Bi-Layer Access Control model (BLAC) to handle insider threats in e-health systems

by

Shruthi Prakash

A Project Report Submitted
in
Partial Fulfillment of the
Requirements for the Degree of
Master of Science
in
Computer Science

Supervised by

Dr. Rajendra Raj

Department of Computer Science

B. Thomas Golisano College of Computing and Information Sciences
Rochester Institute of Technology
Rochester, New York

March 2016
Dedication

To my parents who always encouraged me to excel in whatever I was interested in.
Acknowledgments

I am grateful for Prof. Dr. Rajendra Raj, for guiding me throughout. He has been very helpful in clearing my doubts and guiding me in right direction.
Abstract

Bi-Layer Access Control model (BLAC) to handle insider threats in e-health systems

Shruthi Prakash

Supervising Professor: Dr. Rajendra Raj

Due to the increase in threats in e-health systems, various kinds of access control mechanisms have been followed over the years to protect the privacy and confidentiality of the information related to patients. Though the threats from outer world can be handled, insider threats are very crucial to handle. Insider threats are the threats from any one who is authorized to access data related to patients. RBAC and ABAC were introduced to handle insider threats but they have few disadvantages. The paper "Secure Access Control for Health Information Sharing Systems” written by Dr. Rajendra Raj and Suhair Alshehri explains BLAC (Bi-Layer access control) model which handles the disadvantages of RBAC and ABAC and over comes insider threats. This project is intended towards implementing BLAC. BLAC provides two layer access mechanism which makes the data safe from insider threats. Only if the first layer check passes, the second layer check is done. If both the checks pass successfully, access is granted to the data requested by user. This mechanism proves to be very effective compared to the previously suggested access mechanisms such as RBAC and ABAC.
Contents

Dedication ......................................................... ii
Acknowledgments ................................................. iii
Abstract ............................................................ iv
1 Introduction ....................................................... 1
2 Design ............................................................... 3
3 Implementation ................................................... 7
4 Analysis ............................................................. 11
5 Conclusions ......................................................... 17
   5.1 Current Status ................................................ 17
   5.2 Future Work .................................................. 18
   5.3 Lessons Learned ............................................. 18
Bibliography ......................................................... 19
Chapter 1

Introduction

In e-health systems, privacy and security are always the major concerns. Access control mechanisms play a major role when sensitive information is stored in the database. Role-based access control and Attribute-based access control mechanisms are good to some extent, however they have major disadvantages. Insider threats are threats within the system from authorized users. It’s very important to protect the system from insider threats.

Any authorized user who has access to the entire data in the system is a threat. Suppose a cardiologist has access to all personal data of a patient of another cardiologist, any day this can be a threat if the cardiologist misuses the data. This is where the need for a secure system comes into picture. When RBAC was used to handle insider threats it didn’t cover few scenarios which are crucial like the example mentioned above as the access was given based on the roles of subjects. To handle this ABAC was introduced where the attributes are used to define access rules. In ABAC policies are defined in order to make decisions regarding the access. For every request policies are checked and based on that, the access is denied or granted. However, ABAC has many disadvantages. Due to the huge number of rules defined in the policies which need to be evaluated every single time, the system can get complex and slower. Also, the performance will be affected.

To handle this, a new model BLAC (Bi-Layer access control) is suggested by Dr. Rajendra Raj, Sumita Mishra and Suhair Alshehri in their research paper “Insider threat mitigation and access control in healthcare systems”.

The goal of this project is to create a Bi-Layer access control model to protect the e-health systems against insider threats. This model incorporates the advantages of RBAC and
ABAC, also handles the limitations of these two. The BLAC model gives a 2-layer protection to the e-health systems from the insider attacks. Pseudo roles will be created for the subject (users) and the objects will have policies defined which includes the pseudo rules as well. Any request made will be passed to the decision maker which compares the rules against the rules defined in the policies and also looks for other rules to check if a particular request has to be granted and makes a decision. Performance will be evaluated in the end to see how effective the model is.
Chapter 2
Design

The BLAC model makes use of both the attributes and the policy listing, covering up the advantages of both RBAC and ABAC, handing the disadvantages of the same. BLAC makes use of pseudo-roles where the crucial roles are captured and policies are generated for those roles. Below is the work-flow of the BLAC model.

Like it is mentioned in the above figure, once the access is made, the pseudoroles are checked and if they match, they are further evaluated with the policies and then access is granted. If the pseudoroles don’t match the access is denied.

My design has a decision maker which takes the requests and calls the pseudorole generator and checks the permissions against the pseudoroles. The method returns a boolean
value and if it returns true, the policies are evaluated and then the decision is passed to the decision maker.

The below figure 2.2 best describes the structure of BLAC model. The subject has a set of pseudoroles which will be generated automatically with the algorithm. The object will have policies associated with it and the structure of a policy will be explained layer in this section. The decision engine is the one which communicates between the subject and the object and takes decision whether access has to be granted or not. My implementation doesn’t focus more on the rest of the factors.

Once access request is made from the subject, the pseudorole associated with the subject is picked up and the decision engine runs over the policies and checks for the rows that match the pseudoroles and returns true if they do. This happens to be the first layer check. If it returns true, the second layer check is done where the policies are further checked for subject id, object id and action. Based on this again it returns true if the access has to be granted or false if it has to be denied.

Figure 2.2: The BLAC model

Below is the policy structure in figure 2.4. The policy has the pseudoroles which are the main attributes on it. In the first layer check for access request, the pseudoroles that match
the pseudorole of subject are picked and in the next phase only those data are evaluated for subject id, object id and action specifier. Once the pseudoroles match, the rules are checked. If the subject id, object id and the action specifier match the one requested by subject the access is granted.

Figure 2.3: The policy structure[3]

The below figure shows the components of BLAC model as per defined in the research paper [3].
S, O, E, A, PR, P are subjects, objects, environments, actions, pseudoroles, and policies, respectively.

$SATT_k$ ($1 \leq k \leq K$), $OATT_m$ ($1 \leq m \leq M$), and $EATT_n$ ($1 \leq n \leq N$) are the predefined finite set of attributes for subjects, objects, and environments, respectively.

assigned-attributes(s), assigned-attributes(o), and assigned-attributes(e) are attribute assignment relations for subject s, object o, and environment e, respectively:

assigned-attributes(s) $\subseteq SATT_1 \times SATT_2 \times \ldots \times SATT_K$

assigned-attributes(o) $\subseteq OATT_1 \times OATT_2 \times \ldots \times OATT_M$

assigned-attributes(e) $\subseteq EATT_1 \times EATT_2 \times \ldots \times EATT_N$.

$PR_i$ ($1 \leq i \leq I$), is a set of $g \in SATT$ such that $g \leq K$.

$SPR \subseteq S \times PR$, a subject-pseudorole assignment relation that is a one-to-many mapping from pseudoroles to subjects.

assigned-subjects(pr) = \{s \in S \mid (s,pr) \in SPR \}, the mapping of a pseudorole pr onto a set of subjects.

$P_g$ ($1 \leq g \leq G$), is a set of access policies that determines whether a subject s can access an object o, such that, $P_g = \{FPR, FR_1, \ldots, FR_n\}$, is a policy that consists of a Boolean function of $PR$, and a finite set of Boolean functions of access rules of s, o, a, and e’s attributes.

Figure 2.4: The components of BLAC model [3]
Chapter 3

Implementation

To implement the BLAC model, I analyzed over many e-health systems and generated health data and created spreadsheets of data. I then converted the spread sheets to XML. My java code is intended to take the requests and pass it to decision maker. The decision maker calls pseudorole generator and checks the roles and if it returns true further evaluation of policies are done. If in the end the decision maker returns true, the records are fetched from the XML and displayed.

I have maintained three packages, one to access the subject’s data which has policies associated to it, another to access object’s data which has pseudoroles associated to it, and the last one is the decision maker which fetches the requests from the user and accesses both the subjects data and the objects data based on the pseudoroles. The code comprises of the generation of pseudoroles based on the data taken from the research paper [4].

The pseudorole generation involves combining three factors. The root holds the role of the object, followed by the leaf nodes that has department and the location. These three together comprises and makes a pseudorole. These pseudoroles are then added to policies on the subject side. Whenever an access request is made, the pseudorole associated with that particular object is picked and then checked against the policy associated with the subject.

The decision maker filters out the the rows in XML that matches the pseudoroles. This returns a boolean value after the first level of checking. If the decision maker returns true, the second level check will be processed. In this phase, the filtered out rows are checked to match the Subject id, object id and action. If all the details entered by user matches these the access is granted else it returns no. The above figure explains the pseudorole
Figure 3.1: Pseudorole generation[3]

Figure 6: Pseudorole Generation in the BLAC Model

generation. The root nodes have unique values and the trees are built adding values to the root nodes. Every root leads to a leaf node which together are distinct compared to every other path. The algorithm I have written creates the possible combinations of all the values with the root nodes. The root nodes can be chosen as the actual roles. In any e-health systems there will be various roles and each role will have access to multiple records. Not all cardiologists will have access to all patients who are getting treated by cardiologists, also a physician shouldn’t be able to access data of patients who are getting treated by cardiologists unless he has a role involved. This will be taken care of by diving every role as the root and associating with the leaf nodes which captures the location and departments. This is the major step to filter out loads of data for accessing. This makes it run faster and makes it more efficient unlike ABAC where entire data have to be run across to find a set of records.

The main disadvantages in RBAC were, it is tedious to define the roles and too expensive, and also misleading as all the records that are made visible to a role is visible to everyone in that role which can be a major insider threat. To handle this ABAC was defined where the main focus was on attributes. However, ABAC leads to awful performance as a huge number of records need to be accessed. The BLAC model handles the limitations of both with the mechanisms followed in it.
Below is the algorithm written by me as per the idea in the paper [3] to generate pseudoroles. The algorithm in the figure 3.2 takes the set of vectors which in turn has vectors of data. The data includes roles, department and location. These vectors are converted to 2-D array and then the roles are considered to have distinct values and the tree is built based on the root. Every path will be distinct starting from the root node to leaf. Root is the role followed by department and then the leaf nodes have locations. This algorithm was successfully implemented and pseudoroles of all combinations were generated.
vecOfActualRole $\leftarrow$ Vector of roles
vecOfDept $\leftarrow$ Vector of departments
vecOfLocation $\leftarrow$ Vector of locations
vecOfRoles $\leftarrow$ Vector of all the above vectors
arrayOfRoles $\leftarrow$ convert vecOfRoles to 2D array
sizeOfEachArray $\leftarrow$ length of arrayOfRoles
tracker $\leftarrow$ length of arrayOfRoles
possiblePseudorole $\leftarrow$ 1

For index equals possiblePseudorole to 1
    sizeOfEachArray $\leftarrow$ length of arrayOfRoles[index]
    possiblePseudorole $\leftarrow$ possiblePseudorole * length of arrayOfRoles[index]
    For value equals 0 to length of arrayOfRoles
        Append arrayOfRoles[value][tracker[value]] to storeRoles
    End For
End For
listOfPseuRoles $\leftarrow$ toString(storeRoles)

For newvalue equals (length of arrayOfRoles) -1 to 0
    If tracker[newvalue] + 1 is less than sizeOfEachArray[newvalue] then
        tracker[newvalue] $\leftarrow$ tracker[newvalue] + 1
    Break
End If
End For
End For
Print listPseudoroles

Figure 3.2: Algorithm for Pseudorole generation
Chapter 4

Analysis

I read few papers in order to see how the security mechanisms are actually followed in general. One of those papers [1] says, with improvement of technology, health care is harnessing the brighter side of it. So the data persisted in any e-health system should be safe from potential threats and hacks. Thus it is very important to have a valid security requirements are laid out and validated to make sure the quality of the e-health system is well maintained. This paper discusses Model Oriented Security Requirements Engineering (MOSRE) framework where security requirements and goals are identified at early stages of software development so that any vulnerability and threat to e-health system is known before hand and prevented.

There is a need to analyze, elicit and identify the security goals and objectives for a system in different levels like host level, database level and network level. There are several steps which this paper discusses as part of MOSRE framework for e-health web applications as part of requirements engineering phase of security. Briefly these steps would identify different entities of the e-health system like objectives, stakeholders, assets, threats, risks, security requirements, then come up with use cases, UML and class diagrams to have a broader picture of the security requirement. Business requirements should be procured for a health system, followed by a use case diagram. Once this phase is completed it is very important to identify potential threats and vulnerabilities which can be faced by the system, these threats should be categorized and ordered in priority. Then a use case diagram and sequence diagram should be designed as part of security requirement phase [1].

It often happens that security is under rated during initial phases of development, the after
effect would be really catastrophic and it would be really difficult to add security on top of the development at later stages. Thus it is quite possible that people who code does not take security into concern and the system might fail miserably no matter how good the implementation is, especially for critical systems like e-health system where patient data should be kept safe and secure from hackers and eavesdroppers. So it is notable that using MOSRE framework this issue can be addressed effectively[1].

The research paper [3] focuses on very interesting topic which is handling insider threats in e-health systems. The role based access control was the first mechanism which was used in e-health systems which had many flaws associated with it. It was a very expensive process and was a very tedious task to run over all the data and still give free access to data which would cause insider threats [2]. Here the data was picked only based on the roles. This could easily result in providing access to data that is not of any interest to the user. This was not a good option as the user could as well be a threat though he is allowed to have access to some part of the data. This resulted in the whole new idea of Attribute based access control[5]. The attribute based access control did give good results but still failed when it came to performance. It was time consuming as a huge set of data had to be evaluated in order to grant access. As and when the number of rows increases the performance used to hinder. This was the motivation in the paper [3] to come up with the BLAC model. This handles all the limitations of ABAC and RBAC.

Once the BLAC model was implemented it was analyzed based on the following factors.

- The first thing that was tested was if the BLAC can successfully handle insider threats. Random access requests were given manually to check if the algorithm works as expected. Boundary conditions were checked where the algorithm could actually fail, for example trying to write data to objects where the policy says only read action can be performed and so on.

- Pseudoroles were cross-checked to see if all the root nodes are distinct and every path from the root node to leaf node is distinct.
- The time stamp was recorded to check the performance and was compared against regular access. As and when the size of the data increases the performance depreciates in regular access but in BLAC model the performance proves to be really good.

- The size of the data sets were altered to run multiple times and check the performance.

![BLAC model performance against normal access](image)

**Figure 4.1: BLAC model performance**

The above graph in figure 4.1 describes the performance of BLAC model against regular access for multiple runs. The graph records the average time taken for access.

The graph in figure 4.2 describes performance of BLAC model against ABAC. The graph was plotted taking the time taken to run on y-axis and the number of rows of data in x-axis. For small amount of data the performance was similar and there were no major differences. But as and when the data size increased the performance of ABAC depreciated taking ample amount of time where as BLAC model remained almost constant.
The below table 4.3 describes the characteristics of the quality of BLAC model which was taken from the research paper [3].

- **Flexibility**: BLAC is more flexible when compared to RBAC and RABAC as RBAC and RABAC are more of a static model and BLAC is dynamic as it doesn’t store the roles like in pre-defined models [3].

- **Granularity**: BLAC supports granularity with its two layer access. The pseudoroles defined handles the bigger picture and the policies take care of the minute details. Whereas ABAC and RBAC doesn’t do that[3].

- **Authorization Complexity** BLAC handles bi-layer protection like mentioned earlier. Once the user request access BLAC first picks up the pseudoroles associated with it and then it is checked against the pseudoroles mentioned in the policies. If it matches, the rows are filtered out and further check is done to see if the subject id, object id and action matches. Till both the layers return true no access will be given[3].
Privilege Modifiability Policies defined in the BLAC model can be altered dynamically though it is a tedious job. But ABAC and RBAC makes it more complicated [3].

Permission Modifiability BLAC doesn’t do a great job when it comes to modifying permission as the entire policy structure needs to be modified. This is a tedious process and the same thing happens in ABAC as well[3].

Revocability Once the pseudoroles are generated in BLAC model they can be mapped to the subjects the way they want and hence it is easier to revoke[3].
• **Permission Reviewability** In BLAC model, once the pseudoroles match the ones in policies, only those set of records are considered for further checks to grant permission which makes the process really simple [3].

• **Setup Complexity** Making use of pseudoroles in BLAC model already makes it simple and also if the generation of policies are automated this becomes a very simple task [3].

These are the few things analyzed in the BLAC model and the implementation does whatever was expected.
Chapter 5

Conclusions

Based on whatever was suggested in the research paper[3] and the implementation I conclude that BLAC model is very effective to handle insider threats. The model I implemented was tested across various sets of data both for performance and the correctness. It proved to be the best when it comes to handling the insider threats as it has a set of clearly defined policies. Performance was measures running the model across various sets of data and multiple times. It is much faster compared to the normal access.

5.1 Current Status

Data has been simulated and converted to XML format for easy access. The algorithm to generate pseudoroles is written based on the idea suggested in the research paper[4]. The algorithm is also implemented successfully and pseudoroles are generated dynamically. Policies are generated based on the pseudoroles that are created. Data is increased and altered to compare and test. Testing has been done to check for insider threats. Performance is also evaluated comparing the time taken for BLAC model to run against normal access. After multiple runs the average time taken has been selected and performance is there by calculated.

BLAC model proves to be effective even when there is large set of data. It does handle the limitations of RBAC and ABAC. RBAC would let the user carrying a particular role access all the records visible to that role though most of it may not be needed. For example, a patient whose records should be visible to one of the cardiologists, can actually be accessed
by all cardiologists which was a major flaw. ABAC does fix this issue but it proves to too expensive and tedious and slows down the performance as and when the data increases because due to the large volume of data it lets the user evaluate huge sets of policies which inturn slows down the performance. BLAC handles these two limitations successfully. Due to the bi layer checks, in the first phase the records that hold the particular pseudorole associated with the user gets filtered out, there by cutting down the time it would take in the next phase. In the next phase, the policies are evaluated against the actions the user is requesting and the decision is taken based on the policies.

5.2 Future Work

I would like to use database and migrate the code to it. As of now I work on big chunk of data I simulated. I wrote and implemented algorithm to generate pseudoroles. In future I would also like to automate the policy creation. It would be good if I can work with real data, however e-health services don’t share the data as it does have sensitive information.

5.3 Lessons Learned

Initially I used database but then realized it is not a feasible option in the given time as the main focus was to handle the insider threats and check the performance of the model. Using database was time consuming but still a good option for future work. I also had issues getting the data as e-health systems don’t have open source data, so I had to simulate the data.
Bibliography


