An Opinion Model for Evaluating Malicious Activities in Pervasive Computing Systems

Indrajit Ray, Nayot Poolsappasit, and Rinku Dewri

Department of Computer Science, Colorado State University
{indrajit, nayot, rinku}@cs.colostate.edu

Abstract. Pervasive computing applications typically involve cooperation among a number of entities spanning multiple organizations. Any security breach in any single entity can have very far-reaching consequences. In addition, a number of factors make the task of defending against malicious attacks in pervasive systems even more complex than conventional systems. Foremost among them is that a significant number of the devices deployed in such environments are frequently severely resource constrained. Thus strong security controls cannot be easily deployed on these devices. A second factor is that since a large number of such devices are also involved, attacks can propagate very fast in pervasive environments. These prompt us to propose a model for predicting malicious activities in pervasive systems. Our model is based on a logic of opinion that has been proposed elsewhere. Ours is not an intrusion detection system for pervasive systems but works in tandem with one. The system we propose can be used as a standard interface to analyze pervasive system activities in general and generate an opinion about the possibility of an attack.

1 Introduction

With the growth of mobile and sensor devices, embedded systems, and communication technologies, we are moving towards an era of pervasive computing. Pervasive computing uses numerous, casually accessible, often invisible, computing and sensor devices in addition to conventional computing systems. These devices are frequently mobile and/or embedded in an environment that is mobile. Most of the time these devices are richly inter-connected with each other using wireless or wired technology. Being embedded in the environment and interconnected allow pervasive computing devices to exploit knowledge about the operating environment in a net-centric manner. This enables pervasive computing applications to provide a rich new set of services and functionalities that are not otherwise possible through conventional means and has the potential to impact numerous applications that benefit society. Examples of such applications are emergency response, automated monitoring of health data for assisted living, environmental disaster mitigation and supply chain management.

Pervasive computing applications typically involve many entities that span different organizations interacting in complex and subtle ways. Any attack that causes security breach in a single entity can have very far-reaching consequences. For example, future earthquake monitoring systems are expected to be integrated with electricity grid, gas distribution systems, elevator controls in high rises, traffic monitoring systems etc., that

are to be switched off when a severe earthquake is detected [3]. Imagine the havoc that can be rendered if such a system is maliciously triggered. However, defending pervasive computing applications against malicious attacks is not easy. Traditional techniques cannot be directly applied. This is because the severe resource constraints inherent to a significant number of the devices [9] – limited energy, processing and memory – complicates the adoption of a vast number of conventional security protocols and renders others completely useless. The widespread use of wireless communication technology further aggravates the problem because attackers can easily intercept, fabricate or jam traffic. Moreover, the rich connectivity among devices enables an attack to spread very rapidly from one device to another across the system. To complicate matters further, security threats in a pervasive computing environment are very application dependent. Thus, it is practically impossible to design a solution that is satisfactory for all applications. It is important, for these reasons, that pervasive computing systems be carefully monitored for malicious activities. This allows one to take just-in-time mitigating actions, if needed by dynamically relocating security controls in the application. In the current work, we propose a model by which we can evaluate the chances of attacks occurring. The model assumes the existence of a activity monitoring system for pervasive applications. and is based on our earlier work on predicting threats from malicious insiders [11].

The rest of the paper is organized as follows. Section 2 discusses how an workflow can be used to model the activities and interactions in a pervasive system. Section 3 presents the attack tree model for this work and discusses how an attack tree can be derived for the pervasive application from its workflow. In section 3.2 we present the opinion model for attack trees. Section 4 presents the quantitative framework for evaluating attacks. Finally section 5 concludes the paper.

2 Modeling Pervasive Applications with Workflows

Security threats in a pervasive environment are application dependent. Consequently, business models incorporating any kind of a pervasive computing paradigm will highly benefit if formalisms are derived to enable a “case-by-case” study of the problem. We, therefore, propose to discuss our approach using an example health care application. We emphasize, however, that our formalization of the model is such that the formulated problems remain independent of any property intrinsic to the healthcare domain only.

The pervasive health care environment consists of devices that measure the vital signs of patients, location sensors that locate mobile resources, location-aware PDAs carried by health-care personnels, and back-end systems storing and processing records of patient data. The devices are connected through wired or wireless medium. We classify these devices into three categories – *adapters*, *composers* and *back-end*. Adapters are are devices with low capabilities. They collect raw sensor data and forward them to composers for processing. Composers have medium processing capabilities and may have fixed or battery power sources. Back-ends are high processing capability systems whose power may be tapped by composers.

The application consists of different workflows that get triggered by various events. The following example specifies the workflow that handles the situation when an unanticipated change occurs in a patient's vital signs (VS) monitor.

Case 1: The VS monitor tries to detect the presence of the doctor within a wireless communicable distance. If the doctor is present, he can make suggestions which may or may not be based on the patient report stored at the back-end. He may also decide to request the assistance of a nurse, who is located with the help of the network infrastructure. In case of an emergency, the same network infrastructure is used to notify the emergency service.

Case 2: If a doctor cannot be located nearby, there is a search for a nurse. The nurse may have the requisite skills to take care of the situation, perhaps with information obtained from the back-end system. If not, the nurse requests the network infrastructure to locate a remote doctor. The remote doctor can then make his suggestions to the nurse or directly interact with the monitoring devices using the network. Possibilities are also that the doctor feels the need to be immediately with the patient, and informs the emergency service on his way.

Case 3: If a nearby doctor or a nurse cannot be located, the VS monitor communicates with the network infrastructure to locate a remote doctor. The doctor, once located, can remotely interact with the monitoring equipments, or decide to attend to the situation physically, often asking for assistance from a nurse. Emergency services are notified on a need basis. Also, on the event that the network is unable to locate the doctor, it informs the emergency service.

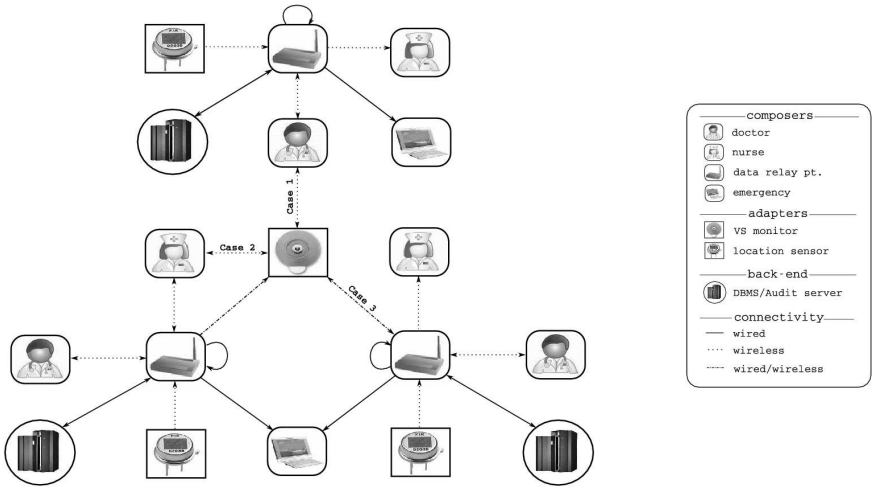

Fig. 1. Constituents of a pervasive healthcare environment.

This scenario is represented by the workflow in Fig. 1 and emphasizes the communication links that are used between the nodes in different contexts. The direction of the

link indicates the direction of the information flow. The "VS Monitor" node, which is the source node for this workflow can initiate a communication with either a doctor, a nurse or a data relay point. The link between a data relay point and the emergency service is in one direction owing to the specification that the data relay point is only used as an intermediate node to inform the emergency service. How the emergency service handles this notification is outside the scope of the application.

For our purose, a workflow is a tuple $\langle N, E, n, \mathcal{N}, \mathcal{D} \rangle$, where N is a set of *nodes*, E is a set of *edge characteristics*, n is a *sink node*, $\mathcal{N}$ is a set of *paths* and $\mathcal{D}$ is called the *path composition*. A node represents an application executing on a wired or wireless device in the pervasive environment. The application can be broken down into simpler operations. For security analysis we assume that vulnerabilities in these operations are know. An edge characteristic represents a communication channel used between the nodes. Note that we are not identifying an edge by the two nodes it connects, but rather by the characteristics of the communication being performed by the two nodes. Such a definition serves better in the context of attack trees since distinct pairs of nodes may be using a similar communication link (for example an ssh connection), failure or compromise of which can result in faulty (or no) communication between all such pairs. The sink node is representative of the objective that the workflow is designed to achieve. The set of paths represent the interactions between the nodes using edge characteristics. Every member $\mathcal{N}_p = N_{p1}E_{p1}N_{p2}E_{p2}\ldots n$ in this set is a sequence of alternate nodes and edge charateristics, starting with a node in N and ending at the sink node. The path composition specify how different paths interact to accomplish the objective of the workflow.

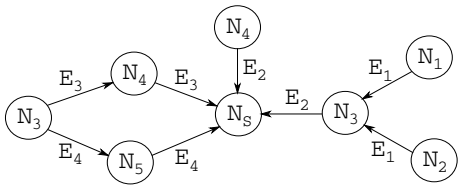

Fig. 2. A simple example workflow

Figure 2 shows a simple example workflow. The workflow consists of the nodes $N = \{N_1, N_2, N_3, N_4, N_5\}$, edge characteristics $E = \{E_1, E_2, E_3, E_4\}$ and the sink node N_s. The set of paths is marked by all possible sequences of alternating nodes and edge characteristics starting at a node with no incoming edges and ending at the sink. Thus, $\mathcal{N} = \{\mathcal{N}_1 = N_1E_1N_3E_2N_s, \mathcal{N}_2 = N_2E_1N_3E_2N_s, \mathcal{N}_3 = N_4E_2N_s, \mathcal{N}_4 = N_3E_3N_4E_3N_s, \mathcal{N}_5 = N_3E_4N_5E_4N_s\}$ is the set of paths. Note that the set of paths is not sufficient in specifying the different ways the objective of the workflow can be achieved. For example, paths $\mathcal{N}_1$ and $\mathcal{N}_2$ cannot individually help realize the objective, but can do so in conjunction. The path composition is thus used to specify any conjunctions, or disjunctions, required among paths to reach the sink. In this particular example we specify this composition as $\mathcal{N}_1 \cdot \mathcal{N}_2 + \mathcal{N}_3 + \mathcal{N}_4 + \mathcal{N}_5$, where we use the "dot" and "plus" notations to denote "AND"

and “OR” logic respectively. Besides capturing all possible ways that the objective of the workflow can be achieved, such an expression also lays the ground for the attack tree representation.

3 Modeling Attacks Using Attack Trees

Attack trees have been previously proposed [2, 8, 11, 12] as a systematic method to specify system security based on varying attacks. They help organize intrusion and/or misuse scenarios by

1. utilizing known vulnerabilities and/or weak spots in the system, and
2. analyzing system dependencies and weak links and representing these dependencies in the form of an And-Or tree.

For every system that needs to be defended there is a different attack tree[1]. The nodes of the tree are used to represent the stages towards an attack. The root node of the tree represents the attacker’s ultimate goal, namely, cause damage to the system. The interior nodes, including leaf-nodes, represent possible system states (that is subgoals) during the execution of an attack. System state can include level of compromise by the attacker (such as successful access to a web page or successful acquisition of root privileges), configuration or state changes achieved on specific system components (such as implantation of Trojan Horses) and other sub-goals that will ultimately lead to the final goal (such as sequence of vulnerabilities exploited). Branches represent a change of state caused by one or more action taken by the attacker. Change in state is represented by either AND-branches or OR-branches. Nodes may be decomposed as

1. a set of events (attacks) all of which must be achieved for a this sub-goal to succeed; this is represented by the events being combined by AND branches at the node; or
2. a set of events (attacks), any one of which occurring will result in the sub-goal succeeding; this is represented by the events being combined by OR branches at the node.

The notion of attack trees is very similar to the notion of attack graphs that have been proposed by other researchers [1, 4, 5, 10, 13, 14] for network vulnerability analysis. The difference is in the representation of states and actions. Attack graphs describe the sequence of actions that leads to attacks whereas attack trees describe attacks in terms of the sub-goals that need to be reached. Thus attack trees are a more concise representation. A often cited criticism of attack trees (vis-a-vis attack graphs) is that they are not able to model cycles. However, we believe that this criticism is valid only in cases where attack trees are used to represent sequence of operations leading to attacks, not when it is used to represent sequence of states reached. A second criticism of using attack tree to model attack scenarios is that they tend to get unwieldy. Earlier, in one of our works [11], we had shown how we can reduce the size of the attack tree so that it is usable.

[1] In real world there can be a forest of trees. However, a forest can be collapsed always to a single tree. So we will assume that there is a single tree

We augment an attack tree by associating a label $\langle b,d,u\rangle$ with each branch and node in the attack tree. The augmented attack tree is defined formally as follows:

Definition 1. *An* augmented attack tree *is a rooted tree defined as* $AAT = (V,E,\varepsilon,L)$, *where*

1. *V is the set of nodes in the tree representing the different states of compromise or sub-goals that an attacker need to reach in order to compromise a system. $\mathcal{V} \in V$ is a special node, distinguished from others, that forms the root of the tree. It represents the ultimate goal of the attacker, namely system compromise. The set V can be partitioned into two subsets, $leaf\_nodes$ and $internal\_nodes$, such that*
   (a) $leaf\_nodes \cup internal\_nodes = V$,
   (b) $leaf\_nodes \cap internal\_nodes = \phi$, *and*
   (c) $\mathcal{V} \in internal\_nodes$
2. *$E \subseteq V \times V$ constitutes the set of edges in the attack tree. An edge $(v_i,v_j) \in E$ represents the state transition (in terms of actions taken) from a child node $v_i \in V$ to a parent node $v_j \in V$ in the tree. The edge (v_i,v_j) is said to be "emergent from" v_i and "incident to" v_j. Further if edges (v_i,v_j) and (v_i,v_k) exists in the set of edges, then v_j and v_k represent the same node.*
3. *ε is a set of tuples of the form $\langle v, decomposition\rangle$ such that*
   (a) $v \in internal\_nodes$ *and*
   (b) $decomposition \in \{AND-decomposition,\ OR-decomposition\}$
4. *L is a set of opinion labels. A label $l \in L$ can be associated with a node or an edge. If $S \in V$ is a node then the opinion label l_S, associated with node* S, *is given by $l_S = w_S^{Vul}$ and is called the* opinion on vulnerability *of* S. *If $e = (v_i,v_j)$ is an edge then the opinion label l_e associated with edge* e *is given by $l_e = w_{v_i}^{Atk}$, and is called the* opinion on attacking activities *of* e. *Each opinion value w, is a tuple of the form $\langle b,d,u\rangle$, where $b,d,u \in$ [0,1] and $b + d + u = 1$, represents respectively, a belief, a disbelief and an uncertainty in the opinion as explained below.*

Definition 2. *Given a node v in an attack tree such that $v \in internal\_nodes$, the node is an* AND-decomposition *if all edges incident to the node are connected by the AND operation.*

Definition 3. *Given a node v of an attack tree such that $v \in internal\_nodes$, the node is an* OR-decomposition *if all edges incident to the node are connected by the OR operation.*

An AND-decomposition on node *v* (shown by a single arc among the edges incident to *v* in figure 3 means that each subgoal of *v* represented by a child of *v* needs to be reached in order to reach *v*. An OR-decomposition (shown by a double arc in figure 3 means that the goal *v* can be reached only if any one of the subgoals is reached. Note that reaching a child goal is only a necessary condition for reaching the parent goal and not a sufficient condition.

Henceforth we will use the terms attack tree and augmented attack tree interchangeably to mean the latter. Intuitively, the opinion on vulnerability tells us to what degree the current state of the pervasive system is vulnerable. The opinion on attacking activities is a measure of a system monitor's belief that the state transition from one vulnerable state to another will occur.

Definition 4. *Given an attack tree, AAT, an* attack scenario, *AS of AAT is defined to be a sub-tree of AAT that is rooted at the root of AAT, and follows one or more branches through the tree to end at one or more leaf nodes of AAT such that*

1. *if the subtree has a node that is an AND-decomposition then the subtree must contain all the children of this node, and*
2. *the sub-tree represents one and only one of the many attacks described by AAT.*

The following figure shows an augmented attack tree. It helps illustrate the notion of attack scenario. The shaded boxes comprise the nodes in the attack scenario.

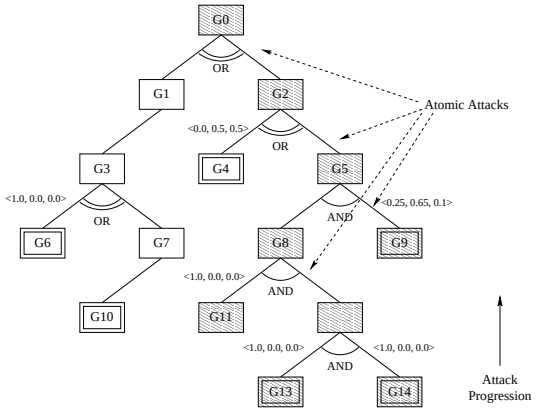

Fig. 3. A possible attack scenario

Definition 5. *An edge* (v_i, v_j) *in an attack scenario is called an* atomic attack. *The node* v_i *represents the precondition for the atomic attack and* v_j *is the exploitation of the atomic attack.*

Referring to figure 3 some of the atomic attacks have been shown by dashed arrows. Note that to achieve an atomic attack, the attacker must execute some operations that exploit one or more vulnerabilities in the system. Once a vulnerability has been identified the attacker executes a set of "attacking operations" that effectuates an atomic attack. This leads us to the following definition.

Definition 6. *A* suspicious operations set, SO^{attk}, *corresponding to an atomic attack attk, is a set of operations on specific objects that may potentially lead to the culmination of the atomic attack attk.* SO^{attk} *is a set of tuples of the form* $\langle action, object \rangle$.

We can identify two different types of operations in a suspicious operations set SO^{attk}. The first subset of operations is the set *Vul* of vulnerable operations. At least one of the operations in the vulnerable set needs to be executed to exploit a vulnerability.

An atomic attack can be launched by exploiting one or more vulnerabilities. Similarly each vulnerability can be exploited by executing one or more vulnerable operations. The second subset of operations is the set *Ao* of attacking operations. All of these need to be executed to accomplish the atomic attack.

3.1 Mapping Workflow to Attack Tree

In order to obtain an attack tree from the workflow corresponding to a pervasive computing application, we assume that the attack trees to compromise a node or an edge characteristic are already known and denoted by $AAT(N_i)$ and $AAT(E_j)$ respectively[2]. We say that $AAT(\cdot)$ is *true* if the root node of the corresponding attack tree (or the goal) has been achieved by an attacker; *false* otherwise. Further, we associate a boolean value $T(\mathcal{N}_p) = \bigvee_i A(x_i)$ to each path, where x_i is a node or an edge characteristic appearing in the path $\mathcal{N}_p$. In other words, a path is disrupted by an attacker if it compromises any of the devices or communication links appearing as part of it. If an attacker is able to disrupt every path, or a subset thereof as necessary, then the application shall have no way of realizing the objective laid down in the workflow. Hence, by replacing the paths by their boolean values in the path composition, we can obtain an expression that is analogous to the attack tree for the workflow. For the example provided in figure 2, the expression for the attack tree would become $T(\mathcal{N}_1) \cdot T(\mathcal{N}_2) + T(\mathcal{N}_3) + T(\mathcal{N}_4) + T(\mathcal{N}_5)$, with the "dot" and "plus" notations now indicating the logical boolean operations. Note that this representation of the attack tree is a boolean expression, and hence standard reduction techniques can be applied to reduce the complexity (node or edge characteristic repetitions) of the attack tree.

3.2 Opinion Model for Attack Trees

The concept of "beliefs, disbeliefs and uncertainty about opinion" is borrowed from the work on subjective logic by Jøsang [6]. In this work, an agent's *opinion* about a proposition x, $w(x)$, is defined in terms of the belief $b(x)$, the disbelief $d(x)$ and the uncertainty $u(x)$, with $b(x) + d(x) + u(x) = 1$. A particular opinion $w(x)$ is represented as a point in the opinion triangle. The triangle itself is defined by the three vertices – [0,0,1], [0,1,0] and [1,0,0] – corresponding to total uncertainty ([0,0,1]), total disbelief ([0,1,0]) and total belief ([1,0,0]) about the proposition.

The following two definitions from [6] help in forming an opinion about the conjunction and disjunction of two propositions x and y. The conjunction of the opinions of two propositions results in a new opinion reflecting the truth of both proposition simultaneously, while the disjunction of the two opinions results in a new opinion reflecting the truth of one or the other or both propositions.

Definition 7. Conjunction of Opinions *Let x and y be two propositions. Let $w_x = (b_x, d_x, u_x)$ and $w_y = (b_y, d_y, u_y)$ represent an agent's opinion about the propositions. Then the conjunction of the opinion w_x and w_y is $w_x \wedge w_y$ given by*

$$w_x \wedge w_y = (b_{x \wedge y}, d_{x \wedge y}, u_{x \wedge y})$$

[2] We have shown earlier in [11] how an attack tree can be build for a system

and satisfies the following equations.

$$b_{x\wedge y} = b_x \cdot b_y \tag{1}$$
$$d_{x\wedge y} = d_x + d_y - d_x \cdot d_y \tag{2}$$
$$u_{x\wedge y} = b_x \cdot u_y + u_x \cdot b_y + u_x \cdot u_y \tag{3}$$

Definition 8. Disjunction of Opinions *Let x and y be two propositions. Let* $w_x = (b_x,\ d_x,\ u_x)$ *and* $w_y = (b_y,\ d_y,\ u_y)$ *represent an agent's opinion about the propositions. Then the disjunction of the opinion* w_x *and* w_y *is* $w_x \vee w_y$ *given by*

$$w_x \vee w_y = (b_{x\vee y},\ d_{x\vee y},\ u_{x\vee y})$$

and satisfies the following equations.

$$b_{x\vee y} = b_x + b_y - b_x \cdot b_y \tag{4}$$
$$d_{x\vee y} = d_x \cdot d_y \tag{5}$$
$$u_{x\vee y} = d_x \cdot u_y + u_x \cdot d_y + u_x \cdot u_y \tag{6}$$

We would like to formalize the notion of an opinion on attack activities based on an opinion on vulnerabilities initially present in the pervasive system. The opinion on vulnerability represents the degree of weakness in the system. For a system to be successfully attacked it must have an initial set of vulnerabilities. As an attack exploits a particular vulnerability it proceeds to the next stage where the system is more vulnerable. This changes our opinion of vulnerability. We represent the vulnerability that results from poor design by the initial subgoal in the attack tree and vulnerabilities that result from the progression of the attack as intermediate subgoals in the attack tree.

The vulnerabilities and the attacking activities are related; the vulnerabilities are the preconditions for an attacker to perform attacking activities. Moreover, a successful attack activity exploits more vulnerabilities. This, in turn, becomes a precondition of a more advanced attack activity. This relationship is formally represented by the following definition.

Definition 9. *Let S be a node in the augmented attack tree* $AAT = (V, E, \varepsilon, L)$. *The opinion on the vulnerability of S,* ${w_S}^{Vul}$ *is defined as follows:*

1. *if S is an AND-decomposition with m branches then*

$${w_S}^{Vul} = ({w_{S_1}}^{Vul} \wedge {w_{S_1}}^{Atk}) \wedge \ldots \wedge ({w_{S_m}}^{Vul} \wedge {w_{S_m}}^{Atk})$$

2. *if S is an OR-decomposition with m branches then*

$${w_S}^{Vul} = ({w_{S_1}}^{Vul} \wedge {w_{S_1}}^{Atk}) \vee \ldots \vee ({w_{S_m}}^{Vul} \wedge {w_{S_m}}^{Atk})$$

3. *if* S *is a leaf node then* ${w_S}^{Vul} = (1,0,0)$, *a constant.*

Definition (9) above represents the relation between two opinions; the opinion on vulnerability in the pervasive system and the opinion on attacking activities. Figure 4 gives the general idea on how we use the augmented attack tree to predict attacks in the pervasive system. When a pervasive computing application is initiated, the intermediate nodes in the augmented attack tree are initiated with total disbelief about any attack.

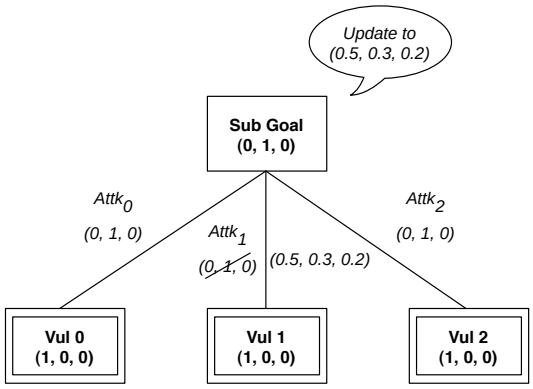

Fig. 4. Attack prediction

However, the leave nodes are initiated with full belief about an attack. This is because the leaf nodes represent the initial vulnerabilities that exist in the system ready for full and immediate exploitation.

At time t_1 say, the monitoring reports suspicious operation. The system maps the operation to the threats database to identify the corresponding atomic attacks. Assuming Attk_1 is the relevant atomic attack, the system computes the opinion on an attacking activity Attk_1. The model to compute this is formulated in Section 4. The monitoring system then starts updating the progression of attack against the attack tree. For this purpose, the system evaluates an opinion value for nodes in the attack tree. The edge associated with Attk_1 is first updated. Then the opinion value of its immediate parent-level subgoal (subgoal 1 in figure 4) is calculated using the above equations (depending on what branch-decomposition this subgoal has). This scenario is shown in figure 4 as the particular node updating the value to (.5, .3, .2). Next comes the next immediate parent-level sub goal. The updating chain continues upwardly to the root of attack tree. Eventually, the updated value at the ultimate goal (the root of the attack tree) announces the current belief in the attack on the system.

4 Quantitative Framework for Evaluating Attacks

Let SO_X be the set of all suspicious operations of an attack in a given attack tree. Let also $SO_X = Vul_X \cup Ao_X$ where Vul_X denotes the set of vulnerable operations and Ao_X is the set of attacking operations. We observe that a monitoring agent that tracks the activities of a given application can reasonably make the following predictions about a malicious attack.

Complete disbelief in an attack If no operation has been executed by the application that belongs to either Vul_X or Ao_X, then the monitor has complete disbelief in an ensuing attack.

Complete belief in an attack If all operations in Ao_X and at least one operation in Vul_X has been executed by the application, then the monitor has total belief in an ensuing attack.

Complete uncertainty about an attack If all operations in Vul_X has been executed and no operation in Ao_X has been executed then the monitor is completely uncertain about an ensuing attack.

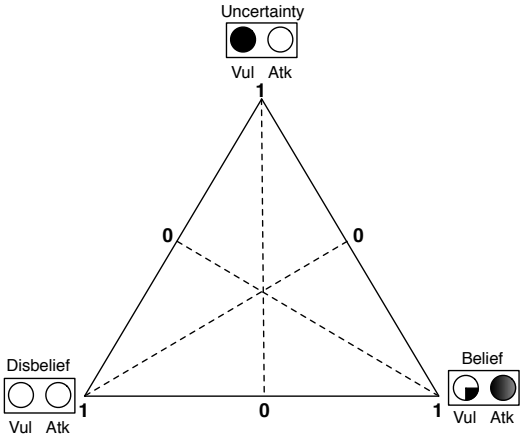

Fig. 5. Opinion thresholds for the model

We represent these three cases as the three vertices of the opinion triangle representing the monitor's opinion about an attack (see figure 5). We use the symbol D for the total disbelief vertex, B for the total belief vertex and U for the total uncertainty vertex. A point within this triangle, which occurs when some operations in both Vul_X and Ao_X has been executed, will give the monitor's opinion about an ensuing attack. The monitor has to compute this opinion based on the fraction of vulnerable operations that has been executed so far and the fraction of attacking operations. We define the following fractions.

$$m = \frac{\text{Number of vulnerable operations executed so far}}{\text{Total number of operation in } Vul_X} \tag{7}$$

$$n = \frac{\text{Number of attacking operations executed so far}}{\text{Total number of operations in } Ao_X} \tag{8}$$

The monitor always starts from the disbelief vertex of the opinion triangle. Since at least one vulnerable operation needs to be executed before any attacking operation can be executed, the opinion of the monitor moves along the $\overline{DU}$ side of the opinion triangle initially. As and when attacking operations gets executed the opinion point begins to move towards the edge $\overline{BU}$. This leads to the following observation.

Observation 1 *The fraction m tends to pull the opinion about an attack towards uncertainty while the fraction n tends to pull the opinion towards belief. The opinion has an initial inertia that tends to keep it at the disbelief end. At any instance the interaction of these three forces keep the opinion in equilibrium.*

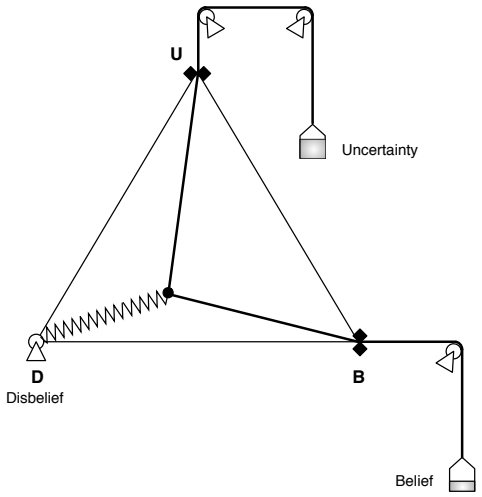

Fig. 6. Analogy of opinion about attacks with equilibrium of forces

This scenario closely resembles the following equilibrium condition arising in mechanics (see figure 6). A ring is attached to a spring which in turn is attached to the vertex D of an equilateral triangle. At the vertices U and B of the triangle are two ideal pulleys that connect two sand buckets to the ring. When the sand buckets are empty the spring keeps the ring at D. As the sand buckets get filled with sand they exert forces on the ring. These forces stretch the spring and move the ring towards U or D. The resulting tension in the string determines the position of the ring within the triangle which can be calculated using the laws of mechanics. In particular, Hook's law gives us that tension in a spring is equal to the product of the stiffness constant of the spring (k) and the elongation of the spring ($\triangle$).

We observe that in our model the analogues of the forces exerted by the sand buckets are the fractions m and n from equations 7 and 8. To model our system we need to define the equivalents of the following things: (i) the perimeter of the triangle, (ii) the force exerted by the two sand buckets and (iii) the tension in the spring. The opinion triangle is an equilateral triangle each of whose three sides are 1 unit in length. The resultant of the forces exerted by the two sand buckets can be computed as follows.

Consider the case when all vulnerable operations in the set Vul_X has been executed and no operation in Ao_X has been executed. This corresponds to the case when $m = 1$ and $n = 0$. In this case the opinion point will be at the vertex corresponding to total

uncertainty. Thus the spring will be extended along the $\overline{DU}$ edge of the opinion triangle and have a displacement $\triangle$ =1.0 unit. Then the effect of full uncertainty can be expressed by the formula $F_u = k \cdot 1.0$ where k is some constant. Thus, at any instance, the effect of a fraction m of total uncertainty is expressed by the equation:

$$F_u = m \cdot k \tag{9}$$

The effect of total belief needs to be modeled a bit differently. This is because in order for some belief to arise in the opinion, not only some attacking operations need to be executed but some vulnerable operations as well. Moreover at least one vulnerable operation needs to be executed prior to execution of any attacking operation. The effect of execution of any vulnerable operation is to move the opinion point towards uncertainty. Thus, following the same reasoning as for the uncertainty case, if $\lambda \cdot k$ is the extension of the spring along the $\overline{DB}$ side of the opinion triangle, where $0 < \lambda \leq 1$ is a known constant, then the following expression holds for the effect of a fraction n of total belief on the opinion.

$$F_b = n \cdot \lambda \cdot k \tag{10}$$

At some instance the opinion point will be somewhere within the triangle. We use polar coordinate systems to represent the point within the triangle. Let the edge $\overline{DB}$ represent the X-axis and the point D be the origin. Then any opinion point P in the polar coordinate system can be represented as $P = (\triangle,\theta)$, where $\triangle$ is the point's displacement from the origin and θ is the angle of the displacement with the X-Axis.

Once the polar coordinates $(\triangle,\theta)$ of an opinion point, P, is known we can calculate the corresponding values for disbelief, uncertainty and belief using the laws of trigonometry (see figure 7).

The three values can be expressed in terms of the following equations.

$$u = \frac{\triangle \sin\theta}{\sqrt{3}/2} \tag{11}$$

$$d = 1 - \frac{\triangle \cos|30-\theta|}{\sqrt{3}/2} \tag{12}$$

$$b = 1 - d - u \tag{13}$$

At any given instance the opinion point, P, is in "equilibrium" within the opinion triangle. In other words, the combined effect of the pull in the uncertainty direction and the pull in the belief direction is balanced by the inertial pull towards the disbelief direction and the resulting effect of the pulls is zero. If we call F_d to be the pull towards the disbelief direction, then the sum of the X-axis components of F_u, F_b and F_d is equal to zero. Similarly the sum of the Y-axis components of F_u, F_b and F_d is zero. We compute these in terms of the angles that the belief, disbelief and uncertainty pulls make with the X-axis, namely α, θ and β respectively (see figure 7).

It can be shown that under equilibrium condition the following equations hold.

$$\lambda\, n\, \cos\alpha \;+\; m\, \cos\beta \;-\; \triangle\, \cos\theta \;=\; 0 \tag{14}$$

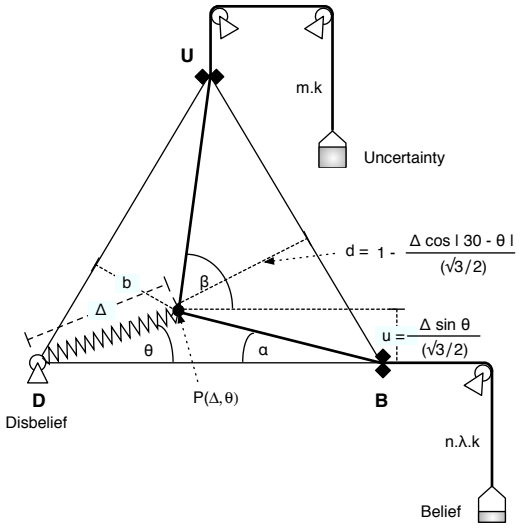

Fig. 7. Computing the values for b, d and u from polar coordinates

$$\lambda\, n\, \sin\alpha \;+\; \triangle\, \sin\theta \;-\; m\, \sin\beta \;=\; 0 \tag{15}$$

Equations 14 and 15 are in terms of the angles α and β. We have to find the value of these angles in terms of $\triangle$ and θ. At this point we note that when the opinion values crosses a certain thershold, executing just vulnerable operations should force the system into high alert state. We set the threshold point to be the midway mark between *D* and *B*. With this observation we can show that α and β can be expressed by the following equations.

$$\alpha = \arctan\frac{\triangle\sin\theta}{1-\triangle\cos\theta} \tag{16}$$

$$\beta = \arctan\frac{\sqrt{3}/2-\triangle\sin\theta}{|1/2-\triangle\cos\theta|} \tag{17}$$

Finally, substituting the values for α and β in equations 14 and 15 gives us the following two equations.

$$\lambda\, n\, \cos(\arctan\frac{\triangle\, \sin\theta}{1\;-\;\triangle\, \cos\theta}) + \;-\; \triangle\, \cos\theta \\ m\, \cos(\arctan\frac{\sqrt{3}/2\;-\;\triangle\, \sin\theta}{|1/2\;-\;\triangle\, \cos\theta|}) = 0 \tag{18}$$

$$\lambda\, n\, \sin(\arctan\frac{\triangle\, \sin\theta}{1\;-\;\triangle\, \cos\theta}) + \;\triangle\, \sin\theta\;- \\ m\, \sin(\arctan\frac{\sqrt{3}/2\;-\;\triangle\, \sin\theta}{|1/2\;-\;\triangle\, \cos\theta|}) = 0 \tag{19}$$

From the above modeling we observe that by counting how many of the different types of operations have executed in the pervasive application we can formulate an opinion about the current state of compromise of the system. The complete framework works as follows. When the application starts executing operations, the monitor determines the values of the fractions *m* and *n*. This gives rise to the opinion values for branches of the attack tree that are emergent from leaf nodes using equations 18, 19, 11, 12 and 13. These values can be computed all the way up to the root of the attack tree using further the theorems on conjunction and disjunction of opinions. Thus at any point in time, while an application is executing operations, we can estimate what the chances are that this particular application can cause a compromise of the pervasive system.

5 Conclusions and Future Work

In this paper, we have presented a quantitative model for evaluating the chances of an attack occurring in a pervasive computing environment. We make two main contributions. First we develop an augmented attack tree model for representing attacks in a pervasive application. Next we build an opinion model for attacks that is based on monitoring, classifying and counting the different activities that is going on in the application. The opinion model provides three measures, a belief value that an attack is ensuing, a disbelief value that there is an attack and an uncertainty value regarding the other two values. We believe that these three values together gives a good picture of the state of compromise of the pervasive application. Since these values are continuously generated based on signals from the monitoring system, an administrator can use these values to adapt the security controls to the changing scenario.

The challenge is to validate the model in a real world scenario and that is our planned next step for this work. The problem we are facing is the lack of real world data for pervasive applications. As a first step we are using the DARPA Intrusion Detection System Evaluation dataset [7] in a simulated pervasive computing environment. The DARPA dataset will be used to provide a stream of activity records for the application. Results from this evaluation will be presented at a future venue.

Acknowledgment

This work has been supported in part by a grant from the U.S. Air Force Office of Scientific Research (AFOSR) under contract FA9550-07-1-0042. The statements and opinions expressed in this work and the conclusions drawn are those of the authors and do not represent those of the AFOSR or any other federal agencies or their representatives.

References

[1] P. Ammann, D. Wijesekera, and S. Kaushik. Scalable, graph-based network vulnerability analysis. In *Proceedings of the 9th ACM Conference on Computer and Communications Security, Washington DC.*, pages 217–224, November 2002.

[2] J. Dawkins, C. Campbell, and J. Hale. Modeling network attacks: Extending the attack tree paradigm. In *Proceedings of the Workshop on Statistical Machine Learning Techniques in Computer Intrusion Detection, Baltimore, MD*. Johns Hopkins University, June 2002.

[3] J. Elson and D. Estrin. Sensor networks: A bridge to the physical world. In C.S. Raghavendra, K.M. Sivalingam, and T. Znati, editors, *Wireless Sensor Networks*. Kluwer Academic Publishers, 2004.

[4] S. Jha, O. Sheyner, and J. Wing. Minimization and reliability analysis of attack graphs. Technical Report CMU-CS-02-109, School of Computer Science, Carnegie Mellon University, February 2002.

[5] S. Jha, O. Sheyner, and J. Wing. Two formal analyses of attack graphs. In *Proceedings of the 2002 Computer Security Foundations Workshop, Nova Scotia*, pages 45–59, June 2002.

[6] A. Jøsang. A logic for uncertain probabilities. *International Journal of Uncertaity, Fuzziness and Knowledge-Based Systems*, 9(3), June 2001.

[7] R. P. Lippmann, D. J. Fried, I. Graf, J. W. Haines, K. Kendall, D. McClung, D. Weber, S. Webster, D. Wyschogrod, R. K. Cunningham, and M. Zissman. Evaluating Intrusion Detection Systems: The 1998 DARPA Off-line Intrusion Detection Evaluation. In *Proceedings of the DARPA Information Survivability Conference and Exposition*, Los Alamitos, CA, January 2000.

[8] A.P. Moore, R.J. Ellison, and R.C. Linger. Attack modeling for information survivability. Technical Note CMU/SEI-2001-TN-001, Carnegie Melon University / Software Engineering Institute, March 2001.

[9] A. Perrig, J. Stankovic, and D. Wagner. Security in wireless sensor networks. *Communicaqions of the ACM*, 47(6):53–57, June 2004.

[10] C. Phillips and L.P. Swiler. A graph-based system for network-vulnerability analysis. In *Proceedings of the 1998 New Security Paradigms Workshop, Chicago, IL*, pages 71–79, January 1998.

[11] I. Ray and N. Poolsappasit. Using Attack Trees to Identify Malicious Attacks from Authorized Insiders. In *Proceedings of the 10th European Symposium on Research in Computer Security*, Milan, Italy, September 2005.

[12] B. Schneier. Attack trees: Modeling security threats. *Dr. Dobb's Journal*, December 1999.

[13] O. Sheyner, J. Haines, S. Jha, R. Lippmann, and J. Wing. Automated generation and analysis of attack graphs. In *Proceedings of the 2002 IEEE Computer Society Symposium on Security and Privacy, Oakland, CA*, May 2002.

[14] L. Swiler, C. Phillips, D. Ellis, and S. Chakerian. Computer-attack graph generation tool. In *Proceedings of DISCEX '0: DARPA Information Survivability Conference and Exposition II*, pages 307–321, June 2001.