Planning to Discover and Counteract Attacks

Tatiana Kichkaylo, Tatyana Ryutov, Michael D. Orosz and Robert Neches
Information Sciences Institute
University of Southern California,
Marina del Rey, CA 90292, USA

Keywords: planning, attack recognition, intrusion detection

Received: September 4, 2009

A major function of a security analyst is to analyze collected intelligence looking for plans, associated events, or other evidence that may identify an adversary's intent. Armed with this knowledge, the analyst then develops potential responses (e.g., countermeasures) to deter the discovered plan or plans, weighs their strengths and weaknesses (e.g., collateral damage) and then makes a recommendation for action. Unfortunately, the collected intelligence is typically sparse and it is not possible for the analyst to initially discover the adversary's specific intent. Under these circumstances, the analyst is forced to look at the range of possible plans/actions an adversary may take. The full range of potential attack scenarios is too rich to generate manually. Its complexity also bars direct analysis and evaluation of the potential impact of alternative actions and countermeasures. To address these issues, we are developing a set of tools that exhibit the following features/capabilities:

- *Using available partial plan segments (referred to as snippets), construct multiple feasible scenarios/pathways that an adversary may take to reach an identifiable end goal*
- *Provide visual tools for exploring sets of possible scenarios under various observables, importance, and likelihood conditions, helping the analyst generate information probes, actions and countermeasures*
- *Compare the potential impact of alternative data probes, actions and countermeasures on an adversary's actions by assessing their discrimination/attack mitigation potential and possible side-effects*
- *Automatically suggest potential data probes, actions and countermeasures based on partial understanding of the adversary's plan and given observable activity*

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

Povzetek: Opisan je nabor orodij za napovedovanje napadov.

1 Introduction

The problem of detection, prevention and/or response to attacks is common to multiple domains. In the cyber community, a goal of an attacker may be to gain access to protected resources or to disrupt service to legitimate users. In counter-terrorism, an attack manifests itself as actions in the physical world. In robotic soccer, an attack is a sequence of actions by a team aimed at outmaneuvering the opponents and scoring goals.

Attacks can be classified along several dimensions. Different techniques and tools for attack detection and prevention/counteraction are applicable to 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 some cyber-attacks (e.g., Denial of Service) a reaction is 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 all vulnerabilities. However, for many of these legacy systems and physical infrastructures, it is possible to identify possible attack strategies for known weak points and design systems that detect such attacks. 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 appear either as 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. Therefore, it is desirable

to minimize the overhead of monitoring and limit disruption of legitimate activity.

In this paper we discuss a set of tools and approaches for identifying vulnerabilities, detecting potential attacks based on observables, and devising detection and/or countermeasures aimed at minimizing disruption of normal operations. We discuss the potential of such tools for human analysts for detection of slow attacks. In addition, we describe an initial prototype system developed for the Intelligence Advanced Research Projects Activity (IARPA) sponsored Proactive Intelligence (PAINT) project. The system 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 collection of partial plan segments (which we call *snippets*). The implemented system was targeted for the counter-terrorism domain. We also present network intrusion detection examples from as a new potential application for our system.

2 The problem

Recognizing attack plans is a key activity 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 [3] [5] are not valid in the security-related adversary recognition domain. In some domains (e.g., RoboCup soccer [16] , computer games [4] , and military simulations [15]) 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 [10] [11] :

Uncertainty and incompleteness. In some cases, the functional limitations of security sensors or a less-than optimum deployment pattern may increase uncertainty by hindering our ability to observe all attacker activities and steps. Moreover, we may have an incomplete knowledge of the possible attack plans.

Partially ordered plans. Often attackers' plans are flexible in the ordering of the plan's 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. For example, 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 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. In addition to actions directly contributing to achieving a goal, an attacker can take actions to mislead plan recognition, or to exploit some of its weaknesses.

Multiple weighed hypotheses. Ranking the possibilities is often more helpful than providing a single (possibly incorrect) explanation for observed activities. Say one observes scanning activity. While this action indicates a hacker is interested in a network, the observation of the action itself 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. The resulting delays could enable an actual attack. As [20] [6] 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 intrusion detection) and targeted countermeasures.

Side-effects of probes and countermeasures. Detection of stealthy attacks, such as malicious insider covert activity and terrorist preparations, can be complex and may require probes (i.e., actions designed to engender a response likely to provide additional useful information). To prevent a situation where a probe or a response action causes more damage than the actual attack, a system must allow analysts to reason about the likelihood and severity of an attack as well as about the effects of candidate alternative probe/response mechanisms.

Existing techniques for predicting adversary actions, include game theoretic approaches and game playing, adversarial planning, and pattern recognition. These provide partial/limited solutions at best. Data mining approaches have been used to find information that helps to detect attacks; however they suffer from a high false alarm rate and do not help analysts connect separate events. In order to minimize damage, new algorithms and tools for security analysts are needed to enable them to further analyze and correlate attack scenarios, make accurate situational assessments and quickly execute appropriate responses.

3 Modelling alternatives

The algorithmic core of our toolkit is a constraint-based planning system [18] . 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 properties (actor, equipment) of the action/state. Variables can be used as arguments to constraints. Constraints define tuples of values their arguments can take [8] . 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 [9] . A snippet captures a modification of a plan as an implication of the form "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. They also can 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 to 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. In the current system implementation, the user has sole responsibility for generating attack snippets. A possible direction for future research is to extract snippets from execution traces using machine learning techniques.

Consider the following example. Figure 1 shows a snippet describing the use of the *lpr attack*[1] 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. Note that the snippet imposes only a partial ordering on its steps, thus allowing for generation of partially ordered plans. 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 *secret.txt* 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.

In addition to describing goal/sub-goal relationships, snippets can describe necessary effects of actions (e.g., "if a port scan detection tool is installed and there is a port scan in progress, an alert will be generated").

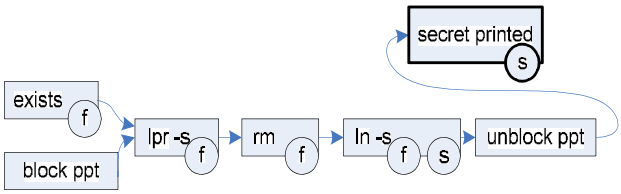


Figure 1: lpr attack snippet.

Representing domain knowledge in the form of snippets allows the system to mix-and-match various steps, making it possible to discover new combinations that could constitute possible new attacks. Further, by using automated planning algorithms instead of having human analysts manually compose scenarios, our approach provides better coverage of both attacks and normal operations. This in turn leads to higher-quality analysis of vulnerabilities, possible attacks and side-effects of countermeasures. The potential downside of this approach is that an operator could be overwhelmed with too many different plans to consider. However, we will discuss techniques which ameliorate that risk by reducing and filtering the set of all *possible* plans down to a more manageable set of *plausible* plans.

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 for (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., first steal a 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, 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 [21] , 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 subset of nodes) in an attack tree. An attack tree representation requires explicit chronological orderings of "exploits" (nodes). Since our approach

[1] To carry out the attack, an attacker requests printing of file *doc.txt* that he is authorized to access. User rights are checked by lpr command, access is granted, and the request is put into the printer queue. Then, before the printing actually starts, the attacker removes the printed file and replaces it by a link to file *secret.txt* he is not allowed to access. As a result, the latter file will eventually be printed and the attacker will be able to get the sensitive information.

supports snippet reuse and captures partial orderings of plans, it offers a more compact representation. Furthermore, attack trees do not facilitate adequate visualization support for analyzing different attack plans. After a set of plans has been constructed, our scheme enables a user/analyst to both 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) program.

4.1 Exploring details of a plan and comparing individual plans

Note that because of reuse, an action may serve as a sub-goal supporting 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: Scan 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 potential goals each action is contributing to. In addition, the user can compare individual plans with each other or explore common features among sets of plans.

Figure 2 shows a graphical interface for visually comparing two plans. Circles and squares represent high-level and atomic actions respectively. Each rhombus represents a constraint. Blue shapes describe elements common to both plans. Green and white shapes show elements present in one plan but not the other. Such comparisons help the analyst identify differentiating points between attacks and benign activity, as well as potential conditions for generating alarms and warnings.

The table interface (Figure 3) is used for comparing group of plans. The user selects a set of plans from the tree of generated plans. The table interface distributes different states and actions in the plans into separate columns. Each row describes one plan. This arrangement allows users to see common and distinguishing

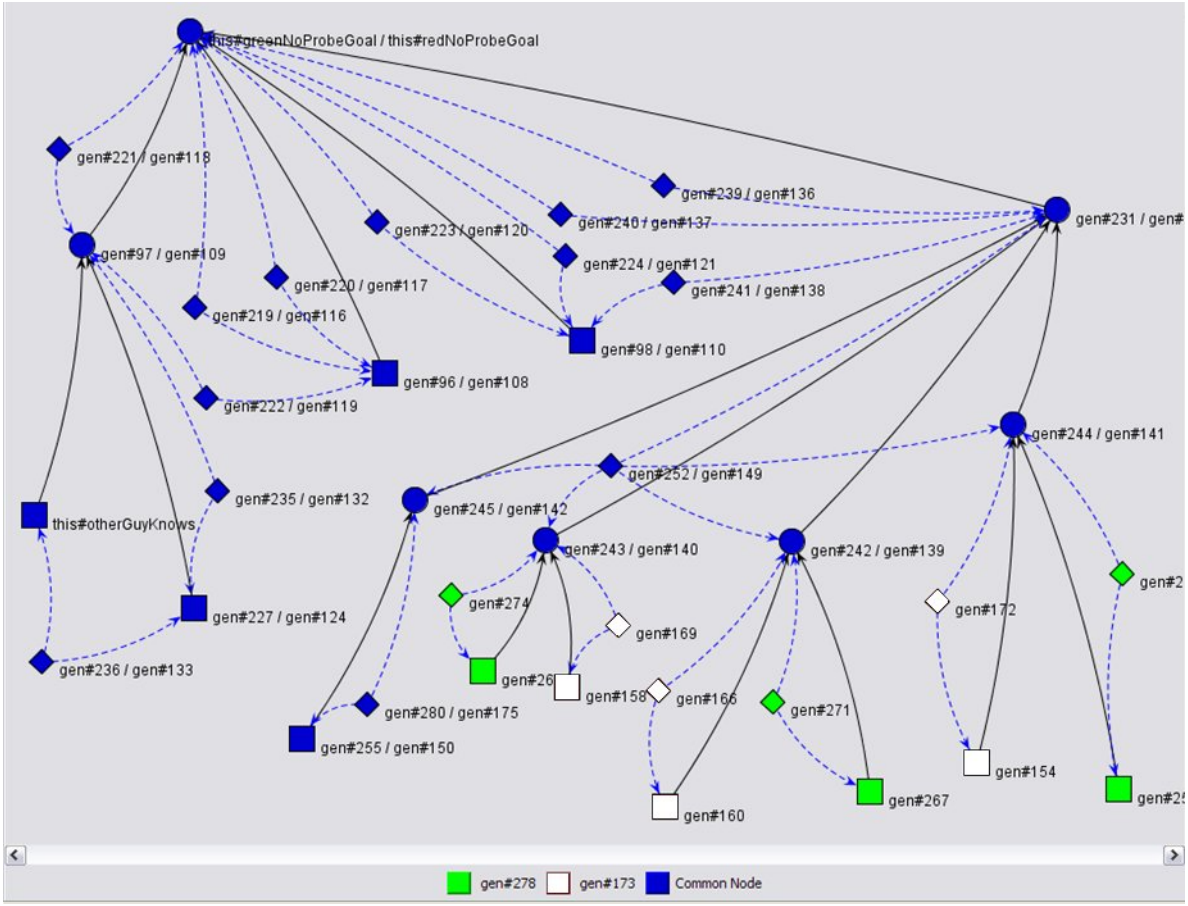


Figure 2: Plan comparison interface.

components of plans. The table can be sorted by any column, for example, to separate plans which contain a particular undesirable action from those that do not.

Any column of the table can be declared observable or hidden. The user can then select a set of rows and use the user interface (the Distinguish button) to determine if the selected plans can be distinguished from those not selected. This function allows one to quickly determine if a given set of observations is sufficient to determine the intent of the adversary regardless of which particular plan in the set is followed. Distinguishable plans are greyed out. In Figure 3, all plans are distinguishable given the chosen set of observables except the scenario identified as **gen#752**.

The graphical version of the group comparison interface (Figure 4) shows the goal-subgoal structure of the plans and visualizes the relative frequency of different actions in plans and presence of individual actions in distinguishable and non-distinguishable plans. Where this approach is helpful is in situations where the set of generated (and possible) plans can be divided into a *benign* group and a *nefarious* group and there exists a certain event (i.e., a node) which, if observable, clearly distinguishes the direction (benign vs. nefarious) the potential attacker is going regardless of the remaining nodes/events/steps in each plan. In such a situation, if a certain event is observed that clearly indicates the potential adversary is implementing one of the benign plans, there is no need to continue monitoring for additional intelligence/observables. On the other hand, if a certain event is observed that clearly indicates that the potential adversary is implementing one of nefarious plans, then further analysis is required to determine the path being taken in order to implement appropriate countermeasures.

4.2 Plan recognition

In addition to building plans from goals to observables, our approach could also be used bottom up (i.e., from observables to goals). This process provides alternative feasible explanations of observed activity (note: this has not yet been implemented, however). 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.txt*. Otherwise, this may indicate unusual, but non-malicious activity.

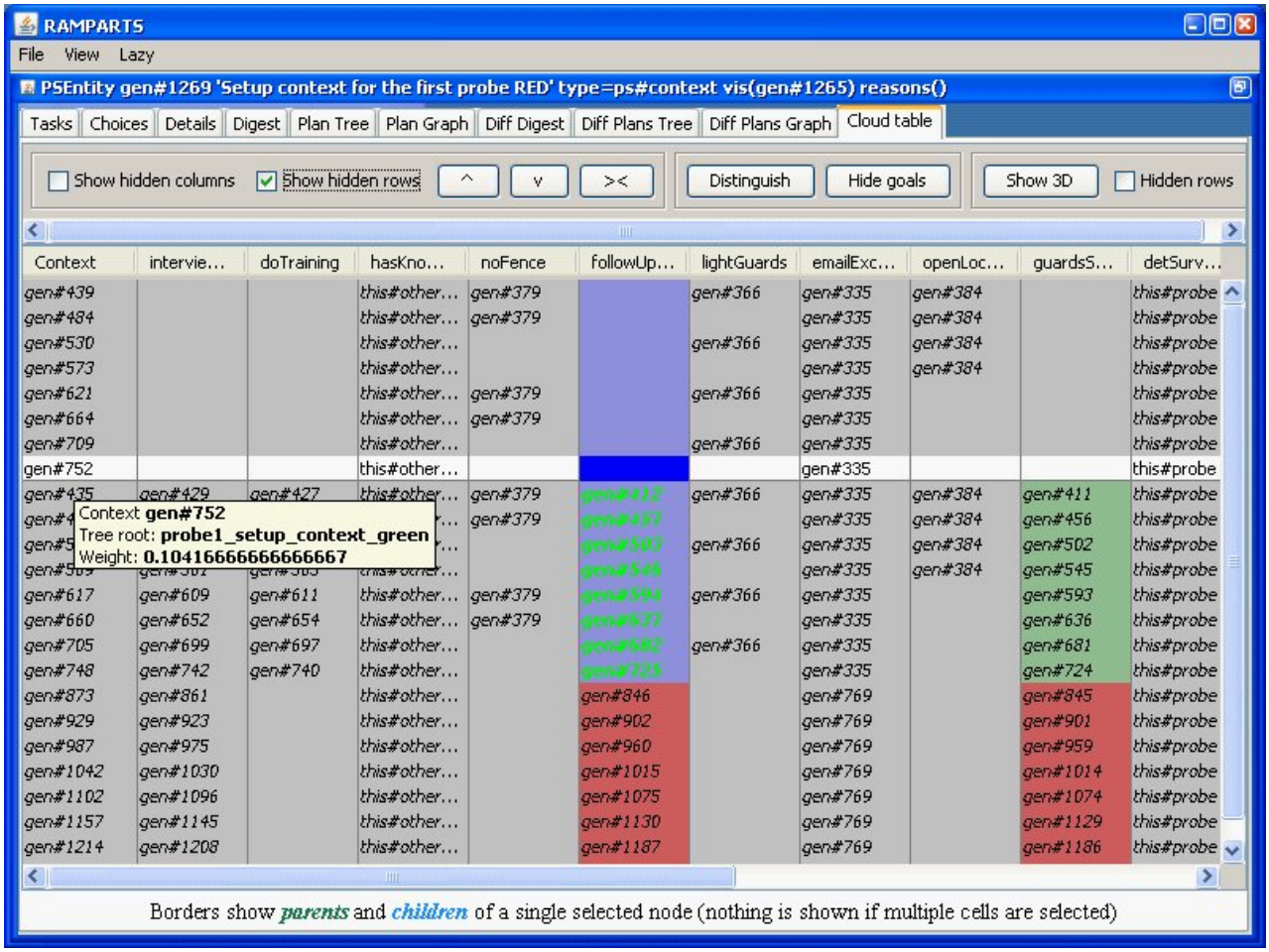
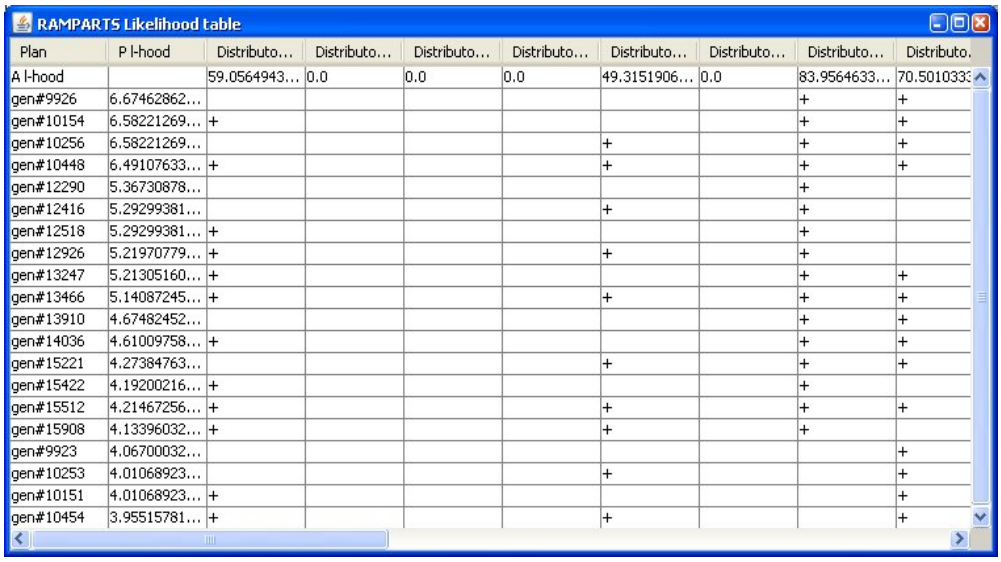


Figure 3: Table-based interface for comparing sets of plans



Plan	P l-hood	Distributo…	Distributo…	Distributo…	Distributo…	Distributo…	Distributo…	Distributo…	Distributo.
A l-hood		59.0564943…	0.0	0.0	0.0	49.3151906…	0.0	83.9564633…	70.501033
gen#9926	6.67462862…							+	+
gen#10154	6.58221269…	+						+	+
gen#10256	6.58221269…					+		+	+
gen#10448	6.49107633…	+				+		+	+
gen#12290	5.36730878…							+	
gen#12416	5.29299381…					+		+	
gen#12518	5.29299381…	+						+	
gen#12926	5.21970779…	+				+		+	
gen#13247	5.21305160…	+						+	+
gen#13466	5.14087245…	+				+		+	+
gen#13910	4.67482452…							+	+
gen#14036	4.61009758…	+						+	+
gen#15221	4.27384763…					+		+	+
gen#15422	4.19200216…	+						+	
gen#15512	4.21467256…	+				+		+	+
gen#15908	4.13396032…	+				+		+	
gen#9923	4.06700032…								+
gen#10253	4.01068923…					+			+
gen#10151	4.01068923…	+							+
gen#10454	3.95515781…	+				+			+

Figure 5: Interface for weighting plans and actions

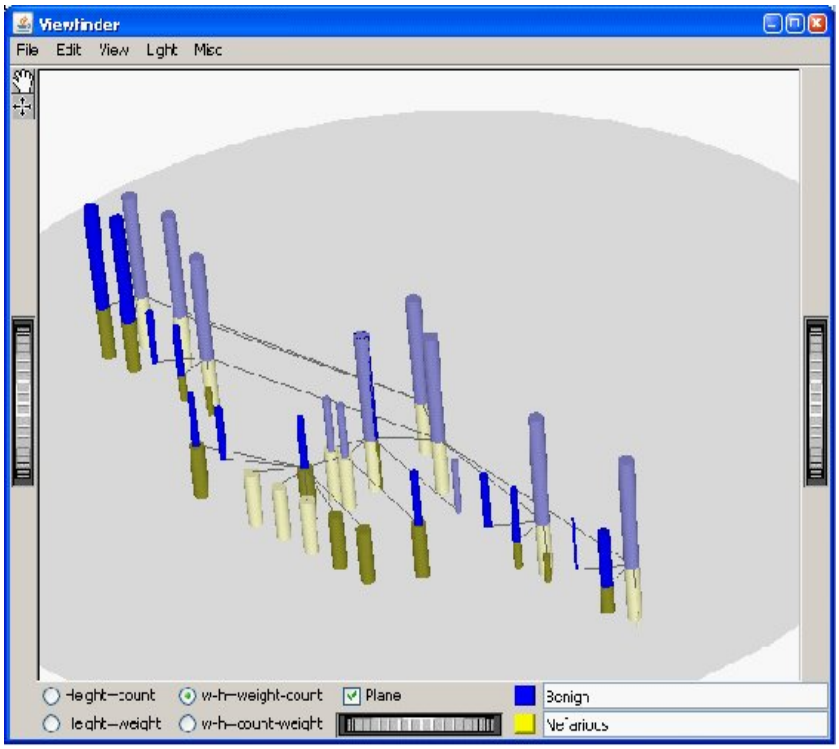


Figure 4: Graphical interface for comparing sets of plans

4.3 Weighting plans and actions

Typically, there are multiple paths that achieve the same result. In these cases, some plans are more preferable (and/or likely) than others. Figure 5 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 event has occurred unobserved, but a related event is observable. For instance, 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., trinoo or TFN).

The weight assigned to an action or observable state (specified in the columns) is the likelihood of it to actually happening, given the set of plans of which it is a part and likelihoods of these plans. The weight of a plan is the likelihood of it being carried out as opposed to alternative plans. The initial weights of plans may be assigned uniformly or using a heuristic function. For example, due to relative likelihood, the same attack plans against a server for an online banking service, might be

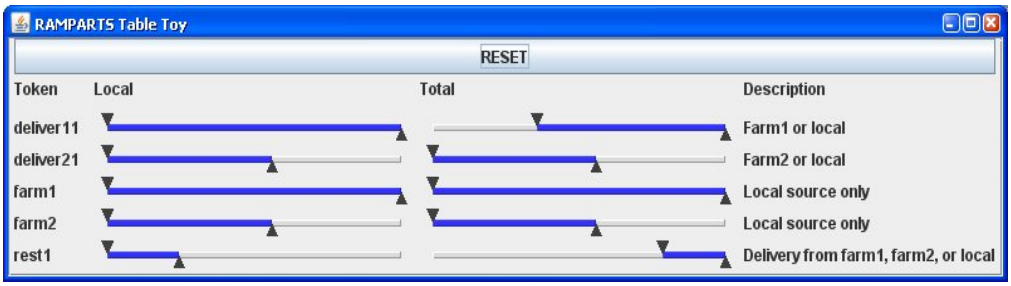


Figure 6: Slider interface for exploring likelihoods.

assigned higher values than they would be against a personal home server.

Users 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 it and decrease likelihood of plans that do not contain it. This, in turn, affects likelihoods of other actions and observables constituting these plans.

4.4 Likelihood propagation within a plan

The logical structure of tokens, constraints, and goal-sub-goal links can be leveraged to reason about the likelihood of violations within each plan. The user interface presented in Figure 6 provides access to such reasoning. This interface also allows the user to treat likelihoods qualitatively, rather than quantitatively. In practice, single numbers describing likelihoods are both hard to obtain and hard to understand. Our user interface instead represents the range of likelihoods from 0 (definitely false) to 1 (definitely true), using double-headed sliders. For example, a range of [0, 1] means "no information", while [0, 0.3] would mean "unlikely, but not yet completely ruled out".

Figure 6 illustrates a simple scenario, where a restaurant buys produce from two farms and delivers using two independent companies. The sliders describe the likelihood of food contamination at each stage of this transportation network. The **Local** column describes introduction of a contaminant at the given location, and the **Total** column corresponds to the presence of the contaminant resulting from all upstream sources. This interface allows the user to see the pruning effect of probes. For example, suppose food contamination has been detected at the restaurant (hence **rest1**'s **Total** slider set to the right), but local contamination at the site has been ruled out (hence **rest1**'s **Local** slider set to the left). These two observations do not give us any additional information. If we further observe that the produce delivered from **farm2** is clean (hence **deliver21**'s slide in the **Total** column being set to the left), the system can deduce that both **farm2** and the delivery company are clean. It thus follows that the contaminant was introduced somewhere along the second transportation chain, although it is not possible to say whether it happened at **farm1** or during transportation (which is why the corresponding sliders show the [0, 1] interval).

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.

False positives are a major problem in alert generation. One way to decrease them is to perform diagnostic actions upon initial detection of a potential problem. The reaction to these alert-triggered diagnostics can, in many cases, provide a clue as to whether or not the alert in question is indeed part of an attack. This allows adjusting the confidence in the original alert, thereby reducing false positives.

By analysing multiple ways to execute an attack and contrasting them with multiple ways to perform benign activities, our tools help to identify potential points for alert generation and intervention. For example, these tools may help an analyst identify an action (i.e., a probe) that could 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.

One challenge for this approach is selection of an action for diagnosis or intervention. Often, such actions may affect not only an attack but benign activity as well. For example, detection of SYN flooding attack may use active probing [23] that collects data on the delay between the server and the client. While timely and reliable, this intervention imposes processing and storage overhead. Techniques already described can be reapplied to address this class of problems.

Earlier we described tools that help an analyst analyze sets of attack plans and contrast them with benign activities. To do so, a scenario is seeded with a set of goals, and the system generates various potential instantiation of the scenario. This same technique can also be used to analyse potential effects of diagnostic actions or countermeasures. 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.

5 Related work

Huang et. al. [16] developed a technique for automated plan recognition in the field of RoboCup simulation soccer games. For each agent representing a player in the game, they translated 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 were identified using a statistical dependency test. These sequences were then retrieved and transformed into formalized plans. Finally the plans were refined as multi-agent teams adopted them. [13] applied Hidden Markov Models (HMMs) for recognizing opponent behaviours in RoboCup soccer simulations. HMM states corresponded to decomposed robot behavior. Uncertainty in recognizing behaviour was represented as probabilistic transitions between the states. [19] adopted case-based reasoning (CBR) for opponent modelling and planning players' strategies in RoboCup competitions. Solutions to problems were found by reusing solutions to similar problems encountered in the past.

None of these techniques are directly applicable to the security domain. In every case, their 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 domain, 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 [10] [7] [22] in the network security domain largely concentrates on correlation of observed actions and alerts produced by intrusion detection systems. [10] 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 a library of fully predefined attack plans and lacks support for reasoning about deceptive actions by an adversary. [7] proposed a method for detecting various steps of an intrusion scenario, casting it as a planning activity based on a declarative description of actions, goals, and plans. The method does not, however, provide additional information to distinguish between more vs. less plausible scenarios. As we noted earlier, this is a very important issue because the number of possible scenarios can be quite large. [1] extends the previous approach by providing the ability to rank possible scenarios. [22] 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, levels of importance (or priorities), and likelihood conditions. These aids are essential for helping analysts generate probes and countermeasures.

[14] proposed a method to analyse and test threats posed by malicious insiders. They used AI planning to automatically generate courses of action an adversary could 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, and no support for comparative analysis of several plans to achieve a given goal. [17] presented an application of plan recognition techniques to support analysts in processing national security alerts by automatically identifying the hostile intent behind them. The system needs a complete library of manually-generated attack templates, a daunting requirement.

6 RAMPARTS Prototype 1.0

As noted earlier, we have developed an initial prototype implementation of a number of the features and capabilities described in this paper. The Risk Analyses and Models of Plans of Attack for Recognizing Terrorist Schemes (RAMPARTS) project was funded by IARPA as part of the ProActive INTelligence (PAINT) program. The objective of this effort was to demonstrate an initial proof-of-concept by developing supporting infrastructure and implementing a subset of capabilities.

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 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 [2] 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.

Acknowledgement

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

References

[1] S. Benferhat, F. Autrel et F. Cuppens (2003). Enhanced Correlation in an Intrusion Detection Process. *In Proceedings of Second International Workshop Mathematical Methods, Models and Architectures for Computer Networks Security*.

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

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

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

[5] CALO (2003). Cognitive agent that learns and organizes, *http://calo.sri.com.*

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

[7] F. Cuppens, F. Autrel, A. Mi`ege and S. Benferhat (2002). Recognizing Malicious Intention in an Intrusion Detection Process. *Soft Computing Systems - Design, Management and Applications*, volume 87, 806–817.

[8] R. Dechter (2003). Constraint Processing. *Morgan Kaufmann Publishers Inc*.

[9] K. Erol, J. Hendler, and D. Nau (2004). UMCP: A sound and complete procedure for hierarchical task-network planning. *Proceedings of AIPS*.

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

[11] Geib, C., Goldman, R. (2002), Requirements for Plan Recognition in Network Security Systems, *Proceedings of the Recent Advances in Intrusion Detection conference.*

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

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

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

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

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

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

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

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

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

[21] Schneier, B., "Attack Trees." Dr Dobbs Journal. December 1999.

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

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