10TH INTERNATIONAL COMMAND AND CONTROL RESEARCH AND TECHNOLOGY SYMPOSIUM

THE FUTURE OF C2

Modeling Insider User Behavior Using Multi-Entity Bayesian Network

Student Paper*

Information Operations/Assurance

*Ghazi A. AlGhamdi[1]
Kathryn Blackmond Laskey [1]
Edward J. Wright [2]
Daniel Barbará [3]
KC Chang [1]

Point of Contact: Ghazi AlGhamdi
galghamd@gmu.edu
Tel: (703) 625 0420

[1] George Mason University
Systems Engineering & Operations Research Department
4400 University Drive
Fairfax, VA 22030-4444
(703) 993-1644
galghamd@gmu.edu ; klaskey@gmu.edu ; kchang@gmu.edu

[2] Information Extraction and Transport, Inc.
1911 N. Ft. Myer Dr., Suite 600
Arlington, VA 22209
ewright@iet.com

[3] George Mason University
Information and Software Engineering Department
4400 University Drive
Fairfax, VA 22030-4444
(703) 993-1638
dbarbara@gmu.edu

ABSTRACT

This paper tackles a key aspect of the information security problem: modeling the behavior of insider threats. The specific problem addressed by this paper is the identification of malicious insider behavior in trusted computing environments. Although most security techniques in intrusion detection systems (IDS's) focus on protecting the system boundaries from outside attacks, defending against an insider who attempts to misuse privileges is an equally significant problem for network security. It is usually assumed that users who are given access to network resources can be trusted. However, the eighth annual CSI/FBI 2003 report found that insider abuse of network access was the most cited form of attack or abuse. 80% of respondents were concerned about insider abuse, although 92% of the responding organizations employed some form of access control mechanism [7]. Therefore, though insider users are legally granted access to network resources, it is essential to protect against misuse by insiders. This paper presents a scalable model to represent insider behavior. We provide simulation experiments to demonstrate the ability of the model to detect threat behavior. Information security objectives can be accomplished through a layered approach that represents several lines of defense. This approach constitutes one of these lines of defense.

1. INTRODUCTION

In today's net-centric environment, information security is a vital concern for command and control systems. Intentional or accidental misuse of C2 system resources may present a major threat to the all three information security objectives: confidentiality, integrity, and availability. Violation of any of these objectives can threaten the operation of a C2 system. Breaches of confidentiality by an insider can occur when data are not handled in a way sufficient to protect the confidentiality of information (i.e. leaking sensitive information). While integrity ensures that the information is authentic and complete, a malicious insider can threaten the authenticity of critical assets by illegally changing their contents. Availability assures that the C2 system resources are accessible and available when needed. A denial-of-service attack, for example, prevents legitimate users from using the system services. The impact of this attack can effectively disable C2 system operations by disabling the system computers or network.

Detecting insider user misuse involves many challenges [10]. Because insiders understand their organization's computer system and how it works, some of the most serious security violations with the most damaging consequences take place through misuse of insider access privileges. Insider users typically have greater knowledge than outsiders do about system vulnerabilities. Therefore, the chances of a successful attack can be greater for an insider attack than for an outsider attack. For instance, the knowledge that a malicious insider has about the sensitivity of information gives him/her a better chance to breach information confidentiality. Even more challenging is the malicious expert insider. This type of user can perform harmful actions while behaving almost indistinguishably from normal users, making detection very difficult. This problem increases the challenge for a variety of analysis, correlation, and data fusion methods. Insider user misuse is different from outsider misuse with respect to the nature of the threats that both cause. High authentication services can significantly reduce many outsider threats, while leaving the system unprotected against many insider threats.[1]

[1] Work for this paper was performed under funding provided by the Advanced Research and Development Activity (ARDA), under contract NBCHC030059, issued by the Department of the Interior. Additional support was provided by the US Navy

2. METHODOLOGY

Because insider user behavior is such a complex problem, a systematic and structured process is required to design, develop, and evaluate systems for mitigating insider threats. This paper provides a methodology, based on multi-entity Bayesian networks, to develop a scalable and extensible behavior model that can be applied in a wide range of computing environments. Our methodology provides an approach to design, develop, and evaluate insider behavior models. This systematic process employs the spiral lifecycle model [22]. Figure 1 shows the spiral lifecycle model. At each phase, we develop a prototype model, evaluate the prototype, and prioritize additional modeling activities for the next phase. This cyclic process of design, development, evaluation, and modification is repeated as necessary.

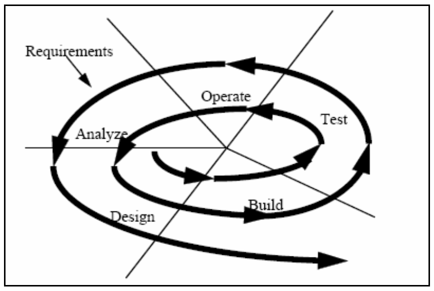

Figure 1: Spiral Model Lifecycle

2.1. Bayesian Networks

Bayesian networks [1,2] have been used in a wide variety of application areas including medical diagnostics, weather forecasting, classification systems, multi-sensor fusion, and legal analysis for trials. Bayesian networks (BN) have become known as a powerful modeling framework that combines graph theory with Bayesian probability. Bayesian networks allow the construction of probability models involving large numbers of interrelated uncertain hypotheses. While many methods make unrealistic independence assumptions, BNs can represent realistic correlation patterns in a given problem domain. Bayesian networks have the ability to combine qualitative expert knowledge with quantitative measures of plausibility and statistical data. This ability facilitates the construction of realistic and practical models.

In Bayesian networks, the graph expresses qualitative structural relationships including conditional independence, cause and effect, and correlation. Local distributions provide quantitative information about the strength of the relationships. Together the graph and local distributions specify a joint probability distribution over all variables in a Bayesian network.

The graph is a *directed acyclic graph* (DAG) that captures relationships between a set of variables which are relevant to specific domain. These relations can be stochastic, uncertain, or imprecise. Each node in the network represents a variable of interest and each edge represents a direct probabilistic relationship between the variables it connects. According to Jensen [2], a Bayesian network consists of:

- A set of random variables.
- A finite set of mutually exclusive states for each variable.
- A directed acyclic graph (DAG) with a node corresponding to each variable[2].

[2] A directed graph is called acyclic if there is no direct path $A_1 \rightarrow A_2 \ldots \rightarrow A_n$ such that $A_1=A_n$.

- A potential table $P(A|B_1 \ldots B_n)$ for each node, where $B_1 \ldots B_n$ are the parents of A in the DAG.

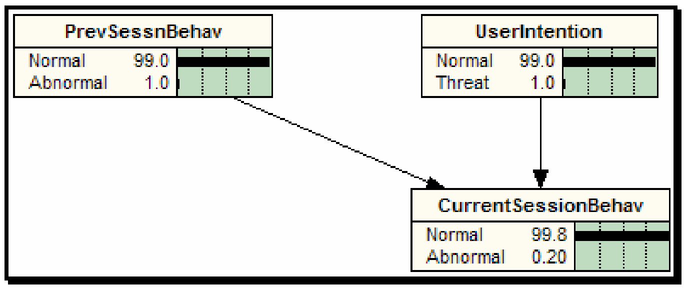

Figure 2: User Intention Bayesian network Model[3]

Table 1: Conditional Probability Table for the CurrentSessionBehav Variable

UserIntention	**PreSessnBehav**	**CurrentSessionBehav Distribution**	
		Normal	*Abnormal*
Normal	*Normal*	*0.99996*	*4.00008e-005*
Normal	*Abnormal*	*0.9*	*0.1*
Threat	*Normal*	*0.9*	*0.1*
Threat	*Abnormal*	*0.8*	*0.2*

Table 2: CPT for the UserIntention Variable

UserIntention	**UserIntention Distribution**
Normal	*0.99*
Threat	*0.01*

Table 3: CPT for the CurrentSessionBehav.PREV Variable

CurrentSessionBehav.PREV	**CurrentSessionBehav.PREV Distribution**
Normal	*0.99*
Abnormal	*0.01*

An edge between two nodes can represent any of the following kinds of dependency:

- The parent node is a cause of the child node;
- The parent node is a partial cause or predisposing factor for the child node;
- The parent and child nodes are functionally related; or
- The parent and child nodes are statistically correlated.

For example, let us consider the BN model shown in Figure 2. The node *UserIntention* is a random variable that represents the overall intention of a user. This variable has two states: *Normal* and *Threat*. The variable *CurrentSessionBehav* represents session behavior for the user in the current session. *PrevSessnBehav* represents session behavior for the user in the previous

[3] Screen shots were taken using Netica^{TM}, developed by Norsys Software Corp. (www.norsys.com)

session. Each of these variables has two states: *Normal* and *Abnormal*. The local distribution of the *CurrentSessionBehav* node depends on both *PrevSessnBehav* and *UserIntention*. The conditional probability table for the *CurrentSessionBehav* node is shown in Table 1.

The conditional probability table represents the quantitative relationships in this model. A user with a threatening intention is more likely to act abnormally while using the system than a user who has a normal intention. Furthermore, if a user acts abnormally in both the previous session (*PrevSessnBehav*) and the current session (*CurrentSessionBehav*) this increases the probability that the user's intention is threatening (*UserIntention).*

2.2. Partially Dynamic Bayesian Networks (PDBN)

Partially dynamic Bayesian Networks (PDBNs), also called temporal Bayesian networks [13], provide a powerful representation framework for temporal reasoning under uncertainty. Temporal reasoning is important in a wide variety of domain including cyber-security. Inference in PDBNs is based on an input stream of reports about observable features of the situation. PDBNs can contain both static and dynamic nodes. A static node is a node whose value is constant over time. A dynamic node is a Bayesian network node whose value changes over time. Static nodes in a PDBN may have dynamic children but dynamic nodes may not have static children.

A user behavior that changes dynamically over time requires dynamic modeling. To illustrate how we to apply PDBNs to modeling the insider behavior we re-examine the user intention model shown in Figure 2.

Figure 3 shows the user intention PDBN. The model contains a static variable and two time steps of a dynamic variable. The static variable is *UserIntention*, which influences the value of the dynamic variable. The figure shows two time steps of the dynamic variable, labeled *CurrentSessnBehav* and *PrevSessnBehav*. The model says the probability distribution of the user's behavior in the current session is influenced both by his/her behavior in the previous session and the his/her intention.

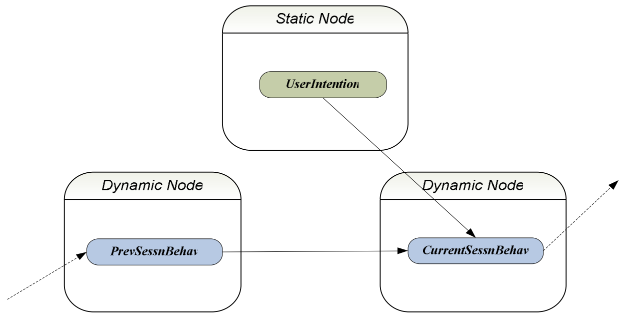

Figure 3: PDBN user intention model for two time slices

Figure 4 shows PDBNs for the user intention model. The *PrevSessnBehav* node is the initial time step. The C*urrentSessnBehav.1* node represents an instance created to capture the behavior during session 1 while C*urrentSessnBehav.0* plays the role of previous session behavior for the same user. In the next session, *CurrentSessnBehav.1* node will be the previous session behavior for the current session behavior variable *CurrentSessnBehav.2*, and so on. The importance of this is that we can extend the model as many sessions as required to represent the user session behavior by only specifying the three belief tables shown in Table 1, Table 2, and Table 3.

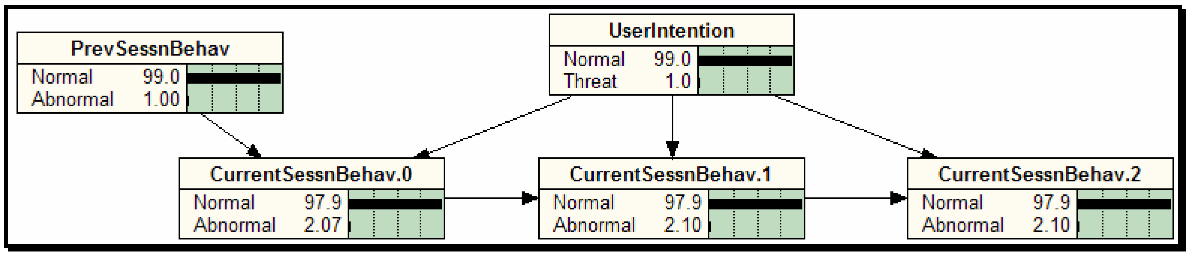

Figure 4: A three-time Steps PDBN for the User Intention

2.3. Hidden Markov Model (HMM)

Hidden Markov Models (HMM) are dynamic models of systems whose behavior depends on an unobservable, hidden, state that changes in time and satisfies a Markov property. The Markov property means that conditional on the present, past states have no influence on future states. HMM is a rich class of models that provide a good balance between tractability and accuracy for many problems involving systems that evolve in time.

In our user intention BN model shown in Figure 2, we define a first order Markov process for the dynamic variable (*CurrentSessnBehav* node) as follows:

- *States*: We define two states for the variable *CurrentSessnBehav:* Normal and Abnormal.
- *Initial state vector*: This vector defines the probability of the states at time zero.
- *State transition matrix*: This matrix defines the probability of the user current session behavior given the previous session behavior.

Figure 5 shows all possible state transitions probabilities for the user's session behavior dynamic variable. The variable will move from one state to another based on a transition matrix that defines the probability of transition conditioned on user intention.

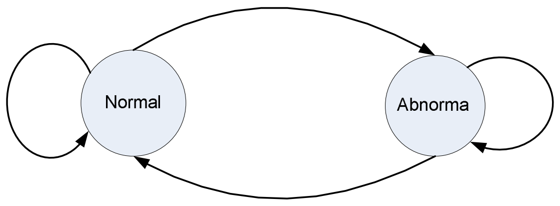

Figure 5: States transition for user session behavior

To specify the state transition probabilities we need to define two transition matrices conditioned on user intention. Thus, our model of session behavior is actually two hidden Markov models, one for normal user and one for threats. The transition matrix in Figure 6 shows the probabilities for all possible transition states for the user intention model conditioned on *Threat* user intention. This state transition matrix specifies that if the user behavior in the previous session was *Normal* then there is a 90% chance that it will be *Normal* in the current session, and 10% chance it will be Abnormal. However, if the user behavior in the previous session was *Abnormal* then the probability is 80% chance that it will be *Normal* in current session is and 20% chance it will be *Abnormal*. These probabilities mean that *Abnormal* behavior in the previous session will result in *Abnormal* behavior in the current session 2 times as often

(0.2/0.1) than if the user's behavior was *Normal* in the previous session. Note that because the numbers are probabilities the sum of the entries for each row is one.

$$\begin{array}{ccc} & & \text{Current Session Behavior} \\ \text{Previous Session Behavior} & \begin{array}{c} \text{Normal} \\ \text{Abnormal} \end{array} & \begin{array}{cc} \text{Normal} & \text{Abnormal} \\ \left(\begin{array}{c} 0.90 \\ 0.80 \end{array}\right. & \left.\begin{array}{c} 0.10 \\ 0.20 \end{array}\right) \end{array} \end{array}$$

Figure 6: States Transition Matrix for User Session Behavior

Conditioned on Threat Intention

The state transition matrix shown in Figure 7 defines the state transition probability conditioned on *Normal* user intention. If the previous session is Normal, the chance of being normal in the current session is 99.996% and abnormal with chance of 0.004%. However, if the previous behavior is Abnormal the probabilities will be 0.9 and 0.1 for current session behavior.

$$\begin{array}{ccc} & & \text{Current Session Behavior} \\ \text{Previous Session Behavior} & \begin{array}{c} \text{Normal} \\ \text{Abnormal} \end{array} & \begin{array}{cc} \text{Normal} & \text{Abnormal} \\ \left(\begin{array}{c} 0.99996 \\ 0.9 \end{array}\right. & \left.\begin{array}{c} 0.00004 \\ 0.1 \end{array}\right) \end{array} \end{array}$$

Figure 7: States Transition Matrix for User Session Behavior

Conditioned on Normal Intention

To initialize a HMM, we need to state what the initial probability of the session behavior is. We define this in a vector of initial probabilities shown below to be *Normal* with probability 0.99 and *Abnormal* with probability 0.01.

$$\begin{array}{cc} \text{Normal} & \text{Abnormal} \\ \left(0.99\right. & \left.0.01\right) \end{array}$$

2.4. Multi-Entity Bayesian Networks (MEBN)

Bayesian networks are well suited for problems in which the same set of random variables apply to all instances, and only the evidence is different from problem to problem. However, insider user behavior is a complex problem that requires reasoning about many different kinds of entities and their relationships such as user access behavior, user skills, and attack tactics. In such complex problems, using a single, fixed Bayesian network to encompass all problem instances is infeasible. The number of objects to be reasoned about may be large and the relationships between these objects differs from one problem instance to another problem instance. Therefore, we need a more flexible representation. Multi-entity Bayesian networks (MEBNs) expand upon standard Bayesian networks in their ability to encode repeated, parameterized argument structures called MEBN Fragments (MFrags).

Multi-Entity Bayesian Network (MEBN) logic [8,18] extends standard BNs to provide first-order expressive power. This permits the kind of replication and combination needed to reason about complex problems. Similar to BN, MEBN logic uses *acyclic directed graphs* to specify joint probability distributions for a collection of interrelated random variables. An MTheory is a collection of Bayesian Network Fragments (called *MFrags*) that satisfy consistency criteria such that the collection specifies a probability distribution over attributes of and relationships among a collection of interrelated entities.

An MFrag consists of *random variables*, a *fragment graph,* and a set of *local distributions*. Each MFrag has an associated set of random variables that are partitioned into *context, input,* and *resident* random variables. While probability distributions for resident random variables are defined in the MFrag itself, probability distributions for context and input random variables are defined in other MFrags. Context random variables specify conditions under which the local distributions for the resident random variables apply. Input and context random variables are represented as root nodes in the fragment graph. There is an implicit arc connecting each context random variable with each resident random variable. For each resident random variable, there is a local distribution to specify how to assign probabilities to possible values of any instance as a function of the values of instances of its parents, given that the context constraints are satisfied.

An MFrag represents characteristics of a type of entity and the relationships among types of entities. For a given type, an MFrag can be instantiated multiple times to represent different entities of a given type. A collection of MFrags can be instantiated and assembled to construct a *situation-specific Bayesian network* [19] to reason about a particular problem. The following section shows how MFrags can be applied to reasoning about insider threats.

2.5. Insider MFrags Knowledgebase

Our model of insider behavior characterizes insider users in terms of attributes and activities that distinguish relevant categories of user from one another. A preliminary version of this model was presented previously [9]. The model was evaluated by expert review and computational experiments, refined on the basis of the evaluation results, and extended to include additional features. The model includes features such as software skill, access privileges, attack tactics, and motivation [14], and activities such as document access and software installation. The insider Bayesian network (IBN) model relates a category of user to the attributes of the category and the activities typically performed by users of the given category. Observable attributes and activities are represented as evidence nodes in IBN. Observable activities include events that can be logged and recorded by a monitoring system.

The fundamental modeling unit in designing IBN is the MFrag. Each MFrag defines general properties that hold for a specific type of entity. The first task in building IBN model is to design the knowledge base of MFrags to represent entities in the domain. The IBN MFrags knowledgebase is shown in Figure 8. This section presents the IBN MFrag knowledgebase that represents user attributes and activities.

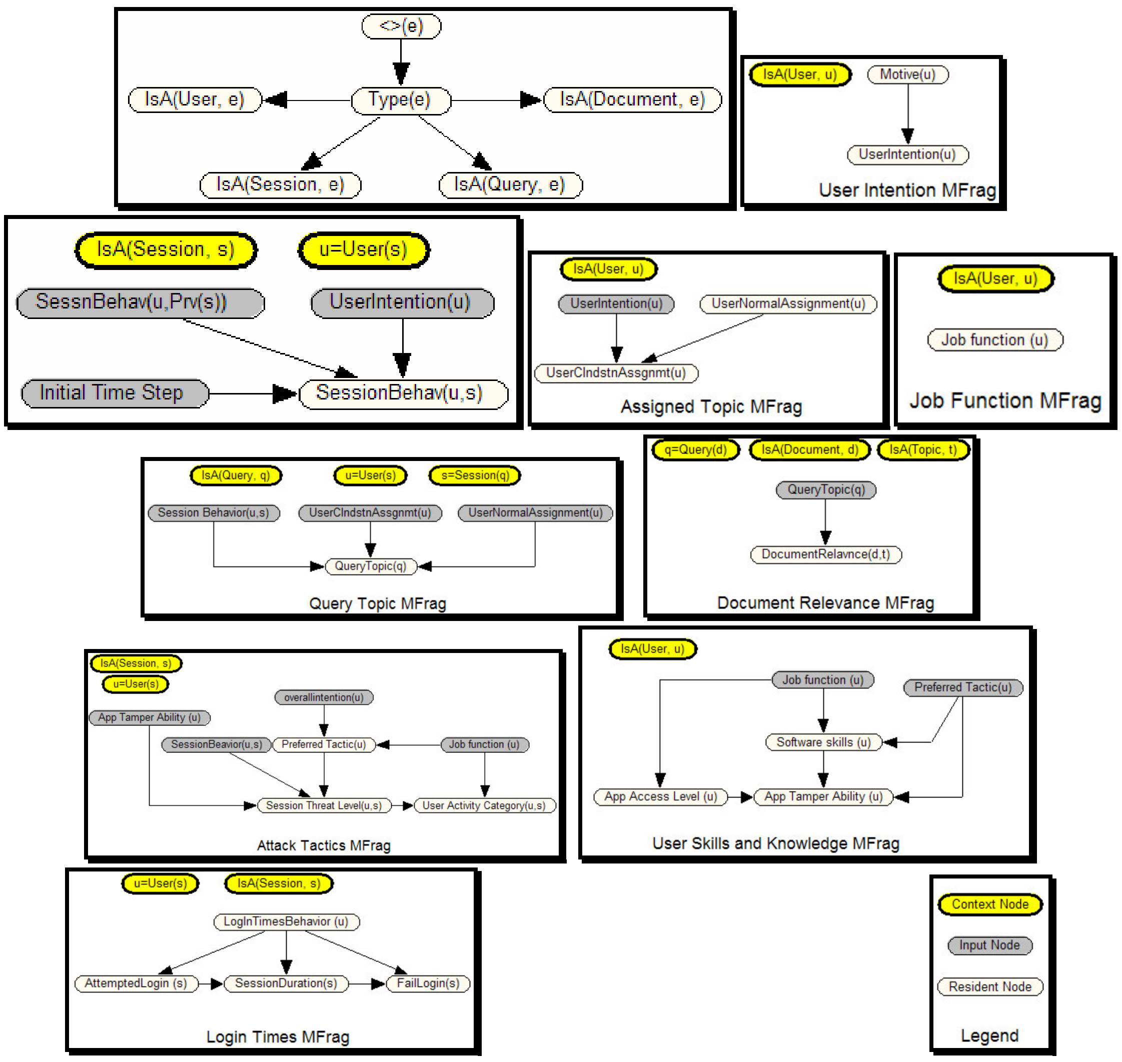

Figure 8: Insider User Behavior MFrags

To compute with MEBN models we used the Quiddity*Suite probabilistic relational modeling toolkit, developed by IET (www.iet.com). Quiddity*Suite uses frames, a commonly used knowledge representation in Artificial Intelligence applications, to specify MFrags and combine them on the fly into of large-scale situation-specific models. In Quiddity, a frame represents a type of entity. Each slot in the frame represents attributes of that entity. An uncertain slot in a frame is a template for a Bayesian network node. Quiddity* Modeler provides a scripting language for defining parents and local distributions of slots in frames. Whenever an instance of a frame is created, a BN node is constructed for each uncertain slot in the frame. In addition, slots can refer to slots in other frames, providing the ability to specify complex interrelationships.

Entity Type MFrag

The first MFrag is the Entity Type. This MFrag is used to formally declare the possible types of entities in the model. We have four entity types: User, Session, Query, and Document. Although our model employs only simple typing, the flexibility of MEBN logic allows accommodating more complex typed systems, with sub-typing, polymorphism, and multiple inheritance [18]. Future versions of our model may make use of these more powerful features.

User Intention MFrag

This MFrag models the user intention toward the system given his/her motive. This MFrag has one context variable and two resident variables. The context random variable Isa(User, u) represents the assumption that the MFrag applies to entities of type *User*. That is, instances of this MFrag can be created by replacing the variable *u* by the identifiers of entities of type *User*, and we can make as many instances as we have users. The assumption of this MFrag is that we expect a *threat* user to be motivated in order to make some sort of adverse affect against the system. This motivation will influence the user's overall intention toward the system. A *normal* user, however, is not expected to be maliciously motivated and, therefore, his/her threat intention against the system is expected to be very low.

Session Behavior MFrag

The session behavior MFrag focuses on modeling the insider user behavior during the logged in sessions. It contains the user overall intention as an input variable that is defined in user intention MFrag, session behavior, and session behavior in the previous session. The context random variables Isa(Session,s) and u=User(s) specify that *s* refers to a session, and *u* refers to the user who is logged in for that session. The input variable *Initial Time Step* specifies if the session is the first session or not. It has two values True or False. When this variable is True then the SessionBehav(u, s) becomes the initial state of this variable (i.e. SessionBehav(u, Prv(s))).

Assigned Topic MFrag

The objective of assigning a topic for each user in the system is to ensure that users are assigned access to the system resources in accordance with the requirements of their job function. Each user is assigned a topic that requires access to several system resources related to his/her topic. The system security policy usually defines which users should be assigned to what topics.

Assigned topic is an essential item of information required to infer user behavior. The assumption is that a normal user will tend to perform activities that are related to his/her assigned job function. We also assume that assignments are constant within a given time step. Examples of these activities can be accessing documents, maintaining software, or installing hardware. This is represented in the assignment MFrag as the *UserNormalAssignment(u)* node shown in Assigned Topic MFrag in Figure 8. This information is provided by the system security policy, and is assumed to be known. Therefore, its value is provided as evidence in this MFrag. Given that the normal assignment node has no parents, we can easily increase the numbers of job functions as necessary. A *threat* user will tend to perform activities that are not related to his/her assigned job function. We call this "*Clandestine Job Function*". This is defined as the type of malicious activities that a *threat* user is covertly pursuing. By definition, a normal user will not have a clandestine job. A threat user clandestine topicis represented in assignment MFrag as the *UserClndstnAssgnmt(u)* node. As before, the context random variable Isa(User,u) specifies that the variable *u* refers to a user.

Query Topic MFrag

The purpose of the Query Topic MFrag shown in Figure 8. is to capture the user interest in any topic in the system. We assume that for any given query that the user is seeking information about a specific topic. The *QueryTopic(q)* variable in this MFrag has three input variables: *SessionBehav*(*u*), *UserNormalAssignment*(*u*), and *UserClandestineAssignment*(*u*). Distributions for these random variables are defined in the previously described MFrags. The context random variables state that *q* refers to a query, *s* refers to the session in which the query was performed, and *u* refers to the user who is logged in for that session.

Document Relevance MFrag

The aim of this MFrag is to model the relevance of documents stored in the system to each assigned topic. Each document *d* has a relevance rating to all assigned topics *t*. The relevance *DocumentRelevance*(*d*,*t*) of document *d* to topic *t* depends on the topic *QueryTopic*(*q*) that the user had in mind when performing the query. The *QueryTopic*(*q*) node in Figure 8 is defined in the Query Topic MFrag. The context random variables state that *t* is a topic, *d* is a document, and *q* is the query that resulted in retrieval of *d*.

User Attack Tactics

This MFrag focuses on modeling the attack tactics that a *threat* user may follow. The *Preferred Tactic(u)* node in the user attack MFrag shown in Figure 8 represents the tactics that insider users may follow to attack the system resources. Its parents are the input variables *OverallUserIntention*(*u*) and *JobFunction(u)*. The underlying assumption is that *normal* users will have no attack tactics against the system while threat users will follow *high sophistication, medium sophistication,* and *low sophistication* types of attacks.

The *UserActivityCategory(u,s)* node shown in the Attack Tactics MFrag is an evidence node that captures users activities during a given session. A *normal* user behavior is defined with respect to assigned job function. For example, an office administration user is expected to access administration documents but running codes or installing software may indicate abnormal behavior for this type of user.

The *SessionThreatLevel(u,s)* identifies the threat level for a particular user *u* and a specific session *s*. It has three states: low, medium, and high; and its value is defined as a deterministic function of its parents. The parents for this variable are *PreferredTactic(u), SessionBehavior(u,s),* and *AppTamperAbility(u)*. The context nodes indicate that *s* refers to a session and *u* refers to the user logged in during that session.

User Skills and Knowledge MFrag

This MFrag focuses on the user skills and the knowledge that the insider user has of the internal system domain. The job function node indicates the role to which the insider user is assigned. In our model, this node has four states: *office administration, analyst, system programmers,* and *system administrator.* The user's software skill level is represented in this MFrag as the *SoftwareSkills(u)* node. The level of the software skills that a user may possess depends on the assigned job function and ranges from *average* to *sophisticated.* This is why we have *JobFunction(u)* variable as a parent to the *SoftwareSkills(u)* variable, which means that the user job function will mostly influence the user skills.

The level of access is different from a one user to another. A system administrator usually has access to most of the network resources whereas an office administrator will have a limited access privileges. The level of access for each user, represented as *AppAccessLevel(u)*, is based on the job function of the user. This is why we have the *JobFunction(u)* as parent of this node.

This variable represents the level of access of the assigned user to modify application code. The access level variable has three states *High, Partial,* and *Minimal*.

The last resident variable in this MFrag refers to the ability of insiders to tamper with system resources such applicant code or operating system kernel. The ability of insider to perform this type of activities is dependent on both the software skills and the level of access to system resources. This is represented via the *App Tamper Ability (u)* node.

Finally, the context constraint for this MFrag specifies that *u* refers to a user.

User Login Times

The purpose of this MFrag is to model user login times shown in the Login Times MFrag in Figure 8. It consists of four resident variables *LogInTimesBehavior(s), AttemptedLogin(,s), SessionDuration(s),* and *FailLogin(s).* The variable *LogInTimesBehavior(s)* is a dynamic node. Its parent is *SessionBehavior* (s). It represents an intermediate node between the log in nodes and is *SessionBehavior*(s) node. It consists of three states *Normal, Abnormal,* and *Masquerader*.

A login session time outside working hours may indicate suspicious activities or abnormal behavior. The variable that captures this event is the *AttemptedLogin(s).* It consists of two states: *within* normal working hours and *outside* normal working hours. System security policy usually determines if it is allowed for users to log in to the system outside working hours. For our model, we assume that a login session time outside working hours indicates abnormal behavior (but not necessarily a masquerader).

The *SessionDuration(s)* variable captures the session duration of the each logged in session time. Abnormal session duration is based on the historical pattern of users. We assume that if the login session duration is abnormal then this increases our belief of *Abnormal* session behavior. For example, based on historical data if the average session duration for a particular user is 20 minutes and the current logged in duration is 90 minutes then this may indicate abnormal session behavior for this user.

FailLogin(*s*) captures the number of failed login attempts at any given session. If a user either succeeds on the first attempt or fails only once, then this does not increase the belief of being abnormal (normal users often misspell their passwords). However, if a user failed to login more than one time then this increases our belief that s/he may be a masquerader.

3. SIMULATION EXPERIMENTS

Computer-based simulation is a popular modeling technique used in engineering design. It enables a representation of the real system to be manipulated when manipulation of the real system is impossible or costly [15]. Simulation can provide us with a better understanding of the real system. In a simulation, model parameters or variables can be controlled and manipulated, which is often not possible with the real system. Simulation also allows us to predict how the system would respond to conditions that have never been observed, but might be anticipated in the future.

Careful advance planning and sound experimental design can greatly improve efficiency of a simulation and utility of the results. Design includes defining the goal of the experiments, developing hypotheses, determining the factors to be manipulated, and planning the data analysis strategy

3.1. Objective of the Experiments

The objective of the computational simulation experiments is to evaluate the detection capability of the IBN model with respect to user activities. Within that constraining scenario, we identified information useful for distinguishing *normal* users from those who pose a *threat*. Our fundamental hypothesis states that given the insider user's set of actions over a period of time, we can identify insider threats using our Bayesian network model

The experiments reported here were conducted on the part of the IBN model related to document relevance problem [23]. Figure 9 shows an example of the document relevance IBN model that we used for the simulation experiments. The model shown in this figure is for 4 topics, 1 query, and 2 document access events. This model was created by making instances of some the MFrags and assembling them into a unified Bayesian network.

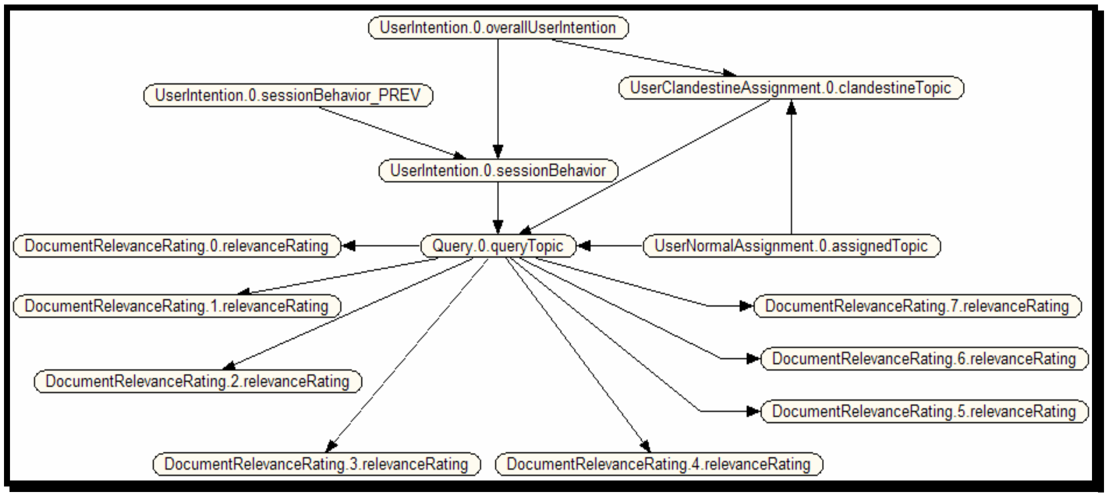
UserIntention.0.overallUserIntention
UserIntention.0.sessionBehavior_PREV
UserClandestineAssignment.0.clandestineTopic
UserIntention.0.sessionBehavior
DocumentRelevanceRating.0.relevanceRating
Query.0.queryTopic
UserNormalAssignment.0.assignedTopic
DocumentRelevanceRating.1.relevanceRating
DocumentRelevanceRating.7.relevanceRating
DocumentRelevanceRating.6.relevanceRating
DocumentRelevanceRating.2.relevanceRating
DocumentRelevanceRating.5.relevanceRating
DocumentRelevanceRating.3.relevanceRating
DocumentRelevanceRating.4.relevanceRating

Figure 9: Document Relevance Model for 4 topics, 1 query, and 2 documents

3.2. Design of Experiments

The simulation concept that we implement is based on comparing the model's inferences against simulated ground truth information about the user intention (normal/threat). Each simulation experiment is implemented by creating two identical Bayesian networks (*generative network* and *inference network*) operating simultaneously. The *generative network* is used to simulate observations of each user given his/her simulated intention. The user's intention is set by design to *normal* or *threat*, and conditional on the user's intention, a sequence of actions is simulated. The inference network is then used to infer on the basis of the simulated actions whether or not the user is a threat. For each complete run, the simulated intentions, the simulated actions, and the model's inferences about users' intentions are stored in output files for analysis.

We conducted three computational simulation experiments using different design factors. For every simulation run, different design factors were used to test the model. These design factors include:

- Number of Normal Users
- Number of Threat Users
- Number of Sessions
- Number of Queries per user/session
- Number of Accessed Documents per query

Table 4 shows the design factors for each computational simulation experiment run.

Table 4: Computational Experiments Design Factors

Experiment Run	**No. of Users**	**Ground Truth**	**Number of Sessions (Each User)**	**No. of queries (each session)**	**No. of documents (each query)**
1	90 Users	45 Normal	100	4	6
		45 Threat	100	4	6
2	90 Users	45 Normal	100	2	5
		45 Threat	100	2	5
3	90 Users	45 Normal	100	6	10
		45 Threat	100	6	10

4. RESULTS

We will first visually examine representative model outputs over 100 sessions. Then, we will plot a ROC curve for probability of detection (PD) and probability of false alarms (PFA) over different threshold values. The area under the ROC curve is used as a threshold-independent measure of the performance of our model.

4.1. Selected Users

Figure 10 through Figure 13 show time series plots[4] of the inferred intention probabilities for a few typical simulated users. The horizontal axis – labeled 1 through 100 - corresponds to 100 sessions for each user. The vertical axis corresponds to probability. The red line indicates the inferred probability of threat intention; the green line indicates the inferred probability of normal intention.

Figure 10 shows sample time series plots for simulated user #13. The ground truth for this user is "threat." During the first 20 sessions, the IBN model did not detect anything suspicious enough to raise an alarm (although note that the probability of threat begins increasing somewhat after a few sessions). The probability of threat (red line) increases dramatically at session 20 and remains high for the remaining sessions. Figure 11 shows a plot for simulated user #33, who is a normal user. For the first 50 sessions, the probability of threat stays low until session 55 where we see an increase in threat belief. Note that the probability of threat decreased to indicate that this user went back to the normal activities. Another typical plot is user 56 shown Figure 12. This is an example of a threat user that the model was not able to detect. Figure 13 shows a typical plot for a normal user where the probability of threat stays low through all 100 sessions.

[4] User behavior and ROC Plots are generated using the statistical computing and graphics R language (www.r-project.org).

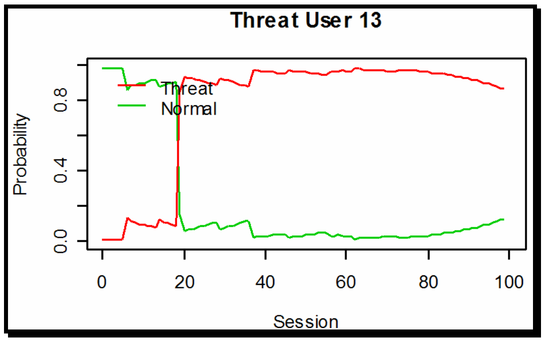

Figure 10: Plot for Detected Threat User

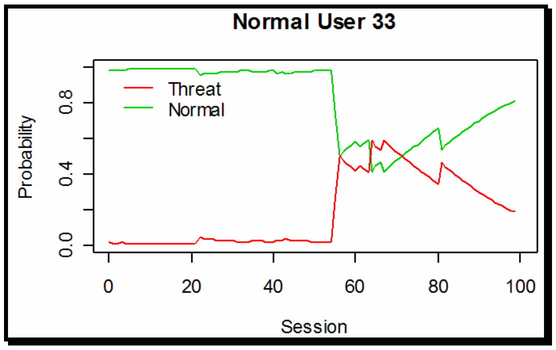

Figure 11: Plot for a Normal User

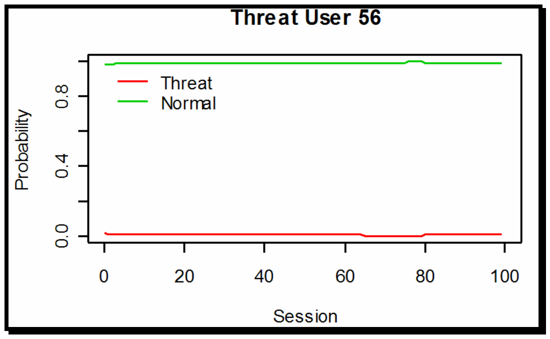

Figure 12: Typical Plot for Undetected Threat User

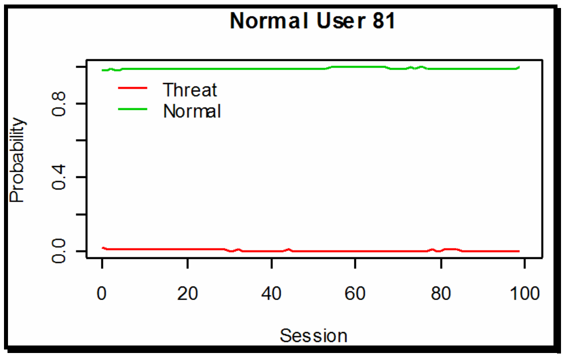

Figure 13: Plot for Normal User

4.2. Statistical Analysis of Threat Probabilities

To declare that a user's behavior is sufficiently suspicious to trigger an alarm, we used the following rule: if the average belief of a threat over a given time window is higher than a given threshold then declare threat, otherwise declare as normal. As our time window, we used the last 80 out of 100 sessions (given the low initial belief in threat, belief in threat for the first 20 sessions tends to be low regardless of the user type). The overall results of the three simulation experiments are shown in Figure 14 through Figure 16. The x-axis is the threshold value for declaring a user to be a threat. The y-axis is the number of users declared to be threats. The red bars are the number of threat users the model correctly detected as threats. The blue bars are the number that the model declared to be threats while their ground truths are normal (false alarms). In the three plots, note that as we increase the detection threshold the number of false alarms decreases. For example, Figure 14 shows that the model was able to detect 37 out of the 45 threat users at threshold 0.6 while Figure 16 shows that the model detected 30 out of the 45 threat users.

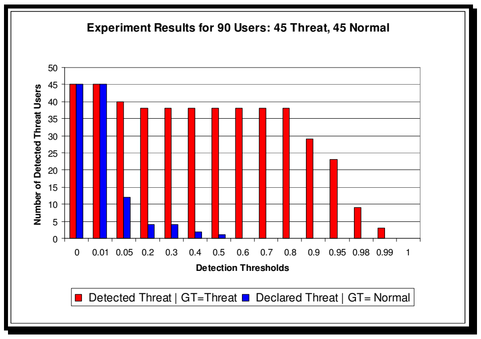

Figure 14: Combined Users Results for Experiment 1

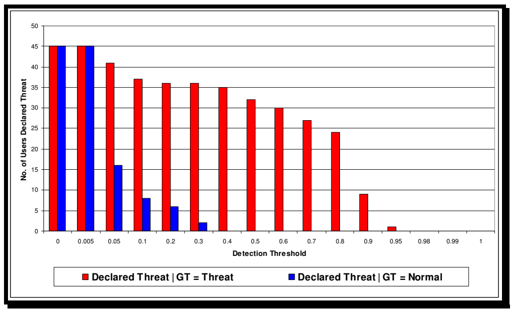

Figure 15: Combined Users Results of Experiment 2

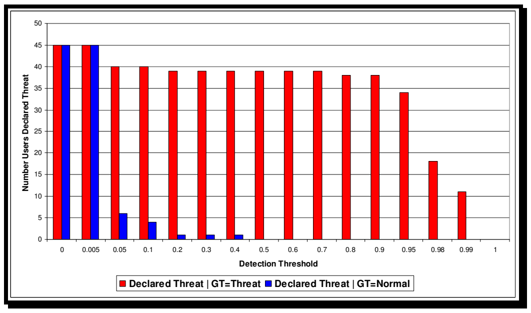

Figure 16: Combined Users Results for Experiment 3

4.3. ROC Curves

Although visualization gives us useful information about the model performance at the user level, it does not provide much information about the overall model detection capability. A receiver operating characteristic (ROC) curve shows the relationship between probability of detection (PD) and probability of false alarm (PFA) for different threshold values [20]. The two numbers of interest are the probability of detection (true positives) and the probability of false alarms (false positives). The probability of detection (PD) is the probability of correctly detecting a Threat user. the probability of false alarm (PFA) is the probability of declaring a user to be a Threat when s/he is Normal. The detection threshold is varied systematically to examine the performance of the model for different thresholds. Varying the threshold produces different classifiers with different (PD) and probability of false alarm (PFA). By plotting PD and PFA for different thresholds values, we get a ROC curve. The area under the ROC curve (AUC) it is a threshold-independent measure of classifier performance. The ROC curves along with the AUC's are used for analyzing the results of the computational simulation.

We have developed two types of ROC curves: *frequency* and *Bayesian* ROC curves. For the frequency ROC, we estimated the probability of detection by counting the number of detected threats for a given threshold value and dividing this number by the total number of threat users. For example, if 30 out of 100 threat users were detected for a threshold of 0.5, then the probability of detection for this threshold value is estimated as 30/45. The probability of false positive is estimated in a similar way.

To provide a conservative estimate of the area under the ROC curve, we calculated the posterior expected value of PD and PFA under the assumption that PD and PFA are uniformly distributed independently of the threshold. The posterior distribution under this assumption is a Beta distribution with expected value given in Table 5.

Table 5: Posterior Expected Values for PD and PFA for Uniform Prior Distribution

Detected Behavior	Ground Truth	
	***Threat* (n)**	***Normal* (m)**
***Threat* (k)**	$E[PD] = \frac{k+1}{n+2}$	$E[PFA] = \frac{k+1}{m+2}$

Figure 17, Figure 18, and Figure 19 show the frequency and Bayesian ROC curves for experiments 1, 2, and 3 respectively. The upper and the further left a curve is, the better it is. The black line shows the frequency ROC values and the red line shows the Bayesian ROC values.

Figure 17 displays ROC curves for the first experiment. While the detection rate increases to reach 80% the false alarms rates stays low. As the detection rate increases (more than 80%), we start seeing an increase in the false alarm rate. Between about 2% and 18% false alarm rates, the detection rate does not change much but starts to increase slightly as the false alarms increases. The area under the ROC curves for frequency and Bayesian were calculated to be 0.9217 and 0.8653.

Figure 18 shows the ROC curve for the second experiment. The model detection rates increases until about 78% while keeping the false alarm rate low. The false alarm rate begins to increase until it reaches 15% while the detection rate stays at the same level. The areas under the curves are 0.9121 and 0.8565.

Figure 19 displays the ROC curve for the third experiment. Note that the false alarm rate remains low as the detection rate increases to more that 85%. The areas under the curves are 0.9358 and 0.8782. Table 6 summarizes the results for the three simulation experiments.

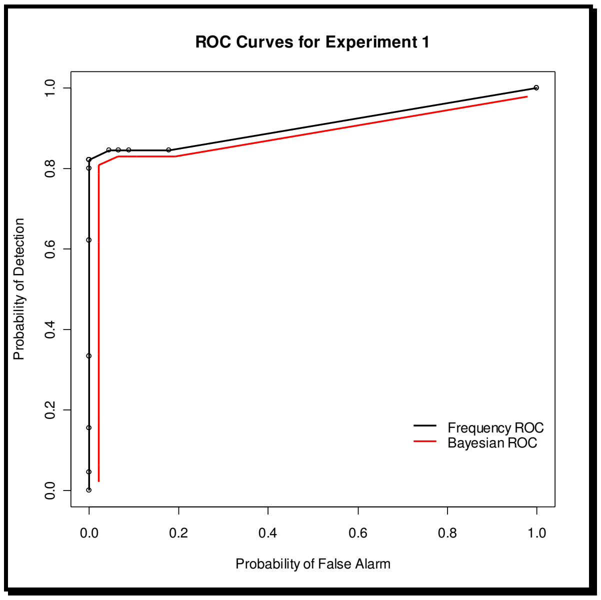

Figure 17: ROC Curves for Experiment 1

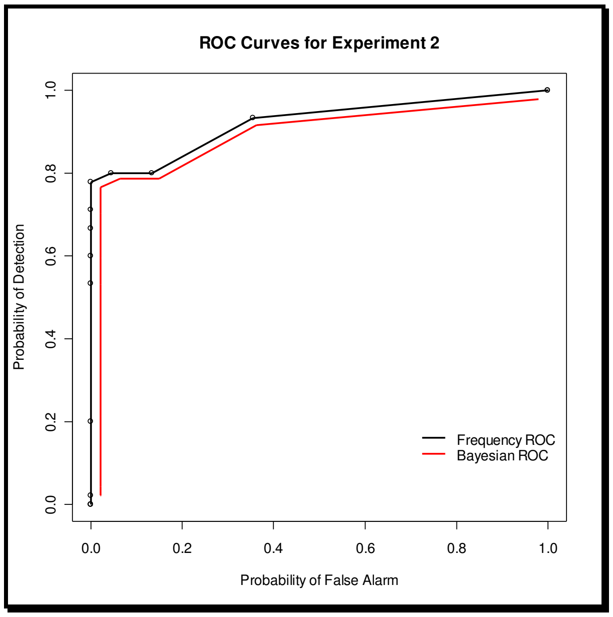

Figure 18: ROC Curves for Experiment 2

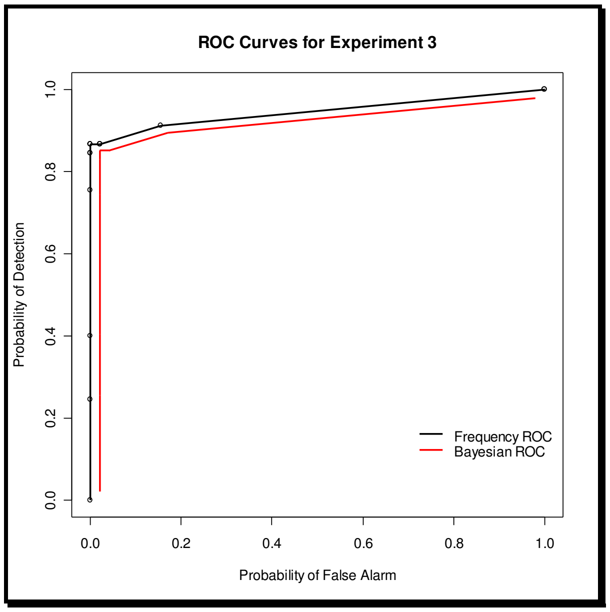

Figure 19: ROC Curves for Experiment 3

Table 6: Summary of Simulation Experiments

Exp Run	No. of topics	No. of queries/ session	No. of document/ query	AUC Frequency	AUC Bayesian
1	6	4	6	0.9217	0.8653
2	6	2	5	0.9121	0.8565
3	6	6	10	0.9358	0.8782

5. RELATED WORK

The three most commonly applied mechanisms for detecting user behaviors related to computer network security are *Statistical anomaly detection, Signature Based Detection,* and *Hybrid Detection* [11]. Statistical anomaly detection usually consists of comparing measurable indicators against a statistical profile estimated from user behavior data recorded over a period of time. Signature based detection, or rule-based detection, is a method that applies sets of generalized rules to represent and store usage patterns. Hybrid Detection combines statistical and rule based detection systems.

In the Ninth Annual IEEE International Conference and Workshop on Engineering of Computer-Based Systems for the year 2002, J. Pikoulas, W. Buchanan, M. Mannion, and K. Triantafyllopoulos published a paper titled: "An Intelligent Agent Security Intrusion System" [16]. Their paper presents a distributed approach to network security. They developed a Bayesian statistical model to predict user actions. Invalid behavior is determined by comparing users' current behavior with their typical behavior. This comparison is based on a set of general rules, obtained from system administrators, that characterize typical user behavior patterns.

In a workshop on preventing, detecting, and responding to malicious insider misuse, approximately 40 researchers and government research met in August 1999 to address and recommend technical research to mitigate the insider threat [21]. The workshop provided key recommendations that include insider threat and vulnerabilities, preventions techniques, detections algorithms, and response measures.

Daniel Burroughs, Linda Wilson, and George Cybenko [17] provided an analysis of distributed intrusion detection systems using Bayesian methods to classify intrusion detection system events into attack sequences. The main concept of their work is to defend computer networks against outsider attackers. Information provided by intrusions detection systems (IDS) are gathered and divided into its component parts such that the activity of individual attackers is made clear. The approach involves the application of Bayesian methods to data being gathered from distributed IDS in order to improve the capabilities for early detection of distributed attacks against infrastructure and the detection of the preliminary phases of distributed denial of service attacks.

Security Situation Assessment and Response Evaluation (SSARE) [12] is a mixed-initiative system for wide-area cyber attack detection, situation assessment, and response evaluation. SSARE is designed to detect a large-scale attack in progress, display an assessment of the situation, and identify responses. Hierarchies of dynamic Bayesian network models were developed to estimate the likelihood of an attack situation. The SSARE models are applied by dynamically supplying evidence to a Bayesian reasoner, which constructs a problem-specific Bayesian network and uses Bayes Rule to compute the probability that an attack is occurring. SSARE provides understanding and timely management of rapidly changing cyber battle space through the application of dynamic, knowledge-intensive, Bayesian and decision-theoretic methods. It dynamically composes models in a data-driven way to develop situation-specific hypotheses about potential breaches in security, thus providing essential support for the central task of cyber command and control.

6. CONCLUSION AND FUTURE WORK

We presented a model to detect insider threat behavior using multi-entity Bayesian networks (MEBN). MEBN technology permits us to represent the knowledge about the insider user behavior in a flexible way. It is essential to mention, however, that while the MFrags that we developed are broadly applicable to users of almost any computing systems, the specific applications may vary from one organization to another and even from one user to another. Therefore, the model will need to be tailored to the organization requirements. The simulation experiments have demonstrated the ability of the IBN model to draw reasonable inferences by using data generated from the model. The IBN model was able to detect threats with reasonably high reliability and low false alarm rate. Additional indicators would increase the ability to detect threats. Future work will evaluate the benefit of including the other factors modeled in our MFrags. Currently we are conducting sensitivity analysis to test the robustness of the IBN mode. The sensitivity analyses include generating observations from one model and do inference using a different model. A future research direction is learning the parameters of the IBN model from field data and test the model against field data not used to train the model.

7. ACKNOWLEDGMENT

Sincere appreciation goes to the Royal Saudi Naval Forces for supporting Ghazi AlGhamdi during his Ph.D. at George Mason University. We thank James H. Jones for his expert evaluation of the IBN model that greatly helped in improving it. The comments and suggestions provided by Paulo Costa, Tomas Shackelford, Mehul Revankar, and the GMU decision theory seminar participants in many discussions are also gratefully acknowledged.

References

1. Pearl, J. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* Series in Representation and Reasoning. Morgan Kufmann Publishers, San Francisco, CA.
2. Finn V. Jensen, "Bayesian Networks and Decision Graphs", Springer-Verlag 2001.
3. Kathryn Laskey and Suzanne Mahoney, "Network Fragments: Representing Knowledge for Constructing Probabilistic Model", 13th Conference on Uncertainty in Artificial Intelligence, 1997.
4. Menezes, P Oorschot, and S. Vanstone, "Handbook of applied cryptography" CRC Press LLC, 1997.
5. Laskey, K.B.; Mahoney, S.M.; "Network engineering for agile belief network models", IEEE Transactions on Knowledge and Data Engineering, Volume: 12 Issue: 4, Jul/Aug 2000, Page(s): 487 -498.
6. Judea Pearl, "Decision Making Under Uncertainty". ACM Computing Surveys, Vol. 28, No. 1, March 1996.
7. The eighth annual CSI/FBI 2003 report: "Computer Crime and Security Survey".
8. Kathryn Laskey. MEBN: logic for Open-World Probabilistic Reasoning. http://ite.gmu.edu/~klaskey/index.html.
9. Kathryn Laskey, Ghazi AlGhamdi, Xun Wangg, Daniel Barbará, Tom Shackelford, Ed Wright, Julie Fitzgerald. Detecting Threatening Behavior Using Bayesian Networks. BRIMS 04.
10. Neumann, Peter. *The Challenges of Insider Misuse*, August 1999, SRI Computer Science Lab.
11. Chris Herringshaw, "Detecting Attacks on Networks", Industry Trends, December Computer Volume: 30 Issue: 12, Page(s): 16 -17 1997.
12. D'Ambrosio, B.; Takikawa, M.; Fitzgerald, J.; Upper, D.; Mahoney, S.; "Security Situation Assessment and Response Evaluation (SSARE)", Proceedings of DARPA Information Survivability Conference & Exposition II, 2001. DISCEX '01. Volume: 1, 2001. Page(s): 387 -394 vol.1.
13. Masami TAKIKAWA, D'Ambrosio, and Ed Wright; "Real-Time Inference with Large Scale Temporal Bayes Nets", Uncertainty in Artificial Intelligence, Page(s) 477-484.
14. Bradley J. Wood. *An Insider Threat Model for Adversary Simulation.* SRI International Cyber Defense Research Center, System Design Laboratory Albuquerque, New Mexico (USA).
15. Ayyub, B. and McCuen, R. Probability, Statistics, and Reliability for Engineers and Scientist. By Chapman & Hall/CRC Press LLC, 2003.
16. J. Pikoulas, W. Buchanan, M. Mannion, and K. Triantafyllopoulos. "An Intelligent Agent Security Intrusion System". Engineering of Computer-Based Systems, 2002. Proceedings. Ninth Annual IEEE International Conference and Workshop, 2002. Page(s): 94 -99.
17. Daniel Burroughs, Linda Wilson, and George Cybenko. "Analysis of Distributed Systems Using Bayesian Methods". Performance, Computing, and Communications Conference, 2002. 21st IEEE International , 2002 Page(s): 329 -334.
18. Paulo C. G. da Costa and Kathryn B. Laskey. "Multi-Entity Bayesian Networks Without Multi-Tears". Draft Version 3/06/2005.
19. Mahoney, S. M. and Laskey, K. B. 1998. Constructing Situation Specific Networks. Uncertainty in Artificial Intelligence: Proceedings of the Fourteenth Conference, San Mateo, CA, Morgan Kaufmann.

20. Marchette, D.J. 2001. *Computer Intrusion Detection and Network Monitoring*. Springer-Verlag New York, Inc.
21. Result of a Three-Day Workshop: *Research and Development Initiatives Focused on Preventing, Detecting, and Responding to Insider Misuse of Critical Defense Information Systems*. August 1999.
22. Andrew Sage, System Management for Information Technology and Software Engineering. Wiley Series in Systems Engineering / Andrew P. Sage, Series Editor 1995.
23. Shacelford, Thomas, 2005, Using Data Mining Techniques to Develop Measures of Document Relevance, PhD Dissertation, George Mason University.