argument is A_1 — that Ω_1 is true — whereas under preferred semantics each argument would be acceptable (with extensions $\{A_1, A_2, A_3\}$ and $\{A_1, A_4, A_5, A_6\}$).

Reviewing the scenario, the administrator has a few different options: select a single preferred extension and act on it, prioritize the different rules to reach a single complete grounded extension, or look for more evidence regarding the status of the external host X. As a simple example of the second category, a preference can be encoded to prefer r_1 over r_4: since r_1 builds on a more specific observation than the one used in r_4, it can be viewed as a *stronger* rule than the latter. Using this preference, the single grounded extension $\{A_1, A_4, A_5, A_6\}$ would emerge, with the conclusion to block traffic from X.

IV. Case Study: An Authorized Penetration Test

A. Scenario

In this section we illustrate a scenario, taken from real world data, that we can apply our framework to. We begin with the following predicate:

$$\rightarrow \mathit{Obs}(\alpha, \beta, \texttt{any}, X, Y, \texttt{in}, \texttt{many}) \qquad (\omega_1)$$

Put into words, ω_1 states that between times α and β, external host X sent many alerts types to internal host Y. This leads to a natural conclusion: X is sending all these alerts to Y in order to break into Y, and therefore X is some sort of malicious attacker. Incorporating this into the knowledge base yields the following rules:

$$\begin{aligned}\mathit{Obs}(\alpha, \beta, \texttt{any}, X, Y, \texttt{in}, \texttt{many}) \quad \Rightarrow \\ \textsc{Malicious}(X)\end{aligned} \qquad (\tau_1)$$

$$\textsc{Malicious}(X) \quad \rightarrow \quad \textsc{BlockTraffic}(X) \qquad (\tau_2)$$

signifying that *all* traffic from X should be blocked, just in case X triggers a vulnerability unknown to the IPS.

Applying this signature to the TippingPoint logs generated a list of potentially malicious attackers (found in Figure 5).

Source	Count	Alerts	Targets
24.7.158.10	309	68	1
54.235.163.229	995	53	1
54.215.13.26	986	53	1
98.255.224.214	621	44	1
67.166.158.11	301	44	1
198.96.129.164	576	44	1
67.207.202.9	5882	41	1
64.37.231.131	205	14	3
123.151.39.34	1675	12	21

Fig. 5: External attackers triggering at least 12 different alerts, with at least 100 alerts total.

Looking at each IP individually, we made a surprising discovery: Googling the address 54.235.163.229 returns a page (https://scanmyserver.com/faq.html) associating the IP address with the ScanMyServer service, intended to perform an authorized penetration test on a given target. In fact, the aforementioned page even goes so far as to request users of the service to white-list their IP so that all vulnerability tests can be performed. However, since these scans were blocked by the UCD IPS, the result of the test is inaccurate (as the test failed at the gateway as opposed to the server) and it is possible that a local server contains vulnerabilities unknown to the operator.

With this in mind, we create an alternative explanation and action for ω_1:

$$\begin{aligned}\mathit{Obs}(\alpha, \beta, \texttt{any}, X, Y, \texttt{in}, \texttt{many}) \quad \Rightarrow \\ \textsc{PenTester}(X)\end{aligned} \qquad (\tau_3)$$

$$\textsc{PenTester}(X) \quad \rightarrow \quad \textsc{AllowTraffic}(X) \qquad (\tau_4)$$

with the predicates $\textsc{Malicious}$ and $\textsc{PenTester}$ mutually exclusive. Using this new knowledge base, any IP that satisfies ω_1 will always create an argumentation framework with two preferred extensions — one with $\textsc{Malicious}(X)$ and $\textsc{BlockTraffic}(X)$ and the other with $\textsc{PenTester}(X)$ and $\textsc{AllowTraffic}(X)$ – but no responsive recommendation. In the remainder of this section, we present two techniques that can be used to extend this scenario towards a clear resolution.

B. Intermediate Conclusions

Both predicates $\textsc{Malicious}(X)$ and $\textsc{PenTester}(X)$ describe the same type of classification, differing only in the intention of X — in either case, since X is sending many alerts, it is likely that X will continue to send alerts, and thus we would want to conclude that X should be monitored in order to identify new potential signatures. Additionally, because we know there will not be a clear resolution for blocking or allowing X, we would want to also conclude that X should be looked up to determine if it is an authorized attacker or not. This leads us to add a *middle-point* between ω_1 and the conclusions $\textsc{Malicious}(X)$ and $\textsc{PenTester}(X)$, updating the rule set as follows:

$$\begin{aligned}\mathit{Obs}(\alpha, \beta, \texttt{any}, X, Y, \texttt{in}, \texttt{many}) \quad \Rightarrow \\ \textsc{Attacker}(X)\end{aligned} \qquad (\tau_0)$$

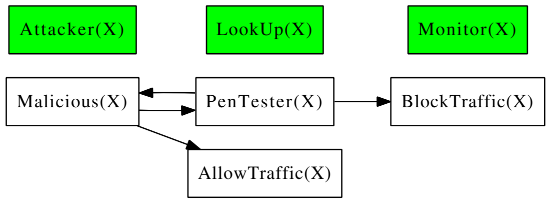

Fig. 6: Arguments from Section IV-B visualized, with each argument labeled by its conclusion.

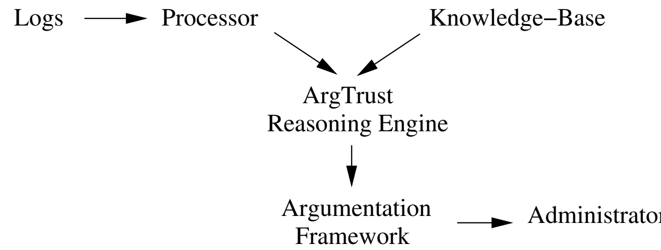

Fig. 7: Implementation workflow.

with

$$\text{ATTACKER}(X) \Rightarrow \text{MALICIOUS}(X) \quad (\tau_1)$$
$$\text{ATTACKER}(X) \Rightarrow \text{PENTESTER}(X) \quad (\tau_3)$$

and added conclusions:

$$\text{ATTACKER}(X) \Rightarrow \text{MONITOR}(X) \quad (\tau_4)$$
$$\text{ATTACKER}(X) \Rightarrow \text{LOOKUP}(X) \quad (\tau_5)$$

Adding these new rules and seeding with ω_1 yields a framework (visualized in Figure 6) with a grounded extension containing ATTACKER(X), LOOKUP(X) and MONITOR(X). As in the initial situation, there will be two preferred extensions (one containing MALICIOUS(X) and BLOCKTRAFFIC(X) and the other containing PENTESTER(X) and ALLOWTRAFFIC(X)) with no clear resolution on allowing or blocking X. However, because this new framework introduces intermediate steps, the administrator will always be able to respond to the initial observation, either by monitoring and learning from X or looking up X and determining if the traffic is part of an authorized penetration test.

C. Adding New Observations

Consider the IP address 123.151.39.34 in Figure 5. Although it satisfies the initial predicate ω_1, its patterns are markedly different than those of the ScanMyServer service — whereas the latter sent alerts to one specific target, the former sent alerts to 21 different targets. Due to the large number of targets contacted by this IP, it seems quite unlikely that it is an authorized penetration tester. To encode this reasoning, we create a new observation:

$$\rightarrow \mathit{Obs}(\alpha, \beta, \texttt{any}, X, \texttt{many}, \texttt{in}, \texttt{any}) \quad (\omega_2)$$

along with a new rule and predicate:

$$\mathit{Obs}(\alpha, \beta, \texttt{any}, X, \texttt{many}, \texttt{in}, \texttt{any}) \Rightarrow \text{SCANNER}(X) \quad (\tau_6)$$

This new rule states that any external host that is sending alerts to numerous internal hosts should be labeled a SCANNER; hosts exhibiting this behavior are likely opportunistic and are attempting to find vulnerabilities in any part of the network. By themselves, scanners are often benign — malicious, perhaps, but unlikely to cause significant damage. However, a scanner that is *also* a known attacker poses a significant risk, and, no longer benign, should be strictly categorized as malicious:

$$\text{SCANNER}(X) \wedge \text{ATTACKER}(X) \rightarrow \text{MALICIOUS}(X) \quad (\tau_7)$$

Introducing this rule yields a new argument for MALICIOUS(X): provided that X is a scanner and attacker, then we can *strictly* conclude that X is malicious. Accordingly, the argument for PENTESTER(X) – which we constructed defeasibly – no longer attacks the conclusion MALICIOUS(X), yielding a single preferred extension for 123.151.39.34 containing SCANNER, ATTACKER, MALICIOUS, MONITOR, LOOKUP and BLOCKTRAFFIC.

V. ARCHITECTURE AND IMPLEMENTATION

The scenarios and examples contained in this paper, while initialized from real data, were created manually. An implementation is currently under development. Figure 7 provides a rough overview of operation: logs obtained from the TippingPoint IPS are first parsed into *Obs* predicates and then combined with a user-defined knowledge base. From there, the predicates and rules are fed to the argumentation engine ArgTrust [14], which outputs a graph containing the acceptability semantics for the framework.

Once complete, the final version of our implementation will be extensible, taking input from multiple log files, customizable, allowing users to define cutoffs for keywords `few` and `many` as well as allowing users to add and remove rules from the knowledge base, and interactive, providing an interface for the administrator to use to view key parts of the argumentation framework (e.g., customizable acceptability semantics, potential responses, consequences, etc.). Additionally, the final version will provide an automated format where the administrator can specify rules preferences and evidence metrics to resolve conflicts and arrive at a single grounded extension.

VI. RELATED WORK

Alert correlation for intrusion detection, similarly motivated to reduce the number of alerts facing administrators, is a well established research area where reports from multiple network sensors/IDSs are combined to identify larger and more complex attacks. A survey of techniques for alert correlation in [2] categorize techniques into five main types, two of which are of particular interest to us: those that correlate alerts based on predefined scenarios and those that correlate alerts by analyzing prerequisites and consequences of alerts. Approaches that fall under these categories are built upon logical models; an example logic can be found in [8], which gives a comprehensive first-order logical model to represent the security of a system.

Attack graphs [13] are similarly themed to alert correlation, with the focus primarily on chaining vulnerabilities as opposed to alerts. The attack graph approach has been featured heavily within the area of cyber situational awareness [4]; two noteworthy use cases are Cauldron [5], which automatically exposes vulnerability paths, uses advanced visualization techniques and provides recommended responses, and [6], which considers ways that uncertainty can be modeled within an attack graph. While similar in motivation, our work lies outside the typical focus of cyber situational awareness as our work is centered around awareness from status-based alerts as opposed to vulnerability analysis. Nonetheless, we do consider our work to fall under the general heading of cyber situational awareness, and hope to reposition our work within this field.

Perhaps most similar to our work are [16] and [9], where the authors' goal is to combat *uncertainty* in intrusion detection. Both works use logical frameworks that start with observations and chain inference rules towards a conclusion, although they differ in the way they handle uncertainty: in the former, uncertainty is quantified by assigning a numerical value (obtained by using Dempster-Shafer theory) to inference rules, while in the latter uncertainty is qualitative, with each rule is assigned one of three labels (possible, likely or certain). This second approach is similar to the strict/defeasible rule dichotomy used in this paper, although our system's use of argumentation allows for the presence of conflicts in the knowledge base.

Cyber security as a whole is a fairly new venue for argumentation. One early focus is using argumentation to prove and diagram security properties [3], which was reimagined by the authors of [10] by using ASPIC+ and considering the logical formalisms of argumentation as a game. Argumentation for identification and analysis of attacks was considered in our prior work [11]. A similar problem was examined in [12], which uses argumentation to tackle the problem of attribution in cyber warfare. To the best of our knowledge, our work is the first attempt to use rule-based argumentation for alert correlation and intrusion analysis.

VII. Discussion and Future Work

In this paper we presented a framework for cyber reasoning in the face of incomplete and conflicting evidence, detailed further by a scenario grounded in real data. Our approach falls outside the traditional security domain, applying formal argumentation to generate a general view of a network as opposed to focusing only on network attacks and intrusions. Initial reception to our approach has been positive — the U.C. Davis staff who deal with network security were intrigued by scenarios that we identified and we hope to work with them further to introduce new features to our framework.

As mentioned in Section V, an implementation is currently in progress. Future work includes incorporation of alternative sources of information, new patterns and signatures (e.g., spikes in traffic, network events, external circumstances, etc.) and an expanded taxonomy of alerts. Additionally, we hope to create a heavily expanded knowledge base which can reason about complex attacks, ideally using some form of learning to determine new attack patterns and signatures. As a last step, to validate our approach, we plan to perform user studies to identify ways in which our framework can interface with an administrator.

Acknowledgements: Research was partially funded by the National Science Foundation, under grants CNS #1117761 and #1118077, and Army Research Laboratory CTA Number W911NF-09-2-0053.

References

[1] P. M. Dung. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artificial Intelligence*, 77:321 – 357, 1995.

[2] H. T. Elshoush and I. M. Osman. Alert correlation in collaborative intelligent intrusion detection systems — A survey. *Applied Soft Computing*, 11(7):4349–4365, 2011.

[3] V.N.L. Franqueira, T. T. Tun, Y. Yu, R. Wieringa, and B. Nuseibeh. Risk and argument: A risk-based argumentation method for practical security. In *Proceedings of the 19th IEEE International Requirements Engineering Conference*, 2011.

[4] S. Jajodia, P. Liu, V. Swarup, and C. Wang. *Cyber situational awareness*, volume 14. Springer, 2010.

[5] S. Jajodia, S. Noel, P. Kalapa, M. Albanese, and J. Williams. Cauldron mission-centric cyber situational awareness with defense in depth. In *MILCOM*, Nov 2011.

[6] J. Li, X. Ou, and R. Rajagopalan. Uncertainty and risk management in cyber situational awareness. In *Cyber Situational Awareness*, pages 51–68. Springer US, 2010.

[7] S. Modgil and H. Prakken. The ASPIC+ framework for structured argumentation: a tutorial. *Argument & Computation*, 5(1):31–62, 2014.

[8] B. Morin, L. Mé, H. Debar, and M. Ducassé. A logic-based model to support alert correlation in intrusion detection. *Information Fusion*, 10(4):285–299, 2009.

[9] X. Ou, S.R. Rajagopalan, and S. Sakthivelmurugan. An empirical approach to modeling uncertainty in intrusion analysis. In *Proceedings of the Annual Computer Security Applications Conference*, 2009.

[10] H. Prakken, D. Ionita, and R. Wieringa. Risk assessment as an argumentation game. In *Computational Logic in Multi-Agent Systems*, pages 357–373. Springer, 2013.

[11] J. Rowe, K. Levitt, S. Parsons, E. I. Sklar, A. Applebaum, and S. Jalal. Argumentation logic to assist in security administration. In *Proceedings of the 2012 Workshop on New Security Paradigms*, 2012.

[12] P. Shakarian, G. I. Simari, G. Moores, S. Parsons, and M. A. Falappa. An argumentation-based framework to address the attribution problem in cyber-warfare. *CoRR*, abs/1404.6699, 2014.

[13] O. Sheyner, J. Haines, S. Jha, R. Lippmann, and J.M. Wing. Automated generation and analysis of attack graphs. In *Proceedings of the IEEE Symposium on Security and Privacy*, 2002.

[14] E. I. Sklar, S. Parsons, Z. Li, J. Salvit, S. Perumal, H. Wall, and J. Mangels. Evaluation of a trust-modulated argumentation-based interactive decision-making tool. *Autonomous Agents and Multi-Agent Systems*, pages 1–38, 2015.

[15] M. Snaith and C. Reed. TOAST: online ASPIC+ implementation. In *Proceedings of the Fourth International Conference on Computational Models of Argument*, 2012.

[16] L. Zomlot, S. C. Sundaramurthy, K. Luo, X. Ou, and S. R. Rajagopalan. Prioritizing intrusion analysis using Dempster-Shafer theory. In *Proceedings of the 4th ACM Workshop on Security and Artificial Intelligence*, 2011.