Towards A Theory Of Insider Threat Assessment*

Ramkumar Chinchani, Anusha Iyer, Hung Q. Ngo, Shambhu Upadhyaya
University at Buffalo
Buffalo, NY 14260, USA
Email: {rc27, aa44, hungngo, shambhu}@cse.buffalo.edu

Abstract

Insider attacks are a well-known problem acknowledged as a threat as early as 1980s. The threat is attributed to legitimate users who abuse their privileges, and given their familiarity and proximity to the computational environment, can easily cause significant damage or losses. Due to the lack of tools and techniques, security analysts do not correctly perceive the threat, and hence consider the attacks as unpreventable. In this paper, we present a theory of insider threat assessment. First, we describe a modeling methodology which captures several aspects of insider threat, and subsequently, show threat assessment methodologies to reveal possible attack strategies of an insider.

1 Introduction and Motivation

Insider threat is typically attributed to legitimate users who maliciously leverage their system privileges, and familiarity and proximity to their computational environment to compromise valuable information or inflict damage. According to the annual CSI/FBI surveys conducted since 1996, internal attacks and insider abuse form a significant portion of reported incidents. The strongest indication yet that insider threat is very real is given by the recent study [2] jointly conducted by CERT and the US Secret Service; the first of its kind, which provides an in-depth insight into the problem in a real-world setting. However, there is no known body of work which addresses this problem effectively. There are several challenges, beginning with understanding the threat.

- **Insider threat is a low base rate problem.** Perpetrators of insiders attacks are users with legitimate authorization, and therefore, it is difficult to predict or protect against these attacks. Consequently, security officers view these attacks as unpreventable, resulting in inaction.
- **Insider threat is misperceived.** Organizations often concentrate on external attacks, almost exclusively, mainly because security audit tools and modeling techniques are readily available which aid in finding vulnerabilities and fixing them. On the other hand, insider threat is not correctly perceived because it is difficult to *measure* it, and the lack of tools and techniques doesn't help the situation. Therefore, any good model or assessment methodology is already a significant advance.
- **Insider threat is high impact.** Although insider attacks may not occur as frequently as external attacks, they have a higher rate of success, can go undetected and pose a much greater risk than external attacks. This is due to the fact that insiders enjoy certain important advantages over external adversaries. They are familiar about their targets and the security countermeasures in place. Therefore, very damaging attacks can be launched with only a short or non-existent reconnaissance phase.

In a nutshell, insider threat is a complex problem involving both computational elements and human factors. As a long-term process to mitigate this threat, steps such as pre-hire screening of employees, training and education can be undertaken. While all these practical measures will reduce the threat, they cannot eliminate it altogether, and some incidents can still occur. A possible solution it would seem is an overall increase in monitoring, logging and security countermeasures. However, it only leads to general inconvenience. Moreover, in an organization, it sends wrong signals of distrust between the management and the employees. We seek a methodology by which very specific and targeted countermea-

*Research supported in part by Telcordia Technologies Subcontract: FA8750-04-C-0249 from the DARPA SRS Program

Proceedings of the 2005 International Conference on Dependable Systems and Networks (DSN'05)
0-7695-2282-3/05 $20.00 © 2005 IEEE

sures can be deployed. This approach occupies a sweet spot between complete inaction and intrusive solutions. Central to such an approach is an effective threat modeling methodology, accompanied by threat assessment and analysis, with the goal of discovering likely tactics and strategy of an adversary so that appropriate countermeasures can be taken.

Insiders can cause damage either by: 1) remaining within their default set of privileges, or 2) exceeding them by seeking new information and capability through a repertoire which contains not only common attacks but also unconventional ones such as social engineering. The problem of insider threat assessment is precisely the problem of evaluating the damage which can potentially occur in these two cases.

Threat assessment methodologies are not new in general and techniques such as attack graphs [16, 19, 21, 20] and privilege graphs [7, 17] are already known. However, these techniques have been proposed to primarily model external attacks, and hence, have a limited appeal to insider threat. Moreover, there are also scalability concerns regarding both model specification as well as subsequent threat analysis. Specifying a model requires information in very exacting detail, making it impractical to generate the model manually. Instead, current approaches generate models automatically [20] via information obtained from live penetration testing of an organization network. However, given the possibility of systemic failures, a large part of the network is typically excluded during testing, resulting in an abbreviated model instance. Consequently, any further inferences drawn from the model are questionable. Also, threat analysis following model specification very quickly runs into the problem of intractability. To summarize, these modeling techniques are not suitable for addressing insider threat both for technical and practical reasons. We seek to devise a more appropriate modeling and assessment methodology.

1.1 Summary of Contributions

There are two prominent contributions in this paper. As our first contribution, we propose a new threat model called *key challenge graph*. The main idea behind insider threat modeling is to focus on a legitimate user's view of an organization's network. In order to estimate the threat when insiders remain within their privilege levels, we only need to represent the basic network connectivity and access control mechanisms that are in place. Additionally, to assess the threat when insiders exceed their privilege levels, we also need to represent knowledge and location of key information and capability, not normally accessible, which may assist him in his attack. The overall goal, like attack graphs, is to understand insider threat from a global perspective rather than just single-point incidents. In terms of threat variety, our model supports not only conventional attacks, but also more complex scenarios such as social engineering. One important design consideration is the granularity of information and the nature of representation. Due to the unavailability of tools to scan for weaknesses in the context of insider threat, a significant portion of the model specification task can fall upon the security analyst. Our modeling methodology allows models to be manually specified and the resulting model instances are only polynomially-sized in the input information. To demonstrate applications of our model, we have constructed some typical scenarios motivated by the CERT/USS insider threat study which analyzed threats in the banking and financial sector.

As our next contribution, we investigate and analyze the problem of automated threat analysis. It turns out that the problem is NP-hard. Nevertheless, we have designed two algorithms for this purpose - one which solves the problem to optimality but takes exponential time, and the other which is a polynomial-time heuristic. We benchmark the algorithms for scalability and quality of threat analysis. The impact of threat analysis is manifold. Given a organization network and its people, it is possible to assess whether existing security countermeasures are adequate. If not, threat analysis allows recommendations to be made to improve security. In the face of insider threat, sometimes the only countermeasure is installing monitoring and logging systems for non-repudiability. Finally, if several intrusion detection systems are installed, threat analysis can also assign appropriate weights to intrusion detection sensors based on the likely areas of insider activity inside the organization.

Paper Organization. The rest of the paper is organized as follows. We present our model in Section 2 and show its application on representative illustrations in Section 3. Next, we describe the threat analysis methodology in Section 4 and also present insights into the complexity of the problem. Related work is discussed in Section 5 and finally, closing remarks are in Section 6.

2 Modeling Insider Threat

In this section, we elaborately discuss our modeling methodology. But before that we state some working assumptions based on generally accepted notions about

insider threat along with results from the recent study [2].

2.1 Background

Defining the term "insider" in an airtight manner is hard because the boundary between insiders and outsiders is fuzzy. We assume that every legitimate user is an insider. Note that the term “insider” can have both physical and logical connotation. Physical outsiders can be logical insiders and vice versa. For example, an authorized user who may be physically far away from an organization but has wireless or VPN connectivity. Similarly, users may be physically inside an organization but have no authorized access to use the computation infrastructure. Insiders are in a unique position with the privileges entrusted to them and the knowledge about their computational environment, and this already translates directly to a certain amount of capability. Insider abuse can occur within this default capability, but more dangerous scenarios occur when an insider widens his realm of capability. Since insiders have access privileges to use the computational infrastructure, it represents resources at their disposal that can be used against the parent organization, so resources for an insider attack are freely available. Unlike external attackers who use the Internet as an umbrella of anonymity and can be sloppy, insiders have a strong incentive to avoid detection. They are a part of an organization and bound by the organization policy, and if caught, an organization has all the necessary information about the insider and the legal resources to prosecute him. External attackers can become insiders too by compromising an internal system and learning about the computers in the neighborhood. However, there is an inherent risk to the attacker that the compromise may be discovered and the corresponding security hole patched.

The insider threat study [2] reports that financial gain is the main motivating factor behind most insider attacks; any other motive is simply not worth the risk. The financial gain can be realized in different ways depending on the organization. In a financial institution such as a bank, likely targets are customer account records or perhaps company accounts, where there is a direct access to funds. In other cases, an insider may not obtain immediate monetary gain, such as in a software company where the real value lies in the proprietary software code. While it is possible to envision several other scenarios, it is not realistic to expect that each and every one of them can be modeled, mainly because it entails a significant effort on the part of the security officer.

2.2 Our Model

In our model, we assume that an attacker is goal-oriented. Also, he is already aware of the location of his potential targets and how to reach them, obviating the need for reconnaissance. These assumptions closely model an insider and this is one of the reasons why our model is most suitable for this class of threats. We also assume that a successful compromise of a target is not possible if there is no channel of interaction. Finally, an attacker may not be able to use an existing channel of interaction with a potential target due to a strong security mechanism in place on that channel. This may force him to seek alternate routes to reach the target. Each sub-target that is compromised requires extra effort but can provide the attacker with additional information/capability and another front to continue the attack. Given a model specification, the goal of vulnerability analysis is to exhaustively find the different ways in which attacker can reach the target.

Preliminaries. Prior to the formal definition of our model, which we call a *key challenge graph*, we describe the various components. Figure 1 shows the basic building block of the key challenge graph.

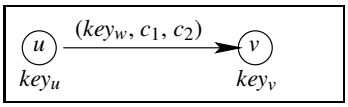

Figure 1. Basic building block of a key challenge graph

- Any physical entity on which some information or capability can be acquired is represented as a *vertex* of the graph. Let the set of vertices be denoted by V. Typically, vertices are points in the network where some information may be gained such as a database server or simply any computer system whose resources can be used or misused.
- Each piece of information or capability that is present at any vertex is represented as a *key*. Let the set of keys be denoted as K. For example, records in a database, passwords stored on a computer, or computational resources of a computer can be represented as keys. When an attacker visits a vertex, he is empowered with this additional information or capability. Note that this should not be confused with cryptographic "keys". A key in our model is only an abstraction.
- If there is a channel of access or communication between two physical entities which facilitates inter-

action, then a *directed edge* is created between the two corresponding vertices, pointing to the direction of the allowed interaction. Multiple channels of communication are possible, hence there can be more than one edge between two vertices. Let the set of edges be denoted by E. For example, assume a ssh server and a client computer. A channel of communication exists from the client to the server and a directed edge is drawn.

- The presence of a security measure or an enforced security policy protects the resources and allows only authorized interaction. This deterrence is represented as a *key challenge* on the corresponding channel of communication. An example of a key challenge is the password authentication required prior to accessing to a server. A key challenge is an abstraction to capture access control.

- If a user does not have the right key to the key challenge, then he incurs a significant *cost* in breaking or circumventing the security policy; legitimate access incurs only a smaller cost of meeting the key challenge. For example, when password authentication is used, if a user knows the password, he incurs little or no cost, while another user who doesn't know the password will incur a higher cost in breaking the password. The cost metric is a relative quantity signifying the amount of deterrence offered by one security measure over another. It has been abstracted as a non-negative integer for the purposes of our model.

- The starting point of an attack could be one or more vertices in the graph, which are assumed to be initially in the control of an adversary. Let this set be denoted as V_0.

- The target of an attack could also be one or more vertices in the graph. In case of multiple targets, the goal is to compromise all of them. Let the set of target vertices be denoted by V_s. An example of a target is a source code repository for a commercial product.

Table 1 provides a summary of all the abstractions captured by our model.

Definition 2.1 (Key Challenge Graph). A *Key Challenge Graph* or KG is a tuple:

$$KG = (V, E; K, V_0, V_s, \pi, \delta),$$

where V is the set of vertices, E is the set of edges, V_0 is the initial set of compromised vertices, V_s is the set of target vertices, $\pi : V \rightarrow K$ is a function that assigns keys to vertices, $\delta : E \rightarrow K \times \mathbb{N} \times \mathbb{N}$ is a function that assigns key challenges and costs to edges, and $\mathbb{N}$ is the set of natural numbers.

Model Component	Abstraction
Vertex	Hosts, People
Edge	Connectivity, Reachability
Key	Information, Capability
Key Challenge	Access Control
Starting Vertex	Location of insider
Target Vertex	Actual target
Cost of Attack	Threat analysis metric

Table 1. Model components and the captured abstractions

For example, $\pi(v_1) = k_0$ means that the key k_0 can be obtained at vertex v_1, $\delta(e_1) = (a, c_1, c_2)$ implies an assignment of a key challenge to edge e_1, which requires an attacker to produce the key a. If he cannot do so, then he incurs a cost c_1, which could be significant depending on the access control mechanism; otherwise, he incurs a smaller cost c_2.

An adversary begins his attack at some point in the set of compromised nodes in the graph and proceeds by visiting more and more vertices until the target(s) is reached. At each visited vertex, the attacker adds the corresponding key to his collection of keys picked up at previous vertices. Once an attacker compromises a vertex, he continues to have control over it until an attack is completed. Therefore, any vertex appears exactly once in the attack description. While a trivial attack can be performed by visiting all vertices until the target is reached, cost constraints occlude such free traversals. We shall now develop useful metrics for analysis of our model.

Lets assume that initially V_0 and V_s are disjoint sets, that is, an attacker has not successfully compromised any targets yet. We can define a successful attack as follows.

Definition 2.2 (Successful Attack). A *successful attack* is defined as a finite ordered sequence of a subset of vertices $(v_1, v_2, \ldots, v_m)$, where $V_s \subseteq \{v_1, \ldots, v_m\}$, and $V_0 \cap \{v_1, \ldots, v_m\} = \emptyset$.

In other words, a successful attack is a sequence of zero or more vertices not in the initial set V_0 but eventually containing all the target nodes in V_s. (Note that this sequence in general does not represent a path or a walk.

We elucidate this point in illustrations in the following sections.)

The next important aspect of the model is the cost metric. Although an attack is defined exclusively in terms of vertices, the cost incurred by the attacker at a vertex is mainly dependent on the edge that he chooses to visit the vertex. We first define the cost of traversing an edge and then the cost of visiting a new vertex. The latter is the basic unit of cost metric in our model.

Definition 2.3 (Cost of Traversing an Edge). Let V^* be the set of visited vertices so far, including the initially compromised vertices, i.e. $V_0 \subseteq V^*$. For $u \in V^*$ and $v \notin V^*$, the *cost of traversing the edge* $e = (u,v) \in E$, given that $\delta(e) = (k, c_1, c_2)$, is c_1 if $k \notin \{\pi(w) \mid w \in V^*\}$. Otherwise, it is c_2. (In general, $c_1 > c_2$.) If $(u,v) \notin E$, for technical convenience we assume that $c_1 = c_2 = \infty$, and that k is some unique key no node has.

Definition 2.4 (Cost of Visiting a New Vertex). Define V^* as above. The *cost of visiting a new vertex* $v \notin V^*$ is defined to be

$$c(v, V^*) = \min\{\text{cost of traversing } (u,v) \mid u \in V^*\}. \quad (1)$$

Let NEW-VERTEX-COST be a procedure that computes this value. (Note that the cost of traversing an edge is implicit in this computation.)

The cost of an entire attack is measured as a sum of the effort required to compromise individual vertices by attempting to counter the key challenges on the edges with or without the keys that an attacker has already picked up.

Definition 2.5 (Cost of an Attack). The *cost of an attack* $v_1 v_2 \ldots v_m$ is defined as:

$$\sum_{i=1}^{m} c\left(v_i, V_0 \cup \{v_1, \ldots, v_{i-1}\}\right). \quad (2)$$

We will call this computation ATTACK-COST.

3 Modeling Methodology And Applications

In this section, we describe the applications of our modeling methodology. First we dispel concerns which are normally attributed to most theoretical modeling methodologies regarding their practicality. Later, we demonstrate through illustrations the relevance of our model in capturing different types of threat scenarios.

3.1 Practical Considerations

One major benefit of using theoretical models is that they are inexpensive and do not require actual system implementation and testing. However, such benefits can be offset if the model is difficult to use or if several facets of the modeling methodology are unclear. We answer some of outstanding questions which may arise.

How is a model generated? Model specification begins by identifying the scope of the threat; it could be a very small part of the organization or the whole organization itself. The algorithm BUILD-MODEL below gives a step-by-step procedure to construct a model instance.

BUILD-MODEL(Network Information)
1. Identify potential target nodes denoted by set T.
2. $\forall v \in T$, identify all hosts/people denoted by u having access to v.
   Add (u,v) to the set of edges E.
3. $\forall (u,v) \in E$, identify key challenges and calibrate costs.
   Add the key challenge to the set δ.
4. $\forall$ keys in δ, identify nodes containing these keys.
   Add each such node to T and goto Step 1.
5. Repeat until no new nodes are added to T.

Who constructs the model? The model is constructed by someone who is aware of the organization network and location of various resources. This is typically the system administrator and/or the security analyst. Note that for the purposes of evaluating security, we assume that whatever a security analyst can model, an insider can model as well. In terms of the time and effort required to construct the model, since our model takes a high-level view of the network, the model instance is not significantly larger than the actual network representation. Given a OPNET-like tool to assist in instantiating a model, we expect that a security analyst will not have to invest a substantial effort. We are currently implementing one such tool.

How are costs defined? Costs in our framework are a metric representative of the resistance put up by an access control mechanism. In cases such as cryptographic access control mechanisms, there are standards to go by. For example, the strength of a cryptographic algorithm is indicative of the time required for an adversary to break it. However, in other cases, the solution may be more systemic such as intrusion detection systems, where the actual costs may not be very clear. In such cases, a value relative to known standards can be used. Note that it is quite possible for two security analysts to assign different costs and it will depend on what each

perceives is the more appropriate value.

3.2 Illustrations

We now turn to application of our model to specific examples. We have performed small scale modeling of banking environments based on [1] and [2].

A Financial Institution Example. Consider the following description. From his workstation, every teller can perform sundry personal accounting tasks. Each personal account transaction cannot exceed US$ 5,000. A manager has to endorse any larger transactions and personally handles accounts involving business organizations. The business accounts database is kept separate from the one that houses personal accounts. Any bank transaction, either by the teller or the manager, which operates on a database is encrypted. The communication between the teller's computer and the personal accounts database is frequent but not of high value, so it uses a lower strength symmetric encryption. On the other hand, the manager authenticates himself to a PKI server with his credentials, and obtains a session key to talk to the database and complete the transaction. Both the databases are protected behind a firewall to prevent any external attacks. Another key piece of information known to the teller is that the manager doesn't apply security patches to his computer frequently, and that the manager's computer could be vulnerable.

Now, consider an insider threat that manifests in the form of a rogue teller and the target being the business account database. Using the modeling methodology described earlier, we convert this description into our model (shown in Figure 2). Using our KG model

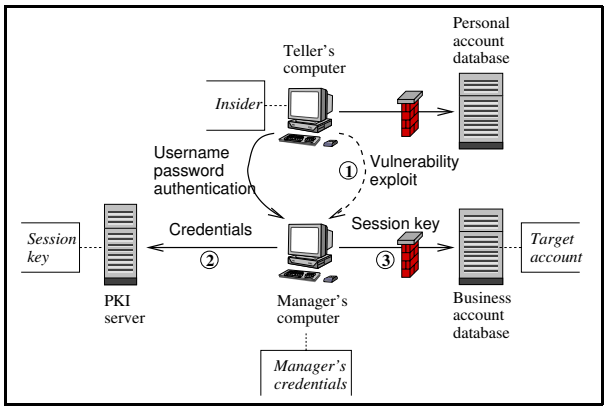

Figure 2. Modeling an example insider threat scenario

representation, the steps taken by an insider (shown numerically in Figure 2) can be easily seen. The most likely sequence of steps is: 1) use a vulnerability exploit to compromise the manager's computer, 2) use the manager's credentials to obtain a session key from the PKI server, and 3) use the session key to attack the business account database. Simply with a key piece of information, an insider is able to launch a very damaging attack, and our KG model is able to provide a very intuitive view of the attack. The solution in this scenario is trivial, i.e., the manager's computer is to be patched.

We point out a few properties based on the example. The sequence of steps taken by the attacker is generally *not* a path. This is a striking departure from the attack graph model, where attacks appear as paths in the graph. Also, note that a very damaging insider attack is possible even with only one vulnerability in the network description.

Social Engineering Attacks. The KG model allows for a very general kind of communication channel. This means that it is possible to represent not only wired and wireless network media, but also channels such as telephone lines, and this still falls within the framework of our model. For example, when a customer calls a credit card company, a representative poses a key challenge in the form of date of birth or social security number, and divulges information only when the right information is presented.

Colluding Insiders. In order to improve the chances of a successful attack, two or more attackers controlling different vertices may collude and share the keys that they possess. In this case, the set V_0 contains all these vertices and $|V_0| > 1$. This situation is no different from the one where a single attacker may initially have control over multiple vertices. This would not complicate the analysis of the model as an attack is represented not as a path but rather as a sequence of compromised vertices.

4 Threat Analysis

In this section, we address various aspects of threat analysis. Looking at this task from a algorithmic complexity viewpoint, a good attacking strategy for an insider is in some sense equivalent to a good algorithm or approximation algorithm [14] to find a minimum-cost attack on the key challenge graph. An insider is unlikely to adopt any other strategy because it will lead to an attack that is easily detected. But first, we are interested in knowing the computational difficulty of analyzing a general instance of a key challenge graph. This provides very useful insight into the problem based on which automated threat analysis algorithms can be developed.

Proceedings of the 2005 International Conference on Dependable Systems and Networks (DSN'05)
0-7695-2282-3/05 $20.00 © 2005 IEEE

4.1 On the Complexity of Analyzing Key Challenge Graphs

Given a key challenge graph, we may consider the problem of finding an attack with the minimum cost. Recall that an attack is a sequence of vertices $(v_1,\ldots,v_m)$ such that $V_s \subseteq \{v_1,\ldots,v_m\}$. The objective function is the cost of orchestrating the attack. We will call this problem KEYSEQ. We shall first show that KEYSEQ is NP-hard by showing that its decision version is NP-complete [10]. The decision version of KEYSEQ asks if there is an attack whose cost is at most some given positive integer C. (In fact, the optimization problem KEYSEQ is difficult even to approximate up to a certain ratio.)

Lemma 4.1. *The decision version of* KEYSEQ *is* NP-*complete.*

Proof. First, we prove that the decision version of KEYSEQ is in NP. Given an instance $G = (V,E,K,V_0,V_S,\pi,\delta)$ of the problem, and a cost upper bound C, one can guess an attack $\mathcal{S} = (v_1,\ldots,v_m)$, where $v_i \in V$. The verification that this is a good attack can be done by checking that $V_s \subseteq \{v_1,\ldots,v_m\}$, and that the total cost of the attack is at most C using formulas (1) and (2). This verification certainly can be accomplished in polynomial time (at most $O(|V|^2)$).

Next we prove that KEYSEQ is NP-hard via a reduction from 3-SAT [10]. In the 3-SAT problem, one is given a set of n boolean variables $X = \{x_1,...,x_n\}$ and a 3-CNF formula ϕ over X. The formula ϕ consists of clauses C_1, ..., C_m of size 3. The problem is to determine if there is a truth-assignment to x_1, ..., x_n such that ϕ is evaluated to be true. We construct an instance of KEYSEQ corresponding to ϕ as follows. Let the set of vertices V consist of the following nodes: 1) the literals $x_1,...,x_n$, and $\bar{x}_1,\ldots,\bar{x}_n$, 2) the clauses C_1, ..., C_m, 3) a starting state v_0, i.e., $V_0 = \{v_0\}$, 4) a success state v_s, i.e $V_s = \{v_s\}$, and 5) $n-1$ intermediate nodes v_1, .. v_n. Let the set of keys be $K = V$, and the vertex to key mapping π be the identity function. The set of edges E and the corresponding key challenges are constructed as follows:

1. for $i = 1,\ldots,n$, construct the following edges: (v_{i-1},x_i) with $\delta(v_{i-1},x_i) = (x_i,1,0)$, $(v_{i-1},\bar{x}_i)$ with $\delta(v_{i-1},\bar{x}_i) = (\bar{x}_i,1,0)$, (x_i,v_i) with $\delta(x_i,v_i) = (v_i,1,0)$, and $(\bar{x}_i,v_i)$ with $\delta(\bar{x}_i,v_i) = (v_i,1,0)$. The main idea is that, to get from v_{i-1} to v_i, one has to pay a cost of at least 2, while obtaining at least one of the keys x_i or $\bar{x}_i$. If both of these keys are obtained, then it must have been the case that a cost of at least 3 was paid.
2. three edges (v_n, C_1) each representing a variable in the clause C_1. The key challenge on each of these edges is of the form $(l_i,\infty,0)$, where l_i a literal in C_1. The infinity cost could be any large enough integer ($\geq 3n$, e.g.).
3. similarly for $j = 2,\ldots,m$, three edges (C_{j-1},C_j), and literals in the clause C_j appear in the key challenge.
4. A final "free" edge (C_m,v_s) signaling that all constraints have been met. ("Free" here means both costs are zero, having the key challenge or not.)

It is now straightforward to see that there is an attack of cost $\leq 2n$ in this instance *iff* ϕ is satisfiable. □

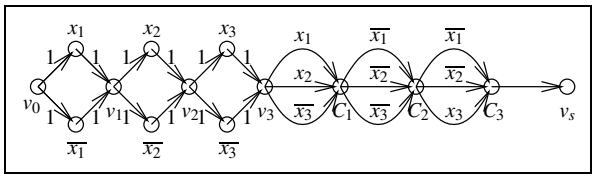

Figure 3. An example reduction from 3-SAT to KEYSEQ

Figure 3 is a short example showing a reduction from a 3-SAT instance $\phi = (x_1 \vee x_2 \vee \bar{x}_3) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee \bar{x}_3) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee x_3)$ to a KG instance. A satisfying assignment for ϕ is $\{x_1 = 1, x_2 = 0, x_3 = 1\}$ which translates to a successful key sequence v_0, x_1, v_1, $\bar{x}_2$, v_2, x_3, C_1, C_2, C_3, v_s of cost 6.

Corollary 4.2. KEYSEQ *is* NP-*hard.*

Given this complexity result, our aim is now to derive efficient heuristics and approximation algorithms to analyze key challenge graphs. We have seen that an optimal attack is in general difficult to obtain in a reasonable amount of time, unless P=NP. Now one may ask: **how close to optimal can one construct an attack in polynomial time?** In other words, we would like to find good approximation algorithms and/or devise inapproximability results [12, 3, 4, 13, 9] for the KEYSEQ problem. An involved approximation-ratio preserving reduction [18, 15] from the LABEL-COVER problem [8] can be obtained to show the following:

Theorem 4.3. KEYSEQ *is quasi*-NP-*hard to approximate to within* $2^{\log^{1-\delta} n}$ *where* $\delta = \log\log^{-c} n$, *for any constant* $c < 1/2$.

See [5] for details of the proof.

4.2 Threat Analysis Algorithms

We have proven that solving KEYSEQ to optimality is very hard. In fact, even approximating the optimal solution to a large factor is already very hard. However, it is possible to get an estimate of the optimal solution using heuristics. We present a brute force algorithm along with a heuristic algorithm for the purposes of comparison.

The brute force algorithm BRUTE-FORCE (see Table 2) generates all possible permutations of attack sequences and finds the minimum cost among them. Without loss of generality, let $V_0 = \{v_0\}$ and $V_s = \{v_s\}$. Given a set S, let PERMUTE(S) signify the set of all possible permutations of elements of S without repetitions. The running time of this algorithm is super-exponential but it solves the KEYSEQ problem to optimality.

```
BRUTE-FORCE(KG)
1  min_cost ← 0
2  for each S ⊆ V − (V_0 ⋃ V_s)
3  do for each s ∈ PERMUTE(S)
4     do cost ← ATTACK-COST(v_0 s v_s)
5        if cost < min_cost
6           then min_cost = cost
7  return min_cost
```

Table 2. A brute force algorithm to find cost of optimal key sequence

We now describe our polynomial-time heuristic called GREEDY-HEURISTIC (see Table 3) which is based on the observation that a key sequence is structurally a path from some initial vertex to a final target vertex with zero or more branches from this backbone path, taken to collect additional keys. We use a greedy approach with the all-pairs shortest path (APSP) as the core decision-making procedure. Given a $n \times n$ adjacency matrix of a graph $G = (V,E)$, the APSP algorithm computes the all-pairs shortest path matrix, which gives the shortest path between any pair of vertices in the graph G. However, we cannot use this algorithm directly since the input that is available to us is a key challenge graph and not a weighted graph. We now briefly describe the algorithm UPDATED-ADJ-MATRIX, which converts a key challenge graph to a weighted graph. The main idea is that when an attacker acquires a new key, then weights on all edges having this key in the corresponding key challenge will reduce to the lower cost, otherwise they reduce to a higher cost. GREEDY-HEURISTIC proceeds by evaluating which neighboring key if acquired would give a shorter backbone path from the source vertex to the target vertex than the one currently seen. After at most $|V|$ rounds of decision-making, the algorithm returns a cost which cannot be reduced further. Invocation of the APSP algorithm inside the loop results in a worst case running time of $O(n^5)$. An insight into the algorithm is that we use both local (neighboring keys) and global (shortest path) factors to find the approximate solution.

```
GREEDY-HEURISTIC(KG)
 1  S ← V_0
 2  M ← UPDATED-ADJ-MATRIX(KG, π(S))
 3  A ← APSP(M)
 4  min_cost ← A[v_0, v_s]
 5  for round ← 1 to |V|
 6  do
 7     flag ← 0
 8     for each v_i ∈ NEIGHBOR(S)
 9     do vertex_cost ← NEW-VERTEX-COST(v_i)
10        M' ← UPDATED-ADJ-MATRIX(KG, π(S ⋃ {v_i}))
11        ∀v_j ∈ S, M'[v_j, v_i] ← 0, M'[v_i, v_j] ← 0
12        A' ← APSP(M')
13        if (vertex_cost + A'[v_0, v_s]) < min_cost
14           then min_cost ← vertex_cost + A'[v_0, v_s]
15                new_vertex ← v_i
16                flag ← 1
17     if flag = 1
18       then
19            S ← S ⋃ {new_vertex}
20       else return min_cost
21  return min_cost
```

Table 3. A greedy heuristic to find cost of near-optimal key sequence

4.3 Algorithm Benchmarking

We have performed empirical evaluations to compare BRUTE-FORCE with GREEDY-HEURISTIC. Our experiments were conducted on a Pentium 4/3GHz/1GB RAM running RedHat Linux 9.1. Since the BRUTE-FORCE algorithm has a prohibitively expensive running time, we have limited the size of the input to only 15 nodes. Our test data set consists of 1000 simulation runs and each run generates a random instance of a key challenge graph. We have compared the quality of the solutions (see Figure 4) computed by the two algorithms as well as their running times (see Figure 5).

When comparing the attack costs returned by the two algorithms, we have used randomly generated key challenge graphs of exactly 15 nodes for all the 1000 runs.

In Figure 4, there are two distinct bands, a lower one which corresponds to the optimal attack costs returned by BRUTE-FORCE and a higher one which represents the attack costs returned by GREEDY-HEURISTIC. The gap is due to the inapproximability results as shown in Section 4. GREEDY-HEURISTIC worked very well when the final attack sequence is very small (3-4 nodes) and this seems to be the best case scenario for the heuristic algorithm. However, when the attack sequence is long (10-15 nodes), the heuristic produces a larger gap from the optimal solution. The explanation we give for this observation is that longer the sequence, greater the chances that the decision-making procedure will make errors which are compounded.

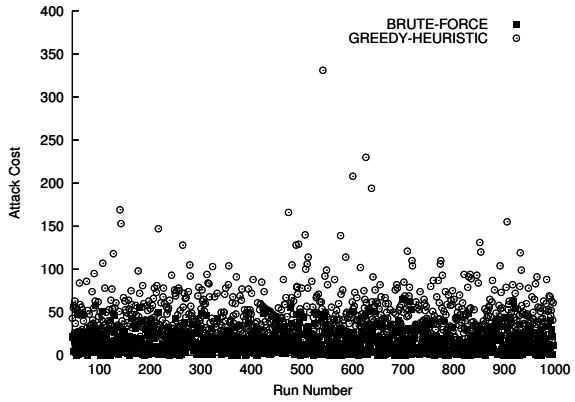

Figure 4. GREEDY-HEURISTIC **vs** BRUTE-FORCE**: Minimum cost of an attack**

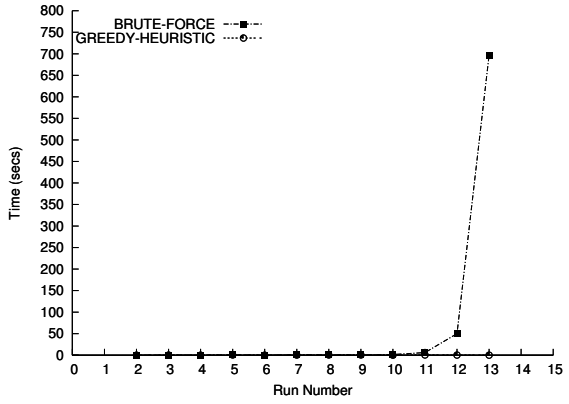

Figure 5. GREEDY-HEURISTIC **vs** BRUTE-FORCE**: Running time behavior**

When comparing the running time, the 1000 runs had a cross-section of varying graph sizes (3-20 nodes). The running time of the BRUTE-FORCE algorithm becomes very large even for small values of 13 to 15. Clearly, this is the expected behavior as the running time of this algorithm is $O(n!)$, which is worse than exponential. On the other hand, the GREEDY-HEURISTIC has polynomial running times for the same input. Even for graphs with a large number of nodes (200-300 nodes), we have observed a running time of only a few minutes (15-20 minutes).

5 Related Work

Theoretical models allow inexpensive security analysis without real experiments or implementation. Fault trees, privilege graphs and attack graphs are the most relevant modeling methodologies in the context of our work. We compare and contrast against these techniques to put our work in perspective.

Fault trees [11] are the first generation of formal models primarily used for system failure analysis. A fault tree is a logic (AND-OR) tree, where nodes are single faults, edges define a combination of them, and probabilities over edges represent the chance of their occurrence. While fault trees are suitable to model a disjunction and conjunction of faults, they lack the expressive power to capture attacks.

The privilege graph, introduced by Dacier et al. [7, 6] as an extension to Typed Access Model (TAM), is a directed graph where each node represents a set of privileges on sets of objects and each arc represents privilege escalation possibly through an exploit. This model also uses a probabilistic metric corresponding to likelihoods of attacks. However, determining these probabilities in a meaningful manner is far more challenging that our approach which measures the effectiveness of access control mechanisms. Ortalo et al. [17] describe an experimental evaluation of the privilege graph framework under certain assumptions on the memory state of an attacker, and their work did address a few insider attack scenarios, but their implementation required a substantial effort.

Philips and Swiler [21] proposed the attack graph model, where nodes in the graph represent the state of a network and edges represent steps in an attack. Building and analyzing such a graph by hand is not practical, and instead, several approaches have been proposed which use model-checking to automatically generate attack graphs [20]. However, model-checking suffers from a well-known problem of state explosion which is not easily solved and these approaches are not suitable even for networks of a reasonable size.

Although in some respects, both privilege graphs and attack graphs appear similar to our work, they are in fact closer to each other than our approach. A major dis-

tinction arises in the details that are captured (see Table 1) and the nature of threat analysis. For example, in their model, the difficulty attributed to edge traversal is a static entity, while in our model it is dynamic. Moreover, in both techniques, there is the problem of exponential state explosion, whereas our approach generates models which are polynomial-sized in the input network information. Also note that attacks in our model can succeed without privilege escalation or the presence of vulnerabilities, which is a distinct possibility for insider attacks.

6 Conclusion And Future Work

Insider threat is a long standing security problem, but so far, without good tools and techniques, there is little that could be done to counter the threat. In this paper, we believe that we have made a significant advance by proposing a usable and generic threat assessment model, and showed its applications to some typical insider threat scenarios. Indeed we do not claim that our model is a replacement for all the existing models, but that it occupies a certain niche and complements other models. We also believe that our modeling methodology is more generic than demonstrated in this paper and may have an appeal beyond just insider threat.

Our future work involves developing automated tools around the modeling methodology and algorithms developed in this paper to empower security analysts with techniques to measure a threat which has otherwise not been possible.

References

[1] *2004 E-Crime Watch Survey: Summary Of Findings*. CERT/United States Secret Service/CSO, 2004. http://www.cert.org/archive/pdf/2004eCrimeWatchSummary.pdf.

[2] *Insider Threat Study: Illicit Cyber Activity In The Banking And Finance Sector*. CERT/United States Secret Service, August 2004. http://www.secretservice.gov/ntac/its_report_040820.pdf.

[3] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof Verification And The Hardness Of Approximation Problems. *J. ACM*, 45(3):501–555, 1998.

[4] S. Arora and S. Safra. Probabilistic Checking Of Proofs: A New Characterization Of NP. *J. ACM*, 45(1):70–122, 1998.

[5] R. Chinchani, D. Ha, A. Iyer, H. Q. Ngo, and S. Upadhyaya. On the hardness of approximating the Min-Hack problem. 2005. Technical Report 2005-05. Computer Science and Engineering, SUNY at Buffalo.

[6] M. Dacier. *Towards Quantitative Evaluation of Computer Security*. PhD thesis, Institut National Polytechnique de Toulouse, December 1994.

[7] M. Dacier and Y. Deswarte. Privilege graph: an extension to the typed access matrix model. In *ESORICS*, pages 319–334, 1994.

[8] I. Dinur and S. Safra. On the Hardness of Approximating Label Cover. *Electronic Colloquium on Computational Complexity (ECCC)*, 6(015), 1999.

[9] U. Feige, S. Goldwasser, L. Lovász, S. Safra, and M. Szegedy. Interactive Proofs And The Hardness Of Approximating Cliques. *J. ACM*, 43(2):268–292, 1996.

[10] M. R. Garey and D. S. Johnson. *Computers and Intractability*. W. H. Freeman and Co., San Francisco, Calif., 1979. A Guide To The Theory Of NP-Completeness, A Series of Books in the Mathematical Sciences.

[11] J. Gorski and A. Wardzinski. Formalizing Fault Trees. *Achievement and Assurance of Safety*, pages 311–327, 1995.

[12] J. Håstad. Some Recent Strong Inapproximability Results. In *Algorithm theory—SWAT'98 (Stockholm)*, volume 1432 of *Lecture Notes in Comput. Sci.*, pages 205–209. Springer, Berlin, 1998.

[13] J. Håstad. Some Optimal Inapproximability Results. In *STOC '97 (El Paso, TX)*, pages 1–10 (electronic). ACM, New York, 1999.

[14] D. S. Hochbaum, editor. *Approximation Algorithms for NP Hard Problems*. PWS Publishing Company, Boston, MA, 1997.

[15] S. Khanna, R. Motwani, M. Sudan, and U. Vazirani. On Syntactic Versus Computational Views Of Approximability. *SIAM J. Comput.*, 28(1):164–191 (electronic), 1999.

[16] C. Meadows. A Representation of Protocol Attacks for Risk Assessment. In R. N. Wright and P. G. Neumann, editors, *DIMACS Series in Discrete Mathematics and Theoretical Computer Science: Network Threats*, volume 38, December 1998.

[17] R. Ortalo, Y. Dewarte, and M. Kaaniche. Experimenting With Quantitative Evaluation Tools For Monitoring Operation Security. *IEEE Transactions on Software Engineering*, 25(5):633–650, September/October 1999.

[18] C. H. Papadimitriou and M. Yannakakis. Optimization, Approximation, And Complexity Classes. *J. Comput. System Sci.*, 43(3):425–440, 1991.

[19] C. Phillips and L. P. Swiler. A Graph-Based System For Network-Vulnerability Analysis. In *Proceedings of 1998 New Security Paradigms Workshop*, pages 71 – 79, Charlottesville, Virginia, 1998.

[20] O. Sheyner, J. Haines, S. Jha, R. Lippmann, and J. M. Wing. Automated Generation and Analysis of Attack Graphs. In *Proceedings of the IEEE Symposium on Security and Privacy*, Oakland, CA., May 2002.

[21] L. P. Swiler, C. Phillips, D. Ellis, and S. Chakerian. Computer-Attack Graph Generation Tool. In *DARPA Information Survivability Conference and Exposition (DISCEX 11'01)*, volume 2, June 2001.