Submission to 2002 International Computer Symposium

(1) Name of the Workshop

Workshop on Cryptography and Information Security

(2) Title of the Paper

Cost Minimization for Thwarting Network Attacks

(3) A Short Abstract

An Attack Graph is a graphical representation of possible ways that an intruder can attack a network system through atomic attacks. Based on such graphs, Sheyer et al. have formulated a problem of thwarting a minimum set of atomic attacks (i.e., set T') to guarantee that the network system won't be attacked. As thwarting different atomic attacks usually involve different cost, we extend their problem to find a minimum-cost T'. Unlike their approach of using the greedy method to obtain approximate solution of this NP-hard problem, we reduce the minimum-cost problem to solving a set of Integer Programming equations. A Branch and Bound Algorithm can efficiently solve these equations for small cases. For larger cases, the algorithm takes the solution of the greedy method as the initial solution, iteratively improving this solution until that the software program is terminated.

(4) Name, current affiliation, postal and e-mail addresses, tel and fax numbers (Contracting Author)

Prof. Wen-Huei Chen
Dept. of Electronic Engineering
Fu Jen Catholic University, Taipei, Taiwan, R.O.C.
Email: PAULTAIPEI@YAHOO.COM.TW
Fax: (02)29042638 Tel:(02) 29031111 ext 2424

(5) Keywords

network security, attack graph, atomic attacks, integer programming

Cost Minimization for Thwarting Network Attacks

Wen-Huei Chen
Department of Electronic Engineering
Fu Jen Catholic University, Taipei, Taiwan, R.O.C.
PAULTAIPEI@YAHOO.COM.TW

Abstract

An Attack Graph is a graphical representation of possible ways that an intruder can attack a network system through atomic attacks. Based on such graphs, Sheyer et al. have formulated a problem of thwarting a minimum set of atomic attacks (i.e., set T') to guarantee that the network system won't be attacked. As thwarting different atomic attacks usually involve different cost, we extend their problem to find a minimum-cost T'. Unlike their approach of using the greedy method to obtain approximate solution of this NP-hard problem, we reduce the minimum-cost problem to solving a set of Integer Programming equations. A Branch and Bound Algorithm can efficiently solve these equations for small cases. For larger cases, the algorithm takes the solution of the greedy method as the initial solution, iteratively improving this solution until that the software program is terminated.

1. Introduction

A network is composed of many *hosts* remotely connected by *links* (i.e., cables, fibers, etc.) A host is a collection of hardware, software, storage media, data and involved people, each of which may have *security holes*. Based on these security holes, an *intruder* may accomplish an evil goal, e.g., causing a critical host to crash down or lose important information. Initially, the *intrude*r is in a specific host and has a limited privilege in using it. He may illegally increase his privilege in using that host or even enter his neighbor host; such an action is called an *atomic attack*. He may conduct a sequence of atomic attacks from his initial host to his *target host* so as to accomplish his evil goal; such a sequence is called an *attack*. As networks grow larger and more complex, more attacks are likely to occur. Today, network administrators want to thwart all attacks so as to guarantee that their networks are *safe*. They begin their works by using tools to scan security holes in each host so as to determine possible atomic attacks. Now, they want to determine how atomic attacks

form attacks [1].

Many administrators construct a graph by hand to see how attacks occur. Generally, such a graph is called the *Attack Graph* [2]. Each node of the graph represents a state of the intruder, and each edge represents an atomic attack that changes the state. Certain nodes are called *init nodes* (or *goal* nodes) because they represent the initial state (goal state) of the intruder. Therefore, an attack can be represented by a path from a init node to a goal node. The Attack Graph becomes quite complex when the network adds more hosts that introduce new security holes. Even the most diligent administrator may construct an incorrect Attack Graph that neglects important attacks. Hence, network administrators desire a tool to automatically construct the Attack Graph [2].

Many methods have been proposed to formally define the Attack Graph and automatically construct it. In [3], Dacier defines a *Privilege Graph* where nodes represents intruder's privileges and edges represent intruder's actions that change the privileges. The Privilege Graph is then expanded into an *Attack State Graph*, which represent different ways that an intruder can reach a goal, such as earning a root privilege of a target host. In [3], Dacier also proposes a "mean effort to failure" metric to evaluate the Attack Space Graph. In [4], Orlato et al. experiment that method . In [5], Philips and Swiler define an Attack Graph where each node represents an atomic attack attempted by the intruder. In [6]. Swiler implements the method of [5] into a tool. However, these construction methods consume a large space even for small examples. Moreover, those Attack Graphs exclude seemingly benign system events. These system events (e.g., link failures, user errors, system recovery actions, etc.) may combine with atomic attacks to form attacks.

Recently, a new Attack Graph is proposed. Jha and Wing proposes a generic state machine to represent a *network environment* specified by a set of variables for the intruder and the network system [2][7]. A state of the machine corresponds to certain values of these variables, and a transition is a change of these values caused by an atomic attack. In [2][7], an atomic attack refers to either the intruder's evil attempt or the system's seemingly begin events. Jha and Wing implements a tool to produce a subset of the state machine which describes how the intruder has accomplished his goal. Such a subset is graphically represented by an Attack Graph where each edge is labeled an atomic attack. Certain nodes of the graph are called *init nodes* (*goal nodes*) because they represent initial and goal states of the intruder respectively. Their tool is based on a symbolic model checker called "NuSmv" [8] [9] which utilizes BDDs [10] and a backward search from a multitude of goal

states [6]; both techniques significally save space and thus that subset. As a result, a large Attack Graph can be produced efficiently. At the same time, Richey and Amman also use model checker SMV for network security analysis which checks if an intruder can reach his single goal state [11].

Shyer et al. consider an optimization problem of the Attack Graph [2]. They assume that a system administrator wants to thwart a minimum number of atomic attacks so that the intruder could not reach his goal state. Put it another way, he wants to remove a minimum set of labels from the Attack Graph (notice that a label corresponds to an atomic attack) so that any init node cannot reach any goal node through a labeled path (i.e., a path which contains of all labeled edges.) They reduce the problem to the minimum cover problem [12]. The latter problem is NP-hard but a greedy approximation algorithm is available. However, their algorithm does not guarantee the optimal solution even for a small-size problem. Moreover, the problem assumes that it costs the same to thwart every atomic attack, which is not true for most network systems.

Our paper has two contributions. First, we generalize the optimization problem by assigning cost to each atomic attack. That is, we want to thwart a minimum-cost set of atomic attacks so that the intruder could not reach his goal state. Second, we reduce the optimization problem to the *Integer Programming Problem* [13]. Though the latter problem is also NP-hard, extensive researches in the literature have provided algorithms which either obtain optimal result for the small case or approximation result for the large case. For example, a *Branch and Bound Algorithm* uses the *Linear Programming Relaxation* as the lower bound [14]. By introducing the solution from the greedy algorithm of Sheyner et al. [12] as the first solution, the algorithm can iteratively obtain solutions that continually improve the first solution till the execution is terminated.

In Section 2, we describe the Attack Graph and the earlier minimization problem. In Section 3, we introduce the new optimization problem and the Integer Programming method. In Section 4, our conclusions are presented.

2. The Attack Graph

Consider an Attack Graph G(V, E) of Figure 1. The node set V= {a, b, c, d, e, f, g, h, i} represents the states of the network environment, where init node a represents the initial state and goal node i represents the goal state of the intruder. (J, K; X) is an edge

from node J to node K that has a label X which represents an atomic attack; atomic attack X changes the network environment from state J to state K. For example, edge (b, e; 6) means that an atomic attack "6" will change the environment from state b to state e. Notice that atomic attack "6" can also change the environment from state g to state h. The set T= {1, 2, 3, 4, 5, 6} is the set of atomic attacks (i.e., labels.)

An attack can be formed by considering labels of the path which starts from an init node and ends at a goal node. For example, an attack [1, 6, 5, 1] can be formed from the path [node a, node b, node e, node f, node i]. Disregarding their orders, atomic attacks of an attack form a specific set called the *realizable set*. For example, the attack [1, 6, 5, 1] forms a realizable set {1, 6, 5}. Rel(G) is a set of possible realizable sets of G(V, E). Thus, Rel(G) = {{1, 6, 5}, {1, 6, 4, 5}, {2, 1, 5}, {2, 1, 4, 5}, {2, 4, 6, 5}, {3, 1, 6, 5}}. In general, the size of Rel(G) can be exponential [15]. In practice, Rel(G) can be significally smaller than the number of possible paths from init nodes to goal nodes because the same label appears in many edges. For example, a reasonable assumption of G(V, E) has made the size of Rel(G) linear.

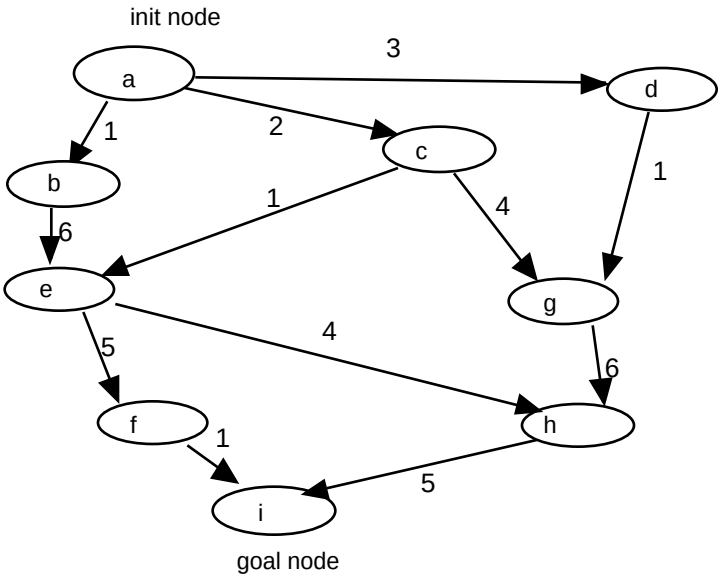

Figure 1. An attack graph G(V, E)

Assume that a realizable set is $\{x_1, x_2, ..., x_j, x_{j+1}, ..., x_n\}$ which represents many attacks. A system administrator can adopt a policy (e.g., installing a firewall) to thwart an atomic attack x_j so as to thwart those attacks. Notice that x_j may appear in another realizable set which represents other attacks. Thus, thwarting x_j may in fact thwarting two sets of attacks. In [2], Sheyer et al. assumes that the system administrator wants to thwart a

minimum set subset T' of the set of atomic attacks T so as to thwart all attacks represented by Rel(G) = {R_1, R_2, ...,R_k, R_{k+1}, . .., R_m}. Put it another way, the administrator wants to find a subset T' such that that T' intersects with R_k (k=1, 2, ..., m) by at least one element. For example, consider T= {1, 2, 3, 4, 5, 6} and Rel(G) = {{1, 6, 5}, {1, 6, 4, 5}, {2, 1, 5}, {2, 1, 4, 5}, {2, 4, 6, 5}, {3, 1, 6, 5}}. T' = {2, 3, 6} intersects with each realizable set of Rel(G) by at least one element. Thus, thwarting atomic attacks 2, 3 and 6 can thwart all attacks.

Sheyner et al. reduces the minimization problem to the minimum cover problem [2]. The problem is NP-hard but a greedy approximation algorithm is available. The greedy algorithm pick an element of T that appears mostly in the sets of Rel(G). Those covered sets are removed from Rel(G), and the same picking process continues until that Rel(G) is empty. For example, the algorithm would pick T' = {6} first as "6" appears mostly in Rel(G). The algorithm finally pick T' = {6, 2}. However, T' is the minimum one because T' = {5} can thwart all attacks.

3. The New Optimization Problem

In Section 2, we have reviewed an optimization problem for constructing a minimum set of atomic attacks to thwart all attacks. In this section, we enhance the problem to construct a minimum-cost set of atomic attacks to thwart all attacks. We will use the Integer Programming method to solve this problem.

Assume that thwarting an atomic attack X takes a cost of cost(X). The administrator wants to find a minimum-cost subset T' = T (which is the set of atomic attacks) so as to thwart attacks. That is, he wants to find T' such that T' intersects with each realizable set of Rel(G) by at least once one element (recall Section 2 that a realizable set represents attack(s)). For example, consider Figure 1. Assume that cost (1)= cost(2) = cost(3)= cost(4) =cost(6) = 10 and cost(5) = 50, T = {1, 2, 3, 4, 5, 6} and Rel(G)= {{1, 6, 5}, {1, 6, 4, 5}, {2, 1, 5}, {2, 1, 4, 5}, {2, 4, 6, 5}, {3, 1, 6, 5}}. The optimal choice of Section 2: T' = {5} takes a cost of 50, which is larger than the cost 30 of another choice: T' = {2, 3, 6}.

We solve this minimum-cost optimization problem by formulating a set of Integer Programming Equations. Consider a set of atomic attacks T= {t_1, t_2, . . .,t_j, . . ., t_n] and a set of realizable set Rel(G) = {R_1, R_2, ..., R_j, R_{j+1}, . ., R_m}. The optimal subset T' of T is

defined by a set of integer variables $x_1, x_2, \ldots, x_j, \ldots, x_k$, where $x_j = 1$ if t_j of T is included in T' and $x_j = 0$ if it is not included. The equations are as follows:

Minimize $cost(t_1) + cost(t_2) + \ldots + cost(t_n)$

such that

$t_{1'} + t_{2'} + \ldots + t_{r'} = 1$ if $\{t_{1'}, t_{2'}, \ldots, t_{r'}\}$ is a realizable set of Rel(G).

$t_1, t_2, \ldots, t_n = 0$ or 1.

The objective function "$cost(t_1) + cost(t_2) + \ldots + cost(t_n)$" means that we want to find a subset T' that is of the minimum cost. The constraint equation means that each atomic attack of T' should intersect with each realizable set at least once, so as to thwart the attack represented by that realizable set. Consider the same example of Figure 1, we thus have the following integer programming equations.

Minimize $10*t_1 + 10*t_2 + 10*t_3 + 10*t_4 + 50*t_5 + 10*t_6$

such that

$t_1 + t_6 + t_5 \geq 1$

$t_1 + t_6 + t_4 + t_5 \geq 1$

$t_2 + t_1 + t_5 \geq 1$

$t_2 + t_1 + t_4 + t_5 \geq 1$

$t_2 + t_4 + t_6 + t_5 \geq 1$

$t_3 + t_1 + t_6 + t_5 \geq 1$

$t_1, t_2, t_3, t_4, t_5, t_6 = 0, 1$

The five constraint equation corresponds to to the five realizable sets of Rel(G). By solving the above equations using the Lindo package over an IBM PC[14], we have the following solution:

$t_1 = 1, t_2 = 0, t_3 = 0, t_4 = 1, t_5 = 0, t_6 = 0$

That is, we must thwart atomic attacks "1" and "4". It can be checked from Figure 1 that after thwarting these atomic attacks, the intruder cannot reach his goal node from the init node. It takes a cost of 20 to thwart the two atomic attacks.

4. Conclusions

In this paper, we have extended the problem of thwarting a minimum set of atomic attacks to the problem of thwart a minimum-cost set of atomic attacks which guarantee that the intruder cannot attack the network system. We have also proposed an integer programming method to solve the problem. As described in [2][7][15], the Attack Graph has been used to analyze the security of a Bank Transaction System and a Firewall Detection System. We plan to implement our method into a tool so that we can see the performance of our method over these large cases.

References

[1] C. P. Pfleeger, *Security in Computing, 2nd Edition*, Prentice-Hall Inc., New Jersey, U.S.A.,1997.

[2] O. Sheyner, J. Haines, S. Jha, R. Lippmann and J. M. Wing, "Automated generation and analysis of attack graphs," *Proc. of IEEE Symposium on Security and Privacy*, May 2002.

[3] M. Dacier, *Towards Quantitative Evaluation of Computer Security, PhD Thesis*, Institu National Polytechnique de Toulouse, December 1994.

[4] R. Ortalo, Y. Dewarte and M. Kanniche, Experimenting with quantitative evaluation tools for monitoring operational security, *IEEE Trans. on Software Engineering*, Vol. 25, No. 5, pp. 633-650, September/October, 1999.

[5] C. Philips and L. Swiler, "A graph-based system for network vulnerability analysis," *New Security Paradigms Workshop*, pp. 71-79, 1998.

[6] L. Swiler, C. Philips, D. Ellis and S. Chakerian, Computer-attack graph generation tool, *Proc. of the DARPA Information Survivability Conference and Exposition*, June 2000.

[7] S. Jha and J. M. Wing, "Survivability analysis of networked systems," *Proc. of Int'l Conference on Sofware Engineering*, May 2001.

[8] NuSMV, "Nusmv: a new symbolic model checker," http://afrodite.itc.it:1024/nusmv/.

[9] SMV, "Smv: a symbolic model checker," http://www.cs.cmu.edu/modelcheck/.

[10] R. E. Bryant, "Graph-based algorithms for boolean function manipulation," *IEEE Trans. on Computers*, Vol. C-35, No. 8, pp. 677-691, August 1986.

[11] R. W. Ritchey and P. Amman, Using model checking to analyze network vulnerabilities, *Proc. IEEE Symposium on Security and Privacy*, pp. 156-165, May 2001.

[12] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-completeness*, W. H. Freeman and Company, San Francisco, U. S. A., 1979.

[13] A. Schrijver, *Theory of Linear and Integer Programming*, John Wiley & Sons, 1998.

[14] L. Schrage, *Lindo 5.0 User's Manual*, Scientific Press, 1991.

[15] S. Jha, O. Sheyner and J. M. Wing, "Minimization and Reliability Analyses of Attack Graphs," *Technical Report, CMU-CS-02-109, School of Computer Science, Carnegie Mellon University,* PA, U.S.A., Feb 2002.