A Markov Decision Model for Intruder Location in IP Networks [1]

T. Darling and M. A. Shayman
Department of Electrical and Computer Engineering and
Institute for Systems Research
University of Maryland
College Park, MD 20742
Email: shayman@eng.umd.edu

IEEE Conference on Decision and Control, Sidney, Australia, December 2000

Abstract

We consider the problem of locating an intruder in an IP domain using dynamic IPSec security associations as proposed in the DECIDUOUS project. We formulate this problem as a Markov decision process that evolves on the set of subtrees of a shortest path routing tree. For small domains, an optimal stationary policy can be determined by dynamic programming. For large domains, the use of neurodynamic programming as well as heuristic policies are examined. Our results indicate that under certain assumptions, a one-feature heuristic policy provides good performance.

1 Introduction

In 1991, the Computer Science and Telecommunications Board released a study that began with the following statement:

> We are at risk. Increasingly, America depends on computers. They control power delivery, communications, aviation, and financial services. They are used to store vital information, from medical records to business plans to criminal records. Although we trust them, they are vulnerable– to the effects of poor design and insufficient quality control, to accident, and perhaps most alarmingly, to deliberate attack. The modern thief can steal more with a computer than with a gun. Tomorrow's terrorist may be able to do more damage with a keyboard than with a bomb.

Nearly a decade later, the National Research Council report on *Trust in Cyberspace* notes that

> Evidence abounds that the Internet and the public telephone networks not only are vulnerable to attacks but also are being penetrated with some frequency. In addition, hackers seeking the challenge and insiders seeking personal gain or revenge have been successful in attacking business and critical infrastructure computing systems.

The NRC report goes on to state:

> The Defense Information Systems Agency (DISA) estimates that DOD may have experienced as many as 250,000 attacks on its computer systems in a recent year and that the number of such attacks may be doubling each year. ...Similarly troubling statistics about private-sector computer breakins have been reported,

2 Intrusion and Misuse Location

In recent years, there has been considerable progress in developing systems for the *detection* of network intrusion and misuse. Most approaches use signature matching to detect known attacks and statistical anomaly detection for novel attacks. Among many others, examples include the IDES, NIDES, and EMERALD systems from Stanford Research Institute (SRI) [1] [6]; the GrIDS system from UC Davis [10] which constructs activity graphs of network operations to look for traffic

[1] This research was supported in part by NSF under Grant ECS-9626399 and by the NSA Laboratory for Telecommunications Science under contract MDA90499C2521

patterns indicative of coordinated attacks on a network; the JiNao system from North Carolina State University and MCNC [7] which uses statistical anomaly detection and signature matching to detect attacks on routing infrastructure.

In contrast to the large amount of work on intrusion/misuse detection, there has been much less research reported on the crucial related problem of locating the source(s) of an attack once it is detected. (One such project is IDIP [8].) Because of *IP spoofing*, the source address in an attack packet cannot be relied upon to disclose the true source of the attack. IP spoofing refers to a variety of techniques used to falsify source IP addresses. This may be done simply to hide the identity of the sender or may be done to obtain unauthorized access privileges. In his comprehensive analysis of the data compiled by CERT, Howard notes [4]

> Mail spam is the most common form of denial-of-service attack.... One way this is accomplished is by sending repeated messages to a mail server with the intent of exceeding the capacity of the system. Attackers will often use mail spoofing to falsify the 'from' address when sending mail spam.

Currently, the victim of an attack must rely on tools that were not intended for attack source identification. These include *traceroute* and *finger* which were developed for other purposes and are at best minimally suitable for tracing an attack source [3].

Recently, researchers from North Carolina State University and MCNC have proposed DECIDUOUS, a security management framework for identifying the sources of network-based intrusions [3]. DECIDUOUS is constructed on top of IETF's IPSEC/ISAKMP infrastructure. For attack source identification in a single administrative domain, it has the advantage of not requiring any new network protocols.

The key idea underlying DECIDUOUS is that of *dynamic security associations*. IPSEC is an extension of the Internet Protocol to include features for authentication and/or confidentiality. The primary mechanism by which these features are obtained is the *security association* (SA). A security association is a one-way relationship between a pair of nodes (hosts or routers) A (viewed as sender) and B (viewed as receiver) established to permit security services to be provided to the traffic traveling from A to B. (See, e.g., [9].) Note that the SA applies to all traffic going from A to B; it applies both to traffic destined for B that originates at A as well as to traffic that originates at an upstream node and is forwarded by A on its way to B. Also, nodes A and B need not be immediate neighbors.

Only the authentication function of IPSEC is relevant to our discussion. The authentication function provides for data integrity and source identification. In particular, since the source and destination addresses are protected, IP address spoofing is prevented. Authentication is based on the utilization of a message authentication code (MAC). In turn, this requires the two parties to the SA to share a secret key. Internet Security Association and Key Management Protocol (ISAKMP) is used for key management. ISAKMP typically uses a key exchange algorithm such as Oakley that is based on the Diffie-Hellman algorithm. Diffie-Hellman is computationally intensive since it requires modular exponentiation.

Obviously one way to prevent spoofing and ensure identifiability of attack sources is to establish SAs between every pair of nodes that might ever need to communicate. However, this would be too expensive from a computational viewpoint. IPSEC processing overhead would occur even when there are no attacks. Static SAs between a limited collection of pairs of communicating nodes would not guarantee attacker identification, especially with the increasing sophistication of attackers who share vast databases of network vulnerability information [5]. The solution proposed by DECIDUOUS is to construct a security management module that dynamically decides when and where to establish IPSEC SAs.

For dynamic SAs to be effective at locating intruders, it is critical to have an efficient, if not optimal, decision algorithm to determine when and where to establish SAs. We will show that this problem can be formulated as a Markov decision process (MDP). This opens up the possibility of using techniques for the solution (and approximate solution) of MDPs to solve the sequential decision problem of SA placement. In fact, the problem is an example of an important class of MDPs that we refer to as *hierarchical diagnosis problems*; see section 2.1 below.

2.1 The Hierarchical Diagnosis Problem

We define the general *hierarchical diagnosis problem* as follows: Let S be the collection of all components of a given system. For component i, let X_i be the indicator function for the component being faulty. For each subset $A \subset S$, it is "possible" to perform a test to determine if A contains a faulty component. The cost of such a test is represented by a random variable C_A. In reality, the structure of the system and the properties of available diagnostic tools will make testing impossible for certain subsets A. This can be modeled by setting $C_A = \infty$. Finally, a joint distribution for the random variables $\{X_i\}$ is given. In the special cases of mutually exclusive or independent faults, it suffices to specify the probability p_i that component i is faulty for each i. Given this model, the problem is to determine the optimal order of subsets to be tested to locate the faulty component(s).

In the standard sequential diagnosis problem, only individual components may be tested. This corresponds to the special case of the hierarchical diagnosis problem in which $C_A = \infty$ whenever $|A| > 1$.

We focus on the important special case of the hierarchical diagnosis problem in which the subsets that may be tested are constrained by a directed tree. A directed spanning tree for S is given. For each $i \in S$, let A_i denote the subset consisting of i together with all its descendents. Only the subsets A_i may be tested; i.e., the testing cost for every other subset is infinite. Thus, there is a test associated with each each node of the tree. A test corresponding to a node i will be positive if node i or any of its descendents are faulty.

We restrict our attention to the case where there is exactly one faulty component. In the literature, this is referred to as the *mutually exclusive fault* (MEF) problem. Suppose that there is a probability p_i that component i is faulty. For a subset A_i of S, let $p(A_i) = \sum_{j \in A_i} p_j$. If subset A_i is tested, the result will be positive with probability $p(A_i)$ and negative with probability $1 - p(A_i)$. If the result is positive, the fault is isolated to the subtree consisting of the nodes in A_i; if negative, it is isolated to the subtree consisting of the nodes in $S - A_i$.

More generally, suppose that a sequence of tests has been performed that has isolated the fault to a subtree S' of the initial tree S. If the subset A_i is tested, the fault will be isolated to either $A_i \cap S'$ or $S' - A_i \cap S'$ depending on whether the result is positive or negative, respectively. The probability of the former is $p(A_i \cap S')/p(S')$, while the probability of the latter is $1-p(A_i \cap S')/p(S')$. Thus, we have a Markov decision process in which the state space consists of all subtrees of the initial tree S. There is an admissible action in state S' for each node $i \in S'$. If action i is taken, there are two possible successor states, namely $A_i \cap S'$ and $S' - A_i \cap S'$. The cost of taking action i in state S' is C_{A_i}. The MDP is a *stochastic shortest path problem* in which the terminal states consist of those subtrees S' that have $|S'| = 1$–i.e., single-node subtrees.

3 Intruder Location as an MDP

We now show that the problem of using dynamic security associations to locate intruders/misusers in an IP-based network can be formulated as a tree-hierarchical diagnosis problem. Hence optimal policies for establishing dynamic SAs can be obtained by using techniques available for solving the type of MDPs introduced above.

As in the DECIDUOUS project [3], we assume that an intrusion detection system is in place that can detect attack packets that arrive at a host or router. We make the assumption that shortest path routing is used; e.g., consider an autonomous system using OSPF in the Internet. This means that there is a spanning tree rooted at j that is used to route all packets with final destination j. When a packet destined for j reaches an intermediate node i, i forwards it to its parent node in the spanning tree. Suppose that a security association is established between i and j. If an attack packet is then received at j, it can be determined whether this packet was forwarded by i. If the answer is positive, then the attack source is localized to the subtree A_i consisting of i together with all of its descendents. (See Figure 1.) If the answer is negative, then the attack source is localized to the subtree consisting of all nodes other than i and its descendents. (This assumes a single source attack.) Consequently, the problem of determining the optimal sequence of SAs is equivalent to the hierarchical diagnosis problem described above and corresponds to an MDP. Note that if the attacker is not continuously sending attack packets, then each time we establish a new SA we will need to wait a random length of time until the next attack packet is received.

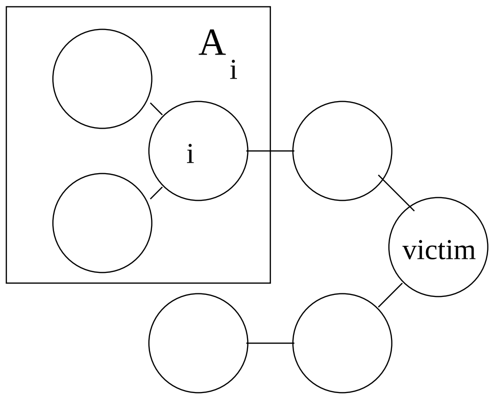

Figure 1: SA with node i isolates attacker to subtree A_i or $S - A_i$

In general, the optimal stationary policy for the stochastic shortest path problem for the tree-hierarchical MEF diagnosis problem will depend on the topological structure of the tree, the fault probabilities p_i, and the single-stage cost function $C_i := C_{A_i}$. For the remainder of this paper, we consider the special case where the probabilities and the single-stage cost function are constant. In other words, we assume that all nodes are equally likely as the attack source and the establishment of each security association incurs a cost which is independent of the node, and hence is taken to be equal to 1. Under these assumptions it is clear that the policy is determined by the graph topology.

As is pointed out in [3], the choice of policy is obvious in the case of a *linear* topology. In this case, an optimal policy (which is also worst-case optimal) corresponds to choosing the security association with a middle node–i.e., such that the difference in the number of nodes in the two successor subtrees is at most one. In this case, the number of SAs required to locate the attacker is deterministic and is a logarithmic function of the number of nodes. At the opposite end of the spectrum is the case where all nodes are directly connected to the victim node. In this case, each subset A_i consists of a single node and the optimal cost depends linearly on the number of nodes. However, in the case of general topology, the solution is not at all obvious. Furthermore, exact solution by dynamic programming becomes impractical for more than about 25 nodes. Consequently, it is natural to consider the use of one of the variants of neurodynamic programming [2] [11].

3.1 Solution Methodology

We explored four methods for computing or approximating the value function and obtaining optimal, or near-optimal, policies:

- Exact solution by dynamic programming
- Linear feature-based approximations of DP tables
- Neurodynamic programming using linear approximation architecture
- Heuristic policy with one feature

The first method considered was *dynamic programming*. The dynamic programming starts with the single tree containing one node, on which the value function takes value 0. From this, it follows that the value of the single tree containing two nodes is 1. Next one computes the value of the two distinct trees with three nodes. This approach can be applied to compute the value of all trees with at most a given number of nodes. Alternatively, if an initial tree of interest is given, the dynamic programming can be restricted to all distinct subtrees of the initial tree.

DP is computationally intensive since the number of subtrees (and hence states) grows exponentially with the number of nodes in the initial tree–i.e., the number of routers in the domain. In addition, storing the optimal policy may require a very large lookup table. To address the storage/lookup issue, we considered 9-feature approximations of DP tables. The features are: number of nodes N, average number of children per node, number of leaf nodes, average path length, the base 2 logarithms of the preceding four features, and $N\log_2(N)$.

We considered approximations for the optimal value function that were of the form of a linear combination of the nine feature functions. The nine coefficient values were determined by minimizing a *weighted* mean square error between the approximation and the true optimal value function obtained by dynamic programming. It must be kept in mind that the ultimate goal is to obtain a good approximation for the optimal *policy*, not merely to obtain a good approximation for the optimal value function. In achieving this goal, the values of the approximate value function on some states are more important than the values on other states. We were guided by the philosophy that the more often a state (subgraph) arises in the calculation of the exact value function by DP, the more important it is to have an accurate approximation of the optimal value function at this state. Consequently, in the mean square criterion, we used a weight factor for each state that was proportional to the frequency with which the state occurred in the DP computation. This was done as follows. We created a counter for each subgraph and incremented it by 1 when that subgraph was used directly in the DP process to compute the value function for a larger graph–i.e., the subgraph is one of the two successor states when the optimal action is chosen for the larger graph. If the optimal action is not unique, one of the optimal actions (and hence a pair of successor states) was chosen randomly. After using DP to compute the exact value function for all graphs with at most N nodes, the approximation was obtained by weighted least squares with weights proportional to the counter values. There are certainly other methods of choosing the weights; the method we have followed is not necessarily the best.

Linear feature-based approximation of the exact value function as described above requires computation of the exact value function prior to the approximation. Thus, it does not overcome the exponential growth of the computation time as a function of the number of nodes. To deal with this issue, we considered using neurodynamic programming [2, 11]. The particular algorithm used in our experiments employed a linear approximation architecture with nine non-binary features for the *value function* together with gradient-descent TD(1) learning. Actions were chosen using a policy that was ϵ-greedy with respect to the current approximate value function. It should be noted that we approximated the value function directly instead of approximating the *action-value function* as is typically done in neurodynamic programming. This was possible since there is a complete model of the transition probabilities for the controlled Markov chain. We used the same nine features used previously for the direct approximation of the optimal value function. The learning algorithm generated values for the nine corresponding coefficients.

We now describe the algorithm more precisely. The only uncertainty in the state transitions is due to the location of the attacker. Consequently, instead of using a random number generator to simulate the uncertainty in the state transitions, we were able to simplify the simulation by rotating the (fixed) position of the attacker at each new trial and using the corresponding *deterministic* state transition function.

Consider the algorithm when the simulation state is given by the tree s. Let $\theta = (\theta_1, \ldots, \theta_9)$ be the current parameter vector. Then the current value function approximation is given by

$$V_\theta(s) = \sum_i \theta_i \phi_i(s),$$

where $\phi_i(s)$ is the value of the i^{th} feature of the state (tree) s. Let a be an admissible action in s–i.e., a node of s selected for an SA. Let s_1 and s_2 denote the two possible successor states (subtrees) of s resulting from the action a. Then the corresponding action-value function approximation is given by

$$Q_\theta(s,a) = 1 + \frac{|s_1|}{|s|} V_\theta(s_1) + \frac{|s_2|}{|s|} V_\theta(s_2),$$

since the single-stage cost is assumed to be 1 and the probability that the next state is s_i is proportional to the number of nodes in that subtree since it is assumed that the location distribution of the attacker is a uniform distribution. Thus,

$$Q_\theta(s,a) = 1 + \sum_i \theta_i \left(\frac{|s_1|}{|s|} \phi_i(s_1) + \frac{|s_2|}{|s|} \phi_i(s_2)\right).$$

Consequently, this can be viewed as a linear approximation architecture for the action-value function in which the i^{th} action-value feature of the state s is $\frac{|s_1|}{|s|}\phi_i(s_1) + \frac{|s_2|}{|s|}\phi_i(s_2)$. Since the action is chosen according to an ϵ-greedy policy with respect to the current value function approximation, with probability $1-\epsilon$ a is chosen to be an action that minimizes $Q_\theta(s,a)$, and with probability ϵ is chosen randomly (with equal probabilities) from the admissible actions in state s.

The next state s' is uniquely determined from (s,a) by the location of the attacker fixed at the start of the simulation run. The general gradient descent update for action-value approximation is given by [11, p. 211]

$$\theta' = \theta + \alpha[v - Q_\theta(s,a)]\nabla_\theta Q_\theta(s,a),$$

where α is the learning rate parameter and v is the 'target output'. The form of the target output is determined by the type of learning algorithm employed. We used $v = 1 + V_\theta(s')$ which corresponds to $TD(1)$ learning. Note that

$$\nabla_\theta Q_\theta(s,a) = \frac{|s_1|}{|s|}\phi(s_1) + \frac{|s_2|}{|s|}\phi(s_2),$$

where $\phi = (\phi_1, \ldots, \phi_9)$.

The final approach we considered was a heuristic policy based on a single feature. In each state, the action (node) is chosen that minimizes the absolute value of the difference between the number of nodes in the two possible successor states (subgraphs). We explored several methods for breaking ties by maximizing or minimizing a variety of other features.

3.2 Results

In the experiments, random trees were generated by starting with a single root node and recursively adding one node to a random node in the existing tree until a tree with the desired number of nodes was obtained. For trees of up to about 25 nodes, we were able to use DP to compute the exact value function; hence, we were able to construct the 9-feature direct approximations as well. In general, we found that the single-feature heuristic policy was slightly worse than the exact optimal value function, but typically as good as the 9-feature direct approximation and significantly better than the 9-feature approximation obtained by neurodynamic programming. Furthermore, the tiebreak method used in the single-feature heuristic policy made little difference. Thus, it is sufficient to randomly choose a node from those involved in a tie.

For more than 25 nodes, it may not be feasible to compute the exact value function. Although we do not have the optimal value functions to use for comparison, we can compare the value function of a given policy with the logarithmic bound–i.e., no policy is expected to isolate the attacker in less than $\log_2 N$ steps, where N is the number of nodes. Using the single-feature heuristic policy (with random tiebreak), we created a dozen random 200-node trees and ran 200 simulations on each, one for each attacker location. By averaging the time-to-isolate over the 200 simulations, we were able to exactly compute the value function for the heuristic policy on each of the 12 initial trees. The values obtained ranged from 7.860 to 7.975. Since $\log_2(200) = 7.64$, the heuristic policy performs reasonably close to the log bound and hence close to optimal.

Given its computational simplicity, the single-feature heuristic policy seems to be adequate for the special case of the intruder location problem on which we have focused–i.e., the problem with a single attack source, a uniform location probability distribution for the attacker, and a single-stage cost for setting up a security association that is independent of node location. If these assumptions are relaxed, we expect more complex policies to be required. This is the subject of current investigation.

References

[1] D. Anderson, T. Frivold, and A. Valdes. Next-generation intrusion detection expert system (nides): A summary. Technical Report SRI-CSL-95-07, SRI International, Menlo Park, CA, May 1995.

[2] D. Bertsekas and J. Tsitsiklis. *Neuro-Dynamic Programming*. Athena Scientific, Belmont, MA, 1996.

[3] H. Y. Chang, R. Narayan, S. F. Wu, B. M. Vetter, X. Wang, M. Brown, J. J. Yuill, C. Sargor, F. Jou, and F. Gong. Deciduous: Decentralized source identification for network-based intrusions. In *Proceedings of International Symposium on Integrated Network Management*, pages 701–714, Boston, MA, May 1999.

[4] J. D. Howard. An analysis of security incidents on the internet 1989-1995. Technical Report Ph.D. thesis, Carnegie Mellon University, Pittsburgh, PA, 1997.

[5] T. Longstaff. Panel: Preparing for internet threats in the years 2001 to 2010. In *National Information Systems Security Conference*, Arlington, VA, October 1999.

[6] P. A. Porras and P. G. Neumann. Emerald: Event monitoring enabling responses to anomalous live disturbances. In *Proceedings of the National Information Systems Security Conference*, 1997.

[7] D. Qu, B. M. Vetter, F. Wang, R. Narayan, S. F. Wu, Y. F. Jou, F. Gong, and C. Sargor. Statistical anomaly detection for link-state routing protocols. In *Proceedings of IEEE Conference*, pages 62–70, September, 1998.

[8] D. Schnackenberg, K. Djahandari, and D. Sterne. Infrastructure for intrusion detection and response. In *Proceedings of the DARPA Information Survivability Conference and Exposition (DISCEX)*, Hilton Head, South Carolina, January 2000.

[9] W. Stallings. *Cryptography and Network Security: Principles and Practice*. Prentice Hall, Inc., Englewood Cliffs, NJ, 1998.

[10] S. Staniford-Chen, S. Cheung, R. Crawford, M. Dilger, J. Frank, J. Hoagland, K. Levitt, C. Wee, R. Yip, and D. Zerkle. Grids–a graph based intrusion detection system for large networks. In *Proceedings of the National Information Systems Security Conference*, pages 361–370, October, 1996.

[11] R. Sutton and A. Barto. *Reinforcement Learning*. MIT Press, Cambridge, MA, 1998.