Using Planning for a Personalized Security Agent

Mark Roberts and **Adele E. Howe** and **Indrajit Ray** and **Malgorzata Urbanska**
Computer Science Dept., Colorado State University
Fort Collins, CO 80524, USA
email: {mroberts,howe,indrajit,urbanska}@cs.colostate.edu

Abstract

The average home computer user needs help in reducing the security risk of their home computer. We are working on an alternative approach from current home security software in which a software agent helps a user manage his/her security risk. Planning is integral to the design of this agent in several ways. First, planning can be used to make the underlying security model manageable by generating attack paths to identify vulnerabilities that are not a problem for a particular user/home computer. Second, planning can be used to identify interventions that can either avoid the vulnerability or mitigate the damage should it occur. In both cases, a central capability is that of generating alternative plans so as to find as many possible ways to trigger the vulnerability and to provide the user with options should the obvious not be acceptable. We describe our security model and our state-based approach to generating alternative plans. We show that the state-based approach can generate more diverse plans than a heuristic-based approach. However, the state-based approach sometimes generates this diversity with better quality at higher search cost.

Planning for a Personalized Security Agent

The average home computer user has little understanding of security and limited time to become educated and to take action to protect their computers. Current security approaches, e.g., anti-virus software, OS patches, malware detectors, require time, money and knowledge to be effectively used. Moreover, the software is designed to be one-size-fits-all which does not accommodate the different needs and preferences that have been observed in studies of home users (Howe et al. 2012). For example, a study of 31 undergraduates hypothetically installing software on a friend's machine concluded that many participants considered file sharing software to be indispensable, even accounting for the risks (Good et al. 2005).

Our research project takes a different approach: develop an agent that can monitor security related activities on a home computer and propose interventions to the user to avoid or recover from security threats. The agent will be personalized to the preferences and experience of the user as well as to the configuration of the home computer. The security model underlying the agent is being developed based on psychological studies to identify factors that influence a home user's decisions about security threats (e.g., perceptions of risk and threats) (Byrne et al. 2012).

A home computer security agent would need to perform all of the following tasks: monitor the user/system for new behavior/state, incorporate new security knowledge from a common security database, adapt to newly installed software, prioritize its actions so as to block the most critical vulnerabilities first, offer suggestions of actions to the user to support achieving his/her goals while not breaching security/privacy, and intervene independently to the extent that the user's trust allows. Several of these tasks involve planning. In this paper, we describe how planning has been used for security, how we have started to extend existing planning techniques to support the security agent and our future plans for further extensions.

The Personalized Security Agent

The two core goals behind our security agent are that its design should be motivated and supported by psychological studies of users and that its behavior should be personalized to a particular user. In support of these goals, we have developed a new security model that is based in part on studies from the literature and part on our on-going studies.

The Security Model: A Personalized Attack Graph

Researchers have modeled security for networked systems using *attack graphs* (Phillips and Swiler 1998; Sheyner et al. 2002) and *attack trees* (Moore, Ellison, and Linger 2001; Dewri et al. 2007). These models capture dependencies among different system attributes such as vulnerabilities and network connectivity and facilitate security risk analysis and management. But these models focus on networked systems rather than home computer users. We developed the Personalized Attack Graph (PAG) security model to characterize the ways that a *home system* can be compromised and add actions for the user as well as the attacker. The PAG is a state-transition system that is instantiated with the state of a particular home computer and user. Figure 1 shows a PAG for a Denial of Service (DoS) exploit that is a subtree of a much larger PAG with 7 exploits, 25 user actions, 38 system states or actions (of which 11 are system vulnerabilities), and 19 attack actions. A complete PAG consists of a set of many such exploit subtrees and paths from leaf nodes to the root represent potential attack paths.

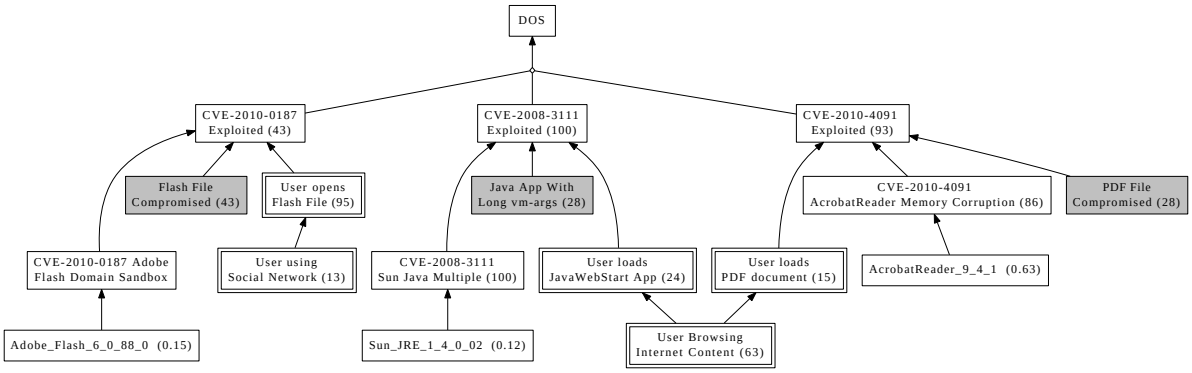

Figure 1: The Denial of Service personalized attack graph. The probabilities for nodes are given in parentheses.

Using Planning to Identify Attack Scenarios

Previous research has shown how a planner can help analysts identify actions that lead to security breaches. Boddy et al. (2005) built a mixed initiative planning system that could identify potential vulnerabilities and countermeasures in cyber security for large organizations. Their PDDL (Fox and Long 2003) domain model allowed them to produce "insider subversion" plans of 40–60 steps. Their work showed that automated planning could find novel attack scenarios.

Attack Graphs quickly become large, computationally expensive to analyze and hard for human analysts to understand. Ghosh and Ghosh (2010) reduce the complexity of instantiating an attack graph by iteratively applying a planner to eliminate unreachable attack scenarios. They use a PDDL model similar to Boddy et al. and generate minimal attack paths. To identify multiple paths that lead to the same scenario, they modified the domain model by eliminating each path (that is, commenting out an action or predicate) as it was discovered. Obes et al. (2010) construct a large PDDL model (1800 actions) from an attack graph and integrate the planner into a penetration testing tool. Although they found an exponential increase in computation time as the number of machines modeled increased, the time was still just 25 seconds to generate a plan involving 480 machines.

In codifying the PAG, we followed a similar approach to the prior work by translating the PAG into PDDL (Roberts et al. 2011). Figure 2 shows portions of the domain and problem description of the leftmost subtree given in Figure 1. The full domain used for the experiments later contains 15 actions, 5 predicates, 18 initial objects, and 12 initial predicates. Plans in this domain highlight potential paths that can be exploited, which allows other portions of the agent to prune and personalize the PAG and remove exploits that cannot happen for a given system/user.

```
(define (domain attack-graph)
    (:requirements :strips :equality
       :disjunctive-preconditions :typing )
    (:types    Object   Action   ExploitState   software )
    (:predicates   (action-observed ?Action - Action)
      (action-taken ?Action - Action)
      (exploit-occurred  ?Exploit - ExploitState)
      (software-installed  ?Software - software) )
    (:action AttackAction_FlashFileCompromised_5
      :parameters ( ?Action5 - Action )
      :precondition (and (action-observed ?Action5 )
               (= ?Action5 AttackAction_FlashFileCompromised_5 ) )
      :effect
        (and (action-taken AttackAction_FlashFileCompromised_5) )))

(define (problem attack-graph-problem1)
    (:domain attack-graph)
    (:objects
      Exploit_DenialOfService_1 - ExploitState
      AttackAction_FlashFileCompromised_5 - Action
      UserAction_UserUsingSocialMedia_7 - Action
      ObeservedState_CVE_2010_0187_Exploited_2 - Action
      UserAction_UserOpensFlashFile_6 - Action
    )
    (:init
      (action-observed AttackAction_FlashFileCompromised_5 )
      (action-observed UserAction_UserUsingSocialMedia_7 )
      (software-installed Adobe_Flash_6_0_88_0)
    )
    (:goal
      (and (exploit-occurred Exploit_DenialOfService_1) )))

PLAN1
UserAction_UserUsingSocialMedia_7
UserAction_UserOpensFlashFile_6
ObeservedState_CVE_2010_0187_Exploited_2
Exploit_DenialOfService_1

PLAN2
UserAction_UserBrowsingInternetContent_13
AttackAction_PDFCompromised_20
UserAction_UserLoadsPDFDocument_21
ObeservedState_CVE_2010_4091_OS_Exploited_17
Exploit_DenialOfService_1

PLAN3
UserAction_UserBrowsingInternetContent_13
AttackAction_JavaAppWithLongVMArgument_11
UserAction_UserStartsJavaWebstartApplication_12
ObeservedState_CVE_2008_3111_SunJavaMultiple_Exploited_8
Exploit_DenialOfService_1
```

Figure 2: Partial PDDL domain and problem descriptions from the CVE-2010-0187 subtree of the DoS exploit followed by the three solutions found by ITA*.

Generating Alternative Plans

In our agent, planning is used to prune the PAG to make it more computationally manageable and to identify interventions (key points to disrupt the plan). Because vulnerabilities can be exploited in many ways, it is essential to be able to generate alternative plans. The prior research in using classical planning to generate attack paths (Boddy et al. 2005; Ghosh and Ghosh 2012; Obes, Sarraute, and Richarte 2010) either generated a single path or iteratively modified the domain/problem description to influence the planner to produce new attack paths. We modify the algorithm.

Alternatives, Plan Diversity, and Plan Sets

A first step in generating alternatives is to define a metric that quantifies the differences in plans. In earlier work, Srivastava et al. (2007) explore how to generate diverse plans in a constraint-based planner. They use actions, states, and causal links to assess differences in plan diversity and find that using an action-based distance usually provides diverse enough plans. More recently, Talamadupula et al. looked at generating plans for execution with the best net benefit in a partial satisfaction planning framework (Talamadupula et al. 2010; Schermerhorn et al. 2009). While this work is not strictly about generating alternatives, it does examine how to evaluate multiple plans with respect to usefulness to a user.

Given a plan π, and a set of plans Π, and a distance metric D, Coman and Muñoz-Avila (2011) define:

$$\text{RelativeDiversity}(\pi, \Pi) = \frac{\sum_{\pi,\pi' \in \Pi} \text{D}(\pi, \pi')}{|\Pi|} \quad (1)$$

During planning, this diversity metric is used in a weighted evaluation function that is maximized by the planner:

$$h_{\text{new}}(\pi, \Pi) = (1 - \alpha) h_{\text{diversity}}(\pi, \Pi) - \alpha h(\pi), \quad (2)$$

where $h_{\text{diversity}}(\pi, \Pi) = \text{RelativeDiversity}(\pi_{\text{relax}}, \Pi)$, π_{relax} is the relaxed plan that discards delete effects, and α balances exploitation of the original heuristic and exploration of more diverse plans. The original heuristic is subtracted because the authors want to minimize h but maximize the diversity.

The general distance metric, D, in Equation 1 allows any quantitative or qualitative distance metric to be substituted in guiding search toward generating diverse plans. In particular, as the authors point out and show, D can contain domain-specific information that may be challenging to incorporate into the domain model.

Many of the planners from the recent International Planning Competition (IPC-2011) find better solutions over time, e.g., LAMA (Richter and Westphal 2010), which uses a multi-queue local WA* search, and CBP (Fuentetaja 2011), which uses branch-and-bound search. Such anytime planning algorithms and satisfying planners could be seen as generating alternative solutions; each new solution is effectively an alternative that improves over the last. Indeed, nearly any planner could be modified to produce alternative solutions, though they may still have a bias toward progressively better solutions under the current IPC metrics for comparing planners. We base our approach on using a Tabu list with restarts; Richter et al. (2010) recently showed that a weighted-A* algorithm performs better with restarts.

In general, all these approaches maximize a "Reward" or "Diversity" metric after subtracting the plan cost. This can have multiple drawbacks. If action costs swamp the reward, then the search trajectory is dominated by the action cost. If two plans have wildly different action costs AND different rewards, then it is not clear how to select one over the other. The state-based approach we propose avoids these issues.

Our Approach: ITA*

We implemented our own version of A* on top of the LAMA-2008 planner. LAMA is based on FastDownward (Helmert 2006), and has been the best planner for two IPCs. For our search, we turned off landmarks but continued to use preferred operators. When the search finds a goal node, it caches each $(state, operator)$ pair that is on the path to the solution into a data structure called the `TabuStateList`. It then calls `restartSearch()`, which clears the open and closed lists, resets the search counts, and puts the initial state back into the open list. When the search encounters a $(state, operator)$ pair that is already in the `TabuStateList`, it adds the g-value of that node to an arbitrarily high value, `gConstant`, (such as 1000) rather than the actual g-value. This ensures that those nodes are prioritized last in the A* open list and thus are very unlikely to be pulled off next. We call this planner *ITA**, for Iterated Tabu A*.

We will show that ITA* supports generating diverse solutions. However, it sometimes generates longer solutions that may be padded with spurious actions as a result of the state-based approach. We explore this trade-off in the evaluation. Another issue is that this change, intentionally, invalidates the optimality of the search algorithm. However, completeness and soundness remain unaffected (space limitations prohibit us from sketching these proofs). Interestingly, the mechanism we use to defer search on already found solutions looks almost like an induced g-value plateau (Benton et al. 2010); we hope to explore this in the future.

The Comparison Planner: DivA*

We also implemented our own version of the planner by Coman and Muñoz-Avila (2011) . To keep the comparison fair, we maintain the use of A* and change only the heuristic used during search. Similar to their planner, we leverage the $\text{D}_{stability}$ metric (Fox et al. 2006). Let π_1 and π_2 be the operator lists for two plans we want to compare. Then $\text{D}_{stability} = |(\pi_1 \setminus \pi_2)| + |(\pi_2 \setminus \pi_1)|$. The symmetry of performing the set difference in both directions is important in order to account for plans that may be reversed.

We also set $\alpha = 0.7$ because this was a setting used in (Coman and Munoz-Avila 2011). But we made some changes to their approach. First, we used the current solution to compare to the set of existing solutions found so far rather than π_{relax}. Also, we minimize the heuristic by subtracting D from a large constant and then adding back the original heuristic. These changes allowed us to maintain as much similarity between the two planners as possible. We call this planner *DivA**, for DiversityA*.

Results on the Small PAG

ITA* can find unique solutions (alternative attack paths) for the small PAG domain generated from Figure 1. Figure 2 (bottom) shows the three solutions found by ITA* in a single planning episode, which correspond to the three subtrees of the PAG. It found these solutions with 53, 46, and 25 node expansions. The plan lengths for these solutions increase from 4 to 5, so ITA* overcomes a key problem identified by Ghosh and Ghosh (2010), that the planner always finds the shortest path. For this problem, DivA* found a single path but was unable to find new plans beyond that. We suspect

that either the weighted heuristic or the diversity measure is not well suited to small plans and domains such as this.

Results on Benchmark Domains

To show how this approach generalizes to other domains, and to support direct comparison to previous work, we also examined how well we can generate alternatives for domains from some IPC benchmarks. We compare to the three IPC3 domains used in (Coman and Munoz-Avila 2011): DriverLog, Depot, and Rover. We also included the cybersec problems from IPC-2008 because these problems are the first security application that identified the gap in classical planners not easily producing alternative solutions. Finally, we include the seq-opt transport problems from IPC-2011 because it is a newer "logistics" style benchmark, makes sense in a mixed-initiative setting, and includes action costs that we can use to examine plan quality in later experiments. We found that the planners could solve more problems from the seq-opt track than the seq-sat track for the transport problems. For our comparison to DivA*, we examine the search cost of ITA*, as well as the characteristics of the alternatives that are found, and how the solutions compare in quality.

Each planner was given 2 hours on a single processor with 4 GB memory. The processes were run on 48 dual quad-core Xeon 5450, 16 GB machines. The planners were run until they found 10 solutions, or exhausted memory or time.

For which domains did ITA* produce unique solutions? Table 1 shows solution counts for ITA* (top) and DivA* (bottom) for each domain. After the domain name, the first column displays how many problems the planner solved. The next column shows how many solutions to those problems the planner generated. Some solutions, while unique, were simply the original solution plus some spurious actions; we called these solutions *padded*. We also found that some of the solutions were simply permutations of each other. The 'Unique' column shows how many solutions exist after removing all padded and permuted solutions across all solutions in a problem. To give a sense of how frequently each planner finds duplicate solutions, we compared the *first* found solution to the remaining solutions found and show the number of padded ('Pad' column) and permuted ('Perm' column) solutions. The last column ('Remain') indicates how many solutions differed meaningfully from that first solution found. Note that these remaining solutions could still be duplicates of each other.

Clearly, ITA* produces alternative solutions though many solutions are duplicates. The IPC3 results show that ITA* produces permutations and some padded solutions. However, in the transport domain ITA* produces a unique solution every time. DivA* solves less problems, but shows a similar result of producing permuted and padded solutions. In the newer cybersec and transport domains, it produces many more permutations. However, as we will show later, DivA* can produce solutions that are not found by ITA*, suggesting that the approaches are complementary.

				Compared to First		
	n	Solns	Unique	Perm	Pad	Remain
Rover	7	70	40	17	4	42
Depot	8	63	52	2	2	51
DriverLog	9	80	49	10	2	59
cybersec	25	244	158	3	42	174
transport	20	200	200	0	0	180
Rover	5	50	7	39	0	6
Depot	4	39	13	10	0	25
DriverLog	8	80	31	22	5	45
cybersec	24	240	42	123	2	91
transport	20	200	91	37	2	141

Table 1: Solution counts for IPC3, cybersec, and transport problems on ITA* (top) and DivA* (bottom).

	DivA*		ITA*		Compare to DivA*		
Domain	Solns	Unq	Solns	Unq	Perm	Pad	Alt
Rover	50	7	70	40	5	0	35
Depot	39	13	63	52	5	0	47
DriverLog	80	31	80	49	11	0	38
cybersec	240	42	244	158	27	1	130
transport	200	91	200	200	18	2	180

Table 2: Assessing solution overlap of DivA* and ITA* for the IPC3 (top) and cybersec (bottom) problems.

Domain	Problem	Min	Max	μ	σ
Depot	01	22	153	86.9	51.9
Depot	02	50	135	95.7	32.9
Depot	07	606	4760	1857.8	1225.9
Depot	13	1017	3143	1748.3	697.6
DriverLog	01	10	28	21.5	6.3
DriverLog	03	62	240	176.3	66.9
DriverLog	11	102	5949	3105.3	2154.1
DriverLog	13	1531	35139	15614.7	13420.3
Depot	01	13	124	24.9	33.1
Depot	02	19	974	130.7	281.9
Depot	07	119	12042	1926.6	3397.5
Depot	13	31	110987	13999.4	34594.1
DriverLog	01	8	26	10.2	5.4
DriverLog	03	14	233	57.2	78.2
DriverLog	11	34	8242	907.2	2448.6
DriverLog	13	39	97774	11156.0	29117.6

Table 3: Search cost for ITA* (top) and DivA* (bottom) on selected IPC3 problems that both planners solved.

How closely do the solutions from ITA* match those of DivA*? To assess solution diversity, we examined the overlap of the plans produced by both planners. Table 2 repeats solution counts from Table 1 and shows solution overlap between the two planners in the last three columns. We took each unique DivA* solution and compared it to each unique ITA* solution (that is, we used only those solutions from the 'Unq' columns). Solutions that were the same or were permutations of each other were counted in the 'Perm' column. Similarly, a padded ITA* solution is one that contained any DivA* solution plus extra actions. Remaining so-

		DivA*		ITA*	
Domain	Problem	n	Diversity	n	Diversity
Depot	01	10	2	10	7
Depot	02	10	12	10	10
Depot	07	10	8	10	10
Depot	13	9	12	10	14
DriverLog	03	10	1	10	2
DriverLog	07	10	5	10	17
DriverLog	09	10	32	10	22
DriverLog	10	10	14	10	20
DriverLog	11	10	12	10	24
DriverLog	13	10	31	9	29
Rover	05	10	6	10	10

Table 4: Comparing the diversity using $D_{stability}$ of DivA* and ITA* solutions from the IPC3 and cybersec problems.

lutions were counted in the 'Alt' column. Clearly, ITA* does produce alternatives to those given by DivA*.

For runs that produce unique solutions, how does search cost change for each new solution? As a proxy for search cost, we examined the time to solution in the number of nodes examined. We show a representative sample in Table 3. The results demonstrate that the search cost can vary greatly between problems as well as between restarts of both planners. DivA* tends to find plans more quickly. This was a surprising result because one might think that the α weight would lead search into non-productive parts of the search tree since it effectively lowers the usefulness of the original heuristic estimate. The variance on cost can be quite high which suggests that both planners can be leveraged to produce even more diverse plan sets.

How do the solutions compare in terms of diversity? Finally, we examined how the two planners compared according the Diversity (Coman and Munoz-Avila 2011) and $D_{stability}$ (Fox et al. 2006) metrics:

$$\text{Diversity}(\Pi) = \frac{\sum_{\pi,\pi' \in \Pi} \text{D}_{stability}(\pi, \pi')}{\frac{|\Pi| \times (|\Pi|-1)}{2}} \quad (3)$$

Table 4 shows selected problems from the domains; in particular, we do not show any problem for which Diversity(Π) was zero using DivA* (ITA* had no problems for which this was true). In the older IPC3 problems, ITA* has a better (higher) diversity metric than DivA* for all but three problems. ITA* dominates with much higher diversity in the cybersec and transport domains (not shown for space reasons).

How do the solutions compare in terms of plan quality? We examined the quality of the plans as measured by VAL (Strathclyde Planning Group 2010); VAL is the program used to validate solutions in the IPCs. We assessed the transport task because ITA* found 10 unique solutions for each problem and because this domain uses action costs. The goal is to move packages while *minimizing* the total cost of the plan, where the cost is a sum of the road distances plus 1 for each unload/load action.

	DivA*				ITA*			
	n	min	μ	σ	n	min	μ	σ
02	3	271	323.7	41.0	10	271	377.6	56.3
04	2	922	1148.0	226.0	10	921	1074.5	139.7
06	2	406	411.0	5.0	10	416	511.1	81.5
08	6	614	836.0	147.5	10	459	668.1	144.7
10	6	733	899.7	136.1	10	700	816.1	88.5
12	4	2490	3253.5	447.4	10	2490	3071.4	394.1
14	8	900	1120.2	258.1	10	551	861.5	155.2
16	10	2093	2601.4	319.8	10	1878	2387.9	273.6
18	2	1367	1506.0	139.0	10	1010	1495.0	189.4
20	4	537	597.5	43.2	10	509	640.0	101.0

Table 5: Comparing the quality of DivA* and ITA* solutions from the transport problems.

Table 5 summarizes the results of running VAL on the solutions discovered by DivA* and ITA* for selected problems in transport. For each planner, we list the number of solutions found, the minimum quality value obtained, the average and standard deviation of the quality of the solutions. The 'Min' and 'μ' columns reveal that neither planner dominates, again suggesting that these two approaches are complimentary.

Future Work

Our goal is to build a security agent to help home computer users. Our future work will focus on improving alternative plan generation, employing planning for security interventions, and incorporating plan quality metrics during the planning process.

Better Alternative Plan Generation ITA* lacks some of the sophisticated enhancements present in other planners and is somewhat limited in what it can accomplish. We would like to improve the performance of ITA* by incorporating many of the enhancements from recent state-of-the-art planning techniques (deferred evaluation, landmarks, other heuristics, etc.). Similarly we could modify other planners to output alternative solutions. This would allow us to make a broader comparison concerning search efficiency.

Backjumping / Backtracking It would be straightforward to modify the algorithm to perform chronological backtracking or heuristically-guided backjumping instead of iterated restarts. For the motivating security domain, we believe this might increase the search cost without producing many more solutions since each solution tends to be isolated from other solutions. But for other domains, it may lead to lower search cost for alternative solutions.

Planning for Interventions In the security domain, we identified paths leading to exploits. But blocking these paths is central to a successful security agent. The next step is to provide the planner with intervention actions that can stop a

viable attack path by negating states that lead to an exploit. So its preconditions are any states that are on the path to an exploit and its effect negates that specific state. Examples of intervention actions include: cleaning attachments of a particular type, updating a software version or installing a firewall. Good intervention actions are those that block multiple vulnerabilities, are low cost to execute and do not interfere with the user's needs or preferences.

Incorporating Negations During Search The intervention action negates important details in the problem space; thus, the intervention planner will need to reason about negative effects. A significant issue in searching for intervention plans is that many state-of-the-art heuristics in the planning literature ignore the delete list to estimate the value of potential solutions; this is alternatively called ignoring negative effects or ignoring negations. Although multi-valued encodings do capture cycles of negative interactions between actions (Helmert 2006), the translation can still ignore some negative effects. But including all negations can be computationally intractable. Richter examined one way to include negated effects into landmarks (Richter and Westphal 2010). She found that recompiling the planning task using the Π^m compilation (Haslum 2009) – a compilation that increasingly includes delete effects by making them add effects using a set of additional actions – generated much more accurate heuristics and decreased search cost. However, she also found that the overall computational cost for searching with this improved heuristic was unjustified (Richter 2010). So we will further study the trade-off of using the computationally tractable relaxed heuristic against including negations that improve search and look for alternative ways to integrate negations. We plan to explore both landmarks (Richter and Westphal 2010) and londexes (Chen, Zhao, and Zhang 2007) as ways to selectively include negations.

Disjunctive Goal Search The search for both attacks and interventions requires that we identify a valid attack path and then seek to negate that path. This can be represented as disjunctive goals in the form of $(A \vee (\neg A \wedge (I_1 \vee I_2 \vee I_3)))$, where A is some attack goal and I_1, I_2, and I_3 are interventions that eliminate the attack and are discovered and added during search. If the disjunctive goals are added at top level of the iterated search, we can then search for alternatives and interventions within the same planning episode and rely on ITA* to produce alternative solutions automatically. Although disjunctive goals are semantically equivalent to the more standard conjunctive representation, they may be computationally different in the search for alternatives.

Plan Quality Since each alternative solution represents a potential security breach/intervention, the security agent needs to consider as many viable alternatives as it can. But it will still be critical to rank the plans so as to focus the resources of the agent to the most important security concerns. For the purpose of this discussion, we consolidate under the term of plan quality a number of measures that can be quantified. The PAG model will be extended to include at least four quality metrics: the likelihood of an attack occurring, the cost of damage, the cost of intervening, and the utility of the specific activity to the user. We believe that the final implementation of ITA* would be more effective by including quality information during search.

Many planners examine plan cost/quality while searching using a modified heuristic to include this information; we plan to leverage such work whenever possible. But most existing planning systems generate solutions that minimize (or maximize) the solution according to a *single* criterion. We are interested in finding alternatives that may vary with respect to *multiple* evaluation metrics that may not be easily comparable. In the security domain, we want to see paths leading to attacks regardless of whether such paths may be more costly or less probable than other paths. The question that we have with respect to including metrics during search is how we can guide the search to produce alternatives that force variation in the metric rather than minimization of the metric. A well-understood way to manage multiple objectives that may have vastly different criteria is to generate solutions along a pareto-optimal front as opposed to a single metric (Ehrgott 2008). We also plan to explore this multi-objective search technique in comparing and searching for alternatives.

Conclusions

As our first step in applying planning to our security domain, we have translated the PAG into PDDL and generated alternative attack paths using ITA*. We have shown that ITA* produces unique solutions to both the PAG example and some IPC benchmarks. When compared to DivA*, ITA* can find alternative solutions that haven't already been discovered. The search cost for both planners varies as search progresses. For the transport domain from IPC-2011, ITA* consistently produces lower quality solutions than DivA*, but it can occasionally find better solutions. Finally, ITA* is well suited to produce alternative solutions for the security application that was our original motivation as well as the security domain that identified this open question; ITA* identified an additional 130 alternative solutions in the cybersec domain (see Table 2).

The home computer security agent application offers both opportunities and challenges to extending classical planning. Planning is integral to the construction and usage of a security model, the Personalized Attack Graph. Planning is also integral to the decision making the agent will need to do to avoid or mitigate security threats.

Acknowledgments

This material is based upon work supported by the National Science Foundation under Grant No. 0905232. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation. We also thank the reviewers of this paper for their insightful comments that helped improve the paper.

References

Benton, J.; Talamadupula, K.; Eyerich, P.; Mattmüller, R.; and Kambhampati, S. 2010. G-value plateaus: A challenge for planning. In *Proc. of the 20th Int'l Conf. on Automated Planning and Scheduling (ICAPS-10). Toronto, Ontario, Canada, May 12-16*, 259–262.

Boddy, M.; Gohde, J.; Haigh, J. T.; and Harp, S. 2005. Course of action generation for cyber security using classical planning. In *Proc. of the 15th Int'l Conf. on Automated Planning and Scheduling (ICAPS-05). Monterey, CA, USA. June 5-10*, 12–21.

Byrne, Z.; Weidert, J.; Liff, J.; Horvath, M.; Smith, C.; Howe, A.; and Ray, I. 2012. Perceptions of Internet threats: Behavioral intent to click again. In *Proc. of the 27th Annual Society for Industrial and Organizational Psychology (SIOP) Conference*.

Chen, Y.; Zhao, X.; and Zhang, W. 2007. Long distance mutual exclusion for propositional planning. In *Proc. of the 20th Int'l Joint Conf. on Artificial Intelligence (IJCAI-07). Hyderabad, India. January 6-12*, 1840–1845.

Coman, A., and Munoz-Avila, H. 2011. Generating diverse plans using quantitative and qualitative plan distance metrics. In *Proc. of the 25th AAAI Conference on Artificial Intelligence (AAAI-11) AAAI Press, Menlo Park, CA. San Francisco, CA, USA. August 7-11*, 946–951.

Dewri, R.; Poolsappasit, N.; Ray, I.; and Whitley, D. 2007. Optimal security hardening using multi-objective optimization on attack tree models of networks. In *Proc. of the 14th ACM conference on Computer and communications security (CCS-07). Alexandria, Virginia, USA*, 204–213. ACM.

Ehrgott, M. 2008. Multiobjective optimization. *AI Magazine* 29:47–57.

Fox, M., and Long, D. 2003. PDDL2.1: An extension to PDDL for expressing temporal planning domains. *Journal of Artificial Intelligence Research* 20:61–124.

Fox, M.; Gerevini, A.; Long, D.; and Serina, I. 2006. Plan stability: Replanning versus plan repair. In *Proc. of the 16th Int'l Conf. on Automated Planning and Scheduling (ICAPS-06). Cumbria, UK. June 6-10*, 212–221.

Fuentetaja, R. 2011. The CBP planner. In Garcia-Olaya, A.; Jimenez, S.; and Lopez, C. L., eds., *The 2011 International Planning Competition: Description of Participating Planners, Deterministic Track*, 21–24.

Ghosh, N., and Ghosh, S. 2012. A planner-based approach to generate and analyze minimal attack graph. *International Journal of Applied Intelligence* 36(2):369–390. Published online: 25 November 2010.

Good, N.; Dhamija, R.; Grossklags, J.; Thaw, D.; Aronowitz, S.; Mulligan, D.; and Konstan, J. 2005. Stopping spyware at the gate: A user study of privacy, notice and spyware. In *Symposium On Usable Privacy and Security (SOUPS) 2005*, 43–52.

Haslum, P. 2009. $h^m(P) = h^1(P^m)$: Alternative characterisations of the generalisation from h^{max} to h^m. In *Proc. of the 19th Int'l Conf. on Automated Planning and Scheduling (ICAPS-09). Thessaloniki, Greece, September 19-23*, 354–357.

Helmert, M. 2006. The fast downward planning system. *Journal of Artifical Intelligence Research* 26:191–246.

Howe, A.; Ray, I.; Roberts, M.; Urbanska, M.; and Byrne, Z. 2012. The psychology of security for the home computer user. In *Proc. of IEEE Symposium on Security and Privacy*, to appear.

Moore, A.; Ellison, R.; and Linger, R. 2001. Attack modeling for information security and survivability. Technical Report CMU/SEI-2001-TN-001, Software Engineering Institute, Carnegie Melon University.

Obes, J. L.; Sarraute, C.; and Richarte, G. 2010. Attack planning in the real world. In *Workshop on Security and Artificial Intelligence (SecArt-10) in Working Notes of the 24th AAAI Conference on Artificial Intelligence (AAAI-10)*.

Phillips, C., and Swiler, L. 1998. A graph-based system for network-vulnerability analysis. In *Proceedings of the New Security Paradigms Workshop*, 71–79.

Richter, S., and Westphal, M. 2010. The LAMA planner: Guiding cost-based anytime planning with landmarks. *Journal of Artificial Intelligence Research* 39:127–177.

Richter, S.; Thayer, J. T.; and Ruml, W. 2010. The joy of forgetting: Faster anytime search via restarting. In *Proc. of the 20th Int'l Conf. on Automated Planning and Scheduling (ICAPS-10). Toronto, Ontario, Canada, May 12-16*.

Richter, S. 2010. *Landmark-Based Heuristics and Search Control for Automated Planning*. Ph.D. Dissertation, Griffith University.

Roberts, M.; Howe, A.; Ray, I.; Urbanska, M.; Byrne, Z. S.; and Weidert, J. M. 2011. Personalized vulnerability analysis through automated planning. In *Workshop on Security and Artificial Intelligence (SecArt-11) in Working Notes of IJCAI-11. Barcelona, Catalonia, Spain. July 16-22*.

Schermerhorn, P.; Benton, J.; Scheutz, M.; Talamadupula, K.; and Kambhampati, S. 2009. Finding and exploiting goal opportunities in real-time during plan execution. In *International Conference on Intelligent RObots and Systems (IROS)*, 3912–3917.

Sheyner, O.; Haines, J.; Jha, S.; Lippmann, R.; and Wing, J. M. 2002. Automated generation and analysis of attack graphs. In *Proceedings of the IEEE Symposium on Security and Privacy*, 273–284.

Srivastava, B.; Kambhampati, S.; Nguyen, T.; Do, M.; Gerevini, A.; and Serina, I. 2007. Domain independent approaches for finding diverse plans. In *Proc. of the 20th Int'l Joint Conf. on Artificial Intelligence (IJCAI-07). Hyderabad, India. January 6-12*, 2016–2022.

Strathclyde Planning Group. 2010. VAL, the automatic validation tool for PDDL, including PDDL3 and PDDL+. Available from `http://planning.cis.strath.ac.uk/VAL/`.

Talamadupula, K.; Benton, J.; Kambhampati, S.; Schermerhorn, P.; and Scheutz, M. 2010. Planning for human-robot teaming in open worlds. *ACM Transactions on Intelligent Systems and Technology (TIST)* 1(2):1–24.