Intrusion Response as a Resource Allocation Problem

Michael Bloem, Tansu Alpcan, and Tamer Başar

***Abstract*— We study intrusion response in access control systems as a resource allocation problem, and address it within a decision and control framework. By modeling the interaction between malicious attacker(s) and the intrusion detection system (IDS) as a noncooperative non-zero sum game, we develop an algorithm for optimal allocation of the system administrator's time available for responding to attacks, which is treated as a scarce resource. This algorithm, referred to as the Automatic or Administrator Response (AOAR) algorithm, applies neural network and LP optimization tools. Finally, we implement an IDS prototype in MATLAB based on a game theoretical framework, and demonstrate its operation under various scenarios with and without the AOAR algorithm. Our approach and the theory developed are general and can be applied to a variety of IDSs and computer networks.**

I. Introduction

One of the primary objectives of enterprise networks is to make information and services available only to authorized users. The increasing complexity and global scale of collaboration between organizations and economic entities results in strict security requirements on contemporary networks.

Moreover, current networks need to be dynamic, distributed, and scalable in order to serve the needs of these collaborating entities. Unfortunately, the distributed nature and complexity of computing and communication networks prevent administrators and organizations from maintaining absolute control over their systems. This leads to an ongoing confrontation with the malicious attackers who aim to compromise the security of the system and gain unauthorized access to information and services.

Meanwhile, the resources allocated towards responding to attacks, such as IT security personnel, firewalls, and patch management systems, are growing slowly. This creates a widening gap between the resources required to respond to attacks and the resources available. In particular, network administrators and security personnel can easily become overwhelmed with the volume of attack responses required [1].

Computing and communication networks also need intrusion detection systems (IDSs) as an integral part of their operation, as static protective measures are no longer sufficient. IDSs enhance security by monitoring the events in the networked system and analyzing them for signs of security problems [2]. Many IDSs have an automatic response component, where the IDS takes some action against detected attacks. Thus attacks can be handled automatically or by administrators.

In order to address the resource allocation problems identified above, the actions and strategies of the intruders must be taken into account, which naturally leads to a game theoretical analysis. Game theory is directly applicable to ID, and several studies have proposed that it be used to theoretically understand and analyze ID [3]–[5]. The game theoretical approach is promising in the context of intrusion detection and response in access control systems, where it is possible to utilize it in attack modeling, analysis of detected threats, and decision on response actions [6], [7].

Access control and authentication systems, such as the policy and role based access control (PR-BAC) server developed by the Boeing company, are the type of computer networks that will be explicitly discussed in this paper. The results however apply to the security of most computer networks.

In this paper, we utilize an existing game theoretical model of an IDS in an access control system [7] to introduce, investigate, and simulate an algorithm for allocating the time a system administrator has available to respond to attacks. This algorithm is entitled the Automatic or Administrator Response (AOAR) algorithm. We demonstrate the algorithm's performance through simulations in MATLAB.

The rest of this paper is organized as follows: In the next section we review the game theoretical model for analyzing an IDS operating in an access control system developed in [7]. Section III introduces the AOAR algorithm and reviews the tools (neural networks and LP) that it utilizes. An implementation of the IDS in MATLAB and simulations are presented in Section IV, and the paper concludes with some remarks in Section V.

II. Game Theoretical Model

This analysis of intrusion response as a resource allocation problem was performed by utilizing a modified version of the game theoretical model of the interaction between an IDS and attacker presented in [7].

A. Continuous Security Game

The model in [7] sets up cost functions that consider the costs of attacking and responding to attacks for the attacker and IDS, respectively. Suppose that there are A_{max} strategies available for attacking a computer network such as an access control system. Clearly it is not possible to enumerate all attacks, so instead we enumerate the output of the IDS by

Research supported in part by a grant from The Boeing Company through the Information Trust Institute, University of Illinois, Urbana, Illinois.

M. Bloem and T. Başar are with the Coordinated Science Laboratory, University of Illinois, 1308 West Main Street, Urbana, IL 61801 USA. (mbloem2, tbasar)@control.csl.uiuc.edu

T. Alpcan was with the Coordinated Science Laboratory. He is now with Deutsche Telekom Laboratories, Technische Universität Berlin, Ernst-Reuter-Platz 7, 10587 Berlin, Germany. tansu.alpcan@telekom.de

simply referring to its alarm characteristics. We later model the imperfections in the particular IDS in use.

Also, suppose that there are R_{max} attack responses available to the IDS or system administrator. Let the strategy space of the attacker be $U^A := \{\mathbf{u}^A \subset \mathbb{R}^{A_{max}} : u_i^A \geq 0,\ i = 1, \ldots, A_{max}\}$ and let the strategy space of the IDS and the administrator be $U^I := \{\mathbf{u}^I \subset \mathbb{R}^{R_{max}} : u_j^I \geq 0,\ j = 1, \ldots, R_{max}\}$. The quantities u_i^I and u_j^A represent the magnitude of response and attack at strategies i and j, respectively.

The model in [7] makes use of a matrix $\bar{P}$ to capture the imperfections in the sensor network and to map attacker actions to sensor outputs. In the case of ideal sensors, $\bar{P}$ corresponds to the identity matrix because then each attacker action perfectly maps to a corresponding sensor output. Moreover, the paper defines

$$P := [p_{ij}] = \begin{cases} p_{ij} = -\bar{p}_{ij} & \text{if } i = j \\ p_{ij} = \bar{p}_{ij} & \text{if } i \neq j \end{cases}. \tag{1}$$

for notational convenience.

Successful attacks are costly to the owners of information or resources in an access control system and lead to a gain for the attacker. The vectors $\mathbf{c}^I := [c_1^I, \ldots, c_{R_{max}}^I]$ and $\mathbf{c}^A := [c_1^A, \ldots, c_{A_{max}}^A]$ represent these costs and gains. Also associated with each attack and response strategy is a cost of effort. Attacking a network takes time and effort, and response strategies usually involve some negative externalities. The cost of responding to an attack is captured in the vector $\alpha := [\alpha_1, \ldots, \alpha_{R_{max}}]$ and the cost of attempting an attack is captured in the vector $\beta := [\beta_1, \ldots, \beta_{A_{max}}]$.

Due to conscious decisions or security imperfections, systems are more vulnerable to some attacks than others. The nonnegative matrix Q with diagonal entries greater than or equal to 1 captures this vulnerability. The $\bar{Q}$ matrix correlates response strategies to the attack strategies they confront; it is made up of ones and zeros and is of dimension $A_{max} \times R_{max}$. Finally, the scalar γ represents the relative value of various cost terms in the equations for the cost for the IDS, $J^I(\mathbf{u}^A, \mathbf{u}^I, P)$, and the attacker(s), $J^A(\mathbf{u}^A, \mathbf{u}^I, P)$, which are

$$J^I(\mathbf{u}^A, \mathbf{u}^I, P) := \gamma(\mathbf{u}^A)^T P \bar{Q} \mathbf{u}^I + (\mathbf{u}^I)^T \mathrm{diag}(\alpha)\mathbf{u}^{\mathrm{I}} + \mathbf{c}^I(Q\mathbf{u}^A - \bar{Q}\mathbf{u}^I), \tag{2}$$

and

$$J^A(\mathbf{u}^A, \mathbf{u}^I, P) := -\gamma(\mathbf{u}^A)^T P \bar{Q} \mathbf{u}^I + (\mathbf{u}^A)^T \mathrm{diag}(\beta)\mathbf{u}^{\mathrm{A}} + \mathbf{c}^A(\bar{Q}\mathbf{u}^I - Q\mathbf{u}^A), \tag{3}$$

where $(x)^T$ represents the transpose of x and $\mathrm{diag}(x)$ refers to a diagonal matrix with diagonal entries containing the corresponding entries in vector x.

The first terms in these equations are zero-sum and represent the cost of false-alarms and benefit of detection for the IDS and the cost of capture and benefit of deception for the attacker. The second terms characterize the cost of effort associated with responding to or generating an attack. The actual benefit or cost of a successful attack is represented in the third of the three terms.

By minimizing the strictly convex cost functions (2) and (3), one can determine the IDS and attacker response functions. They are uniquely given by $\mathbf{u}^I(\mathbf{u}^A, P) = [u_1^I, \ldots, u_{R_{max}}^I]^T$ and $\mathbf{u}^A(\mathbf{u}^I, P) = [u_1^A, \ldots, u_{A_{max}}^A]^T$, respectively, where

$$\mathbf{u}^I(\mathbf{u}^A, P) = [\mathrm{diag}(2\alpha)]^{-1} \left[\bar{\mathrm{Q}}^{\mathrm{T}}(\mathrm{c}^{\mathrm{I}})^{\mathrm{T}} - \gamma \bar{\mathrm{Q}}^{\mathrm{T}} \mathrm{P}^{\mathrm{T}} \mathbf{u}^{\mathrm{A}}\right]^+ \tag{4}$$

$$\mathbf{u}^A(\mathbf{u}^I, P) = [\mathrm{diag}(2\beta)]^{-1} \left[\mathrm{Q}^{\mathrm{T}}(\mathrm{c}^{\mathrm{A}})^{\mathrm{T}} + \gamma \mathrm{P} \bar{\mathrm{Q}} \mathbf{u}^{\mathrm{I}}\right]^+. \tag{5}$$

Note that $[x]^+$ maps all negative values of x to zero.

B. Discretized Security Game

As indicated earlier, this paper focuses on the allocation of the scarce resource of the system administrator's time. The administrator's time is actually allocated in the decision of whether to send an alarm to the administrator or to allow the automatic response to respond to the alarm. This decision is most naturally analyzed for alarms one at a time, so the continuous security game presented in Section II-A was discretized using thresholding. A few other modifications were also introduced, and therefore the final product is not exactly a game theoretical model, but rather a discrete model inspired by a continuous game.

Recall that the quantities u_i^I and u_j^A represent the magnitude of response and attack at strategies i and j, respectively. More concretely, a particular response strategy i could refer to "stop user John Doe from accessing forbidden documents". A low u_i^I might mean that a warning email is sent to John Doe, but a high u_i^I might mean that John Doe is forbidden from accessing the network at all until some action is taken. An even higher u_i^I could indicate that a particular server or resource should be temporarily shut down to prevent damage or data theft. Suppose there are D^R and D^A levels of actual response for each u_i^I and u_j^A, respectively, and that appropriate threshold values have been set to discretize u_i^I and u_j^A values into a particular $d_i^R = 1, \ldots, D^R$ and $d_j^A = 1, \ldots, D^A$, respectively. Let $\bar{\mathbf{u}}^I$ and $\bar{\mathbf{u}}^A$ denote IDS or administrator strategies and attacker strategies that have been discretized in this way. For simulation purposes, time varying quantities will be updated at finite time steps. Let $\bar{\mathbf{u}}^I[t]$ and $\bar{\mathbf{u}}^A[t]$ denote response and attack strategies that have been discretized both in time and in intensity of action.

Assume also that $\mathbf{u}^I$ and $\mathbf{u}^A$ have been discretized such that if an attack of intensity d_j^A is responded to with a response d_i^R with the same or greater magnitude, then the attack will be thwarted and not inflict any damage. Let $\bar{\mathbf{u}}_a^A[t]$ be the attacks that are attempted in a given time step t. Then $\bar{\mathbf{u}}^A[t]$ represents the attacks that survived the response and actually were executed.

This discretized system also leads to new cost equations. The discretized cost for the IDS, $\bar{J}^I(\bar{\mathbf{u}}^I[t], \bar{\mathbf{u}}_a^A[t], \bar{\mathbf{u}}^A[t], \mathcal{D}[t])[t+1]$, and attacker, $\bar{J}^A(\bar{\mathbf{u}}^I[t], \bar{\mathbf{u}}_a^A[t], \mathcal{D}[t])[t+1]$, respectively satisfy

$$\bar{J}^I(\bar{\mathbf{u}}^I, \bar{\mathbf{u}}_a^A, \bar{\mathbf{u}}^A, \mathcal{D})[t+1]:= \gamma(\bar{\mathbf{u}}_a^A)^T \mathcal{D}\bar{Q}\bar{\mathbf{u}}^I + (\bar{\mathbf{u}}^I)^T \mathrm{diag}(\alpha)\bar{\mathbf{u}}^{\mathrm{I}} + \mathbf{c}^{\mathrm{I}}\mathrm{Q}\bar{\mathbf{u}}^{\mathrm{A}} \quad (6)$$

and

$$\bar{J}^A(\bar{\mathbf{u}}^I, \bar{\mathbf{u}}_a^A, \mathcal{D})[t+1]:= -\gamma(\bar{\mathbf{u}}_a^A)^T \mathcal{D}\bar{Q}\bar{\mathbf{u}}^I + (\ \bar{\mathbf{u}}_a^A)^T \mathrm{diag}(\beta)\bar{\mathbf{u}}_{\mathrm{a}}^{\mathrm{A}} - \mathbf{c}^{\mathrm{A}}\mathrm{Qk_c}\bar{\mathbf{u}}_{\mathrm{a}}^{\mathrm{A}}, \quad (7)$$

where the $[t]$ notation has been suppressed to improve readability.

The value computed in each equation is discretized into D^R and D^A levels, respectively, using the thresholds mentioned above. These equations have a few minor differences from the continuous equations that inspired them. Instead of the P matrix, they include a new term $\mathcal{D}[t]$, the distortion caused by the sensor network. This replaces the P matrix but it only captures the distortion of the sensor network. Like P, $\mathcal{D}$ has entries with magnitude between 0 and 1 and has negative entries on the diagonal. It distorts the $\mathbf{u}^A[t]$ vector, representing the operation of an imperfect sensor network. Also, the final term in (7) does not use $\bar{\mathbf{u}}^A$ but rather the attacker's estimate of this value, which is based on the k_c "confidence constant" ($0 \leq k_c \leq 1$). The product of k_c and the attempted attacks $\bar{\mathbf{u}}_a^A$ yields the attacker's estimate of $\bar{\mathbf{u}}^A$, or how successful the attacks will be. By introducing this constant, one can differentiate $\bar{J}^I$ with respect to $\bar{\mathbf{u}}_a^A$, allowing for optimization with respect to $\bar{\mathbf{u}}_a^A$.

The discretized reaction functions are given by $\bar{\mathbf{u}}^I(\bar{\mathbf{u}}_a^A[t], \mathcal{D}[t])[t+1]$ and $\bar{\mathbf{u}}_a^A(\bar{\mathbf{u}}^I[t], \mathcal{D}[t])[t+1]$, where

$$\bar{\mathbf{u}}^I(\bar{\mathbf{u}}_a^A, \mathcal{D})[t+1] = \left[-\gamma[\mathrm{diag}(2\alpha)]^{-1}\bar{\mathrm{Q}}^{\mathrm{T}}\mathcal{D}^{\mathrm{T}}\bar{\mathbf{u}}_{\mathrm{a}}^{\mathrm{A}}\right]^+ \quad (8)$$

$$\bar{\mathbf{u}}_a^A(\bar{\mathbf{u}}^I, \mathcal{D})[t+1] = \left[\mathrm{diag}(2\beta)^{-1}[\mathrm{k_c}\mathrm{Q}^{\mathrm{T}}(\mathrm{c}^{\mathrm{A}})^{\mathrm{T}} + \gamma\mathcal{D}\bar{\mathrm{Q}}\bar{\mathbf{u}}^{\mathrm{I}}]\right]^+. \quad (9)$$

Once again the $[t]$ notation has been suppressed to enhance readability.

These reaction functions uniquely minimize the strictly convex cost functions (6) and (7). After discretization, this becomes an integer optimization problem, so solving it as if it were continuous is only an approximation. When the administrator responds to an attack, the response vector is denoted by $\bar{\mathbf{u}}_{admin}^I(\bar{\mathbf{u}}_a^A[t])[t+1]$ and satisfies

$$\bar{\mathbf{u}}_{admin}^I(\bar{\mathbf{u}}_a^A[t])[t+1] = \left[\gamma[\mathrm{diag}(2\alpha)]^{-1}\bar{\mathrm{Q}}^{\mathrm{T}}\bar{\mathbf{u}}_{\mathrm{a}}^{\mathrm{A}}[\mathrm{t}]\right]^+. \quad (10)$$

Where $\mathcal{D}$ was in (8) is now the negative of the identity matrix. This captures the assumption that the administrator can tell perfectly what is happening in the network.

III. Automatic or Administrator Response Algorithm

System administrators' investigation time and effort per incident is one of the major constraints of any IDS. Although often overlooked in the design of IDSs, this limitation will continue to exist until the development of systems with some of the functionality we associate with "intelligence". For this reason and because attacks themselves are designed by human intelligence, contemporary IDSs have to rely on human intervention for decisions in at least a subset of incidents. On the other hand, forwarding every anomaly or suspected attack reported by the sensors to the system administrator is not realistic, and practically ignores the mentioned constraints. Therefore an automated decision process, however imperfect, is needed. The AOAR algorithm describes one way to decide which attacks are forwarded to the administrator and which ones are not.

More specifically, the AOAR algorithm is based on the classification of attacks into those that resemble previous successful attacks and those that do not. Game theoretical analysis demonstrates that system characteristics and vulnerabilities will be taken advantage of by attackers [7], leading to more intense attacks on certain systems. By monitoring attacks that repeatedly survive the automatic response, and forwarding such attacks to the system administrator, an IDS using AOAR can better allocate the system administrator's scarce time than if attacks are randomly selected to be forwarded to the administrator.

A. Self-Organizing Maps

The AOAR algorithm uses attack classification to determine which attacks to forward to the administrator. Essentially, attacks are classified into two categories: those that resemble previous successful attacks and those that do not. Attacks with different strategies $u_i^A \geq 0,\ i = 1, \ldots, A_{max}$ can be sorted into d dimensions of n_k elements each, such that $\prod_{k=1}^{d} n_k = A_{max}$. For example, attack strategies could be broken down into categories such as resource under attack, IP address of attacker, or time of day of attack.

Using these attack characteristics, attack classification is achieved with Kohonen self-organizing maps (SOM) [8]. The SOMs are trained using a standard *Kohonen learning rule*. Intuitively, "training the network" refers to placing a specified number of neurons on the d dimensional map such that frequent successful attack strategies are more likely to have a neuron near them than strategies that are rarely successful. Neurons are thus nothing more than "cluster centers" for successful attack strategies, and are potentially the result of a particularly valuable resource, an easy attack strategy to attempt, or systematic problems with the IDS. For further details on SOM implementation and training we refer to [9].

A very simple one-dimensional example of neuron placement is shown in Fig. 1. The bars represent the number of successful attacks at each strategy over a period of time and the circles on the horizontal axis are the neurons. Note that few neurons are near infrequent attacks at strategies 4-6.

B. Minimization of Response Cost

We formulate the IDS decision on whether to alert the system administrator or to make an automated decision regarding a reported incident as a resource allocation problem.

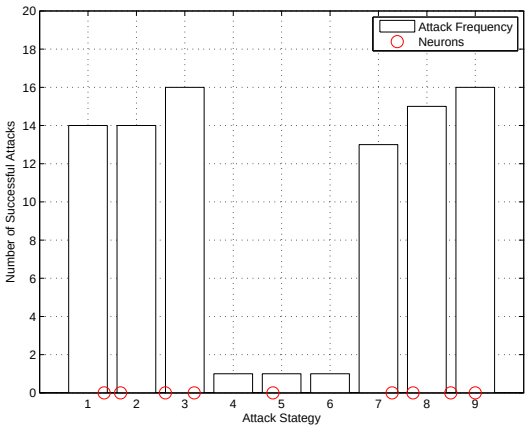


Fig. 1. A bar chart of the one-dimensional attack frequencies and corresponding neuron placement.

In order to optimally make this decision, relative costs are assigned to the two options. $C_1 > 0$ represents the cost of making the decision automatically per incident, which also includes the risk of making mistakes by the automated decision process. The value $C_2 > 0$ quantifies the cost of time and effort spent by the system administrator per incident. The quantities $\bar{J}^I$ and $\bar{J}^I_{admin}$, respectively, are natural choices for these costs.

Let x_2 be the proportion of attacks that are randomly forwarded to the system administrator and let x_1 represent the proportion that are sent to the automatic response mechanism for further analysis and response. Of those sent to the automatic response mechanism, let λ be the proportion that are chosen to be forwarded to the administrator. Finally, let N be the number of incidents per time slot and let L be the number of cases the system administrator can handle per given time slot. The overall cost function is defined for a given time slot as

$$J(x_1, x_2) := Nx_1[(1-\lambda)C_1 + \lambda C_2] + Nx_2C_2. \qquad (11)$$

Thus, we obtain the following constrained optimization problem:

$$\begin{aligned} \min_{x_1,x_2} \; & J(x_1, x_2) = Nx_1[(1-\lambda)C_1 + \lambda C_2] + Nx_2C_2 \\ \text{such that } & N(\lambda x_1 + x_2) \le L \\ & x_1 + x_2 = 1 \\ & x_1,\ x_2 \ge 0. \end{aligned} \qquad (12)$$

C. The Algorithm

For notational convenience, let Z_a denote the distance from a particular attack u_i^A to the nearest neuron. A "time slot" refers to a period of time in between the re-calibration of the SOM and parameters in (12).

Algorithm III.1. *The dynamic algorithm to solve the optimization problem (12) and to update its parameters is:*

1) *Choose the costs C_1 and C_2 so that they accurately represent the relative cost of allowing the automatic response to respond to an attack as opposed to the administrator. Start with best estimates for N and L such that there is a feasible solution to (12).*
2) *When starting a new time slot, measure N and the Z_a values from the last time slot. Use these N, L, the Z_a values from the last time slot, and λ to determine a threshold value Z_a^*. If necessary, update L in order to ensure the existence of a feasible solution to (12).*
3) *Solve the optimization problem (12) using LP to obtain the probabilities x_1 and x_2.*
4) *Forward each alarm from the sensors with probability x_2 directly to the system administrator, and let the automatic response decide how to respond with probability x_1.*
5) *For incidents sent to the automatic response,*
   - *If $Z_a \le Z_a^*$ then this attack is forwarded to the administrator.*
   - *Otherwise, the automatic response responds to this attack.*

Note that when the automatic response makes a decision based on $Z_a \le Z_a^*$, it is actually checking how similar this new attack is to previous attacks. If the attack was a common successful attack last time slot, then it is likely to be near a neuron and fulfill $Z_a \le Z_a^*$.

Fig. 2 graphically depicts the operation of the AOAR algorithm. Note that the solid lines represent attacks and the dashed lines represent the corresponding responses.

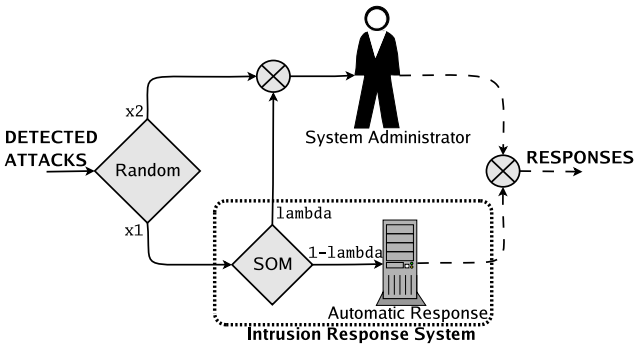


Fig. 2. A flow chart depicting the operation of the AOAR algorithm.

IV. Simulations

Some simulations will be used to demonstrate the operation of the AOAR algorithm and to show its value in a variety of situations. The value of the algorithm will primarily be shown using the cost equations (6) and (7). Also, the "value response ratio" (VRR), a simple metric, will be used to show that the AOAR algorithm works by having the administrator respond to more attacks that the automatic response would not have stopped on its own, thereby leading to less successful attacks. The VRR is defined as $VRR := R^{value}_{admin}/R_{admin}$, where R^{value}_{admin} is the number of "value responses" by the administrator: responses where the administrator halts an attack that the automatic response would not have halted. R_{admin} is the total number of

administrator responses to attacks. A higher VRR indicates that the administrator is stopping more attacks that would have otherwise been successful.

To simulate the imperfections in and dynamics of the sensor grid, the distortion matrix $\mathcal{D}$ is updated at every time step. We define the random matrix $W[t] := [w_{ij}[t]], \quad i = 1, \ldots, A_{max}, \quad j = 1, \ldots, R_{max}$ Hence, W models the transients and imperfect nature of the sensor grid. Let us also define an upper bound, $dt_{max} < 1$, and a lower bound $dt_{min} > 0$ on the elements of $\mathcal{D}$. In doing so we can model the cases where sensors have a limited detection capability. Finally, call ε the "randomization coefficient." Then $\mathcal{D}$ evolves according to $\mathcal{D}[t+1] = [\mathcal{D}[1] + \varepsilon W[t]]$, where the normalization function $[x]^N$ maps entries of x onto the interval $[dt_{min}, dt_{max}]$. In order to generate more realistic results, white noise was added in an identical fashion to the attack vector $\mathbf{u}^A$.

A simple scenario was constructed for simulation purposes. In this scenario, $A_{max} = R_{max} = 9$ and each attack corresponds exactly to one response ($\bar{Q}$ is the identity matrix). Thresholds were established such that $D^R = D^A = 3$. Also, γ was set equal to 10. A uniform system (uniform vulnerabilities, costs, etc.) would thus have $Q = \hat{\mathrm{I}}(9)$, $\mathbf{c}^I = \mathbf{c}^A = 50\hat{\mathrm{o}}(9)$, and $\alpha := 10\hat{\mathrm{o}}(9)$ and $\beta := 10\hat{\mathrm{o}}(9)$. $\hat{\mathrm{I}}(x)$ refers to an $x \times x$ identity matrix and $\hat{\mathrm{o}}(x)$ refers to a one by x vector of ones. The initial $\mathcal{D}$ has 0.75 as its diagonal elements and 0.1 as its non-diagonal elements. The ε parameter is set to 0.25. The parameters λ and k_c were set to 0.15 and 0.45, respectively. The time slot length was set to 10 time steps, and L was set such that the administrator was able to respond to 10 time steps each time slot.

In order to have a control case to compare results against, each simulation was run alongside a corresponding case without the AOAR algorithm implemented. The system is capable of determining x_1 and x_2 such that the optimal number of attacks are forwarded to the administrator, but not choosing those attacks intelligently.

A. Variation in Vulnerability

The first simulation examines how the system responds when the system is particularly vulnerable to one attack strategy. To model this, one entry on the diagonal of Q is set to 3, while the rest are left at the value 1. In this scenario, the AOAR should be helpful because the attacker will have an incentive to attack the more vulnerable attack with greater intensity and frequency. The automatic response was set so that it usually would be unable to stop these attacks. The AOAR algorithm detects this problem and forwards the attacks on this vulnerable system to the administrator, drastically improving the VRR. Table I shows the results of this simulation.

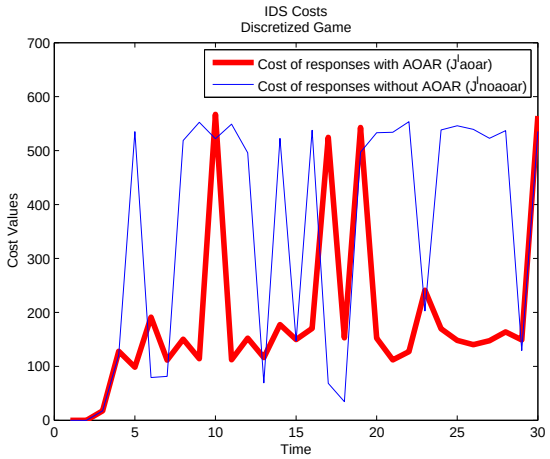
The costs $\bar{J}^I$ and $\bar{J}^A$ are plotted in Fig. 3 (a) and (b), respectively.

Fig. 4 shows the fate of attacks at each time step. Note that the system not using AOAR (a) usually is unable to stop one attack each time step. In most time steps, this is the attack on the more vulnerable system. The administrator in the system using AOAR (b) typically stops this attack on the more vulnerable system because the AOAR algorithm recognizes it and forwards it.

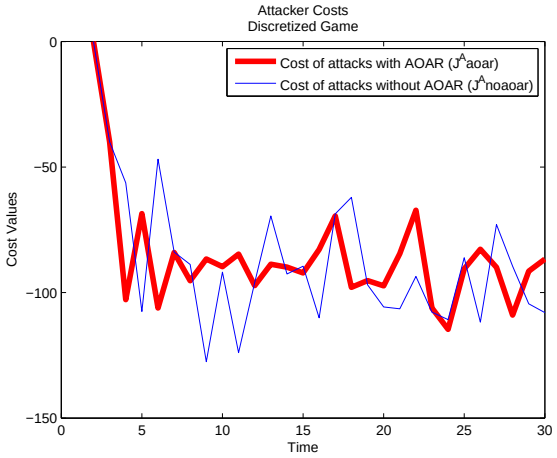
TABLE I

RESULTS OF SIMULATION OF SYSTEM WITH VARYING VULNERABILITIES

IDS	No AOAR	AOAR	% Improvement
J^I_{avg}	376.69	228.35	39.39%
J^A_{avg}	-91.91	-86.47	-5.92%
VRR	0.4700	0.7451	58.53%



(a)

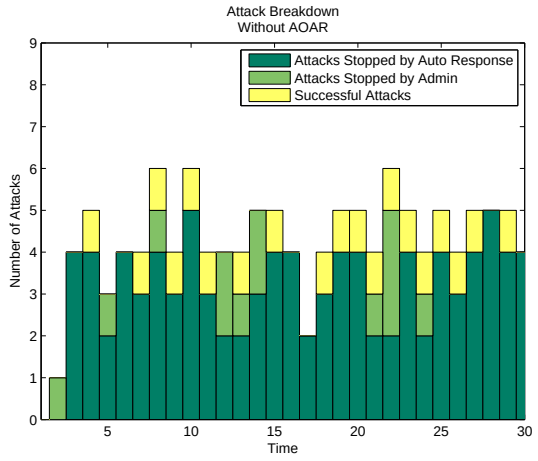


(b)

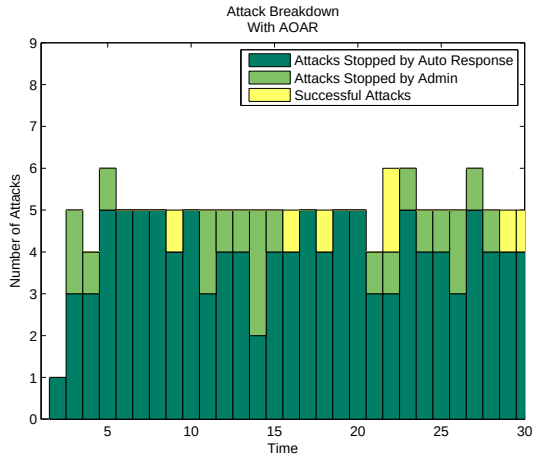
Fig. 3. Costs for IDS (a) and attacker (b) with and without AOAR algorithm when system has variations in its vulnerability to attack strategies

B. Variation in Value

Here we simulate how the AOAR algorithm performs in a situation where the costs resulting from attacks to the system and the gain resulting from attacks for the attacker are not uniform. More specifically, we alter both $\mathbf{c}^I$ and $\mathbf{c}^A$ to no longer be $50\hat{\mathrm{o}}(9)$, but instead have a value of 200 for one strategy. Attacks on this strategy lead to particularly large gains for the attacker and particularly large losses for the systems under attack. See Table II for the results.



(a)



(b)

Fig. 4. Breakdown of attack outcomes without AOAR (a) and with AOAR (b).

TABLE II
RESULTS OF SIMULATION OF SYSTEM WITH VARYING VALUES

IDS	No AOAR	AOAR	% Improvement
J^I_{avg}	529.55	315.45	40.43%
J^A_{avg}	-162.66	-154.05	-5.29%
VRR	0.5151	0.8224	59.64%

C. Variation in Costs of Actions

The effort required to perform a certain response or attack can also vary. Suppose one of the costs in β is 2 while the others remain at 10. This would mean that one particular attack strategy is easy for an attacker. The results of this simulation are shown in Table III. The IDS implementing the AOAR algorithm again performs better than the IDS not using the algorithm, particularly in its ability to increase attacker costs.

V. CONCLUSIONS AND FUTURE WORK

We have utilized a game theoretical model of the interaction between an IDS and attacker to investigate the optimal allocation of the system administrator's time. We

TABLE III
RESULTS OF SIMULATION OF SYSTEM WITH VARYING COSTS OF ACTIONS

IDS	No AOAR	AOAR	% Improvement
J^I_{avg}	167.79	130.77	22.06%
J^A_{avg}	-23.83	-12.92	-45.77%
VRR	0.5268	0.6900	30.97%

presented the AOAR algorithm, which utilizes classification of successful attacks with SOM and also LP optimization, as a way to decide how to respond to each attack. The performance of this algorithm was then simulated under a variety of system and IDS circumstances. In each situation investigated, an IDS implementing the AOAR algorithm performed better than one not using the algorithm.

This work, while laying out a practical implementation of an algorithm and demonstrating its utility, is lacking a formal theoretical framework. Thus the resource allocation problem studied here could be re-formulated as a formal optimal control problem with more sophisticated cost structures and constraints. The costs could be functions of not only the current attacker and IDS actions but also past actions. This algorithm could also be applied to more elaborate models or to real IDS data. Furthermore, the algorithm itself should be improved. More feedback loops could be simulated and studied. Specifically, the λ value could be updated dynamically based on the relative performance of the automatic and administrator responses. Learning algorithms could be used to improve the mapping of attacks to automatic responses, represented by the $\bar{Q}$ matrix. Other classification schemes (aside from SOMs) could be used to determine which attacks should be forwarded to the administrator and which ones should not. Finally, decentralized versions of some of the ideas in this paper should be investigated.

REFERENCES

[1] N. J. Lippis III, "Network-based enterprise threat defense strategy: Returning control to IT departments," white paper, http://www.cosmicegg.com/production/lippis/lr50/network_td.pdf.

[2] R. Bace and P. Mell, "Intrusion detection systems," NIST Special Publication on Intrusion Detection Systems, http://www.snort.org/docs/nist-ids.pdf.

[3] D. A. Burke, "Towards a game theoretic model of information warfare," Master's thesis, Air Force Institute of Technology, Air Force University, Nov. 1999.

[4] K.-W. Lye and J. Wing, "Game strategies in network security," in *Foundations of Computer Security Workshop in FLoC'02*, Copenhagen, Denmark, July 2002.

[5] P. Liu and W. Zang, "Incentive-based modeling and inference of attacker intent, objectives, and strategies," in *Proc. 10th ACM Computer and Communications Security Conf. (CCS'03)*, Washington, DC, October 2003, pp. 179–189.

[6] T. Alpcan and T. Başar, "A game theoretic approach to decision and analysis in network intrusion detection," in *Proc. 42nd IEEE Conf. Decision and Control*, Maui, HI, December 2003, pp. 2595–2600.

[7] ——, "A game theoretic analysis of intrusion detection in access control systems," in *Proc. 43rd IEEE Conf. Decision and Control*, Paradise Island, Bahamas, December 2004, pp. 1568–1573.

[8] N. K. Bose and P. Liang, *Neural Network Fundamentals with Graphs, Algorithms, and Applications*. New York, NY: McGraw-Hill, 1996.

[9] H. Demuth and M. Beale, *MATLAB Neural Network Toolbox User's Guide*, 4th ed., The MathWorks Inc., Jan. 2003.