

# Planning to Discover and Counteract Attacks

Tatiana Kichkaylo, Tatyana Ryutov, Michael D. Orosz, Robert T. Neches

Information Sciences Institute

University of Southern California, Marina del Rey, CA 90292, USA

## Abstract

Recognizing attack plans is one of the goals of security analysts. Attack plan recognition is critical for predicting future actions of attackers, generating possible actions (i.e., probes) to test attacker plans, and planning appropriate responses. The full range of potential attack scenarios is too rich to generate manually, and too complex for direct analysis and evaluation of the impact of alternative probes and defenses. We are developing a set of tools that address these issues by:

- Constructing multiple scenarios/pathways using available partial plan segments (referred to as snippets), possibly discovering new combinations;
- Providing visual tools for exploring sets of possible scenarios under various observables, importance, and likelihood conditions, helping the analyst generate probes and countermeasures;
- Comparing different probes/countermeasures by assessing their discrimination/attack mitigation potential and possible side-effects;
- Automatically suggesting potential probes and countermeasures constructing possible scenarios given observable activity.

These tools can provide decision support for different domains including terrorist activity recognition and network intrusion detection.

## 1 Introduction

The problem of detection, prevention and/or response to attacks is common to multiple domains. In the cyber world, a goal of an attacker may be to gain access to protected resources or to disrupt service to legitimate users. In the counter-terrorism domain an attack manifests as actions in the physical world. In

robotic soccer an attack is a sequence of actions of a team aimed to outmaneuver the opponents and score a goal.

The kinds of attacks can be classified along several dimensions. Different techniques and tools for attack detection and prevention/counteraction are applicable in different regions of this multi-dimensional space.

- Attacks may be fast or slow. Slow attacks, such as terrorism in the physical world, leave sufficient time for human analysts to respond to alerts and warnings. In the case of cyber-attacks a reaction is often required in real time, which often leaves no room for human operators or complex reasoning algorithms.
- Vulnerabilities may or may not be known in advance. In the case of existing physical infrastructures or legacy systems it may be impossible to eliminate vulnerabilities. For many of these legacy systems and physical infrastructures, it is possible to design and develop systems that detect attack strategies for known weak points. In the case of unknown vulnerabilities these external detection systems need to monitor the general health of the legacy system and dynamically devise detection and response strategies if something abnormal is detected.
- An attack may be an expected behavior of other agents or an anomaly. In soccer, one may assume that any action of the opposing team is a part of an attack. In the cyber-world, most actions are part of some legitimate user activity.

In this paper we discuss a set of tools and approaches that may help to identify vulnerabilities, detect potential attacks based on observables, and devise detection or counteraction measures as to minimize disruption of normal operations. We describe the possible use of such tools by human analysts and discuss how the algorithms underlying these tools could be integrated into autonomous detection systems. In addition, we describe an initial prototype system developed for the Intelligence Advanced Research Projects

Activity (IARPA) sponsored Proactive Intelligence (PAINT) project that generates both benign and nefarious pathways (i.e., plans that an opponent or adversary may take to achieve an objective) based on an initial goal and a subset of partial plan segments (which we call snippets).

## 2 The Problem

Recognizing attack plans is one of the goals of security analysts. Plan recognition has been a research area in artificial intelligence (AI) for decades. In AI, plan recognition is a process of inferring the goals of an agent from observations of the agent's activities. In security applications (e.g., intrusion detection), the plan recognition process is concerned with *adversary recognition*, where attackers try to avoid or interfere with recognition process and can take deliberate actions to hide their actions and intentions.

The assumptions used in traditional plan recognition [Blythe1999; Calo2003] are not valid in the security-related adversary recognition domain. In some domains (e.g., RoboCup soccer [Huang2003], computer games [Buro2003], and military simulations [Heinze1999]) the adversary and its high level goals are known. In security domains (e.g., computer security, information warfare, and antiterrorism), the adversary tries to hide its actions and identity. Additionally, its real intentions are not always clearly identifiable.

The challenges in adversary plan recognition and response in security domain include the following [Geib2001; Geib2002].

**Uncertainty and incompleteness.** We may not be able to observe all attacker activities, and can not detect all attack steps due to the limitation or deployment of security sensors. Moreover, we may have an incomplete knowledge of the possible attack plans.

**Partially ordered plans.** Often attackers' plans are flexible in the ordering of their plans steps; therefore we must be able to recognize the multiple possible instantiation orderings created by these plans.

**Multiple concurrent goals.** Attackers can have multiple dynamic attack plans, e.g., a hacker might be interested in stealing sensitive data as well as using computers to launch attacks against other targets.

**Actions used for multiple effects.** Often in the computer security domain a single action can be used for multiple effects. For example, scanning of a domain can be used both for planning a DoS (Denial of Service) attack as well as to identify the web server that a hacker wants to deface.

**Misleading behavior.** Attacker can take actions to mislead plan recognition, or to exploit some of its weaknesses.

**Multiple weighed hypotheses.** Providing a single explanation for the observed actions often is not as helpful as ranking the possibilities. Consider observed scanning activity. While this indicates a hacker is interested in a network, by itself it provides very little evidence about the hacker's intent. Rather than giving just one of the many equally likely answers - it is much more helpful to report the relative likelihood of each of the possibilities.

**Automated vs. human-in-the-loop operation.** In security applications, the purpose of adversarial plan recognition is to predict possible attacks in order to generate effective countermeasures. In many current human-in-the-loop operations, real-time detection and response is not achievable and the resulting delays may result in an actual attack. As [Porras1997; Carver2000] have pointed out, in such applications, an automated attack recognition and reaction system avoids these delays and can stop an attack before the damage is done. Automated systems, however, can produce negative outcomes since some response actions (e.g., changes to firewall rules) can negatively affect legitimate users. Hence, the application of automated techniques is limited to well-known attacks (e.g., signature-based IDS) and targeted countermeasures.

Detection of stealthy attacks, such as malicious insider covert activity and terrorist preparations, can be complex and may require some probes (i.e., actions designed to provoke a response that can be used to gather additional information). To prevent a situation where a probe or a response action causes more damage than the actual attack, a system that allows an analyst to reason about the likelihood and severity of an attack and the effects of a possible probe/response mechanism is necessary.

Existing approaches for predicting adversary actions, such as game theoretic and game playing, adversarial planning, and pattern

recognition at best provide partial/limited solutions. Data mining approaches have been used to find information that helps to detect attacks, however the approaches suffer from a high false alarm rate and do not help analysts connect separate events. It is necessary, therefore, to develop algorithms and tools for security analysts to further analyze and correlate attack scenarios so that they can make accurate situational assessments and take appropriate responses to minimize damage.

### 3 Modeling alternatives

The algorithmic core of our toolkit is a constraint-based planning system [Kichkaylo2007]. The system core maintains alternative plans consisting of *tokens*. A token may represent an action or a state. Both past events (observed or hidden) and future ones (plan projection) are part of a plan. Tokens contain variables representing temporal (start, end, duration) and resource (actor, equipment) properties of the action/state. Variables can be used as arguments to *constraints*, which define tuples of values the constraint arguments can take [Dechter2003]. Constraints enforce partial temporal ordering and/or resource dependencies between tokens.

A plan is seeded with a set of tokens representing high-level goals, target states, and/or observed events. The planning module then refines the seeded plans according to *snippets*. Snippets are similar to HTN methods [Erol1994]. A snippet captures a modification of a plan in the form of “if a certain configuration of tokens is a part of the plan, then the following configuration of tokens is also a part of the plan”. Snippets thus can represent expansion of high-level goals into lower-level actions and states or enforce causality (e.g., if an attacker knows a password, then an action of stealing the password should have preceded the attack). If multiple snippets are applicable in a given situation (e.g., there are multiple ways to achieve the same goal), the planner will create alternative plans using each of the snippets. Identifying and building attack snippets is governed entirely by the user.

For example, Figure 1 shows a snippet describing the use of the lpr attack to achieve a state when an attacker gets access to a protected file. This snippet says that a possible way to explain the presence of a “secret printed” state in

a plan is to add to the plan a partially ordered sequence of actions implementing the attack. Lines on the figure represent temporal constraints. Additional resource constraints (not shown) declare that all variables marked as “f” (or “s”) refer to the same file. Constraint propagation helps to limit the set of possibilities and to define windows of interest. For example, suppose an analyst has an estimate for the earliest time a printed copy of a file *foo.ps* was available to an attacker. Propagation of this file name and time point will limit the set of *ln -s* commands that would be considered as a possible part of the attack. Note also that application of snippets may reuse existing tokens in the plan. For example, *block ppt* command might have been issued by a different user for a legitimate reason and hijacked by the attacker.

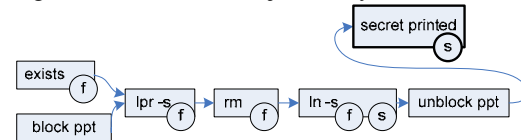


Figure 1 lpr attack snippet

In addition to describing goal-subgoal relationships, snippets can describe necessary effects of actions. For instance, if a port scan detection tool is installed and there is a port scan in progress, an alert will be generated.

Representing the domain knowledge in the form of snippets allows the system to mix-and-match various steps to discover new combinations that may constitute possible new attacks. Further, by using automated planning algorithms instead of employing human analysts to manually compose scenarios, our approach provides better coverage of both attacks and normal operations. This in turn leads to high-quality analysis of vulnerabilities, possible attacks and side-effects of countermeasures. Of course, the downside of this approach is that the operator can be overwhelmed with many different plans to consider. There are, however, techniques (to be discussed) to help reduce/filter the set of all possible plans down to a set of plausible plans to be considered.

We now describe how this algorithmic core may be used in end-user tools for security analysis.

### 4 Discovering potential plans of attack

One way to discover potential vulnerabilities in a system is to think like an attacker. Our tools allow the user to specify multiple high-level

goals, both for attacks and (typically) benign activities. In the network intrusion detection domain, the goals of an attacker can be denial of service (e.g., taking down a web site) unauthorized access to sensitive information (e.g., password or credit card numbers) and unauthorized modification of data (e.g., web site defacement). To stage an attack, multiple objectives may need to be achieved (e.g., steal web site password and then deface the site).

Once the goals/objectives have been established/defined, the planning system then builds a set of plans in the form of partially ordered sets of executable actions and/or observable states that achieve these goals. Since multiple goals are considered together, actions are reused where possible. For example, in the data network domain, the installation of Trojan software can be used to 1) hide an attacker break in to avoid early detection, 2) capture key strokes for stealing passwords, and/or 3) install backdoors that enable an attacker to use the compromised system for a DDoS attack.

The resulting set of plans is similar to a set of attack trees, where each path through an attack tree represents a unique attack with overall attacker's objective placed in the root. A snippet can be seen as node (or a set of nodes) in an attack tree. Attack tree representation requires explicit chronological orderings of exploits (nodes). Our approach supports snippet reuse and captures partial ordering of the plans, and therefore, provides more compact representation. Furthermore, attack trees do not provide adequate visualization support for analyzing different attack plans. After a set of plans has been constructed, a user/analyst may explore individual plans and run multiple analyses on the set of plans as a whole. The rest of this section illustrates some of the ways this can be done. The screenshots are taken from several prototype tools built as part of the IARPA PAINT (Proactive Intelligence) project.

**Exploring details of a plan and comparing individual plans.** An individual plan consists of a set of tokens (representing actions and states) connected with goal-subgoal relations. Note that because of the reuse, an action may be a subgoal to multiple goals. Each token has a set of variables, including a description of the start and end of the action, resources involved, and the agent performing the action. For example, to

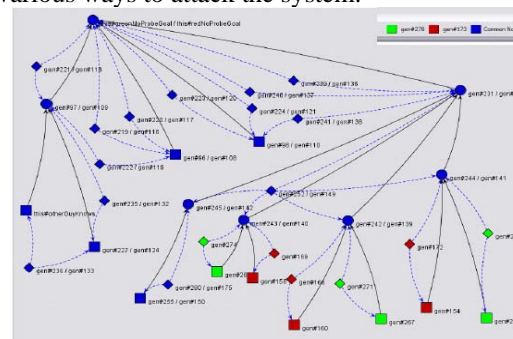
achieve the goal “find vulnerable system to install a Trojan”, several actions are needed:

- Step 1: Look for active IP addresses, open ports, operating systems (OS) and any applications running.
  - Step 2: Create a report.
  - Step 3: Determine the patch level of the OS or applications.
  - Step 4: Attempt to exploit the vulnerability.
- Scanners may either be malicious or friendly. Friendly scanners usually stop at step 2 and occasionally step 3 but never go to step 4.

Constraints, such as temporal ordering and resource restrictions, and semantic relations, connect the variables. In the “Trojan” example above, the temporal constraints are: Step1 before Step2 before Step3 before Step4.

The user can browse details of each plan to discover why each action is added to the plan and which goals each action is contributing to. In addition, the user can compare individual plans (see Figure 2 which displays two plans). In the figure, circles and squares represent high-level and atomic actions respectively, and rhombs represent constraints. Blue shapes describe elements common to both plans, and green and red shapes show elements present in one plan but not in the other. Such comparisons can help the analyst identify differentiating points between attacks and benign activity and potential conditions for generating alarms and warnings.

It is also possible to compare sets of multiple plans at once (Figure 3). For example, one might want to find differentiating points between various ways to conduct normal operations and various ways to attack the system.



**Figure 2 Plan comparison interface**

**Plan recognition.** In addition to building plans from goals to observables, the same approach can be used to apply snippets bottom up (i.e., from observables to goals). This process will

provide an explanation to observed activity. As with the top-down process, bottom-up inference can produce multiple alternatives (note: we have not yet implemented this capability). For instance, if we observe the actions depicted in Figure 1, we may suspect an unauthorized file access scenario exploiting the lpr vulnerability. The list of actions indicates an attack if a user who executes the commands does not have access to the file *secret*. Otherwise, this may indicate unusual, but non-malicious activity.



Figure 3 Interface for comparing sets of plans

**Weighing plans and actions.** Typically, there exist multiple paths to achieve the same result. In these cases, some plans are more preferable (and/or likely) than others. Figure 4 shows a user interface that allows the analyst to adjust weights of individual plans and/or actions. This tool can be used to identify correlations between actions and states. For example, this feature may be useful in cases where an actual event which is not observable has occurred, however, a related event is observable. As a further example, we may not know that our system was compromised and is being used as “zombie” to stage a DDoS attack against other host, but we can observe unusual traffic indicating the presence of a hacker DDoS tool (e.g., trino or TFN).

Plan	P-Hood	Distribute...	Distribute...	Distribute...	Distribute...	Distribute...	Distribute...
A1-Hood	0.7442962...	59.0564943...	0.0	0.0	0.0	49.3151906...	0.0
gen#19928	0.67442962...	*					*
gen#10194	0.58221269...	*					*
gen#10256	0.58221269...	*					*
gen#10948	0.49107633...	*					*
gen#12290	0.36750093...	*					*
gen#12416	0.26299593...	*					*
gen#12519	0.26299593...	*					*
gen#12908	0.21907793...	*					*
gen#13247	0.21305160...	*					*
gen#13466	0.14087245...	*					*
gen#13910	0.157482482...	*					*
gen#14036	0.61009750...	*					*
gen#15201	0.27384763...	*					*
gen#15422	0.19090216...	*					*
gen#15512	0.21481726...	*					*
gen#15908	0.13396032...	*					*
gen#19923	0.06700002...	*					*
gen#10353	0.01066923...	*					*
gen#10151	0.01066923...	*					*
gen#10454	0.95515781...	*					*

Figure 4 Weighing plans and actions interface

Weight of an action (specified in the columns) is the likelihood of that action or observable state to actually happen given the set of plans it is part of and likelihoods of these plans. Weight of a plan is the likelihood of that plan being carried out with respect to alternative plans. The initial

weights for plans may be assigned uniformly or using some heuristic function. For example, for a server which provides a critical online service, a plan corresponding to an attack can be assigned a higher value than for a home server.

The user can adjust weights of plans and actions to simulate various “what-if” scenarios. Once a single value is modified, the implications are propagated to other values. For example, declaring that a particular observable has been detected will increase the likelihood of plans containing this observable and decrease likelihood of plans that do not contain it. This in turn affects likelihoods of other actions and observables constituting these plans.

**Likelihood propagation within a plan.** The logical structure of tokens, constraints, and goal-subgoal links can be leveraged to reason about the likelihood of violations within each plan (Figure 5). In this example, multiple ways exist to introduce a contaminant into food supply. If contamination is detected somewhere in the distribution chain, ruling out parts of the distribution network as potential sources increases the probability that the contamination originated at other sites.

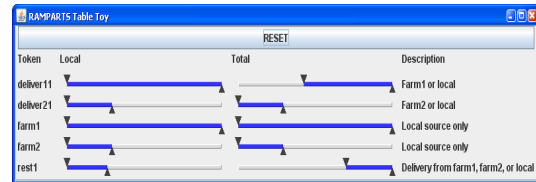


Figure 5 Likelihood propagation interface

## 5 Stopping attacks with minimum collateral damage

The analytical features described in the previous section can be combined to generate more powerful tools for a given domain. One goal of such tools is development of countermeasures that minimize collateral damage.

A major problem in alert generation is false positives. One way to decrease false positives is to perform a diagnostic action upon the initial detection of a potential problem. The reaction to this action will, in many cases, provide a clue as to whether the alert in question may indeed be part of an attack and thus increase the confidence in the original alert and decrease false positives.

By analyzing multiple ways to execute an attack and contrasting them with multiple ways to

perform benign activity, our tools help to identify potential points for alert generation and intervention. For example, these tools may help an analyst determine an action (i.e., a probe) that might force a potential adversary to (unknowingly) proceed down a pathway that produces a predicted benign outcome rather than proceed down a pathway that produces a predicted nefarious outcome.

The difficulty in the aforementioned approach is the selection of an action for diagnosis or intervention. Often, such actions will affect not only an attack but benign activity as well. For example, detection of SYN flooding attack may use active probing [Xiao2005] that collects data on the delay between the server and the client. While timely and reliable, this scheme imposes processing and storage overhead.

Earlier we described tools that help an analyst to analyze sets of attack plans and contrast them with benign activity. A scenario is seeded with a set of goals, and the system generates various potential realization of the scenario. The same technique can be used to analyze potential effects of a diagnostic action or countermeasure. The new action is simply added to the description of the scenario. The resulting set of plans can then be contrasted with the one obtained without the diagnostic action/countermeasure.

This approach can be used to analyze effects of response strategies in addition to individual actions. Multiple actions can be added to the scenario. Further, by introducing a new snippet instead of a fixed action the analyst can model actions triggered by certain activity. As a result, such actions will occur in some realizations of the scenario but not in others.

## **6 Automated detection and counter-action of attacks – the future**

The previous section addressed implementing solutions that rely on a user or analyst to aid in the discovery of potential attack plans and in the exploration of probe effects on those plans. In many domains (e.g., physical infrastructure security) this is sufficient. However, in some domains (e.g., data networks), the delays in response from a human actor are not acceptable and require an automated response.

Automating the process described in the previous sections presents a number of challenges. How

to represent human inference and decision-making in the process and how to guard against false-positives and false-negatives. How to represent choice (as in choosing between different probes or actions) represents a difficult challenge to overcome. Finally, who is responsible when such an automated system responds to a false-positive or false-negative? Although not addressed in this paper, these concerns will need to be considered as plan detection and response technologies evolve.

## **7 Related work**

[Huang2003] developed a technique for automated plan recognition in the field of RoboCup simulation soccer games. For each agent representing a player in the game, the technique translates the actions observed from various adversaries (consecutive or discrete multivariable streams) into behavior queues using prediction and backfill techniques. After populating an agent's behavior queue, frequent and interesting behavior sequences are identified using a statistical dependency test. These sequences are then retrieved and transform into formalized plans. Finally the plans are refined while multi-agent teams adopt them. [Han1999] applied Hidden Markov Models (HMMs) for recognizing opponent behaviors in RoboCup soccer simulations. HMMs states correspond to decomposed robot behavior. Uncertainty in recognizing behavior is represented as probabilistic transitions between the states. [Marling2003] adopts case-based reasoning (CBR) for opponent modeling and planning players' strategies in RoboCup competitions. In CBR, solution to a problem at hand is found by reusing solutions to similar problems encountered in the past.

All these techniques are not directly applicable to the security domain, since the plan recognition process depends on the observations of opponent players, position of ball and gates, and the game state at a particular moment. In security, we may not know who the adversary is, what its goal is, and whether the adversary exists (observed activity can be legitimate).

Attacker plan recognition [Geib2001; Cuppens2002; Queen2004] in the network security domain largely concentrates on correlation of observed actions and alerts produced by intrusion detection systems. [Geib2001] presented a probabilistic model of plan recognition for recognizing and predicting

the intentions of the agents based on the construction of execution traces from raw security alerts. This method requires predefined attack plan library and lacks support for reasoning about deceptive actions of an adversary. [Cuppens2002] proposed a method for detecting various steps of an intrusion scenario, which is seen as a planning activity based on a declarative description of actions, goals, and plans. The method does not provide additional information to distinguish between the most and least plausible scenarios, which is an important feature since the number of possible scenarios can be large. [Benferhat2003] extends Cuppens' approach by providing the ability to rank possible scenarios. [Qin2004] proposed a graph-based technique to correlate isolated attack scenarios derived from low-level alerts. Attack trees define attack plan libraries used to correlate isolated alert sets that are converted into causal networks with assigned probability distributions to evaluate the likelihood of attack goals and predict future attacks.

None of these systems provide visual tools for an analyst to explore sets of possible scenarios under various observables, importance, and likelihood conditions, helping the analyst generate probes and countermeasures

[Harp2005] proposes a method to analyze and test threats posed by malicious insiders. They use AI planning to automatically generate courses of action that an adversary is likely to choose in subverting the system. The analyst can then use this information to evaluate the vulnerability of a system to attacks, and to select the most reasonable defensive measures. There is no notion of uncertainty or likelihood in the generated plans, no support for comparative analysis of several plans to achieve a given goal. [Jarvis2004] present the application of plan recognition techniques to support analysts in processing national security alerts by automatically identifying the hostile intent behind them. The system requires a complete library of attack templates.

## 8 RAMPARTS Prototype 1.0

As noted earlier, we have developed an initial prototype system that implements several of the features/capabilities described in this paper. The *Risk Analyses and Models of Plans for Attacks to Recognize Terrorist Schemes* (RAMPARTS) project was funded by IARPA as part of the Proactive Intelligence (PAINT) program. The

objective of this initial effort was to develop the supporting RAMPARTS infrastructure and develop a subset of capabilities to demonstrate an initial proof-of-concept.

Based on an initial set of goals and a set of plan snippets (generated by subject matter experts), the RAMPARTS prototype (1.0) generates and visually displays possible plans (both nefarious and benign) that a potential adversary/opponent might follow. The RAMPARTS toolkit also allows the user/analyst to explore the plans to help determine which key actions/events – if observed – could be used to help the analyst predict whether the potential adversary is going down a nefarious or benign pathway without actually knowing which exact pathway is being taken. The advantage of this approach is that if it's determined – based on the observable event – that the potential adversary is definitely going down a benign pathway, no further action is required by the analyst or RAMPARTS.

The next step (to be implemented in prototype 2.0) is to determine which “probe” or “probes” (active or passive) to implement to possibly cause the specified event/action to be observed or to cause (or at least attempt to cause) the potential adversary to go down a benign pathway (ideally, without their knowledge). In addition, we plan to use the DHS and NSF funded DETER [Benzel2007] infrastructure for conducting experiments in computer security, as a test bed for further development of the project. Further, we plan to test the next prototype on a number of port security scenarios as part of the DHS sponsored USC Center for Risk and Economic Analysis of Terrorism Events (CREATE) PortSec (Port Security) project.

## 9. Acknowledgement

This research was partially supported by the Intelligence Advanced Research Projects Activity (IARPA) under grant number FA8750-07-2-0177. However, any opinions, findings, and conclusions or recommendations in this document are those of the authors and do not necessarily reflect the views of IARPA.

## References

[Benferhat2003] S. Benferhat, F. Autrel et F. Cuppens. Enhanced Correlation in an Intrusion Detection Process Second International Workshop Mathematical Methods, Models and

Architectures for Computer Networks Security, September 20-24, 2003.

[Benzel2007] T. Benzel, R. Braden, D. Kim, A. Joseph, C. Neuman, R. Ostrenga, S. Schwab, K. Sklower. Design, Deployment, and Use of the DETER Testbed, In Proceedings of the DETER Community Workshop on Cyber Security Experimentation and Test, August 2007.

[Blythe1999] J. Blythe. Decision-Theoretic Planning. *AI Magazine*, 20(2), 1999.

[Buro2003] M. Buro & T. Furtak, RTS Games as Test-Bed for Real-Time Research, Invited Paper at the Workshop on Game AI, JCIS. 2003.

[CALO2003] CALO. Cognitive agent that learns and organizes, <http://calo.sri.com.>, 2003.

[Carver2000] C. A. Carver, J. M. D. Hill, J. R. Surdu, and U. W. Pooch, A Methodology for using Intelligent Agents to provide Automated Intrusion Response, Proceedings of the IEEE Systems, Man, and Cybernetics Information Assurance and Security Workshop, 2000.

[Cuppens2002] F. Cuppens, F. Autrel, A. Miège and S. Benferhat. Recognizing Malicious Intention in an Intrusion Detection Process. In *Soft Computing Systems - Design, Management and Applications*, volume 87, 806–817, 2002.

[Dechter2003] R. Dechter. *Constraint Processing*. Morgan Kaufmann Publishers Inc., 2003

[Erol1994] K. Erol, J. Hendler, and D. Nau. UMCP: A sound and complete procedure for hierarchical task-network planning. In *Proceedings of AIPS*, 1994.

[Geib2001] C. W. Geib and R. P. Goldman. Plan Recognition in Intrusion Detection Systems. In *Proceedings of the Second DARPA Information Survivability Conference and Exposition*, 2001.

[Geib2002] Geib, C., Goldman, R., Requirements for Plan Recognition in Network Security Systems, Proceedings of the Recent Advances in Intrusion Detection conference2002.

[Frank2003] Frank M, Frans V. A formal description of tactical recognition [J]. *Information Fusion*, 2003, 4(1): 47-61.

[Han1999] K. Han and M. Veloso. Automated robot behavior recognition applied to robotic soccer. In *Proceedings of the Workshop on Team Behaviors and Plan Recognition*, 47–52, 1999.

[Harp2005]S. Harp, J. Gohde, Thomas Haigh, M. Boddy Automated Vulnerability Analysis Using AI Planning, 2005 AAAI Spring Symposium on AI for Homeland Security.

[Heinze1999]C. Heinze, S. Goss, and A. Pearce. Plan Recognition in Military Simulation: Incorporating Machine Learning with Intelligent Agents. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence, Workshop on Team Behaviour and Plan Recognition*, pages 53-63, 1999.

[Huang2003] Z. Huang, Y. Yang and X. Chen. An approach to plan recognition and retrieval for multi-agent systems. *Proceedings of AORC*, 2003.

[Jarvis2004] Jarvis, P.; Lunt, T. F.; Myers, K. L. Identifying terrorist activity with AI plan recognition technology. *National Conference on Artificial Intelligence*, AAAI Press; 2004.

[Kichkaylo2007] Kichkaylo, T., van Buskirk, C., Singh, S., Neema, H., Orosz, M., and Neches, R. Mixed-Initiative Planning for Space Exploration Missions, *Workshop on Moving Planning and Scheduling Systems into the Real World*, 2007.

[Marling2003] C Marling, M Tomko, M Gillen, D Alexander, D Case-based reasoning for planning and world modeling in the robocup small size league, *IJCAI Workshop on issues in designing physical agents* 2003.

[Porras1997] P. A. Porras and P. G. Neumann, EMERALD: Event Monitoring Enabling Responses to Anomalous Live Disturbances, *Proceedings of the National Information Systems Security Conference*, 1997, pp. 353-365.

[Quin2004]Qin X. and Lee W., Attack Plan Recognition and Prediction Using Causal Networks, *ACSAC-O4*, 370-379, 2004.

[Xiao2005] B. Xiao, W. Chen, Y. He, E. H.-M. Sha, An Active Detecting Method Against SYN Flooding Attack, *icpads*, vol. 1, pp.709-715, 11th International Conference on Parallel and Distributed Systems (ICPADS'05), 2005.