2015 Annual Conference on Advances in Cognitive Systems: Workshop on Goal Reasoning

Reasoning about Attack Goals for Cyber Resilience

Ugur Kuter UKUTER@SIFT.NET
Mark Burstein MBURSTEIN@SIFT.NET
Robert P. Goldman RPGOLDMAN@SIFT.NET
SIFT, LLC, 319 N. First Ave., Minneapolis, MN 55401 USA

Abstract

This paper describes our approach to anticipating and recognizing potential cyber threats in order to provide timely responses to those threats. Our approach anticipates attacks on distributed systems by generating a diverse set of attack plans on key system components and then determining the probabilities that these attacks may threaten those components or others that are stepping stones to those systems. The results enable our overall system, STRATUS, to defend these systems by preparing backups and controlling communications pathways appropriately. We present a preliminary empirical study of our techniques, demonstrating their promise.

1. Introduction

Strategic resilience in an adversarial environment requires not just the ability to *respond* in a timely fashion to detected problems, but also an ability to *predict* potential threats. A system for strategic resilience must identify potential threats to key mission objectives before they happen, use that analysis to recognize threats as they unfold, and pre-plan so that it can respond quickly when these threats actually occur. Current IDSs do not predict attacks; they do not provide early warning. They report the type and properties of an attack after (sometimes long after) it has happened. While recognition of attacks is an important ability, it falls short of the community's vision for security systems that predict future hacker actions and automatically and correctly respond to attacks in a timely manner.

This paper reports on our work on reasoning a priori about attacker actions and plans for their otherwise dynamic and unexpected effects on a cyber mission and its goals during the mission execution (Vattam et al., 2013; Klenk, Molineaux, & Aha, 2013). In particular, we describe our approach to probabilistic plan recognition using diverse planning to generate a graphical model of hypothetical attack goal/surfaces for a cyber mission and probabilistic plan recognition that uses this diverse graphical model. Our work resides within the context of a larger, multi-component cognitive architecture called STRATUS (Strategic and Tactical Resiliency Against Threats to Ubiquitous Systems) (Burstein et al., 2012) that we are developing to provide resilience for cyber missions in large computer networks. STRATUS's goal is to use modest added overhead in computational resources to diagnose an attack, switch rapidly to computed backup contingencies, and predict downstream events by attackers with enough robustness to make mission critical functions resilient to those attacks. STRATUS uses models of missions, software and network elements and attack types to develop expectations about how observed problems might indicate attacks, and how attacks might

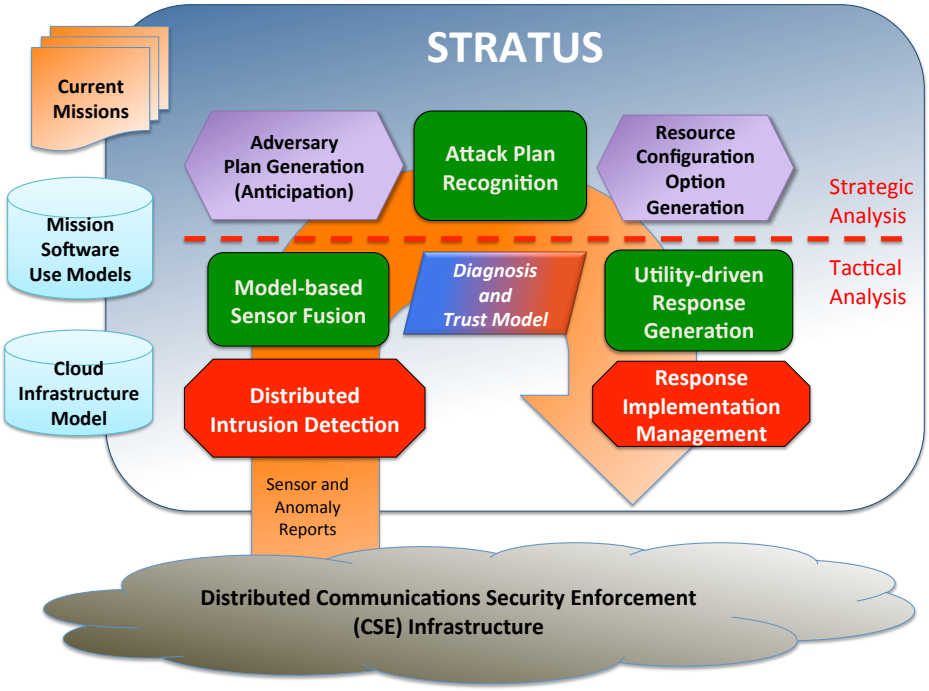

Figure 1. Abstract view of STRATUS information flow architecture.

proceed to adversely impact some mission(s). An abstract functional overall architecture is shown in Figure 1.

Because of the myriad ways an attacker can get into a computer network, and the limited resources available to respond, effective cyber response requires recognizing the attacker's potential goals, and focusing responses on protecting those assets most critical to the defender's own goals. When we are resource-limited, network assets that serve only low-priority goals can be sacrificed to focus on the essentials. Combining information about targets affected by particular attacks and information about the importance of particular resources to a network's mission, we can focus the system's attention on those attacks that would have the greatest impact, and then look specifically for indicators of those attacks. By identifying the critical attacks and their goals, we can pre-plan to ensure we are prepared to respond appropriately.

We are developing two sister systems, RAPPA (Rapid Attack Plan Path Analysis) and SPAR (Simple Probabilistic Attack Recognizer). RAPPA uses a combination of planning and learning techniques to identify likely attack goals and plans, into models needed for attack plan recognition and countermeasure planning. Using the attack plan libraries generated by RAPPA as a model of attacker behavior, SPAR quickly estimates the conditional probability of potential attacks on different mission components, given (noisy) observations of attacker actions. SPAR can explain the reasoning behind its probabilistic goal recognition, using regression techniques over probabilistic

attack paths. We present a preliminary evaluation of RAPPA and SPAR, demonstrating the promise of the attack goal and behavior reasoning and recognition capabilities.

2. Motivating Scenarios

We have been developing a family of scenarios, based on a simplified model of the computational aspects of the planning and execution of missions on large computer networks. Figure 2 shows a set of cloud clusters (LANs), and mission software components for a simple demonstration network. The upper half of the diagram contains a set of cloud clusters for handling imagery and mission-related databases and computational resources, while the lower half shows the human user client systems that access and manage those resources during the mission.

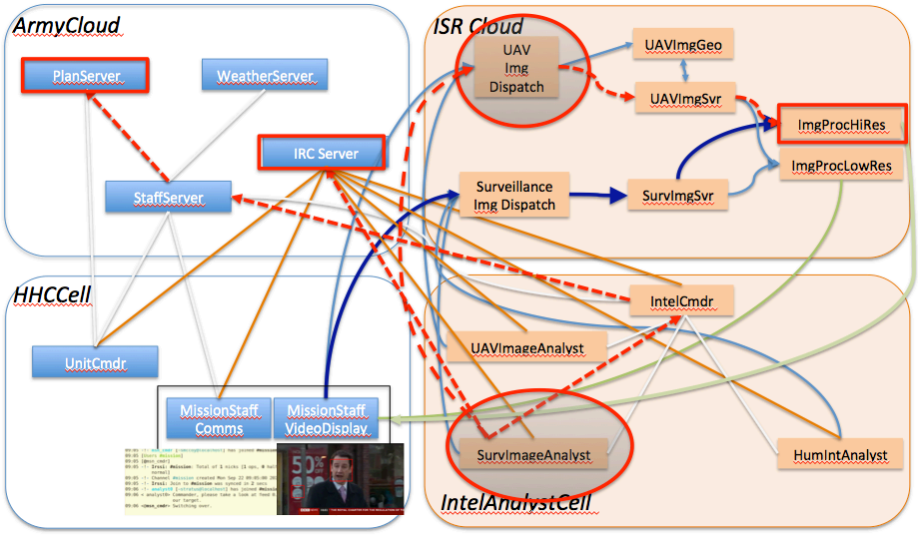

Figure 2. Mission resources and configuration in a computer network. The objective is to recognize distributed attack goals and plans. Circled components denote nodes that have been observed to be compromised. Red-bordered components are the potential goals of attacks. Paths of potential attack are dashed red. Other colored links denote normal mission processing.

In one example of an attack goal against the mission in this scenario, an attacker intends to gain access to the PlanServer database to exfiltrate information. In another, a virus is introduced from the web that 'powns' some of the software resources used and generates an internal DDoS attack on one of the critical Image Processing servers.

A planning problem for RAPPA is a initial situation assessment of the computer network, mission specifications, as well as potential known attacker capabilities, actions, and resources. In this approach, we follow Shrobe (Shrobe, 2002a; Shrobe, 2002b) in driving attack plan generation from models of the defended system, its missions, and known exploits. As an example, such attacker models based on the models of the defended system in Figure 2 would include compromising a component in the network by exploiting a combination of known properties of the user machines,

that are exposed to the outside cyber world, and network internal servers known to be heavily-used for computation and which, therefore, are potential targets for a denial of service attacks.

Using such models, RAPPA generates possible attack goals in the network by exploring the potential attack surface using AI planning techniques. For example, an attack path plan (marked as blue) in our scenario starts from the VideoDisplay user machine, and transmits malware over the network to the Image Hi-Resolution Processing Server to corrupt it.

RAPPA not only generates such attack goals but further categorizes and conceptualizes them into abstract classes. As we describe in the subsequent sections, we use both Hierarchical Task Network (HTN) learning and concept learning algorithms to achieve this capability. Figure 3 shows an illustration of such conceptualizations in terms of HTN methods.

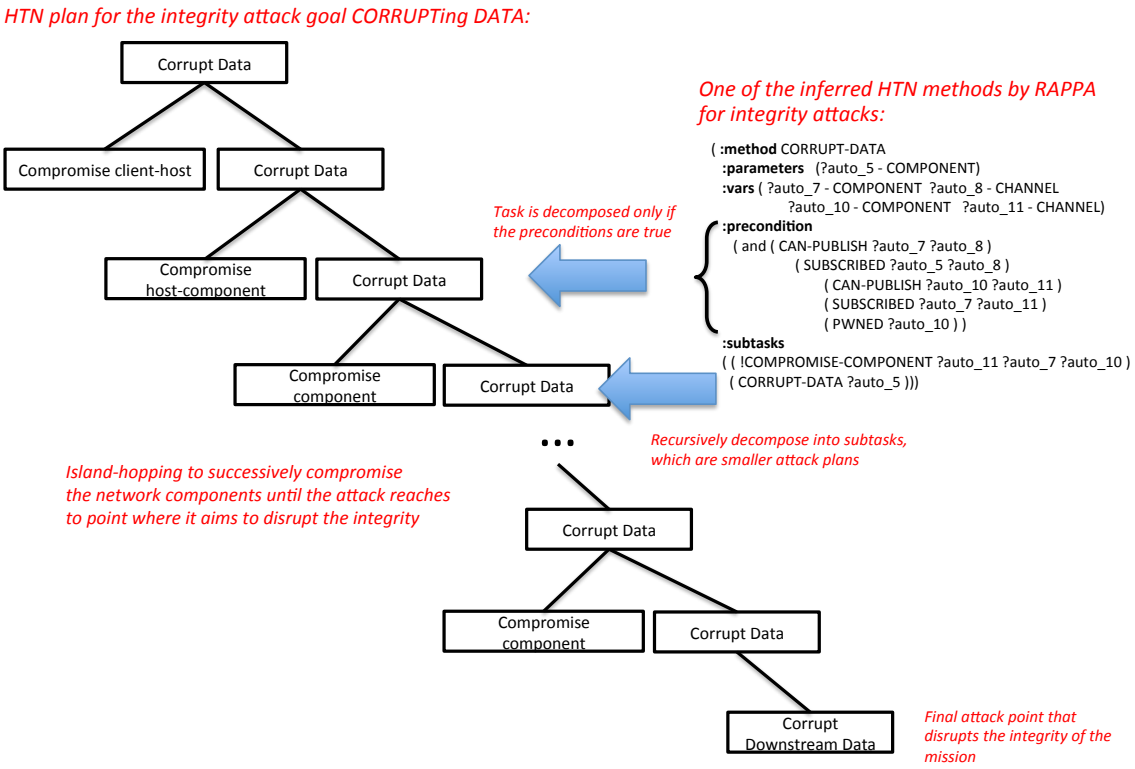

Figure 3. An illustration of an HTN plan library and HTN plan inferred by RAPPA on an Integrity (corrupting data) objective.

The HTN method shown on the right in the figure is one of the methods RAPPA produces from attack plans for the goal of corrupting a component in the network. On the left is an illustration of how to unfold the HTN plan into a hierarchical representation. This represents the CORRUPT-DATA task as consisting of two subtasks: (1) compromising the client host to enter to the system and (2) recursively attempting to accomplish the CORRUPT-DATA task from there. Successive decompositions of the CORRUPT-DATA task enable island-hopping through the network, compromising other

components so as to reach to target. Note that such sequences of attack actions are achieved by recursive definition of the HTN method on the righthand side of the figure.

Once a set of possible attack goals are generated by considering subtask priority in a mission specification, further reasoning estimates the likelihood of these events as we observe some initial set of attack events occuring. We assume that observed events correspond to the abstract attack actions that RAPPA uses to reason about threats. For example, the COMPROMISE-COMPONENT attack action shown in the hierarchy of Figure 3 is also an event that we can indirectly observe in the network by fusing evidence from sensor reports. STRATUS' MIFD component does exactly that.

As observations are obtained from the sensors in the network, and fused by MIFD into hypotheses about the attack events that may have occurred, SPAR uses RAPPA's attack plan and goal libraries to identify which components in the network are threatened, and estimates the likelihood of possible attacks on those components. STRATUS uses that information to help decide where additional backups or other defense maesures might be needed. For example, in Figure 2, if it is observed that Surveillance-Img-Analyst machine is compromised (shown as circled at the lower right portion of the network), then among the many alternative possible attack goals that SPAR considers, three can are deemed more likely than others, namely IRCServer, PlanServer, UAV Image Hi-Resolution Processing server. The rationale for this is that IRCServer is a central component in the network and the other two directly affect how the top-level mission how mission objectives the mission are achieved. These components are given back-ups so that they can be quickly restored if one of those eventualities occurs.

The subsequent sections provide the technical details on RAPPA and SPAR.

3. Adversarial Planning for Reasoning about Attack Goals

When a plan is to be executed in a cyber environment, it has be to protected against potential threats, which may disrupt or prevent the execution of a mission's goals. We developed techniques for recognizing critical attacks and their goals, which typically cause such disruptions, by combining diverse planning, hierarchical learning, and conditional threat reasoning: RAPPA (Rapid Attack Plan Path Analyzer) develops a diverse set of attack plans to prime the plan recognition component, SPAR (Simple Probabilistic Attack Recognizer) (see Section 4), and simultaneously identifies a set of mission-specific attack targets to inform the decision model used to allocate resources to the mission.

3.1 Attack Scenario Modeling in PDDL

We can characterize attacks abstractly as attempts to violate three primary security goals. In the following list, we give the security goal, and the countervailing attack goal/task:

- (C) Confidentiality / exfiltration;
- (I) Integrity / corruption; and
- (A)Availability / denial of service.

We will refer to these as the CI&A goals. Some additional security goals that are sometimes formulated include accountability, authenticity, non-repudiation, and reliability. These may be thought of as partial decompositions of the top three.

Vulnerabilities translate to actions available to the attacker. We use PDDL – Planning Domain Description Language – models for such attack actions. Consider an action example, from our motivating scenarios discussed in the previous section:

```
(:action compromise-component
   :parameters (?ch - channel ?task - task
                ?target-comp ?source-comp - component
                ?target-host - host)
   :precondition
     (and (runs-on ?target-comp ?target-host)
          (runs-on ?source-comp ?source-host)
          (component-task ?target-comp ?task)
          (can-publish ?source-comp ?ch)
          (subscribed ?target-comp ?ch)
          (pwned ?source-comp)
          ;; additional items
          (machine-os ?target-host WINDOWS))
   :effect (and (pwned ?target-comp)))
```

Through its effect and precondition expressions, this action model describes that an attacker that controls (pwned) a component, source-comp, can gain control of a target component target-comp if source-comp can publish to a channel, ch, to which target-comp subscribes, as long as the target-comp is running on Windows. This action allows an attacker to establish control over a component, recorded as (pwned component). The establishment of control allows the attacker to do "island-hopping," establishing more and more control until she can reach her real target(s). The real targets, recall, are to compromise one of the CI&A security goals. We have developed PDDL models of different attacker actions for all of the CI&A goal categories as above. These models are used in our planning systems that generate *diverse* plans, described below.

3.2 Diverse Plan Generation

Given a mission model and a network specification, RAPPA treats every possible condition that the mission's execution might depend on as a potential attack goal. Thus, RAPPA we initially considers a disjunctive goal, in which each disjunct simply is the negation of a mission condition. SPAR, later on, generates a conditional probability ranking over this disjunction, which can be used to focus on only a small subset of the disjuncts in the original goal expression, eliminating those attack goals that are not likely.

Given a model of possible attack actions, RAPPA's objective is to generate attack plans that are *interestingly-different* than each other so as to provide the recognizer with as broad as possible a set of attack plans it might recognize. This set should ideally *cover* the possible attack space to be able to reason, plan, learn, and recognize the widest set of attacks.

We developed a set of new algorithms for diverse planning, each of which translates the diversity metric into a cost measure for planning alternatives and describes how to update iteratively the costs of the actions in the domain model, in order to generate diverse plans. This approach allows *any* classical planner to generate diverse plans, without modifying the planner, as long as the classical planner can reason with simple numeric costs on actions. For this, we integrated LPG-d (Srivastava et al., 2007) for diverse planning in RAPPA. LPG-d also serves as a baseline diverse planner for our ongoing experiments .

For measuring diversity, we have started with existing plan-plan distance measures developed previously:

- **Action Counting.** This is the most common diversity measure used by the existing works mostly because it is the most easy one to compute as a search control mechanism. Basically, one uses the length of the plan in number of ground actions as the similarity measure (Nguyen et al., 2012).

- **Edit Distance.** The *edit distance (a.k.a., Levenshtein Distance)* between two strings is defined as the minimum number of edits needed to transform one string into the other, with the allowable edit operations being insertion, deletion, or substitution of a single character (`http://en.wikipedia.org/wiki/Levenshtein_distance`).

We demonstrated that these diversity measures can provide inconsistent results on the same plans. Based on this observation, we developed a plan-plan distance metric based on Kolmogorov (Algorithmic) complexity, where we define the diversity of plans in terms of how surprising one plan is given another or, its inverse, the conditional information in one plan given another (Goldman & Kuter, 2015). Kolmogorov complexity provides a domain independent theory of conditional information. While Kolmogorov complexity is not computable, a related metric, Normalized Compression Distance (NCD), provides a well-behaved approximation. Thus, we introduced NCD as an alternative diversity metric, and analyzed its performance empirically, in comparison with previous diversity measures, showing strengths and weaknesses of each. We also examined the use of different compressors in NCD.

In (Goldman & Kuter, 2015), we also demonstrated how NCD can be used to select a training set for HTN learning, giving an example of the utility of diversity metrics. The details of this work can be found in (Goldman & Kuter, 2015); however, this result forms the foundation of our approach in RAPPA: by generating diverse sets of attack plans and goals, and categorizing them into hierarhical representations, RAPPA is able to provide a reasonable coverage to the potential vulnerabilities and attack events in a computer network.

3.3 Inferring Plan Libraries for Recognition

RAPPA uses a combination of techniques to translate attack plans it generates into the models needed for attack plan recognition and countermeasure planning, building on our previous work on HTN-Maker (Hogg, Muñoz-Avila, & Kuter, 2008; Hogg, Kuter, & Muñoz-Avila, 2009; Hogg, Kuter, & Muñoz-Avila, 2010), for generalizing linear plans into hierarchical task networks. HTN-Maker generalizes from single examples of attack plans by recognizing partial orders: places where

actions in the plan can be executed in multiple different orders. It will also recognize the underlying goals that explain why actions have been incorporated in the attack plans.

To consider multiple concurrent and interleaved attacks, we developed PDDL models that can support this kind of planning and showed the ability of RAPPA to generate diverse plans with those models. These test domains revealed a crucial shortcoming in the RAPPA architecture with respect to the plan library generation. In particular, although HTN-Maker was able to generate plan libraries as shown in Figure 3 for simple attack formulations, the algorithm was limited to learning strictly less expressive grammars than needed for recognition in our scenario domains.

To alleviate this shortcoming, we developed a concept learning algorithm, called Anchored Hierarchical Learner (AHL), for RAPPA that produces attack categories for recognition. In particular, given attack plans, AHL first attempts to identfy possible plan anchors in those plans, by extracting the causal structure of an attack plan given the PDDL models of the attack actions and the particular attack goals the plan accomplishes. It then generates the causal structure for every attack plan in its input diverse plan set. Then, the algorithm walks over these causal structures and compares them to identify those actions, called *anchors* (Geib, 2009; Geib & Goldman, 2011), that are highly distinctive in those plans.

Once the anchors are identified, AHL starts from the plan anchor actions in the attack plan, and infers a plan library around those anchors successively. Given a plan anchor, the algorithm first learns its parent and possible siblings. The anchor action and siblings constitute a portion of the plan from which the system learns. The learner then replaces this portion with the parent learned for the anchor action, creating an abstract plan, and marks that parent as the new anchor in that abstracted plan. The learning process continues in this manner until no new parents can be learned (or if given, the root nonprimitive task is reached).

We are currently testing this implementation and investigating ways to inductively generalize the hierarchical structures learned to produce more efficient plan libraries for recognition.

4. SPAR: Simple Probabilistic Attack Recognizer

SPAR is a new probabilistic recognition algorithm we developed in order to probabilistically recognize the potential goals of an attack as it progresses. SPAR attempts to quickly but directly estimate the conditional probability of attacks on all mission components, given the observations seen so far.

Algorithm 1 shows a high-level description of SPAR. The input includes a plan library Π of possible attacks (generated by RAPPA). D is the planning domain which describes the set of planning operators. G is the set of *probabilistic goals*, i.e., a partial mapping from possible attack goals in the network to probability values. O is the set of *probabilistic observations*, i.e., a partial mapping from possible intrusion event hypotheses to probability values.

Some explanations about G and O are in order. As in attacks in the physical world, some components in a cyber network are higher value targets compared to others. SPAR can take as input such information as a priori beliefs on how likely it is that a network component will be attacked. The input G models these beliefs as probability values and the algorithm incorporates this probability into its conditional estimation for that component as a possible attack goal. Note that the probabilistic belief over goals does not necessarily specify a probability distribution over

Algorithm 1: The SPAR algorithm.

Procedure SPAR(Π, G, O, D);
begin
  $\mathcal{P} \leftarrow \emptyset$;
  foreach *plan* $\pi \in \Pi$ **do**
    foreach *action* $a \in \pi$ **do**
      $pos \leftarrow$ position index of a in π;
      if $\exists o \in O$ *such that* $o \models a$ **then**
        $apriori \leftarrow p_0$;
      else
        $apriori \leftarrow 0.5$;
      $cond\_aposteriori \leftarrow apriori \times$ expdecay$(pos, |\pi|)$;
      insert $(a, \pi, cond\_aposteriori)$ into $\mathcal{P}$;
  foreach *goal* $(g, p_g) \in G$ **do**
    $C \leftarrow \{$all of the conditions $(a, \pi, p) \in \mathcal{P} | \pi$ achieves $g\}$;
    $probability(g) \leftarrow$ noisyOR $(C) \times p_g$;
    insert $(g, probability(g))$ into $\mathcal{T}$;
  return $\mathcal{T}$;
end

the network components. Instead, the probability values attached to some of the components model *weights*, denoting how much the user believes a component is a likely target of an attack. If the input does not specify such a probabilistic weight for a component, SPAR uses 0.5 (no bias) as a default weight for that component.

Given this input, SPAR iterates over every attack plan π in the plan library generated by RAPPA. SPAR checks which actions of a plan were observed in the network. If an action was observed then SPAR uses the probability value specified in the input as an a priori belief in that observation. If no such probability is given, SPAR again uses 0.5 as default, which models that the algorithm does not have a bias towards whether the observation really occurred or not.

Once SPAR decides on a value for the likelihood of the observation, it uses an exponential decay function to propagate that probability value over the rest of the actions in the attack plan. We used exponential decay functions for probabilistic inference in this manner since the plan libraries are causal pathways in a graphical model for an attack surface and they do not include conditional probability information. If we have conditional probabilities (i.e., a Bayesian Network model), then this computation can be performed by Bayesian inference and/or chain rule. By default, SPAR uses a decay rate of 0.9; but this is a hyper-parameter that can be tuned empirically based on the attack planning domain.

SPAR then computes the probabilities, propagated from the observation to the end action (i.e., goal) of the plan, for each plan. Then, the algorithm extracts those plans for each given goal as input and uses a Noisy-OR computation to aggregate the propagated probabilities over the set of plans that achieve the same goal. The outcome of this Noisy-OR operation is an estimate for the conditional probability of that attack goal will happen, given the set of observations.

5. Preliminary Evaluations and Experiments

We conducted several preliminary experiments with plan recognition using the outcomes of RAPPA's diverse planning and plan library learning capabilities, coupling them with SPAR. In all of our experiments, we used LPG-d (Srivastava et al., 2007) as our diverse planner within RAPPA. We used the STRATUS scenarios described in Section 2 for our experiments. We varied the size of the input set of observations, in such a way that we started by the earliest possible observation in an attack and incrementally added new observations in the order they would be observed in an attack execution.

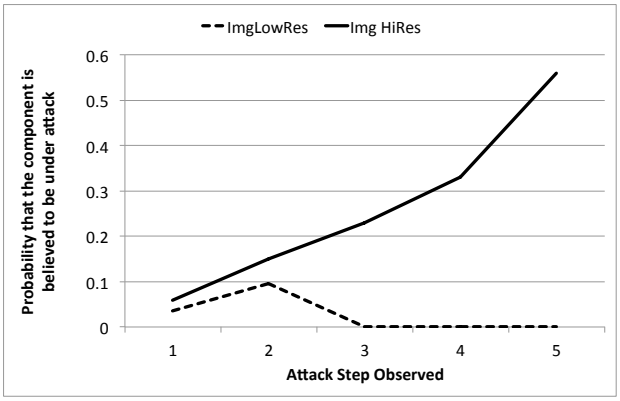

Figure 4. Probability of recognition of low-value vs. high-value attack targets, as a function the progress of the attack.

Figure 4 shows the results of an experiment exploring how SPAR differentiates between high- vs low-value targets given the a priori weights on those as the attack progresses. In this case, an attack on a Hi-Resolution Image Server is considered a more valuable target than the one with Low Resolution. In the first two steps of the attack, observations do not provide significant distinction between the two components, and SPAR believes both components could be targeted. As the attack progresses, and the evidence piles up towards Hi-Resolution Server, SPAR is able to differentiate and increases its beliefs that the Hi-Resolution Image Server is much more likely to be attacked, as opposed to the other component.

Figure 5 shows the results for a similar experiment but the targets in consideration in this case reside in different clouds in the network.

Finally, we observe that SPAR is fast – on average its run times are within a few milliseconds for a plan library of size 1296 attack plans, for each event hypothesis. The reason for its efficiency is because it does not attempt to explain the conditional probabilities; instead, it approximates them via statistical reasoning over input attack plan libraries.

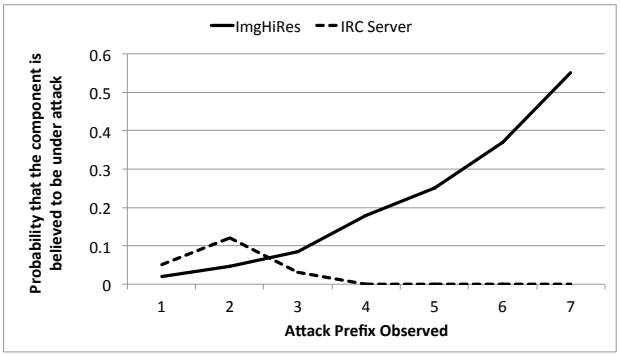

Figure 5. Probability of recognition of an attack target as a function of the direction of the attack in the network.

6. Related Work

6.1 Diverse Planning

The most influential work on plan diversity measures is that of Srivastava, et al (Srivastava et al., 2007), refined in Nguyen, et al (Nguyen et al., 2012). They propose three different distance measures for comparing plans, and for measuring the diversity of a set of plans: action distance (AD), causal-link distance (CLD), and state distance (SD). AD and CLD both project the plan, an *ordered* set of actions, down to an unordered set, and then compute a set-difference based distance between these sets.

Nguyen et al., actually provide *two* alternative definitions for state distance, which differ slightly in how they handle a difference between $l(p)$ and $l(p')$; for details see their paper. They provide two planning algorithms: GP-CSP, which can generate plans that attempt to maximize either action distance, causal-link distance, or state distance; and a more efficient method, LPG-d, which only uses action distance. Using GP-CSP, they provide experimental results on several domains to argue that action distance is the hardest to maximize. In general, later work in diverse planning confines itself to using action distance; we don't know of other work that uses CLD or SD.

The measures defined by Nguyen et al all have some problems. While they have the advantage of being domain-independent, no strong motivation is given, aside from their ready computability. Another problem is that they are not plan distance metrics, in the mathematical sense. Neither action distance nor causal-link distance satisfy the identity property, since different plans can give rise to the same action and causal link sets through reordering (note that action distance and causal-link distance *are* metrics over action and link sets, just not over plans). Similarly, in some problems it is possible for two different action sequences to give rise to the same state sequences. Action distance does not take into account information in the lifted representation of a domain. To action distance, the plans $p_1 =$ *drive(t1, a, b), drive(t1, b, c)* and $p_2 =$ *drive(t2, a, b), drive(t2, b, c)* look every bit as different as p_1 and $p_3 =$ *fly(a1, a, g), drive(t3, g, c).* Causal link and state distances also fail to take such generalizations into account. Further, in the case of state distance, correlated state fluents – fluents that change in lockstep – can artificially drive up the state difference between plans.

Other existing approaches to diverse plan generation either used intensive domain specific knowledge (e.g., (Myers & Lee, 1999; Coman & Muñoz-Avila, 2011)) or they are purely domain independent (e.g., (Boddy et al., 2005)). In many practical applications, it is hard, and sometimes not possible, to author the domain theories the former approaches require, and the latter typically suffers from shallow diversity metrics that may have pathological behavior in comparing plans.

Conceptually, the opposite of diversity is *similarity*. Similarity analyses has been the foundation of case-based reasoning from the field's beginning days (Aamodt & Plaza, 1994).Recently, the works reported in (Sánchez-Ruiz & Ontanón, 2014; Vattam, Aha, & Floyd, 2015) propose plan recognition approaches using case-based similarity measures. These works are closely related to our purposes in this paper: (1) although we have not reported in this paper, we are interested in hierarchical conceptualization and categorization of possible attacker behaviors in our scenarios and (2) case-based similarity measures provide a similar conceptualization as diversity measures as in RAPPA. In fact, we have recently started to develop a case-based similarity, or more generally, an analogical hierarchical planning, reasoning, and recognition approach for our next steps in RAPPA and SPAR. We will conduct an extensive comparative study for this work and the existing research in case-based reasoning.

6.2 Plan Recognition

Kautz' foundational, graph covering based work on plan recognition (Kautz, 1991), in fact did not assume perfect observations, but instead fit the best vertex cover to the plan graph. However, this makes it unable to address a number of issues that SPAR does address including multiple interleaved goals. Other early work using logic based reasoning algorithms (Carberry, 1990; Litman, 1986), while invaluable for formalizing the kinds of inferences that are necessary for efficient plan recognition. However, being logic based, they were not amenable to extension with probabilities.

ELEXIR (Geib & Steedman, 2007; Geib, 2009; Geib & Goldman, 2011) follows the early work of (Vilain, 1990) on plan recognition as parsing. However this early work does not actually present an algorithm or implemented system for plan recognition. Pynadath and Wellman (Pynadath & Wellman, 2000) formalize plan recognition based on probabilistic context free grammars (PCFGs). They use the structure of plans captured in a PCFG to build a fixed Dynamic Bayes Net (DBN), an use the resulting DBN to compute a distribution over the space of possible plans that could be under execution.

There are several recent works on probabilistic plan recognition generalized the foundational works by (Ramırez & Geffner, 2009; Ramırez & Geffner, 2010; Ramirez & Geffner, 2011) on planning for probabilistic goal recognition. The current very successful work on probabilistic activity recognition using HMMs and CRFs (Hoogs & Perera, 2008; Liao, Fox, & Kautz, 2005; Vail & Veloso, 2008) is probabilistic however it is addressing a different problem than the plan recognition problem. These systems are looking for a single label for a sequence of observations. (Ramirez & Geffner, 2011) have proposed using POMDPs that would directly address the partial observability of the world state. Like theHMMs and CRFs they attempting to infer a single high level goal that is the objective of an agent specified by a POMDP.

Targeting planning and human-robot coordination areas for probabilistic plan recognition specifically, the plan recognition approaches introduced in (Chakraborti et al., 2015a; Chakraborti et al.,

2015b; Talamadupula et al., 2014) build on the observation that probabilistic plan recognition typically does not commit to a plan, which pre-assumes a particular plan for the other agent, and therefore, it might be possible to minimize suboptimal (in terms of redundant or conflicting actions performed during the execution phase) behavior of the autonomous agent. Our approach is similar to these works in that RAPPA's diverse plan libraries provide a basis for us to tone down the redundant and conflicting actions in the plans and therefore, SPAR's probabilistic reasoning mechanisms are not affected by them. However, a more deeper comparison with RAPPA/SPAR and these new ground-breaking works are necessary and in order.

6.3 Learning

RAPPA's AHL algorithm described in the paper uses, in essence, a variant of chart parsing (e.g., (Charniak, Goldwater, & Johnson, 1998)) for learning the hierarchy of the plan library from an attack plan. A *chart parser* is a type of parser suitable for ambiguous grammars (including grammars of natural languages). It uses a dynamic programming approach - partial hypothesized results are stored in a structure called a chart and can be re-used. This eliminates backtracking and prevents a combinatorial explosion in the search space since the parser does not generate the same grammar rules multiple times in different parts of the search space.

HTN-Maker (Hogg, Muñoz-Avila, & Kuter, 2008) uses explanation-based reasoning techniques to learn the method's hierarchy and preconditions while HTN-Learner uses constraint-satisfaction techniques for this purpose. They both require the task semantics to be given in the form of (preconditions,effects) pairs. These systems exhibit characteristics beyond the scope of this paper: HTN-Maker and variants are able to learn in domains were actions have non-deterministic effects and take into account plan quality factors. HTN-Learner (Zhuo et al., 2009), a variant of HTN-Maker, is also able to learn the operators' preconditions and effects under partial state observability (i.e., as usual for most learners, the input is a collection of plan traces annotated with the intermediate states; in HTN-Learner's case these states might have missing information).

Camel (Ilghami et al., 2005) assumes that not only the annotated plan traces is given but it also requires two more inputs: the complete task structure and samples of incorrect uses of the task structures. Camel uses these inputs and a version space algorithm to learn the method's preconditions. Another HTN learner in this category is DiNCAT (Xu & Muñoz-Avila, 2005) (not shown in the table). Like Camel, it requires the hierarchical structure to be given as input. Unlike Camel it does not require the incorrect uses of the hierarchy to be given as input; it uses case-based generalization techniques to learn the method's preconditions.

ICARUS (Langley & Choi, 2006) and its variant LIGHT (Nejati, Langley, & Könik, 2006) learn both method's preconditions and the hierarchical structure. It uses teleoreactive learning techniques whereby STRIPS planning is used to fill gaps in the knowledge of the methods; when the methods cannot be used to solve a subproblem, STRIPS planning is used to solve the subproblem and methods are learned from the subplan generated. Icarus and LIGHT require the skill definitions to be given as input. These skills provide the semantics of the tasks, which are STRIPS goals.

Algorithms for learning grammars can be capable of generating hierarchical structures if we map the input plans into strings (i.e., mapping the plan's actions into the string's symbols) and the tasks into the grammar's variables. The grammar learned could be used to generate the hierarchi-

cal structures (i.e., by mapping the grammar's parse trees to HTNs). Grammar learning algorithms (Oates, Desai, & Bhat, 2002; Sakakibara, 1997) generate a grammar that best maps the input strings. These techniques are exploited by the Greedy Structure Hypothesizer (GSH), which uses probabilistic context-free grammars learning techniques to learn a hierarchical structure of the input plan traces. This hierarchy reflects user's preferences. GSH does not learn preconditions since its goals are not to generate the grammars for planning but to reflect user's preferences. In general, grammar learning algorithms does not consider applicability conditions. Furthermore, the learned hierarchical structure is a function of commonalities between subsequences in the input traces, regardless of which tasks they achieve.

X-Learn (Reddy & Tadepalli, 1997) uses bootstrapping learning techniques in which first simple traces are given to learn methods to achieve these traces. Progressively more complex traces are given where the knowledge learned in previous iterations can be exploited to simplify the trace and learn new task decompositions. Tasks correspond to one of the goals that is achieved at the end of the trace. Hence, its task semantics are defined as the usual STRIPS semantics for goals.

7. Conclusions and Future Work

We have described a system composed of two sister algorithms, RAPPA and SPAR, for reasoning and recognizing attacker goals in cyber systems. Although such goals are not necessarily part of the goals of a mission itself, reasoning about them enables mission resilience since we can recognize and respond to those attack goals in advance. We have presented a preliminary evaluation of the two algorithms, demonstrating that they are promising for the cyber scenarios of our interest.

We currently working on a new approach which we call R2S2 (RAPPA2 / SPAR2). Unlike action-based diverse planning techniques in RAPPA, our new approach builds a hierarchical planning graph, in which actions could be either primitive or abstract. In the latter case, an abstract action represents a set of plans that are semantically equivalent. Thus, we reduce and replace our diverse planning capability to generating this compact conceptual planning graph, which represents the set of all possible diverse abstract plans in a planning domain.

SPAR2 algorithm then performs a backward breadth-first search over this graph, not only generating conditional probabilities for possible attacker goals, but also explaining those conditional probabilities causally. We believe that this recognition approach using hierarchical plan graphs, will also address a limitation in SPAR: larger plan libraries containing longer plans poses a tractability issue for SPAR to deal with its joint probability distributions. Hierarchical plan graphs, on the other hand, as compact and abstract representations of such plan libraries, will naturally alleviate this issue for SPAR2. This is also important for another direction with SPAR2: reasoning with partially-observable observation states.

Acknowledgments. Thanks to the entire STRATUS project and our funders. This work was supported by Contract FA8650- 11-C-7191 with the US Defense Advanced Research Projects Agency (DARPA) and the Air Force Research Laboratory. Approved for public release, distribution unlimited. The views expressed are those of the authors and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

References

Aamodt, A., & Plaza, E. (1994). Case-based reasoning: Foundational issues, methodological variations, and system approaches. *AI Communications*, *7*, 39–59.

Boddy, M. S., Gohde, J., Haigh, T., & Harp, S. A. (2005). Course of action generation for cyber security using classical planning. *International Conference on Automated Planning and Scheduling* (pp. 12–21). AAAI.

Burstein, M., Goldman, R., Robertson, P., Laddaga, R., Balzer, R., Goldman, N., Geib, C., Kuter, U., McDonald, D., Maraist, J., Keller, P., & Wile, D. (2012). Stratus: Strategic and tactical resiliency against threats to ubiquitous systems. *Proceedings of SASO-12*.

Carberry, S. (1990). Plan recognition in natural language dialogue. *ACL-MIT Press Series in Natural Language Processing*.

Chakraborti, T., Briggs, G., Talamadupula, K., Scheutz, M., Smith, D., & Kambhampati, S. (2015a). Planning for serendipity. *ICAPS Workshop on Planning and Robotics*.

Chakraborti, T., Zhang, Y., Smith, D., & Kambhampati, S. (2015b). Planning with stochastic resource proïňĄles: An application to human-robot co-habitation. *ICAPS Workshop on Planning and Robotics*.

Charniak, E., Goldwater, S., & Johnson, M. (1998). Edge-based best-first chart parsing. *Association for Computational Linguistics*, 127–133.

Coman, A., & Muñoz-Avila, H. (2011). Generating diverse plans using quantitative and qualitative plan distance metrics. *AAAI*.

Geib, C. (2009). Delaying commitment in probabilistic plan recognition using combinatory categorial grammars. *IJCAI-09*.

Geib, C., & Goldman, R. (2011). Recognizing plans with loops represented in a lexicalized grammar. *AAAI-11*.

Geib, C., & Steedman, M. (2007). On natural language processing and plan recognition. *IJCAI-2007*.

Goldman, R. P., & Kuter, U. (2015). Measuring plan diversity: Pathologies in existing approaches and a new plan distance metric. *AAAI/IAAI-15*.

Hogg, C., Kuter, U., & Muñoz-Avila, H. (2009). Learning hierarchical task networks for nondeterministic planning domains. *IJCAI-09*.

Hogg, C., Kuter, U., & Muñoz-Avila, H. (2010). Learning methods to generate good plans: Integrating htn learning and reinforcement learning. *AAAI-10*.

Hogg, C., Muñoz-Avila, H., & Kuter, U. (2008). HTN-MAKER: Learning HTNs with minimal additional knowledge engineering required. *AAAI-08*.

Hoogs, A., & Perera, A. A. (2008). Video activity recog- nition in the real world. *AAAI-08*.

Ilghami, O., Nau, D. S., Muñoz-Avila, H., & Aha, D. W. (2005). Learning preconditions for planning from plan traces and HTN structure. *Computational Intelligence*, *21*, 388–413.

Kautz, H. (1991). *A formal theory of plan recognition and its implementation*. Doctoral dissertation, University of Rochester.

Klenk, M., Molineaux, M., & Aha, D. W. (2013). Goal-driven autonomy for responding to unexpected events in strategy simulations. *Computational Intelligence*, *29*, 187–206.

Langley, P., & Choi, D. (2006). Learning recursive control programs from problem solving. *Journal of Machine Learning Research*, *7*, 493–518.

Liao, L., Fox, D., & Kautz, H. (2005). Location-based activity recognition using relational markov networks. *IJCAI-05*.

Litman, D. (1986). Understanding plan ellipsis.

Myers, K. L., & Lee, T. J. (1999). Generating qualitatively different plans through metatheoretic biases. *Proc. National Conf. on Artificial Intelligence (AAAI)*.

Nejati, N., Langley, P., & Könik, T. (2006). Learning hierarchical task networks by observation. *ICML*.

Nguyen, T. A., Do, M. B., Gerevini, A., Serina, I., Srivastava, B., & Kambhampati, S. (2012). Generating diverse plans to handle unknown and partially known user preferences. *Artificial Intelligence*, *190*, 1–31.

Oates, T., Desai, D., & Bhat, V. (2002). Learning k-reversible context-free grammars from positive structural examples. *ICML '02: Proceedings of the Nineteenth International Conference on Machine Learning* (pp. 459–465). San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Pynadath, D., & Wellman, M. (2000). Probabilistic state-dependent grammars for plan recognition. *UAI-00*.

Ramırez, M., & Geffner, H. (2009). Plan recognition as planning. *Proceedings of the 21st international joint conference on Artifical intelligence. Morgan Kaufmann Publishers Inc* (pp. 1778–1783).

Ramırez, M., & Geffner, H. (2010). Probabilistic plan recognition using off-the-shelf classical planners. *Proceedings of the Conference of the Association for the Advancement of Artificial Intelligence (AAAI 2010)*.

Ramirez, M., & Geffner, H. (2011). Goal recognition over pomdps: Inferring the intention of a pomdp agent. *AAAI-11*.

Reddy, C., & Tadepalli, P. (1997). Learning goal-decomposition rules using exercises. *Proc. International Conf. on Machine Learning (ICML)*.

Sakakibara, Y. (1997). Recent advances of grammatical inference. *Theoretical Computer Science*, *185*, 15–45.

Sánchez-Ruiz, A. A., & Ontanón, S. (2014). Least common subsumer trees for plan retrieval. In *Case-based reasoning research and development*: Springer.

Shrobe, H. E. (2002a). Computational vulnerability analysis for information survivability. *Proceedings of the Eighteenth National Conference on Artificial Intelligence and Fourteenth Conference on Innovative Applications of Artificial Intelligence (AAAI/IAAI-02)* (pp. 919–926). Menlo Parc, CA, USA: AAAI Press.

Shrobe, H. E. (2002b). Computational vulnerability analysis for information survivability. *AI Magazine*, *23*, 81–91.

Srivastava, B., Nguyen, T. A., Gerevini, A., Kambhampati, S., Do, M. B., & Serina, I. (2007). Domain independent approaches for finding diverse plans. *Proceedings of the International Joint Conference on Artificial Intelligence*.

Talamadupula, K., Briggs, G., Chakraborti, T., Scheutz, M., & Kambhampati, S. (2014). Coordination in human-robot teams using mental modeling and plan recognition. *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on* (pp. 2957–2962).

Vail, D. L., & Veloso, M. M. (2008). Feature selection for activity recognition in multi-robot domains. *AAAI-08*.

Vattam, S., Aha, D., & Floyd, M. W. (2015). Error tolerant plan recognition: An empirical investigation. *The Twenty-Eighth International Flairs Conference*.

Vattam, S., Klenk, M., Molineaux, M., & Aha, D. W. (2013). *Breadth of approaches to goal reasoning: A research survey* (Technical Report). DTIC Document.

Vilain, M. B. (1990). Getting serious about parsing plans: A grammatical analysis of plan recognition. *AAAI-90*.

Xu, K., & Muñoz-Avila, H. (2005). A domain-independent system for case-based task decomposition without domain theories. *AAAI-05*.

Zhuo, H. H., Hu, D. H., Hogg, C., Yang, Q., & Munoz-Avila, H. (2009). Learning htn method preconditions and action models from partial observations. *Proc. Internat. Joint Conf. on Artificial Intelligence (IJCAI)*.