Planning Attack Graphs

Charles F. Bevan
North Carolina State Univeristy
890 Oval Drive, Campus Box 8206
Raleigh, NC 27695-8206
cfbevan@ncsu.edu

R. Michael Young
North Carolina State Univeristy
890 Oval Drive, Campus Box 8206
Raleigh, NC 27695-8206
young@csc.ncsu.edu

ABSTRACT

Classically, attack graphs have been used to represent all possible paths an attacker could take from one computer to another through a network. While having all possible paths viewable is useful, attack graphs do not lend themselves to viewing the path of a specific attack. We propose the use of plans to represent possible paths through a network instead of attack graphs. This paper covers initial progress on a larger project to create a visualization program that uses plans to visualize data equivalent to that represented by attack graphs.

Categories and Subject Descriptors

I.2 [**Computing Methodologies**]: Artificial Intelligence

General Terms

Algorithms, Verification

Keywords

Attack Graph, Planning

1. CURRENT WORK

This project attempts to represent the vulnerabilities within a network in such a way that it is more straightforward to visually highlight specific attacks and animate the attacks in such a way that major network exploitations become apparent to the user. We exploit existing representations and algorithms used in the area of *automatic planning* to create *attack plans* that characterize the sequence of exploits that compose a given attack sequence in a network. In order to demonstrate the equivalence between the attack plans created by our algorithm and the data structures representing attack graphs, we describe below an algorithm that will combine the space of plans for a specific network attack problem into a graph that we argue is isomorphic to an attack graph created for the same network configuration.

1.1 The Case for Isomorphism

In the section below, we argue that attack graphs and graphs created by combining solution plans from the plan space generated by the DPOCL [5] planning algorithm are isomorphic.

Previously, the term *attack graph* has been used to refer to two distinct types of graph structure:

1. Graphs where the set of vertices V is the set of machine states and the set of edges, E, is the set of possible attacks that will transfer a machine from state to state (e.g.[4]).
2. Graphs where the set of vertices V is the set of possible attacks and the edges E represent connections between attacks that show ordering dependencies between attack pairs (e.g. [2]).

In this work, we adopt the second representation for comparison. In this representation, attack graphs represents all possible attacks between all possible computers in a single graph. In general, a program that generates this type of attack graph takes as input a representation of a network configuration and specification of the computer whose access in the network that is the goal of the attack. Our approach first uses the DPOCL planner to create the space of all possible attack plans leading from one specific node in a network to another. Each plan in this plan space is itself a graph where vertices represent individual attack actions and edges represent the independent causal and temporal relationships between the actions associated with the participating verteces. Verteces are annotated with specifics of the attack they model (e.g., specific exploits, vulnerabilities, user paramters) and edges are labeled with the conditions in the network established or required by the respective actions (e.g., changes in user permissions, port open/closed status).

In our approach, individual attack actions are modeled using STRIPS-style [1] operators. Each action operator includes a) a list of preconditions indicating the conditions that must be true in the network in order for the action to correctly execute, b) a list of effects indicating those conditions in the network that change as a result of the action's successful execution, and c) a list of typed variables that are bound to objects or state descriptions at run time to indicate the context in which a specific attack executes. DPOCL has been shown to be both sound (that is, any individual plan it produces in the plan space is guaranteed to be executable and achieve the goal conditions at the end of the plan) and, when restricted to its non-hierarchical version, complete (that is, it generates all possible solutions to a given planning problem). As a result, the resulting plan space s contains all possible attacks for a given start node/end node pair.

A single plan represents only a single attack, comparable to a single path through an attack graph. Our argument for the equivalence of the coverage of a planning system and attack graphs proceeds by combining all plans produced for a given planning problem and then demonstrating that the nodes and vertexes in each are themselves isomorphic. It is straightforward to obtain the full set of solution plans for a given planning problem; DPOCL's plan construction process involves a search through the space of all possible plan [3]. Rather than terminating the search when a solution is first found, we allow the process to run until all plans have been created.

We then create an empty graph we call the Master Plan Space Graph ($MPSG$). We next iterate over the set of solution plans, adding elements from each plan one at time to the MPSG. We add steps to the MPSG based on their ordering in the plan in which they occur. Because we are including steps from multiple plans, it's possible that each step being added may correspond to a step from another plan that already is in the MPSG. To avoid duplicating steps in the MPSG, we first check if this is the case. A step is considered a duplicate just when another step is in the MPSG that has the same act-type and the same variable bindings and has incoming causal connections from actions executing on the same host. If an action is not a duplicate, we add the action to the MPSG as well as all its incoming causal connections.

What we need to show now is that the MPSG and an attack graph generated by another analysis program contain the same data. This can be done by showing that the attack plans steps and nodes in an attack graph are isomorphic and that the causal links in an attack plan and the edges in an attack graph are also isomorphic. Consider first the case of the mapping between plan steps and attack graph nodes. Assume that the attack graph has data that our generated a node with no corresponding step in the plan. For this to be the case, this would require that DPOCL failed to produce a plan containing the step even though, as indicated by its presence in the attack graph, a single course of action including that attack exists. Since DPOCL is complete, producing all possible plans, this cannot occur. Assume that the plan contains a step with no corresponding node in the attack graph. For this to be the case, either DPOCL would have to produce a step that had no role in the plan (and was thus spurious) or the attack graph would have to have failed to include the corresponding node when it should have. DPOCL creates steps only based on the need to use the new step to establish a later step's preconditions. Consequently, no steps are ever added to a DPOCL plan that do not link directly to another step as required to preserve the plan's soundness. Since attack graphs have been shown to be constructed correctly, we assume that every node that should appear in the attack graph does appear in it. Consequently, we conclude that the steps in DPOCL plans and the nodes in attack graphs are isomorphic.

Consider now the second case – the mapping between causal links in DPOCL plans and vertices in attack graphs. Attack graph edges are constructed specifically from the same representation of enablement bewteen actions used by DPOCL. Because DPOCL produces sound plans, no edge in an attack graph will fail to have a correlate causal link in a DPOCL plan. Conversely, because DPOCL only creates a causal link into a step because it is required as a precondition, no causal link in DPOCL will not also have a correlate edge in the attack graph.

2. FUTURE WORK

Now that we have an initial set of data that is comparable to data used by other products on the market, the next step will be to create the visualization of said data.Our intended approach will again use the DPOCL planning algorithm, but this time as camera manager, determining the type and sequence of camera shots used within a 3D virtual environment to film the animation of an individual attack plan. The idea is that animating an attack will make problem points more obvious to the user. We also intende to leverage the underlying plan representation of an attack to facilitate interaction between the user and the plan space. By reasoning about potential changes that the user might make to the network configuration (e.g., upgrading software, closing ports), the planning system can re-evaluate the space of possible attacks based on the changes to the space of possible plans under the new constraints. This will facilitate a kind of "what if" style of interaction, allowing the user to change parts of the worldand immediately observe the changes to the potential for attack.

3. REFERENCES

[1] R. Fikes and N. Nilsson. Strips: A new approach to the application of theorem proving to problem solving. In J. Allen, J. Hendler, and A. Tate, editors, *Readings in Planning*. Morgan Kaufmann, 1990.

[2] S. Jajodia. Topological analysis of network attack vulnerability. *Proceedings of the 2nd ACM symposium on Information, computer and communications security - ASIACCS '07*, page 2, 2007.

[3] S. Kambhampati, C. Knoblock, and Y. Qiang. Planning as refinement search: a unified framework for evaluating design tradeoffs in partial-order planning. *Artificial Intelligence*, 76:167–238, 1995.

[4] C. Phillips and L. Swiler. A graph-based system for network-vulnerability analysis. *Unknown*.

[5] R. M. Young, M. E. Pollack, and J. D. Moore. Decomposition and causality in partial order planning. In *Proceedings of the Second International Conference on AI and Planning Systems*, pages 188–193, 1994.