I. INTRODUCTION

In modeling system response to security threats, researchers have made extensive use of state space models, notable instances including the partially observable stochastic game model proposed in [9]. The drawback of these state space models is that they may suffer from state space explosion. Our approach in modeling defense makes use of a combinatorial model which helps avert this problem. We propose a new attack-tree (AT) [7] model named attack-countermeasure trees1 (ACT) based on combinatorial modeling technique for modeling attacks and countermeasures. ACT enables one to (i) place defense mechanisms in the form of detection and mitigation techniques at any node of the tree, not just at the leaf nodes as in defense trees (DT) [2] (ii) automate the generation of attack scenarios [5] from the ACT using its mincuts and (iii) perform probabilistic analysis (e.g. probability of attack, attack and security investment cost, impact of an attack, system risk, return on attack (ROA) and return on investment (ROI)) in an integrated manner (iv) select an optimal countermeasure set from the pool of defense mechanisms using a method which is much less expensive compared to the state-space based approach ([9]) (v) perform analysis for trees with both repeated and non-repeat events. For evaluation purposes, we suggest suitable algorithms and implement an ACT module in SHARPE [8]. We demonstrate the utility of ACT using a practical case study (BGP attacks) [3].

II. ATTACK COUNTERMEASURE TREES

![ACT Diagram](image)

Fig. 1. ACT with (a) one attack event (b) one attack and one detection event (c) one attack and multiple detection events (d) one attack, one detection and one mitigation event (e) multiple detection and multiple mitigation events

In ACT, we can have three distinct classes of events: attack events (e.g. install keystroke logger), detection events (e.g. detect keystroke logger) and mitigation events (e.g. remove keystroke logger). Figures 1(a)-(e) represent different ACT nodes and the corresponding probability of attack success are given in Eq. 1- Eq. 5, where \( p_{\text{goal}} \) is the probability of attack success at the goal. In case (v), the nature of the mitigation technique triggered depends on the nature of intrusion detected. It is assumed that the detections can occur simultaneously.

\[
\begin{align*}
\text{total cost} & = \text{attack cost} + \text{mitigation cost} \\
\text{impact} & = \text{attack impact} + \text{mitigation impact} \\
\text{probability of attack success} & = \prod_{i=1}^{n} (1 - p_{i}) \\
\end{align*}
\]

The output probability of AND, OR and k-out-of-n gates in an ACT are enumerated in Table I.

<table>
<thead>
<tr>
<th>Gate type</th>
<th>Formuale for probability of attack success</th>
</tr>
</thead>
<tbody>
<tr>
<td>AND &amp; OR</td>
<td>[ p_{\text{goal}} = p_{A} ]</td>
</tr>
<tr>
<td>&amp; k-out-of-n*</td>
<td>[ p_{\text{goal}} = p_{A} \prod_{i=1}^{n} (1 - p_{i}) ]</td>
</tr>
<tr>
<td></td>
<td>[ p_{\text{goal}} = p_{A} \prod_{i=1}^{n} (1 - p_{D}) ]</td>
</tr>
</tbody>
</table>

\*For k-out-of-n gate, it is assumed that \( C_{1}, C_{2}, ..., C_{n} \) are sorted in the decreasing order of their cost values and \( I_{1}, I_{2}, ..., I_{n} \) are sorted in the decreasing order of their impact values.

III. ACT ANALYSIS

ACT analysis can be either qualitative or probabilistic.

Qualitative Analysis. Qualitative analysis of ACT investigates different attack scenarios using mincuts and relative importance of individual attack, detection and mitigation events using importance measures (IM). Mincuts help enumerate the various minimal combinations of attack events in the ACT that lead to attack success. We also use mincuts to perform cost, impact analysis and optimal countermeasure selection. Importance Measures are significant in determining the most critical component in a system. Detection/mitigation events for the most critical component can be given higher priority. We consider structural importance \( I_{ST}^{C} \) when the probability of attack and detection/mitigation of ACT nodes are unknown. If probability of attack/defense for ACT nodes are known, Birnbaum importance measure \( I_{BT}^{C} \) [1] is used. We use \( I_{BT}^{C} \) for sensitivity analysis and for ROI, ROA computation.

Probabilistic Analysis. The metrics such as attack cost and ROA

---

1This research was supported by US National Science Foundation grant NSF-CNS-08-31325.
reflect the attacker’s viewpoint whereas the metrics security investment cost, risk, impact and ROI represent the defender’s viewpoint. Cost and impact of attack were used as measures by Schneier [7] for analysis of AT. In ACT, cost can be of two types: cost of attack and security investment cost. In ACT without repeated events, we use the formulae in Table I to compute attack cost and impact of attacks for the gates. When one or more events are repeated in the ACT, we use Rauzy’s algorithm [6] to construct the binary decision diagram (BDD) corresponding to the ACT. We select the ACT mincut with lowest cost as the attack cost of the ACT. We select the ACT mincut with highest impact as attack impact for the ACT. Security investment cost is computed by summing the cost of defense mechanisms in the ACT. Risk computation (probabilistic risk assessment) follows from probability of attack and impact computation as system risk is given by the product of probability of attack and impact i.e. the expected value of impact. ROA and ROI [2] quantify the nature of the competition between the attacker and the defender. ROA is a measure that quantifies the benefit obtained from a particular attack (Eq. 6). Unlike attack cost, ROI for a defense mechanism changes with the order of applying the defense mechanisms. ROI is the profit obtained by the implementation of the k-th defense mechanism by defender. For an ACT, ROI is given by Eq. 6 where I_pgoal(k), Igoal and C_Dk are the change in pgoal, impact due to attack success and the cost for implementing defense mechanism D_k respectively.

\[
ROA = \frac{I_{goal} \times p_{goal}}{\text{total cost of attack}}
\]

IV. IMPLEMENTATION & OPTIMAL COUNTERMEASURE SELECTION

A. Implementation

We have implemented cost, impact and risk analysis of ACT’s in SHARPE [8]. ROA and ROI computation can be done by defining functions in SHARPE. We may also compute feasible attack scenarios (mincuts) subject to attacker resource constraints (attack cost constraint). Figure 2a shows how ROI varies with security investment cost and impact of attack for the ACT sol `resetting a BGP session’ [3].

B. Optimal Countermeasure Selection

Given the ACT of a system, it is generally desirable to enforce the subset of the whole set of countermeasures, which is most cost efficient while covering as many attack events as possible. Then the problem reduces to finding out the smallest possible set of countermeasures (\(C'\)) that contains at least one countermeasure from each mincut. A greedy algorithm for solving this problem is given in Table II. We implemented all the optimization algorithms in a MATLAB toolbox. The algorithm (Table II) is directly reducible to the set cover optimization algorithm [4]. Runtime = (total number

<table>
<thead>
<tr>
<th>Table II: Algorithm for Finding the Optimal Set of Countermeasures</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Input the mincuts of the ACT (from SHARPE)</td>
</tr>
<tr>
<td>2. Find (U_A) and (U_C), where, (U_A) = set of all attack events of the ACT and (U_C) = set of all countermeasures (C'=\phi) (m =</td>
</tr>
<tr>
<td>3. Initialize an (m \times n) matrix ((A)) with all ‘0’ elements. (A_{ij} = 1) (boolean value indicating that attack event (A_j) is covered by countermeasure (C_i)) else 0 for every (C_i \in U_C)</td>
</tr>
<tr>
<td>Traverse the set of mincuts and enumerate the attack events covered by (C_i) set the corresponding elements in the matrix to 1 end for</td>
</tr>
<tr>
<td>4. While (U_A! = \phi) select the countermeasure (C_p) that covers the maximum number of attack events set the corresponding elements in the matrix to 1 end if</td>
</tr>
<tr>
<td>find (S_{AC_p}) (set of attack events covered by (C_p)) (C' = C' \cup C_p) (U_A = U_A \setminus S_{AC_p}) end while</td>
</tr>
<tr>
<td>5. output (C')</td>
</tr>
</tbody>
</table>

Fig. 2. (a) Variation of ROI with the sum of impacts from attack and the security investment cost of ACT for BGP attack (b) Runtime variation with increasing tree size

of attack events in the ACT)*(total number of defense mechanisms in the ACT)*min(no. of attack events, no. of defense mechanisms)) \(= O(mn*min(m,n))\) (say) = worst case \(O(m^2)\) (n=m in worst case i.e.- we add a distinct countermeasure for every attack event). With incrementally larger tree size, runtime for optimization(y-axis) is plotted against the number of nodes in the tree (x-axis) in Figure 2b. From our analyses, we observe that an optimal set of defense mechanisms for a system can be obtained in comparatively lesser time by using our optimization techniques on ACT than by using attack response trees (ART) [9].

V. CONCLUSION

We have shown that ACT not only allows us to perform qualitative and probabilistic analysis based on combinatorial models but also provide us with methods for the computation of optimal defense strategies in large systems.

REFERENCES CITED