Computers & Security (2005) ■, ■–■

ELSEVIER

Computers & Security

www.elsevier.com/locate/cose

PAID: A Probabilistic Agent-Based Intrusion Detection system

Vaibhav Gowadia, Csilla Farkas*, Marco Valtorta

Information Security Laboratory, Department of Computer Science and Engineering, University of South Carolina, Columbia, SC 29208, USA

Received 9 August 2004; revised 16 May 2005; accepted 16 June 2005

KEYWORDS
Intrusion detection;
Network security;
Computer security;
Computer attack;
Agents;
Bayesian Networks

Abstract In this paper we describe architecture and implementation of a *Probabilistic Agent-Based Intrusion Detection* (*PAID*) system. The PAID system has a cooperative agent architecture. Autonomous agents can perform specific intrusion detection tasks (e.g., identify IP-spoofing attacks) and also collaborate with other agents. The main contributions of our work are the following: our model allows agents to share their *beliefs*, i.e., the probability distribution of an event occurrence. Agents are capable to perform *soft-evidential update*, thus providing a continuous scale for intrusion detection. We propose methods for modelling errors and resolving conflicts among beliefs. Finally, we have implemented a proof-of-concept prototype of PAID.

Introduction

As the complexity of computer systems increases and attacks against them become more and more sophisticated, high-assurance intrusion detection techniques need to be implemented. During the last two decades, many strategies and methods for intrusion detection have been developed (for a survey see Axelsson, 2000).

The main goal of any IDS is to detect all intrusions and only intrusions in an efficient way. Correctness of an IDS is measured by the rate of false positives and false negatives over all events. A false positive warning occurs when a non-intrusive event is labelled intrusive. A false negative warning occurs when an intrusive activity is not detected. Negative effects of false positives include false accusations, reduced system availability, and subsequent disregard of IDS warnings. The negative effects of false negatives include reduced trust in IDS and damages caused by the intrusions. For effective intrusion detection, it is necessary that IDSs reduce the number of misclassifications and find an acceptable balance between false positive and false negative rates. Dacier (2002) found that most of the false positives are generated due to under-specified attack signatures, intent-guessing signatures, or lack of abstraction. Therefore, it is important to specify signatures

* Corresponding author.
E-mail address: farkas@cse.sc.edu (C. Farkas).

0167-4048/$ - see front matter © 2005 Published by Elsevier Ltd.
doi:10.1016/j.cose.2005.06.008

precisely and develop IDSs that can process these signatures efficiently.

Moreover, network-based distributed attacks are difficult to detect because their detection requires coordination among different intrusion detection components or systems (Snapp and Brentano, 1991; Neumann and Porras, 1999). Failure to recognize these attacks leads to false negatives. Therefore, the development of models and protocols for information sharing among intrusion detection components is critical. IDSs are often categorized as distributed or centralized. Spafford and Zamboni (2000) defined centralized IDSs as those where the analysis of data is performed at a fixed number of locations, independent of how many hosts are monitored. Distributed IDSs are defined as those where the analysis of the data is performed at a number of locations proportional to the number of hosts monitored. Centralized IDSs are able to use all available audit data to form a decision, but they create large communication overhead, require a powerful central processor, and represent a single point of failure. To overcome this problem, distributed IDSs process audit data at multiple locations. Distributed IDSs, like DIDS (Snapp and Brentano, 1991) and AAFID (Balasubramaniyan et al., 1998; Spafford and Zamboni, 2000), can share filtered raw data or binary (i.e., yes/no) decisions among their components. However, they cannot share probability distributions of intrusion beliefs. Moreover, existing distributed IDSs do not support selective sharing of published data among peers. In this work, we propose a middle ground between centralized and distributed IDSs, where each IDS component shares its data or results only with those agents that subscribe for these results.

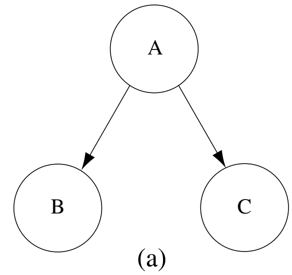
We believe that precise representation of attack signatures in probabilistic intrusion detection model requires: (1) ability to process observations (hard findings) and beliefs (probability distributions) about system parameters, and (2) flexibility in specifying threshold value of probability, above which an alarm is generated.

In this paper, we focus on distributed IDSs based on Bayesian technology and multiagent technology. IDSs based on Bayesian technology may allow sharing of raw data and results (probability distributions) among IDS components. A *Bayesian Network* (BN) is a graphical representation of the joint probability distribution for a set of discrete variables. The representation consists of a directed acyclic graph (DAG). Nodes of the DAG represent variables and edges represent cause–effect relations. The strength of each effect is modelled as a probability. These probabilities are represented by a conditional probability table (CPT). CPTs specify conditional probability of the variable given its parents. For variables without parents, this is an unconditional distribution. Inference in Bayesian Network means computing the conditional probability for some variables given information (evidence) on other variables.

The traditional Bayesian inference (Jensen, 2001; Pearl, 1988) can be performed only with hard findings or observations as input. However, in intrusion detection scenarios modelled with Bayesian Networks, we often find that the input variables cannot be measured directly. Only a belief (probability distribution) in the current state of these variables may be computed. A typical example of such input is a probabilistic result computed by another IDS component. To accept results of other IDS components, existing IDSs (DuMouchel, 1999; Valdes and Skinner, 2000; Sebyala et al., 2002; Cho and Cha, in press) based on the traditional Bayesian inference technique have to coerce the result into a binary decision. Such coercions are performed by assuming occurrence (or non-occurrence) of represented event if the input probability is greater (or smaller) than a threshold value. We believe that IDSs that utilize such binary decisions have limited flexibility and have difficulty in removing false positives and false negatives.

We illustrate our above observation by a simple example given in Fig. 1. Fig. 1(a) shows a Bayesian Network that accepts two inputs: B (hard finding) and C (soft finding), and computes belief in A. Fig. 1(b) shows conditional probability tables for the example BN. Fig. 1(c) shows the likelihood of A being in ''abnormal'' state with respect to the likelihood of C being in ''abnormal'' state. Likelihood of C is computed with the two methods, with and without coercion. In both calculations we have assumed that B is observed to be in ''abnormal'' state. We observe that the likelihood graph of A is continuous when soft-evidential update is used. In this case the security officer has large flexibility in choosing a warning threshold for A. We also observe that the likelihood graph of A is not continuous with traditional probability update. Moreover, the likelihood of A has only discrete values that depend on the threshold set for C. We have developed a Bayesian Network-based technique that allows the IDS components to share results of their analysis in the form of beliefs. Such sharing enables our model to perform intrusion detection on a continuous scale.

Agents are software systems that function autonomously to achieve desired objectives in their environment. Recent research (Spafford and



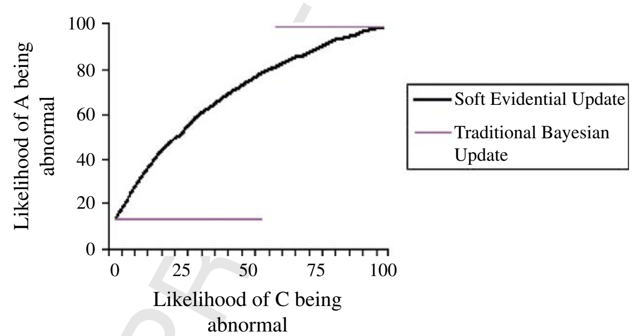
(a)

P(A = normal)	P(A = abnormal)
0.97	0.03

B	P(A = normal)	P(A = abnormal)
normal	0.98	0.02
abnormal	0.01	0.99

C	P(A = normal)	P(A = abnormal)
normal	0.95	0.05
abnormal	0.03	0.97

(b)



(c)

Figure 1 (a) Example BN, (b) CPTs for the example BN, and (c) likelihood of A being abnormal calculated with soft-evidential update and traditional probability update ($C_{\text{threshold}} = 60\%$).

Zamboni, 2000; Jansen et al., 1999; Carver et al., 2000; Helmer et al., 2003) shows that agent-based technology seems to be a promising direction for developing collaborative intrusion detection systems. We propose an agent-based, cooperative architecture where each IDS component is able to process its own data and to integrate local findings with the findings of other IDS components. Each agent acts as a wrapper for a Bayesian Network. Agents in our model utilize the communication protocols and languages (Bellifemine et al., 1999; FIPA, 2002a) developed by multiagent research community. In addition, they use Bayesian inference with soft-evidential update to support integration of beliefs and observations. We refer to this multiagent architecture as Agent Encapsulated Bayesian Network (AEBN) (Bloemeke and Valtorta, 2002). Although, Bayesian Network-based architectures have been considered for intrusion detection (DuMouchel, 1999; Valdes and Skinner, 2000; Barbara et al., 2001; Sebyala et al., 2002; Cho and Cha, in press), these models use traditional probability-update methods (Jensen, 2001; Pearl, 1988). They can effectively utilize only those parameters that result from an actual measure. This limitation often results in under-specified signatures. To solve this problem, in our model agents are enabled to share their beliefs (*soft findings*) in addition to measured values (*hard findings*).

More specifically, we propose an agent-based and cooperative architecture, called Probabilistic Agent-Based Intrusion Detection (PAID), to analyze system information and estimate intrusion probabilities. Agents in PAID accept facts and derived values as inputs. Agents may share their beliefs with, or request information (belief or data) from the other agents.

Our model uses three types of agents: system-monitoring agents, intrusion-monitoring agents and registry agents. System-monitoring agents are responsible for collecting, transforming, and distributing intrusion specific data upon request and evoking information collecting procedures. Each intrusion-monitoring agent encapsulates a Bayesian Network and performs belief update as described in Valtorta et al. (2002) using both facts (observed values) and beliefs (derived values). Intrusion-monitoring agents generate probability distributions (beliefs) over intrusion variables that may be shared with other agents. Each belief is called a soft finding. Soft findings can indicate that a system is in an abnormal state. Even in the absence of hard findings, soft findings can affect the probability of intrusion occurrence or attack against the monitored system. A probabilistic representation of attacks, using hard and soft findings makes our model capable of identifying variations of known intrusions. Coordination between system-monitoring and intrusion-monitoring agents is provided by a registry agent. Within an IDS collaborative model, there may exist several registry agents that, upon failure, can compensate for each other. However, each intrusion-monitoring and system-monitoring agent is registered with a registry agent central to the monitoring agents.

Currently our model detects known intrusions by using well-documented patterns of attacks. Each intrusion-monitoring agent is looking for a particular intrusion pattern. If such a pattern is found, a possible intrusion is indicated. Distributed intrusion detection is achieved by enabling agents to share their beliefs. Depending on the level of collaborations and privacy concerns of the collaborating entities, each component may be able to build the full, global decision stage or only a partial one.

The organization of this paper is as follows. Next section gives a brief introduction to Bayesian Networks, and agent technology. Then, background information and related work followed by the description of the proposed framework (PAID) are given. Further, methodology for building BNs for intrusion detection is presented which is followed by implementation of the proposed intrusion detection framework. Finally, we conclude and recommend future research in last section.

Background

Bayesian Networks

A *Bayesian Network* (BN) is a graphical representation of the joint probability distribution for a set of discrete variables. The representation consists of a directed acyclic graph (DAG), prior probability tables for the nodes in the DAG that have no parents and conditional probability tables (CPTs) for the nodes in the DAG given their parents. As an example, consider the network in Fig. 2.

More formally, a Bayesian Network is a pair composed of: (1) a multivariate probability distribution over n random variables in the set $V = V_1,\ldots, V_n$, and (2) a directed acyclic graph (DAG) whose nodes are in one-to-one correspondence with $V_1,\ldots, V_n$. (Therefore, for the sake of convenience, we do not distinguish the nodes of a graph from variables of the distribution.)

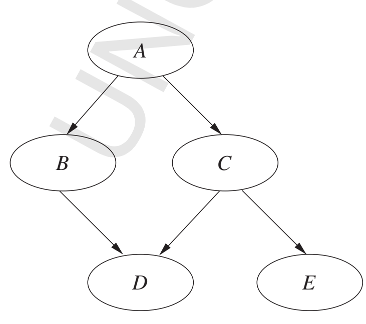
Bayesian Networks allow specification of the joint probability of a set of variables of interest in a way that emphasizes the qualitative aspects of the domain. The defining property of a Bayesian Network is that the conditional probability of any node, given any subset of non-descendants, is equal to the conditional probability of that same node given the parents alone. The chain rule for Bayesian Networks (Neapolitan, 1990) given below follows from the above definition.

''*Let $P(V_i \mid \pi(V_i))$ be the conditional probability of V_i given its parents. (If there are no parents for V_i, let this be $P(V_i)$.) If all the probabilities involved are nonzero, then $P(V) = \prod_{v \in V} P(v \mid \pi(v))$*''.

Three features of Bayesian Networks are worth mentioning. First, the directed graph constraints the possible joint probability distributions represented by a Bayesian Network. For example, in any distribution consistent with the graph of Fig. 2, D is conditionally independent of A given B and C. Also, E is conditionally independent of any subset of the other variables given C.

Second, the explicit representation of constraints about conditional independence allows a substantial reduction in the number of parameters to be estimated. In the example, assume that the possible values of the five variables are as shown in Fig. 2(b).

Then, the joint probability table $P(A, B, C, D, E)$ has $2 \times 3 \times 2 \times 4 \times 4 = 192$ entries. It would be



(a)

A	A_1, A_2
B	B_1, B_2, B_3
C	C_1, C_2
D	D_1, D_2, D_3, D_4
E	E_1, E_2, E_3, E_4

(b)

	A_1	A_2
B_1	0.2	0.1
B_2	0.6	0.6
B_2	0.2	0.3

(c)

Figure 2 (a) An example Bayesian Network, (b) variable states, and (c) conditional probability table for B given A.

very difficult to assess 191 independent parameters. However, the independence constraints encoded in the graph permit the factorization $P(A, B, C, D, E) = P(A) \times P(B \mid A) \times P(C \mid A) \times P(D \mid B, C) \times P(E \mid C)$ which reduces the number of parameters to be estimated to $1 + 4 + 2 + 18 + 6 = 31$. The second term in the sum is the table for the conditional probability of B given A. This probability is shown in Fig. 2(c); note that there are only four independent parameters to be estimated since the sum of values by column is one.

Thirdly, the Bayesian Network representation allows a substantial (usually, dramatic) reduction in the time needed to compute marginals for each variable in the domain. The explicit representation of constraints on independence relations is exploited to avoid the computation of the full joint probability table in the computation of marginals both prior and conditioned on observations. Limitation of space prevents the description of the relevant algorithms; see Jensen (2001) for a discussion of the junction tree algorithm.

The most common operation on a Bayesian Network is the computation of marginal probabilities both unconditional and conditional upon evidence. Marginal probabilities are also referred as *beliefs* in the literature (Pearl, 1988). This operation is called probability updating, belief updating, or belief assignment.

We define evidence as a collection of findings. A *(hard) finding* specifies which value a variable is in. A *soft finding* specifies the probability distribution of a variable. These definitions of finding and of evidence may be generalized, for example, by allowing specifications of impossible configurations of pairs of variables (Cowell et al., 1999; Lauritzen and Spiegelhalter, 1988; Valtorta et al., 2002). However, applications rarely need the power of the more general definitions, and most Bayesian Network software tools support only the definition of (hard) evidence as a collection of (hard) findings given here.

Agent Encapsulated Bayesian Networks

Although there is no universally accepted definition of agent, most authors agree that agents share the following properties: each agent is autonomous, has a set of goals, and has a local model of the part of the world that affects the achievement of its goals, and has a way of communicating with other agents. In an Agent Encapsulated Bayesian Network (AEBN) (Bloemeke and Valtorta, 2002), each agent uses a single Bayesian Network (which is also called an AEBN) as its model of the world. The agents communicate via passing messages that are distributions on variables shared between the individual networks.

The variables of each AEBN are divided into three groups: those about which other agents have better knowledge (input set), those that are used only within the agent (local set), and those for which the agent has the best knowledge and which the other agents may want to use (output set). The variables in the input set and the output set are shared with other agents. The variables in the local set are not. An agent subscribes to zero or more variables in the input set and publishes zero or more variables in the output set.

The mechanism for integrating the view of the other agents on a shared variable is to replace the agent's current belief (which is a probability distribution) in that variable with that of the communicating agent. The update of a probability distribution represented by a Bayesian Network upon receipt of a belief is called a soft-evidential update and is explained in detail by Valtorta et al. (2002). In this work, we have used the Big Clique algorithm for soft-evidential update, implemented in the BC—Hugin system (Kim et al., 2004).

When a publisher makes a new observation, it sends a message to its subscribers. The subscribers in turn adjust their internal view of the world and send their published values to their subscribers. Assuming that the graph of agent communication (which we simply call agent graph) is a directed acyclic graph (DAG), equilibrium is reached, and a kind of global consistency is assured because the belief in each shared variable is the same in agents that subscribe to that variable.

The restriction that an agent has correct and complete knowledge of the variables it publishes forces unidirectional communication, and it may seem excessive. However, there is a good reason to insist on this requirement. The alternative (i.e., to allow bidirectional communication between agents) requires that the agent graph be a tree, as shown in Xiang (2002). Most agent-based systems demonstrate acyclic graph communication model. For example, it is possible to have multiple views of the same parameter. That is, two agents may publish variables that correspond to their measurement (or belief) of the same parameter. Moreover, nothing prevents another agent from integrating the published values of these two agents, thus obtaining a new (and possibly more accurate) view of the parameter.

Table 1 summarizes some features of AEBNs and other related representation formalisms. AEBNs have very good scalability and shared variables are independent of variables in descendant BNs.

Table 1 Agent Encapsulated Bayesian Networks and related representation formalisms

Name	Granularity	Topological restrictions	Constraints on independence relations	Purpose	Scalability
Bayesian Network (Jensen, 2001)	Individual variable	DAG of variables	Local Markov condition (*d*-separation)	Efficient representation of multivariate probability distribution	Poor
Multiply Sectioned Bayesian Network (MSBN) (Xiang, 2002)	Bayesian Network (BN)	Tree (of BNs)	*d*-Separation on composition of BNs;	Efficient distribution of computation among processors	Good: distributed computation, if tree decomposition is possible
Multiple Entity Bayesian Networks (MEBN) (Laskey et al., 2001)	Bayesian Network Fragments (BNFrags)	DAG (of BNFrags)	*d*-Separation on composition of BNs; encapsulation	Distributed representation of Bayesian Networks	Mediocre: representation decomposed, computation centralized
Agent Encapsulated Bayesian Networks (AEBN) (Bloemeke and Valtorta, 2002)	Bayesian Network (BN)	DAG (of BNs)	Shared variables independent of variables in descendent BNs given parent BNs; encapsulation	Construction of interpretation models by collaborating agents	Very good: distributed computation, distributed representation
Decentralized Sensing Networks (DSN) (Utete, 1998)	Sensor	Undirected graph (of sensors)	None: non-probabilistic approach	Distributed sensing and data fusion	Poor: rumor problem is unsolvable in DSNs

Therefore, we chose to use the AEBN organization for the work described in this paper.

We now briefly overview related research work on intrusion detection with the help of Bayesian networks or agent technology.

Bayesian Networks based intrusion detection

IDSs using Bayesian Networks have been proposed by many researchers (DuMouchel, 1999; Valdes and Skinner, 2000; Barbara et al., 2001; Sebyala et al., 2002; Cho and Cha, in press). However, these IDS models use only hard findings in their Bayesian models. We now briefly overview their IDS architectures.

DuMouchel (1999) proposed an anomaly detection technique using the Bayes classifier. They keep a profile of commands issued by each user and compute command transition probabilities. Their IDS detects abnormal behavior based on the observed command transitions.

Valdes and Skinner (2000) proposed an adaptive model that detects attacks using probability theory. Their architecture analyzes the traffic from a given client's TCP sessions. This analysis is done by Bayesian inference at periodic intervals in a session, and the interval is measured in number of events or elapsed time. Between inference intervals, the system state is propagated according to a Markov model. After each inference, the system may give alerts for suspicious sessions.

Sebyala et al. (2002) have incorporated Bayesian Network in their IDS as anomaly detector. They keep a profile of CPU and memory utilization by proxylets in active networks. They use a Bayesian Network to compute state (good or bad) of proxylet. A proxylet is in bad state if the CPU and memory utilization is anomalous.

Cho and Cha (in press) proposed a technique to detect anomalies in web sessions. A web session consists of sequence of page requests. Anomalous request in given web session may correspond to request for secured pages without accessing the login page, or repeated access to a same page. Their model utilizes Bayesian parameter estimation technique (Friedman and Singer, 1998) to compute probability that a user may request certain pages in given sequence. A web session may consist of multiple sub-sessions. To combine the anomaly scores of these sub-sessions, they suggest use of either maximum value (for high sensitivity) or average value (for low sensitivity).

In the above models, Bayesian inference is performed when a hard evidence is received like a command sequence, TCP parameters, CPU or memory utilization, or a page request. None of the above models describe Bayesian model for attacks when the input observation is a probability distribution over the states of a system parameter. For example, there is 80% chance of a DOS attack on the file server, or there is a 70% chance for command sequence to be anomalous. Unlike existing models, our model allows accepts beliefs (probability distributions) as input to the Bayesian models.

Agent-Based Intrusion Detection systems

Agent-based systems require a communication infrastructure. Agents in our system communicate with each other by sending messages in the Agent Communication Language specified by FIPA (FIPA, 2002a,b). JADE (Bellifemine et al., 1999) is a software framework to aid the development of agent applications in compliance with the FIPA specifications for inter-operable intelligent multiagent systems. The purpose of JADE is to simplify development while ensuring standard compliance through a comprehensive set of system services and agents. To achieve such a goal, JADE offers a distributed agent platform, directory facilitator (DF), and library of interaction protocols. The agent platform includes an agent management system that allows monitoring and logging of agent activities and performs life-cycle operations (start, suspend, and terminate) on agents. Interaction protocols (e.g., request, query, subscribe, etc.) are used to design agent's interaction, providing a sequence of acceptable messages and semantics for those messages.

Agent communications can be divided into two categories: communication among agents at the same host and communication among agents at different hosts. Balasubramaniyan et al. (1998) examine these methods in the context of intrusion detection.

Spafford and Zamboni (2000) and Balasubramaniyan et al. (1998) presented a framework called AAFID in which autonomous agents report their findings to entities called transceivers. Each host has a unique transceiver that collects information from all other agents on its host machine. Agents also perform data reduction and send data to monitors that oversee the operation of several transceivers. Monitors have the capability to detect events that may be unnoticed by the transceivers. In mobile agent-based systems, like the ones presented by Helmer et al. (2003) and Asaka et al. (1999), mobile agents collect, integrate, and analyze data from different components of a distributed system. The agent's findings are recorded in a database and/or reported to the users.

System design goals

One of the design goals of our IDS is to enable it to function as a stand-alone system or to support existing IDSs. Our main goal is to improve upon existing IDS technologies by allowing flexible information sharing among system components in a way that the shared data are easily incorporated in the analysis of the components. Our model supports the calculation of intrusion probabilities on a continuous scale of [0, 1]. A probability of zero means it is certain no intrusion has occurred, and one means that an intrusion has definitely occurred. For each intrusion type there is an associated variable that represents the probability of that intrusion. Each Bayesian network is able to modify its own belief (probability distribution over an intrusion variable) and to import or export beliefs from or to other Bayesian networks. These input variables are accepted during all states of processing.

Analysis of distributed attacks on a large network may require monitoring of numerous hosts and large volumes of network traffic. Thus a large amount of data is generated that must be analyzed. Our model supports local analysis of collected data and sharing of results (and partial results). We also allow agents to share probability distributions (beliefs) of intrusion occurrences and system states. This ''belief sharing'' carries more information than sharing a binary decision and also has a lower overhead than raw data sharing.

Each intrusion-monitoring site or network may have different sensitivity and selectivity requirements. Our model allows security officers to customize these parameters according to the local requirements. This customization does not affect the probability distribution values shared among the agents.

Finally, we address some of the issues related to reliability and ease of maintenance. Based on the distributed nature of our model and the possibility of replicated Bayesian Networks for monitoring intrusion, our model remains functional even if some of the IDS network nodes are unavailable.

Since the detection of an intrusion is based on several parameters, including local findings and findings from several other agents. The misleading data from compromised agents are small if the number of non-compromised agents is large and the number of compromised agents is small. Each Bayesian Network is responsible to monitor a particular intrusion; therefore the modification of an intrusion pattern will affect only those networks that monitor the intrusion. Similarly, protection against new types of attacks can be added easily to the model.

Probabilistic Agent-Based Intrusion Detection

In our model, we use agent graphs to represent intrusion scenarios. Each agent is associated with a set of input variables, a set of output variables, and a set of local variables. The agent at each node of the graph encapsulates a Bayesian Network. Nodes of the Bayesian Network are variables that represent suspicious events, intrusions, or system and network parameter values. A variable can have any number of states, and the belief in the variable is the distribution on its states. The encapsulated Bayesian Network is used to model intrusion scenarios. It is also able to incorporate measurement errors and handle multiple beliefs on input variables.

PAID architecture

The PAID architecture uses agent technology to collect and analyze data and to distribute information among the PAID components. PAID supports three types of agents: (1) system-monitoring agents, (2) intrusion-monitoring agents, and (3) registry agents.

(1) *System-monitoring agents*: The system-monitoring agents perform either online or offline processing of log data, communicate with the operating system, and monitor system resources. These agents publish their output variables (facts and beliefs derived from observations) that can be utilized by other agents.
(2) *Intrusion-monitoring agents*: Each intrusion-monitoring agent computes the probability for a specific intrusion type. These agents subscribe to variables and/or beliefs published by the system-monitoring agents and other intrusion-monitoring agents. The probability values for each agent are updates, calculated according to the values of input variables and beliefs.
(3) *Registry agent*: The registry agent maintains information about the published variables and monitored intrusions for each system-monitoring or intrusion-monitoring agent. It is required that all agents of PAID must register with the registry agent. The registry agent also maintains the location and current status of all the registered agents. Agent status is a combination of two parameters *alive* and *reachable*. The status of a communication link between any two agents is determined by attempting to achieve a reliable UDP communication between them. The registry agent is used to find information (e.g., name and location) about agents who may supply required data. The PAID architecture can support multiple registry agents also as described later in section 'Scalability and complexity analysis'. For simplicity, we describe the examples with a single registry agent.

Agent communication

The interactions among the components of PAID are shown in Fig. 3. The messages are sent in XML syntax (Bray et al., 2001) among the agents. These messages correspond to registration requests, information requests and other agent actions. A brief overview of agent actions and the corresponding messages is given below.

1. *Registration of an agent with the registry agent*: Each agent in PAID must register with

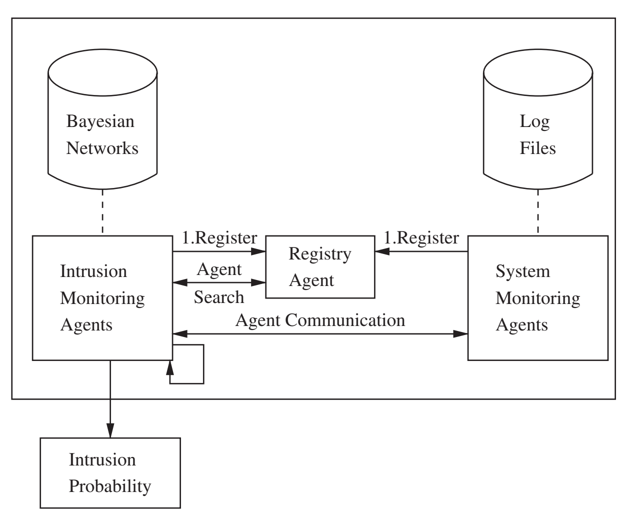


Figure 3 Probabilistic Agent-Based Intrusion Detection (PAID).

the registry agent. A registration message includes the registering agent's agent-id, IP address, list of published variables and their possible states, digital signature, and digital certificate. The registry agent issues an acknowledgment message upon successfully entering the new agent in its database.

2. *Information request by an agent about other agents*: Each intrusion-monitoring agent has a set of input variables (determined from the encapsulated Bayesian Network). To find agents capable of providing required input data, the intrusion-monitoring agent sends a search request to the agent registry. The search request includes the requester's agent-id, IP address, and the required input variables. The message is digitally signed by the requester.
3. *Registry agent's reply to an information request*: Upon receiving a search request, the registry agent verifies that the request is legitimate before searching its database to determine which agents can supply the requested variables and the status of these agents. The message from the registry agent to the requester includes the requested variable name, the agent-id of the agent publishing the variable, its IP address, and status. The message is digitally signed by the registry agent.
4. *Request for belief subscription*: Upon receiving the list of agents capable of providing the required input from the registry, the subscribing agent sends requests directly to these agents. A subscription request consists of the requester's agent-id, requester's IP address, requested input variable name, the duration of subscription time, the desired time interval between subsequent updates, a request-id, and the timestamp of the request. The message is digitally signed by the requester.
5. *Belief-update messages*: Upon receiving a belief subscription request the publishing agent sends regular updates within the agreed intervals and duration of the subscription. The message contains the request-id, the sender's id, and the probability distribution of the requested variable. The message is signed by the publisher.

Communication security and reliability

Reliable and secure communication is achieved by using commercially available encryption techniques to achieve communication security and authentication. Reliability is supported by periodic status update of the active agents. We use secret key encryption for message content to reduce encryption overhead. Message and agent authentication is guaranteed by public-key cryptosystem and the use of digital certificate. Each message is digitally signed by the sending agent. In addition, we require that agents authenticate themselves to the registry by their digital certificates.

Status probes of registered system-monitoring agents and network links are periodically performed by the registry agent. Responses to the probing messages carry information about the state of the system-monitoring and intrusion-monitoring agents. The status of a communication link between any two agents is determined by attempting to achieve a reliable UDP communication between them. Compromised agents can be identified by periodically launching attacks over the monitored network and verifying that the expected results are generated. This approach was proposed by Dacier (2002).

Scalability and complexity analysis

The factors affecting the scalability of our model are the costs of data transfer, belief updates and registry operations (i.e., register, deregister, and query). During normal operation, agents share their beliefs; thus, PAID has a low bandwidth requirement. Sharing of data or partial data is required only to analyze suspicious events.

Pearl (1988) has shown that belief update can be performed in linear time in trees and (more generally) singly connected networks. Unfortunately, belief update in general Bayesian Networks is NP-hard (Cooper, 1990). The computational complexity of the algorithm found to be the best in practice, the junction tree algorithm, is exponential in a graphical parameter called the treewidth of the Bayesian Network. This negative result holds, even for some notions of approximation and for many restrictions on the structure of the Bayesian Network. Despite these negative theoretical results, update in most Bayesian Networks, using the junction tree algorithm Lauritzen and Spiegelhalter (1988) is very fast because most practical Bayesian Networks compile (after an intermediate step that converts them into an indirected graph) into a junction tree where the largest clique is small. The process is described in detail in the literature, for example in Neapolitan (1990). More precisely, the computational complexity of the junction tree algorithm, which is widely found to be the fastest algorithm in practice is exponential in a graphical parameter called the treewidth of the Bayesian Network.

PAID can provide scalability by supporting multiple registries. Each subnet may have its own agent registry. The agent-registries can forward requests and replies to neighboring registries based on the IP address of the receiving agent. Dynamic routing algorithms for IP networks (Moy, 1997; Perlman, 1992) are applicable for this purpose.

Modelling Bayesian Networks for PAID

To assess the probability of an intrusion, we use Bayesian Networks. Modelling a domain with Bayesian Networks involves two major steps. First, a domain expert needs to specify the qualitative structure of the network, which depends solely on the independence relation among the variables of the domain of interest. Second, the numerical parameters need to be assessed; these parameters are the prior probabilities of variables that have no parents, and the conditional probabilities of every other variable given its parents. The graph and the probabilities uniquely and completely specify the joint probability of the variables in the domain of interest.

We now present methodologies for modelling Bayesian Networks for attack patterns, system parameters, incorporating errors, and resolving conflicts.

Bayesian Network building methodology

There are two methods of building Bayesian Networks for a particular application domain. The first method consists of asking the domain expert to construct the network (DAG) and assess the prior and conditional probabilities manually. This is how we build our networks. The second method builds the network from data. There are several algorithms available to accomplish this learning task. These are: BIFROST (Lauritzen and Spiegelhalter, 1988), K2 (Cooper and Herskovits, 1992), and CB (Singh and Valtorta, 1993, 1995). The prior and conditional probabilities can also be computed from data. The models are validated by comparison with the performance of an expert (Spiegelhalter et al., 1993; Neapolitan, 2004). We plan to extend our model to incorporate these algorithms to build Bayesian Networks.

We now illustrate our method of building Bayesian Network model for attack patterns, with an example of a Mitnick attack.

Modelling computer attacks with Bayesian Networks: an example

The Mitnick attack (see Fig. 4) is difficult to identify due to its distributed nature. In a network vulnerable to Mitnick attack, the victim host authenticates a trusted host using an IP address only. The trust relationship between the victim and trusted hosts implies that the users logged in on the trusted host or applications running on the trusted host can access resources on the victim host without secure authentication. Mitnick attack exploits the weakness of IP based authentication systems and a flaw in TCP packet sequence number generation algorithm. An attacker launches a distributed denial of service attack on the trusted host making it temporarily unavailable. The attacker is then able to gain access to the victim host by pretending to be a user from the trusted host. Hence, the identification of a Mitnick attack requires evidence of both IP spoofing and DOS attacks on different machines in the victim network. Soft and hard findings detected on the victim's network can be used to identify the attack. We now examine the Mitnick attack in detail and model possible findings and their dependencies with a Bayesian Network model.

In preparation for the attack, the intruder installs malicious programs (zombies) on many vulnerable computers over the Internet. Meanwhile, the intruder gathers information about the real victim. This information will allow the attacker to successfully guess TCP sequence numbers of the victim host. At a specific time, the attacker activates the zombies to launch a denial of service (DOS) attack against the host trusted by the victim. As a result, the trusted host is unable to reply to packets sent by the victim host. Under such a situation, the intruder tries to open a TCP connection with the victim host by spoofing the IP address of the trusted host. The victim host sends

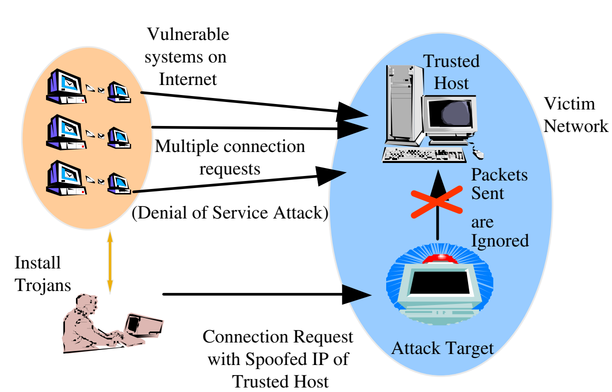


Figure 4 Mitnick attack.

an SYN-ACK packet as a reply to the trusted host. The trusted host drops this packet due to the DOS attack. The attacker now sends an ACK packet to the victim with appropriate sequence number. Upon receipt of this packet, the victim computer assumes that the other party is the trusted host. The attacker is now in a position to use the services provided by the victim host. We observe that the belief of a Mitnick attack's occurrence depends on the belief in occurrence of an IP-spoofing attack and a DOS attack on a trusted server. We add these dependencies to the qualitative structure (Fig. 6) of Mitnick attack's Bayesian model.

During an IP-spoofing attack, the network firewall and/or routing nodes on a network may discover incoming IP-packets with local source IP addresses. There is a high probability of an IP-spoofing attack when such packets are observed. This finding can be detected directly by a network device. The output of network device may say that either such packet was observed, or was not observed. Therefore, we model this finding as a hard finding (called *Local-SrcOnExternalInterface)* supporting the hypothesis of an IP-spoofing attack. Another possible evidence of an IP-spoofing attack is an abnormal variation of the TTL value in the IP header of packets from a trusted host. This variation can be detected against recorded TTL values from the same host. However, such an observation is subjective and there is no clear demarcation between normal and abnormal values. We model this type of finding as a soft finding, and represent them by probability distributions over normal and abnormal states.

In a situation when a server is under a TCP-SYN attack, the server receives a greater number of SYN packets than the number of connections it can handle. An increased ratio of SYN to SYN-ACK packets when observed along with decreased outgoing data packets increases the probability of a TCP-SYN DOS attack and distinguishes it from the normal busy hours of the day.

During a Mitnick attack, applications at a victim host are not able to open new connections with the trusted host, but the victim host is still receiving

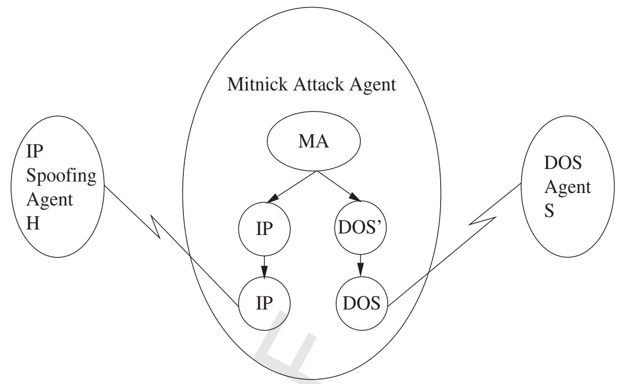


Figure 5 Multiagent system for Mitnick attack.

packets with the source IP address of the trusted host. The former observation by itself may not be sufficient to make an inference about the Mitnick attack, but is very useful when combined with other information. We model this observation as a soft finding called *OneWayCommunication*.

Fig. 5 shows the agent graph to model a simple Mitnick attack. There are three intrusion-monitoring agents: IP Spoofing Agent, Mitnick Attack Agent, and DOS Agent. The Mitnick Attack Agent subscribes to the beliefs of the IP Spoofing Agent and the DOS Attack Agent. Note that the IP Spoofing Agent and DOS Attack Agent may further subscribe to beliefs published by other agents as suggested by the Bayesian Network of Mitnick attack shown in Fig. 6. For simplicity, we do not show these subscriptions.

In addition to the facts and beliefs received from other agents, an agent may have local variables that are used only by this agent. For example, variable *S* is a local variable of the DOS Attack Agent and *H* is a local variable of the IP Spoofing Agent. In addition, the DOS Attack Agent subscribes to the variables corresponding to the number of received SYNs (connection requests) and number of sent SYN-ACKs (connections handled) in a given time period. These values are obtained from system-monitoring agents. Local and input variables are used to calculate the probability distribution of a DOS attack.

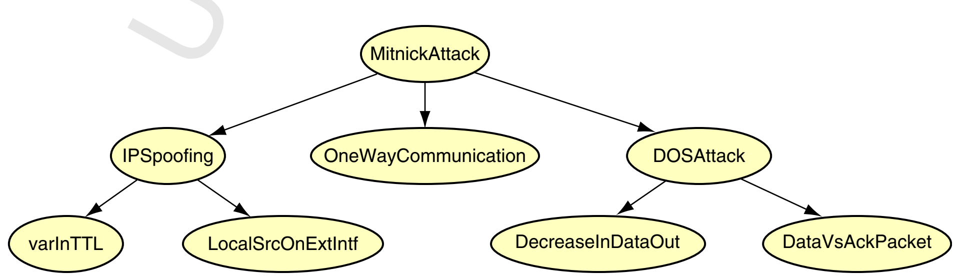


Figure 6 Bayesian Network for Mitnick attack.

Based on the value of S, belief about a DOS attack is computed. We distinguish between three states: low, medium and high probability. If all connection requests are handled, S is equal to zero or has a very small value, thus the probability of the attack is either zero or low. When the system is under attack, with the result that many connection requests cannot be handled, the probability of the attack is high. Actual values that differentiate between high and low states can be determined by applying data mining techniques on log data.

Belief calculation by system-monitoring agents

System-monitoring agents perform simple processing and querying on log files and compute beliefs on variables they publish. These agents use the *method of counts* (Jensen, 2001; Cowell et al., 1999) to estimate the prior marginal probability of a variable being in a certain state by dividing the number of cases in which the variable is in that state by the total number of cases. The method of counts estimates the prior conditional probability of a variable being in a certain state given that its parents are in a certain configuration by diving the number of cases in which the child variable is in that state and the parent variables are in that configuration by the number of cases in which the parent variables are in that configuration.

Modelling errors in measurement

The calculation of a belief depends on factors such as accuracy of measurement and conflicts among beliefs reported by various agents. In our model, if an agent is not able to accurately determine the state of a published variable, the agent publishes a probability distribution (belief) over the possible states of the variable. The publishing agent determines this distribution by incorporating measurement errors. Errors in the measurement of a variable state are modelled within an agent with help of the Bayesian Network shown in Fig. 7. This is achieved by representing the state of a variable with a belief or soft finding. The parent node S represents the actual value of interest. The prior distribution of the actual values is $P(S)$. The measured value is represented by variable S_{obs}. The measurement error is modelled by the conditional probability $P(S_{obs} \mid S)$. In the absence of error, this is a diagonal matrix. The magnitude of non-diagonal entries is directly proportional to the measurement errors. In the special case of a 2×2 matrix, the two entries on the main diagonal quantify the specificity and sensitivity of the measurement, and the other entries quantify the false positive and false negative ratios (Vomlel, 2004). When the actual value is propagated to parent node S, we get a probability distribution over different states of the variable. The agent can publish this distribution as its belief on the state of the measured variable.

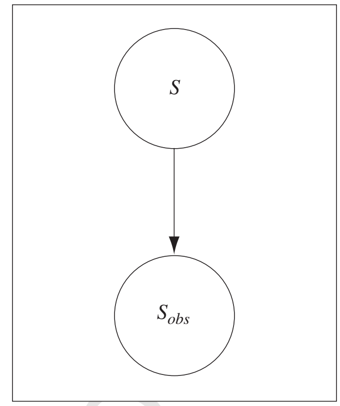


Figure 7 Incorporating error in measurement of variable.

Conflict resolution

Conflicts among beliefs on a state of variable, due to information provided by multiple agents on the same underlying quantity, can be resolved using soft-evidential update. For example, let A_1 and A_2 be two agents that measure a variable v. The values measured by them are B_1 and B_2, respectively. We design a Bayesian Network as shown in Fig. 8. The computed posterior probability of v effectively fuses the information provided by the two agents in the context specified by variable CR.

This approach requires estimating the prior probabilities of B and CR. In most practical uses

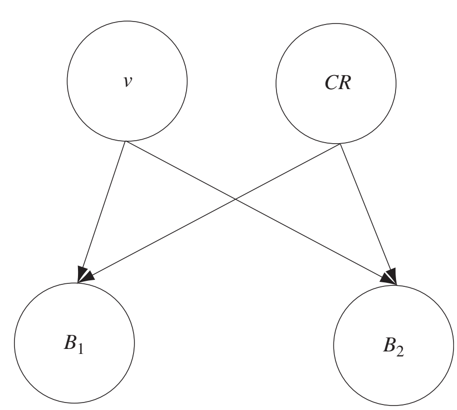


Figure 8 Conflict resolution.

of the Bayesian Network, the value of CR is known, so the assessment of the prior probability of CR does not need to be accurate. The prior probability of B needs to be more accurate than CR. It is normally possible to estimate B by using counts of the values of B in past cases. A similar technique (based on counts) can be used for the conditional probability tables $P(B_1 \mid v, \mathrm{CR})$ and $P(B_2 \mid v, \mathrm{CR})$. See Jensen (2001) and Cowell et al. (1999) for a discussion of the technique in general and Valdes and Skinner (2000) for an application of the technique in an intrusion detection scenario. For the situation involving complete cases, i.e., cases in which all variables are observed, the technique consists simply of replacing the (prior or conditional) probability of interest with the corresponding observed frequency (in the case of prior probabilities) or with a ratio of frequencies (in the case of conditional probabilities). In the more interesting and realistic case in which some variables are not observed, a similar approach, called *fractional updating* (or one of its improved variants) is used, see Jensen (2001) and Cowell et al. (1999) for details. To apply the technique in intrusion detection situation, we require that cases be labelled by attack type.

In special cases, B_1 and B_2 are statements that v is in a particular value. In general, they are probability distributions representing each agent's belief that the variable v has a particular value. The unique feature of the AEBN approach is to allow such general situations, whereas other approaches require the beliefs of the two agents to be hard findings. The process of updating v in the presence of the probability distributions on B_1 and B_2 is called soft-evidential update. In this work, we have used the Big Clique algorithm for soft-evidential update, implemented in the BC—Hugin system (Kim et al., 2004).

Implementation

In this section, we describe our implementation of the proposed architecture from two different perspectives. First, we explain the developer's view of the system, and then we describe how the user (System Administrator) can interact with the IDS.

Developer's perspective

The PAID system uses a behavior-based agent model. In this model, agents are characterized by certain behaviors. A behavior class describes the action that an agent will perform during its life time. Domain specific behaviors are developed by extending class *Behavior* defined in JADE API. These behaviors may be either one shot behaviors or cyclic behaviors. Once a behavior completes its task, it may change its state to inactive by setting the instance variable *done* to true. The underlying agent management system in the agent platform (JADE is this case) invokes agents' active behaviors in each simulation cycle. We now describe the constituent modules of the PAID system:

- Main IDS agent (*IDS*): A singleton agent to supervise the working of the entire system and provide results. IDS agent provides the administrative interface. It also controls other tasks in the PAID system including creation and termination of system-monitoring and intrusion-monitoring agents. IDS agent exhibits *StartAgentsBehavior* and *StopAgentsBehavior*.
- System-Monitoring Agents (*SMAgent*): A class representing the system-monitoring agents in the IDS. This class is responsible for registering itself with the JADE DF, and for executing *PublishingBehavior* and a custom behavior to query log files or measure system performance. The name of custom behavior class is determined by the main IDS agent from the system-audit configuration file and is invoked during runtime with the help of Java Reflection Class API.
- Intrusion-Monitoring Agents (*IMAgent*): A class representing the intrusion-monitoring agents responsible for detecting intrusions. This class is responsible for registering itself with the JADE DF. A Bayesian Network model of intrusion to be monitored by this agent is provided as an argument to this agent on startup. From the input intrusion model, the agent determines required input beliefs and queries the directory facilitator to locate agents publishing those beliefs. This agent then subscribes to beliefs of other agents and updates its belief on intrusion periodically. In other words, intrusion-monitoring agents exhibit *SubscriptionBehavior*, *BeliefUpdateBehavior*, and *PublishingBehavior*.

Administrator's perspective

The administrative interface provided by our implementation is shown in Fig. 9. As described earlier in section 'PAID architecture', the PAID system contains several system-monitoring agents and intrusion-monitoring agents that utilize the directory facilitator (DF) provided by JADE as the

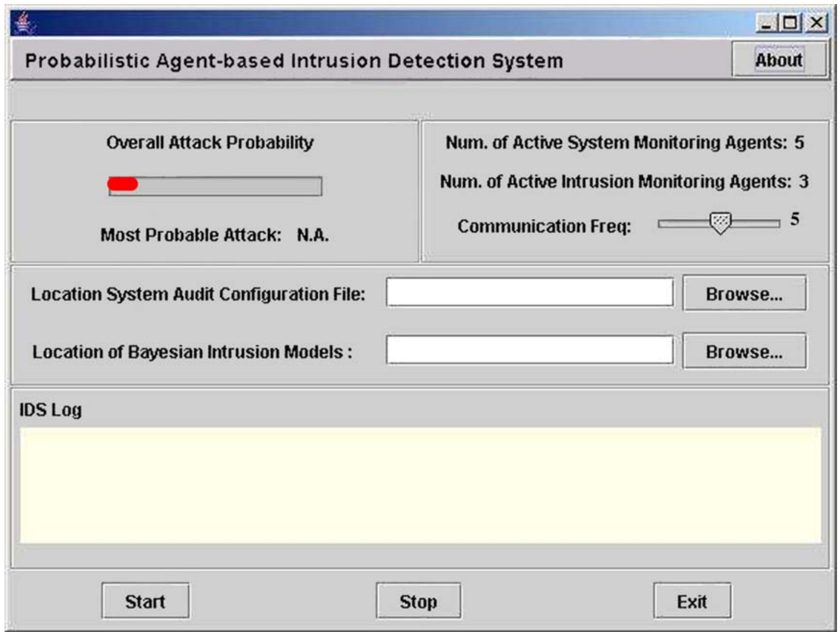


Figure 9 Administrative interface.

registry agent. The interface allows the administrator to choose communication frequency among these agents. To start the IDS, administrator must specify a system-audit configuration file and directory where Bayesian network models for various intrusions are stored. The system-audit configuration file provides bootstrap information (name of Behavior class and input log files) for system-monitoring agents. Multiple intrusion-monitoring agents are started, each with a different Bayesian network model as input. The output of the PAID system is the overall probability for the host computer to be under attack. This value is graphically shown in the user interface. When IDS identifies a probable attack, it brings up detailed probability information of that attack. For example, Fig. 10 shows detailed information for Mitnick attack.

The PAID system goes through the following four phases:

(1) *Initialization phase*: All the agents register themselves with the agent registry on boot-up. JADE provides APIs for enabling the agents to register themselves with the AMS and DF agents for the system. This provides every agent with a globally unique identifier, the Agent-ID (AID), through which the other agents can interact by taking advantage of the white page services provided by the JADE AMS. In addition, each agent has to provide its service description during registration, which the JADE DF uses to provide yellow page services to other agents.

(2) *Analysis phase*: After initialization, agents enter analysis phase. In this phase agents execute *SubscriptionBehavior, BeliefUpdateBehavior*, and *PublishingBehavior*. These behaviors are cyclic, i.e. they are repeated indefinitely after every few seconds determined by the communication frequency set for the session.

(3) *Resetting phase*: The Administrative interface allows the user to reset all the agents by stopping all agents with the *Stop* button and starting the system again. When the system is stopped, all agents are deregistered and terminated. The administrator may then start all the agents again by pressing the *Start* button, or exit the system. This feature can be useful if some agents terminate abnormally during execution and need to be restarted.

(4) *Termination phase*: When the administrator *Exits* the system, all the agents are deregistered and the IDS shuts down.

Conclusions

In this paper, we demonstrated the feasibility of probabilistic intrusion detection technique using soft-evidential updates. We developed and implemented an intrusion detection architecture called Probabilistic Agent-Based Intrusion Detection (PAID). The advantages of our framework over existing models follow.

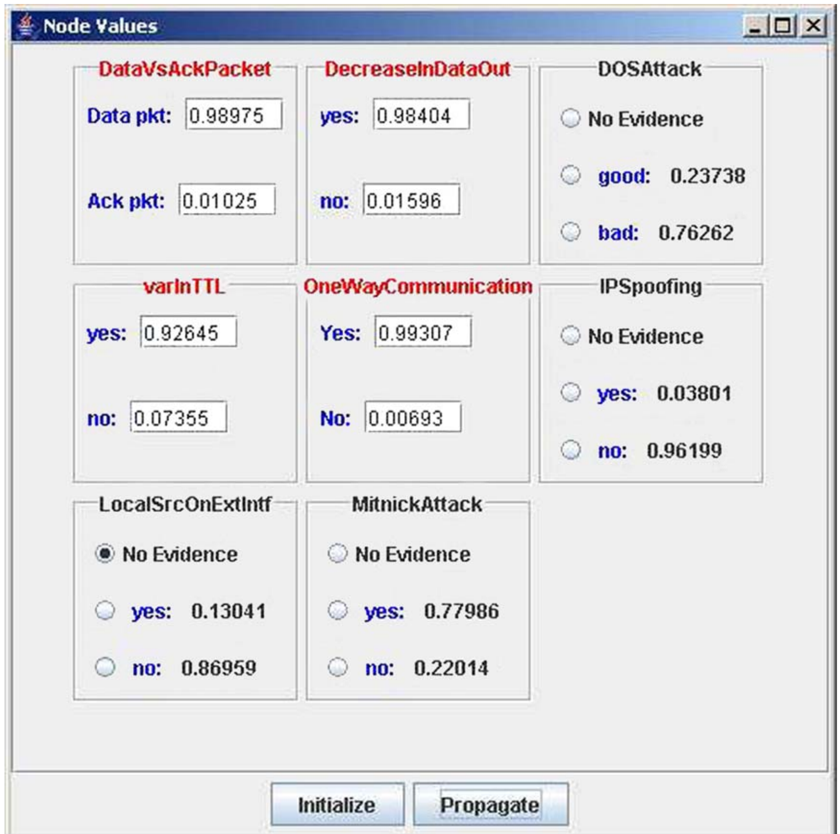


Figure 10 Calculation of attack probability using Big Clique algorithm.

PAID requires low volume of data sharing over network in contrast to centralized data analysis. Although communication overhead is higher than in IDS that allow only binary decision sharing, the improved processing power makes PAID more suitable for sophisticated intrusion detection. PAID also provides a continuous scale to represent event probabilities. This feature allows easy exploration of the trade-off between sensitivity and selectivity that affects the rate of false positive and false negative decisions.

The current version of PAID was illustrated in misuse detection mode, but the same principles can be applied for anomaly-based intrusion detection. Distributed intrusion detection is achieved by allowing each agent to cooperate with others and to build full or partial, global intrusion graphs. Distributed processing not only increases efficiency but also eliminates single point of failure.

A proof-of-concept prototype of our model has been developed using agents developed with Java and C alone. At present we are migrating the complete agent model to JADE framework. We are planning to improve and fine-tune our current model to address agent trust management and dynamic agent-activation protocols.

Acknowledgments

This material is based upon work supported by National Science Foundation under Grant No. IIS-0237782. This work was supported in part by the Advanced Research and Development Agency (ARDA), an entity of the U.S. Government. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the U.S. Government.

References

Asaka M, Taguchi A, Goto S. The implementation of IDA: an intrusion detection agent system. In: Proceedings of 11th annual FIRST conference on computer security incident handling and response. Brisbane, Australia; 1999.

Axelsson S. Intrusion detection systems: a taxonomy and survey. Tech. Rep. 99-15. Göteborg, Sweden: Department of Computer Engineering, Chalmers University of Technology; 2000.

Balasubramaniyan J, Garcia-Fernandez J, Isacoff D, Spafford E, Zam-Boni D. An architecture for intrusion detection using autonomous agents. Coast 98-05. West Lafayette, IN: Department of Computer Science, Purdue University; 1998.

Barbara D, Wu N, Jajodia S. Detecting novel network intrusions using bayes estimator. In: Proceedings of first SIAM conference on data mining; 2001.

Bellifemine F, Poggi A, Rimassa G. JADE — a FIPA compliant agent framework. In: Proceedings of the fourth international conference and exhibition on the practical application of intelligent agents and multi-agents. London; 1999.

Bloemeke M, Valtorta M. The rumor problem in multiagent systems. Tech. Rep. 2002–006. Columbia, SC: Department of Computer Science and Engineering, University of South Carolina; 2002.

Bray T, Paoli J, Sperberg-McQueen C, Maler E. Extensible Markup Language (XML) 1.0 specification. W3C Recommendation, Retrieved October 16, 2002 from, <http://www.w3.org/TR/2000/REC-xml-20001006>; 2001.

Carver C, Hill J, Surdu J, Pooch U. A methodology for using intelligent agents to provide automated intrusion response. In: Proceedings of the IEEE systems, man, and cybernetics information assurance and security workshop. WestPoint, NY; 2000.

Cho S, Cha S. SAD: web session anomaly detection based on parameter estimation. Computers and Security, in press.

Cooper G. The computational complexity of probabilistic inference using Bayesian networks. Artificial Intelligence 1990;42(2–3):393–405.

Cooper GF, Herskovits E. A Bayesian method for the induction of probabilistic networks from data. Machine Learning 1992; 9(4):309–47.

Cowell RG, Lauritzen SL, David AP, Spiegelhalter DJ, Nair V, Lawless J, et al. Probabilistic networks and expert systems. Springer-Verlag New York, Inc.; 1999.

Dacier M. Design of an intrusion-tolerant intrusion detection system. Tech. Rep. D10. IBM Zurich Research Laboratory; 2002.

DuMouchel W. Computer intrusion detection based on Bayes factors for comparing command transition probabilities. Tech. Rep. 91. National Institute of Statistical Sciences; 1999.

FIPA. FIPA ACL Message structure specifications. Retrieved October 16, 2002 from, <http://www.fipa.org/specs/fipa00061>; 2002a.

FIPA. FIPA-OS Developers guide. Retrieved October 16, 2002 from, <http://fipa-os.sourceforge.net/docs/Developers-Guide.pdf>; 2002b.

Friedman N, Singer Y. Efficient Bayesian parameter estimation in large discrete domains. In: Proceedings of neural information processing systems (NIPS 98). MIT Press; 1998.

Helmer G, Wong JSK, Honavar V, Miller L, Wang Y. Lightweight agents for intrusion detection. Journal of Systems and Software 2003;67(2):109–22.

Jansen W, Mell P, Karygiannis T, Marks D. Applying mobile agents to intrusion detection and response. Tech. Rep. 6416. Gaithers-burg, MD: Computer Security Response Center, National Institute of Standards and Technology; 1999.

Jensen F. Bayesian networks and decision graphs. Springer Verlag; 2001.

Kim Y, Valtorta M, Vomlel J. A prototypical system for soft evidential update. Applied Intelligence 2004;21(1):81–97.

Laskey KB, Mahoney SM, Wright E. Hypothesis management in situation-specific network construction. In: Proceedings of the 17th annual conference on uncertainty in artificial intelligence (UAI-01). Seattle, WAAugust 2001. p. 301–9.

Lauritzen S, Spiegelhalter DJ. Local computations with probabilities on graphical structures and their application to expert systems. Journal of the Royal Statistical Society, Series B (Statistical Methodology) 1988;50(2):157–224.

Moy J. OSPF Version 2. Internet draft, RFC-2178; 1997.

Neapolitan RE. Probabilistic reasoning in expert systems: theory and algorithms. New York, NY: John Wiley and Sons; 1990.

Neapolitan RE. Learning Bayesian networks. Upper Saddle River, NJ: Pearson Prentice Hall; 2004.

Neumann P, Porras P. Experiences with EMERALD to date. In: Proceedings of the first USENIX workshop on intrusion detection and network monitoring. Santa Clara, CA; 1999.

Pearl J. Probabilistic reasoning in intelligent systems: networks of plausible inference. San Mateo, CA: Morgan-Kaufmann Publishers; 1988.

Perlman R. Interconnections: bridges and routers. Reading, MA: Addison-Wesley Professional; 1992.

Sebyala AA, Olukemi T, Sacks L. Active platform security through intrusion detection using nave Bayesian network for anomaly detection. In: Proceedings of London communications symposium; 2002.

Singh M, Valtorta M. A new algorithm for the construction of Bayesian network structures from data. In: Proceedings of the ninth annual conference on uncertainty in artificial intelligence (UAI-93). Washington, DC; July 1993. p. 259–64.

Singh M, Valtorta M. Construction of Bayesian belief networks from data: a brief survey and an efficient algorithm. International Journal of Approximate Reasoning February 1995;12(2):111–31.

Snapp S, Brentano J. DIDS (Distributed Intrusion Detection System) — motivation, architecture, and an early prototype. In: Proceedings of the 1991 national computer security conference; 1991.

Spafford EH, Zamboni D. Intrusion detection using autonomous agents. Computer Networks 2000;34(4):547–70.

Spiegelhalter D, Dawid A, Lauritzen SL, Cowell R. Bayesian analysis in expert systems. Statistical Science 1993;8(3): 219–83.

Utete SW. Local information processing for decision making in decentralized sensing networks. In: Proceedings of the 11th international conference on industrial and engineering applications of artificial intelligence and expert systems (IEA/AIE-98). Benicassim, Castellon, Spain; 1998. p. 667–76.

Valdes A, Skinner K. Adaptive, model-based monitoring for cyber attack detection. In: Proceedings of recent advance in intrusion detection. LNCS, No. 1907; 2000. p. 80–92.

Valtorta M, Kim Y-G, Vomlel J. Soft evidential update for probabilistic multiagent systems. International Journal of Approximate Reasoning January 2002;29(1):71–106.

Vomlel J. Probabilistic reasoning with uncertain evidence. Neural Network World. International Journal on Neural and Mass-Parallel Computing and Information Systems 2004; 14(5):453–6.

Xiang Y. Probabilistic reasoning in multiagent systems: a graphical models approach. Cambridge: Cambridge University Press; 2002.

Vaibhav Gowadia is a doctoral student at the Department of Computer Science and Engineering and Research Assistant in Information Security Laboratory at University of South Carolina, Columbia, SC. His current research interests include information systems security, intrusion detection, security protocols, threshold cryptography, and Semantic Web security. He obtained his Master of Science degree in Computer Engineering at the University of South Carolina in 2003, and Bachelor of Technology degree in Instrumentation and Control Engineering at the National Institute of Technology, Jalandhar, India in 2000.

Csilla Farkas is an Assistant Professor at the Department of Computer Science and Engineering, and Director of the Information Security Laboratory at the University of South

Carolina. She received a B.S. degree in Geological Sciences form the Eotvos Lorand University (1985), Hungary, a B.S. in Computer Science from SZAMALK (1989), Hungary, and a B.S. in Computer Science (1993) and Ph.D. in Information Technology (2000) from George Mason University, VA. Dr. Farkas' research interests include information security, data inference problem, economic and legal analysis of cyber crime, and security and privacy on the Semantic Web. She is a recipient of the National Science Foundation Career award. The topic of her award is ''Semantic Web: Interoperation vs. Security — A New Paradigm of Confidentiality Threats.'' Dr. Farkas actively participates in international scientific communities as program committee member and reviewer.

Marco Valtorta is an Associate Professor at the Department of Computer Science and Engineering at the University of South Carolina. He obtained a Laurea in Electrical Engineering from the Politecnico di Milano in 1980, an M.A. in Computer Science from Duke University in 1984, and a Ph.D. in Computer Science from Duke University in 1987. Between 1985 and 1988, he was a project officer for ESPRIT at the Commission of the European Communities in Brussels, where he supervised projects in the Advanced Information Processing area. As a faculty member at the University of South Carolina since 1988, he has conducted research funded by ARDA, SPAWAR, DARPA, the Office of Naval Research (ONR), the U.S. Department of Agriculture (DOA), CISE (an Italian laboratory controlled by ENEL, the state electricity company), and the South Carolina Law Enforcement Division. He is the author of over 30 refereed publications, an associate editor of the International Journal of Approximate Reasoning, and a member of the editorial boards of Applied Intelligence and of the International Journal of Applied Management and Technology. His research interests are in the areas of normative reasoning under uncertainty (especially Bayesian Networks, influence diagrams, and their use in stand-alone and multiagent systems), heuristics for problem solving, and computational complexity in artificial intelligence.