An adaptive risk management and access control framework with obligation to mitigate insider threats

Annamma Monisha.I
PG Scholar
Department of computer science and engineering(PG)
Sri Ramakrishna engineering college
Coimbatore, Tamilnadu
India
monishaimmans@gmail.com

Dr.A.Grace Selvarani
Prof&Head
Department of computer science and engineering(PG)
Sri Ramakrishna engineering college
Coimbatore, Tamilnadu
India
hod-mecse@srec.ac.in

ABSTRACT:

An Insider Threat is a malicious threat to an organization it actually comes from people within the organization, such as employees, former employees, contractors or business associates, who have access to the confidential information of the organization. Preventing insider attacks is a strenuous task. Finding a an intermediate position, where the necessary privileges are provided and as well as malicious usage are avoided, is necessary. In this paper, we propose a framework that expands the role-based access control (RBAC) model by assimilating a risk assessment process, and the trust the system has on its users. Our framework revamps to unsure changes in users behavior by removing privileges when users trust falls below a certain threshold. This threshold is computed based on a risk assessment process that includes the risk due to inference of unauthorized information. We use a Colored-Petri net to detect inferences. We also redefine the subsisting role activation problem, and propose an algorithm that reduces the risk exposure. we propose the concatenation of trust-and-obligation methodology with the framework that reduces the risk exposure of an organization associated with a posteriori obligations. We use the trust values of the users to indicate how reliable they are with regards to fulfilling their obligations. When access requests that trigger a posteriori obligations are accessed, the requesting users trust values and the criticality of the associated obligations is being used. The framework discovers and mitigates insider attacks and unintentional damages that may result from infringing a posteriori obligations.

INTRODUCTION:

According to the Computer Crime and Security Survey, insider attacks accounted for 33% of the total incidents reported in 2010 (C. S. Institute, 2010). An insider attack is exceuted by people who are rightfully authorized in the system to perform certain tasks. The outcomes of insider attacks may be catastrophic, and may include economic losses, negative impact on the stature, loss of customers. According to Moore et al. (2008), the economic losses due to insider attacks wandered from five hundred dollars to tens of million dollars, around 75% of the organizations had a negative impact to their business operations, and 28% experienced a negative impact to their stature.

Regrettably, as the statistics show, insiders do perform attacks! In addition, even if the users could be trusted, malware can be by accidently installed and a user account be compromised. The trust the system has on a user should be restructured according to the user's behavior. When a user's behavior falls out of the probable pattern in a doubtful fashion, the trust the system has on him should be lowered. A trust threshold is typically used to limit access to resources depending on their importance to the organization. On the other hand, existing approaches do not present a wide-ranging analysis of the way in which trust thresholds should be assigned, nor do they include the separation of duty constraints or hybrid hierarchy. They also do not detail how to impose such policies or reduce the risk exposure automatically.

An obligation is an action that needs to be performed by a user before or after accessing a resource1. For illustration, systems that need to safeguard data privacy may need to impose an obligation to the users after they access a exacting resource. Supervising a posteriori obligations is a challenging task, as there is no promise that after granting access to a resource, the user will full the forced obligation

Hence, organizations would benefit from a framework intergrated with obligation that helps mitigate accidental and planned damages performed by insiders. Since organizations suppose that the risk of having users dodging on (i.e., ignoring or forgetting to perform) a posteriori obligations every time the system imposes them, it is essential to control such risk exposure. If these obligations are not fulfilled, it may cause in, penalties, delays, lawsuits, loss of proceeds and goodwill, among other negative consequences for the organization.

Depending on the criticality of an obligation, the risk exposure may differ from low to severe. Regrettably, current approaches [7, 14] allot a posteriori obligations to users without considering their affinity to fulfilled or evade on obligations.

We recommend an obligation-based risk management to be intergrated with the framework that is able to lessen the risk exposure of an organization caused by a posteriori obligations. The higher the trust value of a user, the more the system trusts him to fulfilled a posteriori obligations.

2. PREMILINARIES

In this section, we overview RBAC, risk, trust and Coloured Petri-nets.

2.1. RBAC, constraints and hybrid hierarchy

Our work is based on Role Based Access Control (RBAC) model(Ferraiolo et al., 2001), because of its remuneration. It encompasses optional and obligatory access control models and supports organization or user-specific necessities. Additionally, RBAC uses roles which are a natural idea for most organizations, and it provides organizations with profitable benefits due to the reduction on the administration cost (Osborn and Sandhu, 2000). In RBAC, permissions are assigned to roles, and roles are assigned to users. In order to get hold of the permissions authorized for a role, users need to turn on the role in a session. Sets U, R, and P represent the set of users, roles and permissions in the system. Separation of duty constraints (SoD) are used to avoid false activities within an organization by preventing a unique user from assuming two or more conflicting roles. There are two types of SoD constraints: Static (SSoD) and the Dynamic (DSoD). SSoD restricts the sanction of users to conflicting roles (Ahn and Sandhu, 2000). Each constraint is denoted as SSoD(RS,k)¸SSoD, where RS,R with 2 <k <n. This constraint states that a user can be authorized to at most k-1 roles in RS. Similarly, a DSoD constraint DSoD(RS,k)¸DSoD states that a user can activate at most k-1 roles in RS simultaneously.

2.2. Risk and trust

We agree to the following trust description: “Trust is a slanted expectation that an agent has about another’s future activities based on the history of their encounters” (Mui and Mohtashemi, 2002). Trust may be based on the context in which the interface between entities takes place. We denote the set of contexts as C. For example, the type of service and the network connection used by the user may describe a context. Risk is defined by the probability of a perilous situation and its consequences if it occurs. The probability of occurrence can be lowered through the accomplishment of controls and mechanisms in the system that aim to mitigate threats. The risk exposure after all the controls and mechanisms are in place is called residual risk, and preferably it is the risk that the organization is willing to accept. Risk can be calculated using the expected value formula (Celikel et al., 2009), where the probability of incident is multiplied by the cost of the event.

2.3 Posteriori obligation

The privileged the trust value of a user, the extra the system trusts him to fulfill a posteriori obligations. We use a threshold-based risk management method in which the criticality of an obligation agree on how much a user desires to be trusted in order to allocate the obligation to him. When a user ask for an access, the user's diplomas, his trust value and the criticality of the obligations related with the requested permissions are used to resolve whether he should be granted the requested access and assigned the connected obligations or whether the request should be denied. If the criticality of an obligation is high while the assigned user's trust value is low, granting the requested access would create a noteworthy risk to the organization, and for this reason it should be denied. We recommend and estimate a methodology to calculate the obligation-based trust values for each user. The methodology is durable against users who know how the system computes the trust values and try to exploit this acquaintance. Our methodology is also able to separate among users who by mistake do not fulfilled an obligation, maliciously avoid the accomplishment of obligations and those who deliberately oscillate their behavior to maintain their trust value within an acceptable threshold to start on an attack later.

3. OVERVIEW OF THE FRAMEWORK

The proposed system architecture is shown in Fig. 1. The Monitoring Module monitors the users in the system. The Trust and Context Module (TCM) uses the monitored information to identify the context, and calculate the trust value of each of the users accordingly. These trust values are stored in the Trust Repository.

The Access Control Module is collected of the Enforcement Module the Administration Module and the Policy Information Point (PIP). The policy of the system is piled up in the PIP. The Enforcement Module is in accuse of evaluating access requests and has several components, the Policy Enforcement Point (PEP), the Policy Decision Point (PDP), the Risk Module and the Inference Module. An right to use request consists of the set of permissions a user wants to acquire. The PEP interrupts all these requests, and ensures that the resources of the system can be entréed only if the policy authorizes it. The access requests are interrupted by the PEP, which sends them to the PDP. The

PDP estimates the policy according to the trust the system has on
the user, the context, and the inference risk. The inference risk is calculated by the Inference Module and the computations connected to trust thresholds are performed in the Risk Module. In case the trust value of a user diminishes while a user has a session open, the TCM sends a announcement to the PDP, which re-evaluates whether the privileges the user is exercising should be revoked. In this way, the system is able to reject access to misbehaving users before they can perform extensive damage to the system.

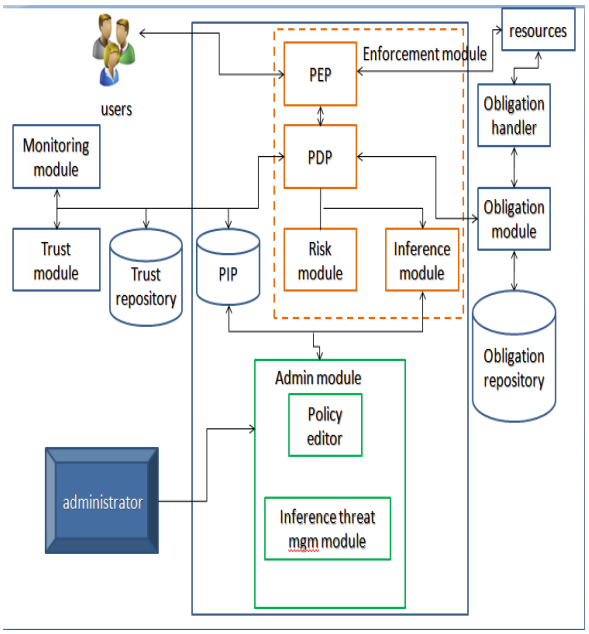


Fig 1 The proposed system architecture

The Administration Module allows the administrator to define, filter and analyse the policy. It is collected of two modules. The Policy Editor permits the specification of the policy and the Inference Threat Risk Management Module produces information that identify the lively inference threats of a scrupulous policy arrangement. Using this information, an administrator may iteratively modify the policy to reach the desired risk exposure. In this way the administrator can recognize the ideal policy, with respect to inference risk, before he understands the policy in the production system. We discuss in detail this procedure in Section 6.

The obligation module included in the framework finds the insiders who are likely to become an attacker.

4. RISK AND THRESHOLDS

In this section, we present the proposed methodology to calculate the risk exposure of an organization. We show how the risk is calculated for different roles that a user wants to activate based on the risk of the permissions they can acquire. Finally, we show how to compute the trust threshold.

4.1. Risk associated with permissions

A permission is a tuple (obj,act) where the obj is an benefit in the organization such as a file or other resource, and the act communicates to the action that a user can perform on the object. Objects are vulnerable to different threats. Among these are object's loss of veracity, loss of confidentiality, and loss of accessibility. Intuitively, different objects have dissimilar security requirements that depend on the business functions of a particular organization. For instance, some objects necessitate that their integrity be well guarded, other objects are sensitive (their leakage would result in a lot of damage to the business), while others may be critical and sensitive concurrently. Hence, the risk exposure of the organization depends on the action that is performed on the object and the significance of the object. The risk value of a permission p is the likelihood that p is misused multiplied by the corresponding dent cost. We are interested in the residual risk which means that the chance of a particular misuse depends on the mitigation mechanisms and controls that the organization has in place to diminish the vulnerabilities that can lead to the misuse.

$$rs(p,s) = \sum_{xp \in malicious\ usage\ c} \Pr[xp|c] * cost(xp)$$

4.2. Risk associated with role sets

Unsurprisingly, the risk associated with a set of roles is a function of the risk of the permissions that can be take up through those roles. When maneuvering such risk values, we comprise the risk of the permissions that can be clearly attained through those roles, as well as those that can be gathered from them. We show, how to compute the risk of activating a set of roles.

4.2.1. Calculating the role set risk

The risk revelation of given that access to a set of roles $R' \subseteq R$ to a user u depends on the state of the CP-net. The following formula presents the risk exposure.

Definition 4. The risk revelation of the system if user u set in motion a set of roles $R' \subseteq R$ in context c is given by

$$rs(R', c, u) = \sum_{p \in \Phi_{\wp}} rs(p, c)$$

Where $\wp$=Pau(R')U inferred(Hu,u,R').

4.3. Trust thresholds associated with role sets

The trust threshold hooked up with a set of roles symbolizes how trusted a user needs to be in order to utilize those roles. unsurprisingly, this threshold needs to emulate the risk exposure of the organization when the roles are turned on by a user. We identify the trust threshold as follows.

Definition 5. The trust threshold of the set of roles $R' \subseteq R$, in context c meant for user u is defined as follows:

$$\tau(R',c,u) = \frac{rs(R',c,u)}{\sum_{p \in p}(rs(p,c))}$$

Where $0 \leq \tau(R',c,u) \leq 1$. When $\tau(R',c,u) = 0$, it means that user u does not require to be trusted to trigger R' in context c; when $\tau(R',c,u) = 1$, it means that user u needs to be entirely trusted consecutively to activate R' in context c.

4.4. Trust of users

We allocate each user in the organization a trust level. The trust intended for a user u in context c is denoted by trust(u,c) and is defined in the interval [0,1], where 1 means the user is entirely trusted and 0 means the user is perfectly untrusted. The Trust and Contexts Module in Fig. 1 considers the activities of users over time and the context to calculate the trust value for each user; e.g., if the user is by means of an untrusted connection, the trust in the user may be reduced. Solutions such as Bertino et al. (2005) and Chung et al. (1999) can be used to construct profiles and latter calculate a trust value supported by the activities of a user.

5.TRUST AND RISK AWARE ROLE ACTIVATION

The role activation procedure is influential in our framework. It is in prosecute of identifying when a user should be shorn of to activate roles due to lack of trust or other policy limitations. It also allows us to lessen the risk exposure by selecting the roles that have a smaller amount risk in the system.

5.1. Role activation algorithm

We put forward Algorithm 1 to find a way out for the Trust-and-Risk Aware Role Activation dilemma. Our algorithm take for granted that the policy is well-formed. The algorithm first get rid of from the search space the roles that cannot be activated due to trust apprehensions (line 4).

This takes place when a candidate explanation provides all the permissions requested (line 16). If the candidate elucidation is less risky than the existing best solution, it turns out to be the new best solution. If both solutions have the same difficulty, the algorithm selects the one that has a smaller quantity of roles. Before the algorithm accomplishes the base case, it trims the search space by taking away the roles that cannot be activated due to DSoD and cardinality limitations (lines 25 and 28). Roles that do not bring in the missing permissions in the candidate solution are also eradicated (line 32). In case no candidate roles are absent after the pruning (line 34), the algorithm backtracks as that search path did not illustrate the way to a valid solution. Otherwise, the next role to be additional to the candidate set is chosen in line 36; this function only selects a role r if adding it to Rsel fulfills s(Rsel∪{r},c,u)≤trust(u,c) (line 44). The selected role is denoted as r_{best}.

Algorithm 3 Trust and Risk aware Role activation
Precondition: The policy is well formed
Postcondition: Rq contains the solution of the problem specified

1. **FirstTrustandRiskAwareActivationSet**(u,Ps,c)
2. R_{avail} ← authorized(u)
3. **For all** rϵ R_{avail} **do**
4. **If** $\tau(r,c,u) > trust(u,c)$ ***then***
5. $R_{avail} \leftarrow R_{avail} \backslash r$
6. R_{sel} ←∅ (selected roles so far)
7. P_{rem} ← PS (set of permissions that haven't been found)
8. R_q ←∅ (Global variable)
9. selectRoles ($P_{rem}, R_{avail}, R_{sel}, u, c$)
10. **if** $R_q \neq \emptyset$ **then**
11. **return** R_q
12. **else**
13. **return** ∅ (request denied)

14. **SelectRoles** ($P_{rem}, R_{avail}, R_{sel}, u, c$)
15. **If** $P_{rem} = \emptyset$ **then**
16. **If** $R_q = \emptyset$ **then**
17. $R_q \leftarrow R_{sel}$
18. **Else**
19. **If rs**(R_q,c,u) > rs(R_{sel}, c, u) **then**
20. $R_q \leftarrow R_{sel}$
21. **return**
22. **for all** dsod(RS,k) ϵ DSoD **do**
23. **if** | $R_{sel} \cap$ RS |= (k-1) **then**
24. $R_{avail} \leftarrow R_{avail} \backslash$ [RS\ ($R_{sel} \cap$ RS)]
25. **For all** card(r_c,k) ϵ CARD ∩ $r_{c,} \epsilon R_{avail}$ **do**

26. **If** activated($r_{c,}$) + 1 = k-1 **then**
27. $R_{avail} \leftarrow R_{avail} \backslash r$
28. **For all** $r_i \in R_{avail}$ **do**
29. **If** $P_{rem} \cap P_{au}(r) = \emptyset$
30. $R_{avail} \leftarrow R_{avail} \backslash r$
31. **If** $R_{avail} = \emptyset$ **then**
32. **Return**
33. **If** $r_{best} = \tau$ **then**
34. **Return**
35. **SelectRoles** ($P_{rem} \backslash P_{au}(r_{best}), R_{avail}, R_{sel}, u, c$)
36. **SelectRoles** ($P_{rem}, R_{avail}, R_{sel}, u, c$)

6. INFERENCE THREAT ANALYSIS AND ADMINISTRATION

In this section, we propose a CP-net based approach to find the information a user may infer after a exacting admittance, and a technique to supervise inference threats associated to a meticulous policy.

6.1. Finding inferred permissions

The set of inference tuples, I, that are pertinent to an organization were defined previously. We begin by characterize an algorithm to locate the inferred permissions.

Algorithm 2 Given an inference CP-net 'H, a user u ϵ U and a set of roles R' that he can trigger, return the set of inferred permissions P'.

1. **FindInferredPermissions** (H',u,R')
2. M ← H'.tokensAt (u,w_{end})
3. Wait for H' to distribute the tokens
4. N← H'. tokensAt (u,w_{end})
5. Q← tokens in N that are not contained in M
6. P'={P_x |<u, P_x >Q}
7. **return** P'

In Algorithm 2, we demonstrate the process to find the possible new permissions P0 a user u may infer after activating a set of roles R0 as per Definition 3. First, in line 2, the current inferred acquiescence of the user are saved in M. Thus M contains the set of tokens contingent by user u prior to activating R0. At that moment, we place one token hR0; ui at ws. After the transitions shoot and all tokens are in a place different than ws, we confirm the state of the CP-net. We symbolize this new state as H0. Afterward, in line 5,we store up in N all the tokens placed at wend. Since we merely need to identify the permissions that u will be able to gather if he activated R0, in line 6, Q is initialized to include only newly inferred information. To conclude, in line 7, we extract from Q the set P0 of newly inferred permissions.

6.2. Finding active inference threats

In this section, we recommend a methodology to improve the presentation of the Inference CP-net. We assume that I surrounds all the existing inference threats. Make a note of that the set of inference tuples I does not depend on the user-to-role or the permission-to-role assignments. Inference tuples are distinctively dependent on the types of objects that survive on the organization (this is the case for existing methodologies to automatically find inference tuples (Chen and Chu, 2008; Yip and Levitt, 1998)). The rationale is that if I contains all the existing supposition tuples, even if the user-to-role and the permission-to-role assignments transform, the framework can still incarcerate the risk disclosure due to inference threats. In contrast, if I only contains the tuples for a particular strategy configuration, for each possible policy modification, the administrator would need to authenticate and perhaps include or eradicate new tuples in the set I. This improvement consists on finding the active inference threats for a given strategy and uniquely including the pertinent inferences tuples in the deployed Inference CP-net.

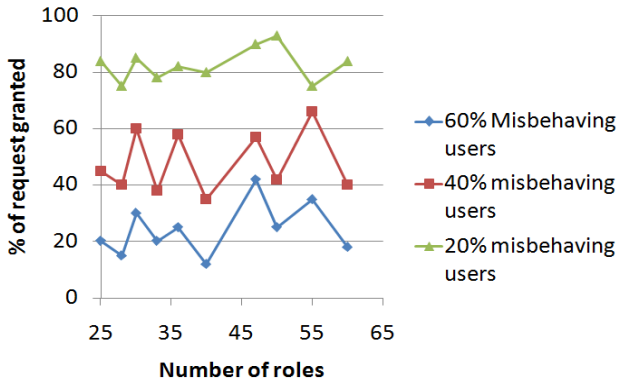
Algorithm 3 Find active inference threats given a strategy configuration PL=(R,U,P,I,C,DSoD,SSoD)

1. **FindActiveInferenceThreats(**PL)
2. 1stactive← [List with active inference, initially empty]
3. I_y ← ∅ [Initialize active inference tuples as an empty set]
4. H'_s = create CPNet (PL)
5. U_p = SameProfile(U)
6. **For all** uϵU_p **do**
7. R_u = authorized(u)
8. 1stTokens(r,u)
9. t = createTokens(r,u)
10. 1st$Tokens_u$.add(t)
11. **For all** Tϵ nextPermutation (1stTokens) **do**
12. H'_s .PlaceAt(w_s,T)
13. runSimulation()
14. if H'_s.tokensAt(u,w_{end}) ≠ ∅
15. 1stActive.add(t)
16. I = identifyInferencePath()
17. Add I to the set of active inference tuples
18. **Return** (1sAtctive,I_y)

The comprehensive process to spot active inference dangers is shown in Algorithm 3. Its input is the strategy that is going to be tested, PL, and its output is the inventory of users that are able to infer unauthorized information and the user-to-role assignments that allow the inference. This list is initialized in line 2. The set of dynamic inference tuples that we symbolize as Iy is initialized as an drain set in line 3. In line 5, we make a representative user for each

sketch. For each of the authorized roles a token is produced in line 10 and added to the list of tokens of that user. Subsequently, the simulation is run for quite a few arrangement of roles. In line 12 all the potential permutations of the way in which roles can be activated are establish. Later, Hs is inspected to see whether the organization allowed any suppositions. For this rationale, in line 15, we validate if each user was able to conjecture information, and if so, we store the inference in the variable lstActive. In conclusion, to know which is the inference tuple that was make active, in line 18 we use the method make out InferencePath() that identifies the path from beginning to end which a token arrived to end. Then, the tuple identified is supplementary to Iy**.**

7.RESULT AND DISCUSSION



Comparing percentage of giving way the requests for different proportions of unruly users: This experiment gives you an idea about how the system behaves as the percentage of users that misbehave amplifies, and their trust thresholds are reduced. Primarily, we randomly generated policies, and requests that were all established. Intended for the same policies and requests, we indiscriminately selected some users, and reduced randomly again their trust thresholds; then the framework shows how many requests were shorn of because of decreases in trust of some users. The results for 20%, 40% and 60% of users misbehaving are shown in Fig. 6. As the number of misbehaving users amplifies, the number of requests granted diminishes. The lines are not flat, as the number of requests shorn of depends on the trust threshold of the selected roles, in addition to the random reduction of the user's trust value. The results show that our framework is able to reject access to misbehaving users, thus adapting to avoid possible insider attacks.

8.CONCLUSION

In this paper, we have offered an approach to perform access control considering the behavior of the users, and the risk experience that an organization is ready to recognize when granting access to certain roles in the organization. The approach reacts to harmful behaviors of users by denying admittance to permissions whose mishandling would negatively contact an organization. In this fashion, the approach is able to dissuade possible attacks when there are technical precursors that point out a user is behaving maliciously. Sequentially to reduce the risk exposure further, we have also defined an optimization problem, and an algorithm that decreases the risk exposure. The combination of the obligation module benefits the organization. It is considered that the features offered by our framework make it difficult for the insider attackers to exploitation the privileges. As future work, It is planned to investigate in point of the integration of the work accessible here with the Monitoring Module, and Trust and Context Module.

REFERENCE

[1] Aagedal JO, Braber F d, Dimitrakos T, Gran BA, Raptis D, Stolen K, (2002) 'Model-based risk assessment to improve enterprise security', In: Proceedings of the 6th International enterprise distributed object computing conference, pp. 51.

[2] Ahn G-J, Sandhu R, (2000) 'Role-based authorization constraints Specification', ACM Transactions on Information and System Security, pp.207-26.

[3] Baracaldo N, Joshi J, (2012) 'A trust-and-risk aware RBAC framework: tackling insider threat', In: Proceedings of the 17th ACM symposium on access control models and technologies, SACMAT'12, New York, NY, USA: ACM , pp. 167-76.

[4] Baracaldo N, Joshi J, (2013) Beyond accountability: using obligations to reduce risk exposure and deter insider attacks.. In: ACM symposium on access control models and technologies (SACMAT). The Netherlands: Amsterdam, pp. 213-24.

[5] Brodsky A, Farkas C, Jajodia S, (2000) 'Secure databases: constraints, inference channels, and monitoring disclosures', Knowledge and Data Engineering, IEEE Transactions, PP.900-19.

[6] Chakraborty S, Ray I, (2006) 'Trustbac: integrating trust relationships into the RBAC model for access control in open systems', In: Proceedings of the eleventh ACM symposium on access control models and technologies, SACMAT'06. New York, NY, USA: ACM, pp. 49-58.

[7] Chen Y, Chu W, (2008) 'Protection of database security via collaborative inference detection', Knowledge and Data Engineering, IEEE Transactions on,20(8):1013-27.

[8] Delugach HS, Hinke TH, (1994) 'Using conceptual graphs to represent database inference security analysis', Journal of Computing and Information Technology, pp.291-307.

[9] Dimmock N, Belokosztolszki A, Eyers D, Bacon J, (2004) 'Using trust and risk in role-based access control policies, In: Proceedings of the ninth ACM symposium on access control models and technologies SACMAT'04.

[10] Feng F, Lin C, Peng D, Li J, (2008) 'A trust and context based access control model for distributed systems' In: Proceedings of the 10th IEEE International conference on high performance computing and communications, HPCC'08. Washington, DC, USA: IEEE Computer Society, pp. 629-34.

[11] Jabbour G, Menascé D.A, (2009) 'The insider threat security architec-ture: a framework for an integrated, inseparable, and uninterrupted self-protection mechanism', In: The International Conference on Computational Science and Engineering.

[12] Ma J, Adi K, Mejri M, Logrippo L, (2010) 'Risk analysis in access control systems', In: Privacy security and trust (PST), Eighth annual international conference on, pp. 160-6.

[13] Moore A, Cappelli D, Trzeciak R, (2008) 'The "big picture" of insider it sabotage across U.S. critical infrastructures', CERT.

[14] Nissanke N, Khayat EJ, (2004) 'Risk based security analysis of permissions in RBAC', In: Proceedings of the 2nd International Workshop on security in information systems, security in information systems. INSTICC Press, pp. 332-41.

[15] Yip R, Levitt E, (1998) Data level inference detection in database systems, In: Computer security foundations Workshop Proceedings, 11th IEEE, pp. 179-89.

[16] Wickramaarachchi GT, Qardaji WH, Li N, (2009) An efficient framework for user authorization queries in RBAC systems, In: Proc. of the 14th ACM SACMAT technologies, SACMAT'09, ACM, pp. 23-32.

[17] Stoneburner G, Goguen A, Feringa A, (2002) 'Risk management guide for information technology systems', recommendations of the national institute of standards and technology.

