A Distributed and Privacy-Preserving Method for Network Intrusion Detection

Fatiha Benali[1], Nadia Bennani[2], Gabriele Gianini[3], and Stelvio Cimato[3]

[1] CITI, INSA-Lyon, F-69621, France
[2] Université de Lyon, CNRS, INSA-Lyon, LIRIS, UMR5205, F-69621, France
[3] Universitá degli Studi di Milano, Milano, Italy
fatiha.benali@insa-lyon.fr, nadia.bennani@liris.cnrs.fr,
{gabriele.gianini,stelvio.cimato}@unimi.it

Abstract. Organizations security becomes increasingly more difficult to obtain due to the fact that information technology and networking resources are dispersed across organizations. Network intrusion attacks are more and more difficult to detect even if the most sophisticated security tools are used. To address this problem, researchers and vendors have proposed alert correlation, an analysis process that takes the events produced by the monitoring components and produces compact reports on the security status of the organization under monitoring. Centralized solutions imply to gather from distributed resources by a third party the global state of the network in order to evaluate risks of attacks but neglect the *honest but curious* behaviors. In this paper, we focus on this issue and propose a set of solutions able to give a coarse or a fine grain global state depending on the system needs and on the privacy level requested by the involved organizations.

1 Introduction

Today, information technology and networking resources are dispersed across an organization. Threats are similarly distributed across many organization resources. Therefore, the Security of information systems (IS) is becoming a crucial part of business processes. Companies must deal with open systems on the one hand and ensure a high protection on the other hand. As a common task, an administrator starts with the identification of threats related to business assets, and applies a security product on each asset to protect an IS. Then, administrators tend to combine and multiply security products and protection techniques such as firewalls, anti-virus, Virtual Private Network (VPN), Intrusion Detection System (IDS) and security audits. Usually attacks against open and or distributed systems are difficult to detect as attackers acts independently on different resources to accomplish a full attack scenario. On a previous work, Saraydaryan and al [29] propose an efficient solution to detect Abnormal Users Behavior. In this solution, users' behaviors are modeled through a Bayesian network. The

R. Meersman et al. (Eds.): OTM 2010, Part II, LNCS 6427, pp. 861–875, 2010.

Bayesian Network modeling allows a great detection effectiveness by injecting incoming events inside the centralized model and computing all the associated conditional probabilities.

Collaboration between organizations to detect security threats allow organizations to enforce the organization's security policies, to stop attacks in progression and to react at the right time in order to prevent information, financial or reputation losses. However, when several organizations decide to collaborate in order to detect intrusive activities, every organization resource manager is requested to send the events log to a central unit that analyses them and calculates the Bayesian graph. In this case, the central unit is supposed to act as a trusted entity. Indeed, when the analyzer receives the event description from the participant, a lot of private information about resources, IP addresses, is communicated. Moreover, it could be embarrassing for a participant to be pointed out by the third party as a particular weak participant. In this paper, different solutions to preserve local privacy while detecting an overall intrusion attempt are proposed. The solutions assume still the existence of the central node starting the distributed probability calculations, which are done locally, and whose result is obtained by executing a privacy preserving protocol. At the end, the central node will be able to recover the global state of the network, and take the appropriate countermeasures, but the information on the state of each single node will be obfuscated according to the starting assumptions and the requested privacy level.

The paper is outlined as follows: section 2 reminds the centralized solution. Section 3 covers some related work and preliminary information on the cryptographic solutions adopted. Section 4 describes the solutions offering a distributed intrusion detection protocol with different privacy-preserving techniques. Section 5 concludes and presents future work.

2 The Centralized Intrusion Detection System

As more security products (such as intrusion detection systems, antivirus, firewall,etc.), security mechanisms (such as access control, authentication, etc.), and critical assets (such as web server, mail server, etc.) are deployed in an organization, security administrators are faced with the hard task of analyzing an increasing number of events trigged by these monitoring components.

To address this problem, researchers and vendors have proposed alert correlation or more generally events correlation, an analysis process that takes the events produced by the monitoring components and produces compact reports on the security status of the organization under monitoring. Several alert correlation techniques have been proposed to facilitate the analysis of these events. A probabilistic method [16,20,32,33] was used to correlate alerts based on feature similarity between some selected events attributes such as source IP address, login name and/or port number. Events with higher degree of overall feature similarity will be linked. Several work propose event correlation based on prerequisites and consequences of known attacks [17,28]. The prerequisite of an

attack is the necessary condition for the attack to be successful, the consequence of an attack is the possible outcome of the attack. The correlation method uses logical formulas, which are logical combinations of predicates, to represent the prerequisites and the consequences of attacks. Ning et al. [27] integrate these two types of correlation in order to improve the performance of intrusion alert correlation and reduce the impact of missed attacks. Another correlation technique proposed to correlate events based on the known attacks scenarios [26,**?**]. An attack scenario is specified by an attack language or learned from training datasets using data mining approaches. New approaches have emerged recently to learn event correlation models by applying machine learning techniques to training data sets with known intrusion scenarios [19,30]. This approach can automatically build models for alert correlation, but a training in every deployment is required to learn the new behavior of the system.

Event correlation methods need that the information security are normalized. There is a high number of alerts classification proposed for use in intrusion detection research. Three approaches were used to describe security information: list of terms, taxonomies and ontologies. The easiest classification proposes a list of single terms [15] covering various aspects of attacks. A variant of such an approach is listing of categories that regroup many terms under a common definition [14,31]. To avoid ambiguity between categories, a lot of taxonomies were developed to describe attacks. Lindqvist and Jonson [25] have proposed the intrusion results and the intrusion techniques as dimension for classification. John Howard [23] This process-driven taxonomy consists in five dimensions: attackers, tools, access, results and objectives. Howard extends his work by refining some of the dimensions [22]. Undercoffer and al [24] describe attacks providing an ontology as a new effort for describing attacks in intrusion detection field. Initially, they developed a taxonomy defined by the target, means, consequences of an attack and the attacker. The taxonomy was extended to an ontology, by defining the various classes, their attributes and their relations. A recent work [13] uses an ontology which describes in a uniform way all the actions that can be performed by users of an IS based on the action theory [21]. The ontology is based on four concepts: the intention, the movement, the target and the gain. The intention is the objective for which the user carries out his action, the movement is the means used to carry out the objective of the user, the target is the resource in the IS to which the action is directed to, the gains is the effect produced by the action on the system. This modeling is validated by a correlation of events trigged in a real organization [29].

2.1 Architecture

Cooperation between multiple monitoring product in an organization in term of security is achieved by means of cooperation modules in an architecture. A centralized architecture for the analysis of distributive knowledge in the IS relies on one central point for the collection and the analysis of the collected information. The architecture is composed of three modules as presented on figure 1: a collect, an analysis, and a response module. First an agent is installed on each asset. An

Asset can be a security product (IDS, Firewall, etc.), a management product (router, switch, etc.) or a sensitive service (web server, mail server, etc.). The security administrator assigns names to the sensitive assets. The collect module uses installed agents on each asset. The agent collects information from the target system on the activities performed on each asset and stores them as raw data in the asset log file. The agent maps the collected event in the same representation in order to give the same syntax to all the events, and the same security information semantics modeling. Once the data are unified, they are saved in the same database. The analysis module implements the processes to perform the collected such as the correlation methods that construct scenarios and classify them as normal or intrusive, reports generation of activities which were undertaken in the system and risk-value quantization of behaviors that exceeds a certain threshold. The response module takes care of the alarms or labeled events from the analysis module and decides how to respond to them.

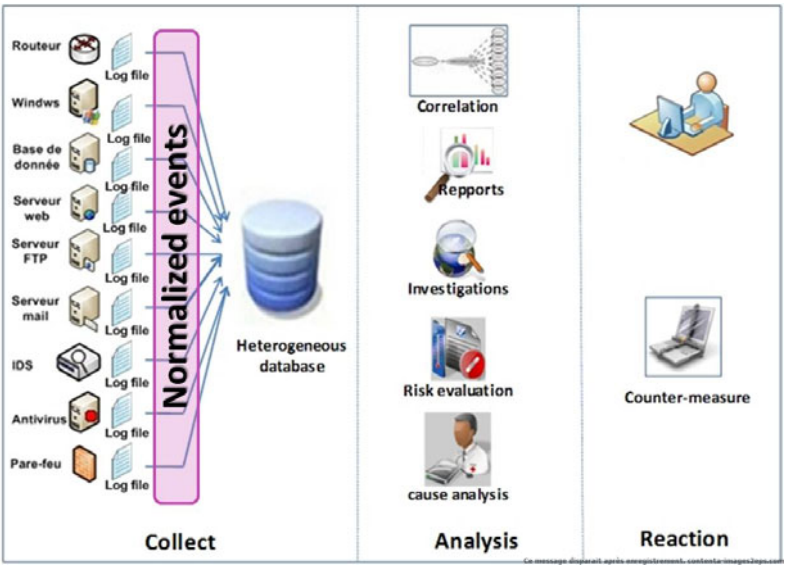

Fig. 1. Centralized analysis for distributive security information

2.2 Attack Scenario in Centralized Analysis

This section present an example of events correlation based on normalized events in centralized architecture. We dispose of a corpus of events generated by a company activities during 11 days (To preserve confidentiality, the details on the company are not published). These events are heterogeneous. The corpus consists of 6351565 raw events generated by the deployed products in the company.

IETF have proposed the Intrusion Detection Message Exchange Format (IDMEF) [18] as a way to set a standard representation for the syntax of intrusion alerts. IDMEF became a standard format with the RFC 47652[1]. We use the IDMEF as data model for security information. This format dispose of a class named *Classification text* to put the name of the detected event. We redefine the classification text class by a category of the ontology developped in [13] in order to describe the security information semantics. In this way, all events are normalized: same semantics using the ontology, and same syntax using the IDMEF data model. Then, we apply the behavioral correlation engine of autors [30] on normalized data. This behavioral correlation aims to learn a normal profile of the IS and to detect all deviances from this profile. This correlation method allows the automatic discovery of relations between the collected events on the IS. The proposed approach by the author is a supervised approach. The relations between the events are discovered from prior knowledge, and are presented by a graph of events which defines normal behaviors scenarios. To complete this model, the author transformed the events graph into a Bayesian network [30] in order to represent all the conditional probabilities of the graph events. This model allows comparison of future events to the normal behavior models and detect abnormal behaviors like user errors or intrusive activities.

We present here an attack scenario, a variant of website defacement (is usually the substitution of the original home page by a system attacker). We used the ontology developed in [13] to normalize the raw events.The different steps of this attack are represented by the following categories of events:

- E1-System_Attack.InformationGathering.Web. Detected (Information gathering on the website)
- E2- Rights_Vulnerability.Gain.web.Detected (vulnerability exploitation)
- E3-Authentication_Activity.Session.Admin.Success (acces to the website as the admin)
- E4-Rights_Activity.WEB_Command.Executed (have more rights on the web)
- E5- Rights_Activity.Modify.Web_Homepage.Success (modification oh the home page)

The engine has generated graphs of events performed on the IS use. The figure 2 illustrates a graph of events among the graphs which were detected by the behavioral correlation.

When multiple organizations decide to collaborate for intrusion analysis, they do not reveal some sensitive information contained in some attributes of events like attacks name or IP destination address because of privacy concerns. Alert correlation methods will be affected due to the lack of precise data provided by other organizations. In distributed secure intrusion detection, we can use the same approaches already presented in literature for centralized intrusion detection to normalize events and the same correlation methods but it is necessary to have techniques to perform privacy preserving alert correlation such that the privacy of participating organizations is preserved, and at the same time, alert correlation can provide correct results.

[1] `http://www.rfc-editor.org/rfc/rfc4765.txt`

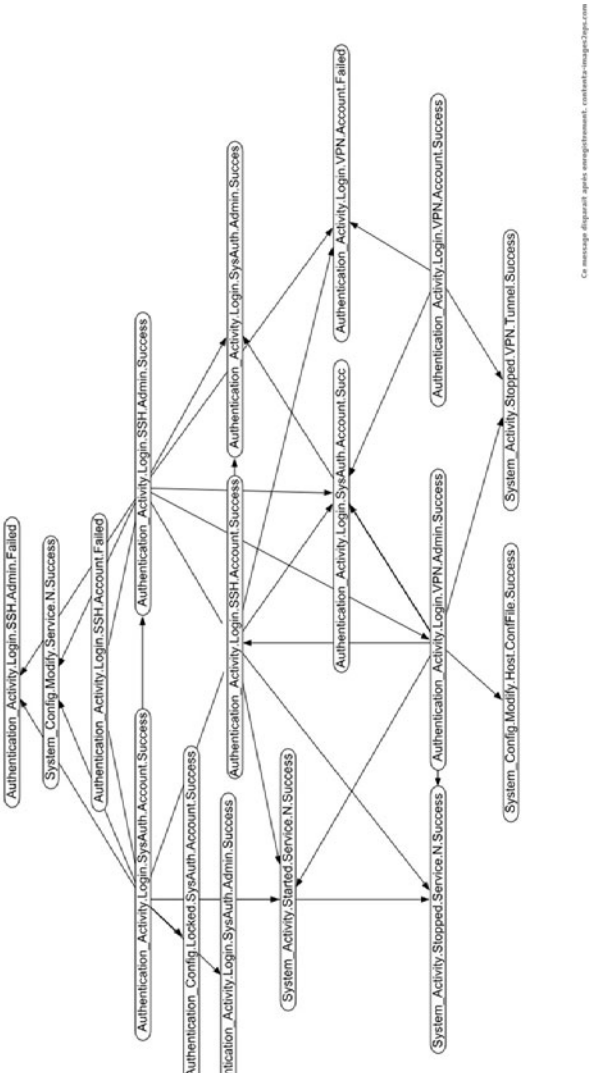

Fig. 2. A graph of events on the IS use

3 Background

3.1 Secure Multi Participants Computation(SMC)

The paradigm of secure two party computation was introduced by Yao [12], and successively generalized to include multi participants [5]. The resulting protocol assumes the representation of the function F to be computed via a boolean circuit and enable the parties to engage in a protocol to securely compute the output of each gate till the completion of the result. While the theoretical result is strong (any function can be computed in a secure way), the resulting protocol is not very practical, since its complexity depends on both the number of the inputs and the size of the combinatorial circuit representing the function F. Different applications of SMC have been presented in the literature, however in many cases, the main results in this area remain of theoretical interests, since even simple computations involving few parties still require several rounds of interaction. Much of the research work in this area is devoted to the design of efficient SMC protocols for the computation of particular functions (such as set operations, or scalar product, and so on) in different application scenarios.

3.2 Privacy Preserving Computation of Bayesian Network

Privacy preserving computation of Bayesian Network (BN) has been previously addressed in literature. The focus of previous work has been the computation of the parameters of the BN when the data are partitioned in different databases. In [9,4] the problem of distributed Bayesian network learning has been addressed for both the cases of horizontally and vertically partitioned data. The solution however, do not really preserves the privacy of the data, since the parts are required to share some information with the other parties. A more complete solution for learning both the structure of the BN and the parameters on vertically partitioned data has been presented in [10,11] which improves on another solution presented in [8]. The computation of the structure of the BN is obtained by modifying the learning algorithm widely used to extract the BN form a set of data (the K2 algorithm) in a privacy preserving way. At the same time the BN parameters are computed using a secure two party computation protocol between the owners of the database.

3.3 Preliminaries

We assume that the nodes are *honest but curious* parties, meaning that they will follow the steps of the protocol in which they are engaged, but could be motivated to access information they observe on the communication channel. We also assume that an ordering is imposed among the nodes and each node is able to communicate with the successive node in the list of all the participants to the protocol. The notion of an one-way accumulator has been introduced by Benaloh and de Mare in [1]. Basically a *one way accumulator* is a cryptographic primitive giving the capability of accumulating a set of values into a unique output value.

It has been defined as a family of one way hash functions $h : X.Y \rightarrow X$ with the additional property of quasi-commutativity, meaning that if the initial value x is selected, for any values y_1, y_2, it holds $h(h(x, y_1), y_2) = h(h(x, y_2), y_1)$. The basic functionality of an accumulator is that of constructing a compact representation of a set of values, providing at the same time method for constructing a proof that an element has been accumulated into the set. In [1], a time stamping and a membership testing application have been proposed exploiting such a primitive. Successively, dynamic accumulators have been proposed, providing the capability of adding and deleting elements from the set [2]. Accumulators have also been used to propose authenticated dictionaries, where untrusted directories mirror source contents, providing verifiable answers to membership queries [7]. An example of accumulator function is the RSA accumulator defined as

$$f(y, x) = y^x mod\ N$$

Such function is quasi commutative and is also one-way if some conditions on the domain and the choice of the modulus are met.

4 The Distributed Intrusion Detection System

The privacy properties of secure multi party computation require that the participants learn no more about other parties' input at the end of the protocol than it would if they get the computed result. Security in this context is often defined with respect to an ideal model [6], where a Trusted Third Party (TTP) is requested to do all the computation needed to return the requested result.

In this work, we are proposing a privacy preserving protocol for the computation of the global state of the network. The scenario we are considering is that of a network composed of n nodes, $P_1, \ldots, P_n$, collaborating in intrusion detection system. A querying node Q, is evaluating the probability of an attack state for the network, by analyzing the Bayesian model of the network. The attack probability is computed , Q assigns different probability values to the possible configurations of the network. Then the knowledge of the actual state of the network, is needed to compute correctly the attack probability. In a centralized scenario where a TTP is enabled to collect all the information about the collaborating nodes, the computation is an easy task, since the TTP could return the state of each node of the network and the querying node could correctly compute all the associated probability value in the Bayesian graph.

As regards the state representation, we assume that the state can be associated to a k-bit binary string. In the simplest scenario $k = 1$, i.e. the state of each node is represented by 1 bit, where 0 means *normal* status, and 1 means *attack*.

Referring to the sample network configuration and the attack scenario given in section 2.2, each node has a 3-bits state (0 is equivalent to a normal state, 1, to E1 state, ...,5 to the E5 state and lastly state 6 and state 7 are not assigned configurations).

4.1 Privacy Requirements

A distributed computation that does not rely on a TTP is secure if the involved parties do not learn anything more that they would during the execution of the protocol in the ideal model. In particular, the privacy requirements are the following:

- The querying node should be able to retrieve the associated probability value with the returned state from the protocol, without knowing the local state of each node;
- During the protocol, each node should not be able to compute the state of other nodes involved in the computation;

To better clarify the scenario we are considering, here we briefly discuss the notion of state of the network, and the level of requested privacy w.r.t. the previous requirements.

The privacy requirements above listed seem to contrast somehow with the knowledge the querying node needs to collect. Indeed the intrusion detection system, should be able to recognize the current status of the network in the most detailed way, in order to take the most appropriate countermeasures. For example, the recognition of a corrupted node inside the network could lead the network manager to isolate the node and move critical activities away from the node. Such kind of actions, can only be taken when the network manager knows exactly the topology of the network and the current condition of each node. In this work, we are concerned with privacy preserving techniques for the recognition of the network status, where the nodes are collaborating in the intrusion detection mechanism. According to the amount of information that the nodes want to release to the querying node, the kind of analysis that can be performed on the network and the corresponding countermeasures change. We take into account three basic scenarios, distinguishing on the basis of the information that the querying node can retrieve after the execution of the protocol and the corresponding privacy level requested:

- *centralized knowledge*: The state of the network depends on the condition of each individual node;
- *topology dependent knowledge*: The state of the network depends on the number and the position of the nodes;
- *summary knowledge*: The state of the network depends on the number of nodes in a given condition;

We do not deal here with the first scenario, where nodes collaborate with the intrusion detection system releasing all their private information. In the second case, the computation of the global state of the network takes into account the positions of the nodes within the network graph, but a certain degree of obfuscation is guaranteed, since the querying node will know the state of the network up to a permutation of the nodes. To this purpose we assume that all the information collected by the querying node is not directly linkable to a node.

In the last scenario, the computation of the global state returns the number of nodes in a given condition. The IDS system will trigger some reactions based on the occurrence of some threshold condition on the nodes, independently by their positions in the network graph.

4.2 The Privacy Preserving Protocol

The aim of a distributed privacy preserving protocol for the computation of the global state of the network is to let the querying node correctly compute the attack probability values in the BN. In this section, we discuss different solutions all returning a state string representing the global status of the distributed network under different assumptions. All the proposals preserve in some degree the privacy of the involved nodes, and are applicable in one of the above presented scenarios.

4.3 Counting the Corrupted Nodes

In this setting, we assume that the state of each node is represented by a single bit and the computation aims to return the number of nodes in attack conditions. The solution is offered by the execution of a secure sum protocol, assuming that more than three nodes are collaborating and that they do not collude [3]. The querying node computes a random number r in the range $[1, \ldots, n]$, and sends it to P_1. Each node will simply add its state value ($s_i = 0$ or $s_i = 1$) and forward it to the next node in the network, till the result $r + \sum_{j=1}^{n} s_i$ is sent from the last node P_n to Q. Q can now retrieve the number of corrupted nodes by subtracting r from the received summation. Notice also that each node can't learn anything on the state of the previous nodes, since she does not know the random value r and cannot know if the previous node changed the received value. A collusion however between nodes P_{i-1} and P_{i+1} could determine the state of node P_i. To this purpose the basic protocol can be extended to work if a majority of honest nodes exists in the network.

4.4 Collecting the State String

In this scenario we assume that the state of each node is represented by k bits and the returned state string is obfuscated (second scenario above discussed). The distributed protocol simply consist in retrieving the state of each node in the network, by accumulating the encryption of the k-bit strings. To this purpose, we assume that nodes know the public key of the querying node and use a semantically secure asymmetric encryption system. Each node simply forwards to the successive node the encrypted state string, adding its own local status to the string previously received. The querying node, will retrieve the global status of the network, by decrypting with the private key the state string. Also in this case, each node cannot retrieve any information on the previous nodes, and the querying node can only reason on the returned obfuscated information about the

nodes in the network. Let us illustrate this method by mean of the example in section 2.2: Node P_1 calculates S_1 as $E_Q(e_i)$ and sends it to P_2 having state e_j. Node P_2 calculates the state string $S_{1,2} = S_1||S_2$, where $S_2 = E_Q(e_j)$ and $||$ is the concatenation operation. The protocol goes on till P_n returns the computed global state string to Q.

4.5 Traversing the State Tree

We present here a very simple protocol, which enables the querying node to retrieve the state string, by intelligently traversing the state tree, still preserving the privacy of each node. For the sake of clarity, we assume that the state of each node is represented by 1 bit (but the reasoning can be easily extended to k bits). The global state of the network is retrieved by traversing the binary state tree. Starting from the root associated to the first node P_1, it is possible to descend on the left or on the right subtree, according to its state. Each nodes behaves in the same way, till one of the 2^n leaves is reached. Then the state of the system is described by the associated n-bits long string.

The traversing of the tree is simulated by privacy aware distributed protocol by asking each node to propagate the state string, starting from the $n \cdot 2^n$ bit long sequence, composed by the sequence of all the leaves of the state tree. To preserve the privacy of each node and of its local choice, we assume that the string is encrypted with a private key, known only to Q. Each node will split the string and pass to the successive the node the right or the left half of the string, according to its own state. At the end, the last node will return to the querying node the leaf state string, n-bits long, corresponding to the actual state of the system. Figure 3 illustrates the traversing tree state protocol for the configuration of section 2.2.

Notice that, the privacy of each node is maintained, since the local choice does not disclose the state to the successive node, which cannot read the state string and know the state of the previous nodes. The propagation of the string, however could cause some problem, since for a large number of nodes, i.e. if n is great, the communication costs are high.

In a variant of the protocol the network could circulate not the string with the state of the network, but rather a string with the probabilities of attack corresponding to each state. The querying node Q can pre-compute locally all probabilities of attack corresponding to all the possible states of the network, then each probability value can be approximated by a representation using only r bits. At this point, the querying node transmits the probability string consisting in $r \cdot 2^n$ bits, and each node will drop the left or the right half of the string depending on her state being 0 or 1. In the end, the querying node will receive a string of r bits representing a single probability value, the one corresponding to the state of the network. This protocol has the advantage of keeping the state of the network private also to the querying node, furthermore the number of circulating bits can be reduced with respect to the previous protocol if $r < nk$.

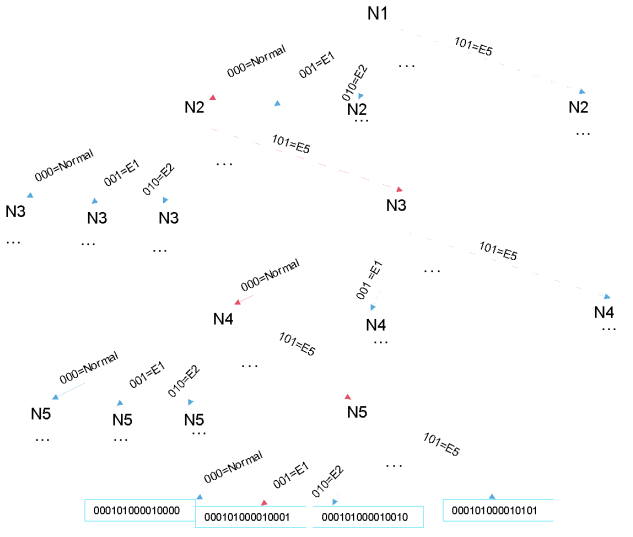

Fig. 3. Illustration of the traversingtree state method on the example of section 2.2

4.6 Accumulator-Based Protocol

Finally we present two privacy preserving protocols based on the use of a cryptographic accumulator A. In the first version, each node is simply requested to add to the received value, the value corresponding to its own state, and forward the obtained value to the successive node. In particular, let Q select random values q, r, and pass to P_1 the value $A_0 = A(q, r)$. Each node P_i is requested to compute the value $A_i = A(s_i, A_{i-1})$, and forward it to the successive node, where s_i is the k-bit string representing its state. At the end of the protocol, Q will obtain the value A_n returning the global status of the network. Due to the commutative property of the accumulator, Q will be able to determine the number of nodes in a given state, independently of the topology of the network, and match the returned value with the one obtained during the construction of the BN.

In a second version, it is possible to add topological information, by adding the encryption of the state string. Here we assume the existence of a semantically secure symmetric encryption scheme E, and the possibility of using the accumulated value received during the execution of the protocol as key. In particular during the execution of the protocol, each node P_i

- Uses the value A_{i-1} to encrypt its state, obtaining $S_i = E_{A_{i-1}}(s_i)$;
- Accumulates its own state computing $A_i = A(s_i, A_{i-1})$;
- Forwards A_i to the successive node and concatenates S_i to the state string.

At the end of the protocol, Q gets A_n, representing the global status of the network, and the global state string $S = S_1, \ldots, S_n$. Notice that in this case

no secret information or key is assumed to be known by the participants to the protocol. Each node is able to add her status to the global status, but cannot compute the state of the previous node.

5 Discussion

The solutions presented above are efficient methods for preserving nodes privacy, while ensuring that the network manager is able to collect information enough on the state of the network and react to abnormal situations. Some solutions rely on the exploitation of secure computing protocols while others use cryptographic primitives for achieving the requested privacy level.

If the IDS needs only to know the number of nodes under a given attack condition, both solutions illustrated in sections 4.3 and 4.6 can be used. Notice that the second solution improves on the first one, since it is possible to count nodes laying in different kinds of attack conditions. Due to the commutativity property of the accumulator, each node is able to add its local state to the global count, letting Q able to distinguish how many nodes stay in a given condition.

If on the other side, the IDS needs to retrieve more detailed knowledge on the network state, Q can adopt one of the solutions presented in 4.4, 4.5, or 4.6. The solution presented in 4.4 relies on the basic assumption that some shared knowledge is owned by all the nodes in the network; in particular all the nodes should know the updated public key of the querying node. The other two solutions release also this assumption. For the solution presented in sec. 4.5, however, it is requested that Q pre-computes all the possible network states, and assign a probability value to each of them. The solution presented in 4.6, finally, returns a very compact representation of the network state, computing the accumulated value that can be matched with some of values that Q can compute and associate to potential dangerous situations. Such values will trigger the reaction from the network manager, who can adopt the requested countermeasures.

6 Conclusions and Perspectives

The recognition of attack scenarios in open and or distributed systems is a complex and difficult task since attackers acts independently on different resources to accomplish a full attack scenario. Collaboration between organizations to detect security threats often relies on a central entity that analyses logged data and computes a Bayesian graph, where probabilities are assigned to the possible different configurations of the system under monitoring. In this work we discussed some efficient solutions for computing the state of network in which nodes collaborate with an intrusion detection system, but would like to avoid disclosing all private information to a trusted entity, and would like to retain a given privacy level. In future, we would like to evaluate the performance and the effectiveness of the proposed protocols in practical situations involving a given number of nodes. Furthermore, in the proposed solutions we have to consider situations where several abnormal behaviors could be detected simultaneously on

the same node. Finally, in the proposed solutions, a common assumption is that all the nodes adopt the same behavior. An interesting investigation challenge is to consider different types of participants with different kinds of requested privacy level. In this case, the computation of the network state should be adapted in order to collect the maximum amount of information that nodes are willing to disclose.

References

1. Benaloh, J., de Mare, M.: One-way accumulators: A decentralized alternative to digital signatures. In: Helleseth, T. (ed.) EUROCRYPT 1993. LNCS, vol. 765, pp. 274–285. Springer, Heidelberg (1994)
2. Camenisch, J., Lysyanskaya, A.: Dynamic accumulators and application to efficient revocation of anonymous credentials. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, p. 61. Springer, Heidelberg (2002)
3. Clifton, C., Kantarcioglu, M., Vaidya, J., Lin, X., Zhu, M.Y.: Tools for privacy preserving distributed data mining. SIGKDD Explor. Newsl. 4(2), 28–34 (2002)
4. Chen, R., Sivakumar, K., Kargupta, H.: Learning Bayesian Network Structure from Distributed Data. In: Proc. SIAM Int'l Data Mining Conf., pp. 284–288 (2003)
5. Goldreich, O., Micali, S., Wigderson, A.: How to Play ANY Mental Game. In: Proc. 19th Ann. ACM Conf. Theory of Computing, pp. 218–229 (1987)
6. Goldreich, O.: Foundations of Cryptography, vol. II: Basic Applications. Cambridge Univ. Press, Cambridge (2004)
7. Goodrich, M.T., Tamassia, R., Hasic, J.: An efficient dynamic and distributed cryptographic accumulator. In: Chan, A.H., Gligor, V.D. (eds.) ISC 2002. LNCS, vol. 2433, pp. 372–388. Springer, Heidelberg (2002)
8. Meng, D., Sivakumar, K., Kargupta, H.: Privacy-Sensitive Bayesian Network Parameter Learning. In: Proc. Fourth IEEE Int'l Conf. Data Mining, pp. 487–490 (2004)
9. Yamanishi, K.: Distributed cooperative Bayesian Learning strategies. Information and Computation 150(1), 22–56 (1999)
10. Wright, R.N., Yang, Z.: Privacy-Preserving Bayesian Network Structure Computation on Distributed Heterogeneous Data. In: Proc. 10th ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining, pp. 713–718 (2004)
11. Yang, Z., Wright, R.N.: Privacy-Preserving Computation of Bayesian Networks on Vertically Partitioned Data. IEEE Transactions on Knowledge and Data Engineering, 1253–1264 (September 2006)
12. Yao, A.: How to Generate and Exchange Secrets. In: Proc. 27th IEEE Symp. Foundations of Computer Science, pp. 162–167 (1986)
13. Benali, F., Legrand, V., Ubéda, S.: An ontology for the management of heteregenous alerts of information system. In: The 2007 International Conference on Security and Management (SAM 2007), Las Vegas, USA, pp. 374–380 (June 2007)
14. Cheswick, W.R., Bellovin, S.M.: Firewalls and Internet Security Repelling the Wily Hacker. Addison-Wesley, Reading (1994)
15. Cohen, F.B.: Information system attacks: A preliminary classification scheme. Computers and Security 16(1), 29–46 (1997)
16. Cuppens, F.: Managing alerts in a multi-intrusion detection environment. In: ACSAC 2001: Proceedings of the 17th Annual Computer Security Applications Conference, Washington, DC, USA, p. 22. IEEE Computer Society, Los Alamitos (2001)

17. Cuppens, F., Miège, A.: Alert correlation in a cooperative intrusion detection framework. In: SP 2002: Proceedings of the 2002 IEEE Symposium on Security and Privacy, Washington, DC, USA, p. 202. IEEE Computer Society, Los Alamitos (2002)
18. Curry, D., Debar, H.: Intrusion detection message exchange format
19. Dain, O., Cunningham, R.K.: Fusing a heterogeneous alert stream into scenarios. In: Proceedings of the 2001 ACM Workshop on Data Mining for Security Applications, pp. 1–13 (2001)
20. Dain, O.M., Cunningham, R.K.: Building scenarios from a heterogeneous alert stream. In: IEEE Workshop on Information Assurance and Security, pp. 231–235 (June 2001)
21. Davidson: Actions, reasons, and causes. Journal of Philosophy 685–700 (1963) (Reprinted in Davidson 1980, pp. 3–19)
22. Howard, J., Longstaff, T.: A common language for computer security incidents. Sand98-8667, Sandia International Laboratories (1998)
23. Howard, J.D.: An Analysis of Security Incidents on the Internet -normalement phd dissertation. PhD thesis, Carnegie Mellon University, Pittsburgh, Pennsylvania 15213 USA (April 1997)
24. Johi, A., Pinkston, J., Undercoffer, J.: Modeling computer attacks: an ontology for intrusion detection. In: Vigna, G., Krügel, C., Jonsson, E. (eds.) RAID 2003. LNCS, vol. 2820, pp. 113–135. Springer, Heidelberg (2003)
25. Lindqvist, U., Jonsson, E.: How to systematically classify computer security intrusions. In: Proceeding of the IEEE Symposium on Security and Privacy, pp. 154–163 (1997)
26. Lindqvist, U., Porras, P.A.: Detecting computer and network misuse through the production-based expert system toolset(p-best). In: Proceeding of the 1999 Symposium of Security and Privacy, Oakland, CA, USA. IEEE Computer Society, Los Alamitos (May 1999)
27. Ning, P., Xu, D., Healey, C.G., Amant, R.S.: Building attack scenarios through integration of complementary alert correlation methods. In: Proceedings of The 11th Annual Network And Distributed System Security Symposium (NDSS 2004), pp. 97–111 (2004)
28. Peng, N., Yun, C., Reeves Douglas, S.: Constructing attack scenarios through correlation of intrusion alerts. In: Proceedings of the 9th ACM Conference on Computer and Communications Security, CCS 2002, pp. 245–254. ACM, New York (2002)
29. Saraydaryan, J., Benali, F., Ubéda, S., Legrand, V.: Comprehensive security framework for global threads analysis. International Journal of Computer Science Issues IJCSI 2, 18–32 (2009)
30. Saraydaryan, J., Legrand, V., Ubéda, S.: Behavioral anomaly detection using bayesian modelization based on a global vision of the system. In: NOTERE (2007)
31. Stallings, W.: Network and internetwork security: principles and practice. Prentice-Hall, Inc., Upper Saddle River (1995)
32. Staniford, S., Hoagland, J.A., McAlerney, J.M.: Practical automated detection of stealthy portscans. J. Comput. Secur. 10(1-2), 105–136 (2002)
33. Valdes, A., Skinner, K.: Probabilistic alert correlation. In: Proceedings of the 4th International Symposium on Recent Advances in Intrusion Detection, RAID 2000, London, UK, pp. 54–68. Springer, Heidelberg (2001)