Next Generation Access Control for the Mobile Health Cloud

Rejina Basnet*, Vignesh M. Pagadala†, Subhojeet Mukherjee ‡ Indrakshi Ray§
Department of Computer Science, Colorado State University
Fort Collins, CO, 80523, USA
Email: *Rejina.Basnet@colostate.edu, †Vignesh.Pagadala@colostate.edu,
‡Subhojeet.Mukherjee.@colostate.edu, §Indrakshi.Ray@colostate.edu

***Abstract*—In today's era health informatics is a major contributor to the advancements in ubiquitous computing. Of late, the concept of mobile health (mHealth) systems has attracted considerable attention from both medical computer science communities. mHealth devices generate a significant amount of patient data on a timely basis. This data is often stored on cloud based EHR and PHR systems to aid in timely and better quality healthcare service. However, as has been seen lately, stored personal records act as honeypots for malicious entities and the internet underground. It is thus imperative to prevent unauthorized leakage of mHealth data from cloud based E/PHR systems. As observed from some of our preliminary research, NIST's policy machine (PM) framework [1] suits the access control modeling requirements posed by mHealth systems. Moreover, the graph-based model [2] adopted by this framework allows efficient policy management through advanced graph search techniques. In this paper, we leverage the policy machine model to propose a cloud-based service that achieves secure storage and fined-grained dissemination of mHealth data. The primary goal of this work is to demonstrate the applicability of the PM framework to the mHealth domain and illustrate the workflow of an algorithm to resolve access decisions in theoretically faster time than achieved in [2].**

I. Introduction

A. Mobile Health Systems

Recent years have witnessed significant a growth in the field of healthcare information technology. This, coupled with advances made in mobile computing, have essentially laid the foundation for the advent of mobile-healthcare systems, abbreviated as 'mHealth'. We could be on the verge of a new revolution in healthcare-technology, with mHealth being the dominant player. According to a report from Berg Insight, by late 2012, home-monitoring healthcare systems (based on monitors with integrated connectivity) were being made use of by an estimated 2.8 million people around the world. The report goes on to predict that the number of such systems would experience a compound annual growth rate of 26.9 percent between 2012 and 2017 eventually resulting in 9.4 million connections by the end of 2017 [3]. Also, healthcare related mobile applications have also seen considerable growth, with one study predicting a total of 500 million patients making use of healthcare-related mobile applications by the end of 2015 [4].

mHealth can be broadly described as a system which involves the use of mobile computing devices in discharging healthcare-related services [4]. The architecture of this system, as described by Avancha et al. [5], comprises of a set of sensor nodes, mobile nodes, and a remote, centralized repository. Sensor nodes (SN) are basically attached to the patient's person in different configurations (based on what type of health-data is required), and are responsible for detecting and relaying information in real-time. For example, a sensor node attached to a person's wrist could be used in assessing the pulse rate of the user and relaying it to the appropriate end-point. Mobile nodes (MN), on the other hand, capture the information sent by the SNs and transmit the same to the remote repository, possibly after performing some processing, on the received data, and aggregating the same. These MNs could either be a mobile phone or a Mobile Internet Device (MID), and this is, in addition to the SN's, carried by the patient. The remote repository is referred to as the Health Records System (HRS), which is the centralized storage system holding the records of all the patients who have registered to the mHealth service, where data received from different MNs are stored after performing some preprocessing. Any entities such as physicians, researchers, insurance companies or lawyers who required access to a patients data, can obtain it from the HRS, with suitable permissions from the patient.

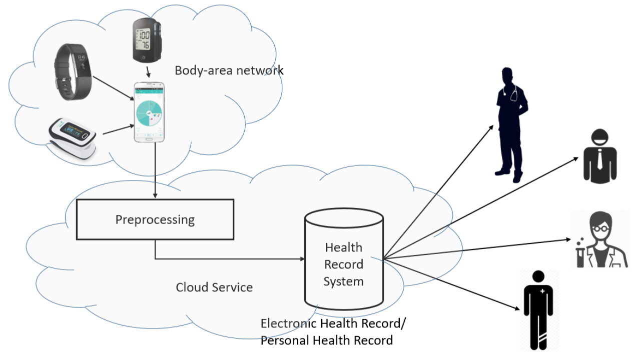

Fig. 1: mHealth Architecture

B. Threat Model and Problem Description

The mHealth system could be under threat by a myriad of sources in different ways. The sources of threat can, however, be broadly categorized as (1) Patient (the patient himself or herself), (2) Internal entities like like doctors, insurance companies and HRS cloud service providers who can gain authorized access to some portion of the patient's data and

(3) External entities like hackers and thieves who do not have authorized access to any portion of the patient's data [6]. As seen from Fig. 1, the assets in consideration are the MID, SNs, the preprocessing unit and the HRS itself. Any or every of threat sources can get unauthorized access to stored records, perform malicious alterations or deny services provided by the critical units in the mHeatlh infrastructure (from Fig. 1). Although, all of the above mentioned cases are possible, in this paper, we address the privacy concerns raised by unauthorized access to the PHR records stored on the HRS. As observed by Kotz e al. [6], the concerned threat agents can be any of the three entities listed in the beginning of this paragraph.

From above mentioned threat model it is evident that it is imperative to prevent unauthorized access or disclosure of stored mHealth data. There are, however, two critical aspects to this problem that need attention. Firstly, an authorized entity may only require data access for a limited number of features in the patient's data record, which means fine-grained access to data is required. As an example, an insurance company, for example, might only have a need to peruse cardiovascular data only. In case a flaw in the system exists, an insurance company employee might be able to illegally gain access to other, personal details of the patient and use to extort money. This would constitute the case of an insider attack. Secondly, in the context of healthcare applications, fast access decisions are an absolute requirement. A slow HRS server can lead to catastrophic consequences for a patient in an emergency situation.

C. Related Work

Role-Based Access Control is made use of in traditional mHealth systems [5]. Kulkarni et al. [7] have implemented the use of an RBAC mechanism for their mobile-healthcare solution. They argue that, RBAC would be more meaningful in the context of healthcare when compared with Mandatory Access Control (MAC). MAC's mechanism involves designating varying levels of sensitivity to an object based upon the information contained within, and granting access to any subject based upon the subjects clearance. This mechanism not suitable in the context of healthcare applications, since in this case, it is not only necessary to consider the information's sensitivity, but it's also required to ensure control over individual action, which leads to preference for RBAC. Kotz [6] suggests RBAC as well, for controlling access to a patient's Personal Health Information (PHI). The same justification as given by Kulkarni et al. is provided by Kotz for the use of RBAC in a healthcare setting. However, as the scale of an organization grows, trying to properly fit roles to individuals becomes more difficult and is very challenging to coordinate. RBAC, however, suffers from its own share of pitfalls, Firstly, it's very difficult to manage. In the context of a large organization such as a healthcare institute with many different administrative domains, RBAC is going to complicate things when it comes to making an access decision based on a role [8]. This is because, the concept of a role might differ across organizations or different domains withing an organization. Furthermore, the process of assigning roles and managing them is very difficult in a large organization [5]. Finally, RBAC does not consider object level attributes in making access decision and thus lacks in the level of granularity [9].

Lu et al. [10] suggests the use of Attribute-Based Access Control (ABAC) in implementing an efficient access-control methodology to achieve privacy in their mHealth system. ABAC successfully overcomes the drawback presented by RBAC by making access decisions based upon different attributes of the user, object and the environment [11]. But it is still not a possibility to enforce multiple policy classes in ABAC. This very much is a possibility when we design our access-control system using PM. Using PM also allows us to apply graph algorithms which may help speed up the process of making an access-control decision, which is significantly important in the context of an mHealth system.

D. Policy Machine for mHealth Systems

The problem domain addressed in this paper requires effective policy representation and enforcement in the mHealth cloud service's access control system. Policy Machine [1] provides us with a suitable framework for defining policies and implementing the same. In the context of mHealth, using PM can help us overcome to drawback with respect to a Role-Based Access Control methodology where it's difficult to identify and manage roles. Enforcement of policies are greatly assured using PM. Then again, as observed by Mell et al. [2], the PM framework can be represented using directed acyclic graphs, thereby opening up a wide range of possibilities for optimizing search and retrieval techniques This, in turn, can speed up the process of evaluating access decisions. Considering these advantageous aspects of PM, we propose using PM to model the access-control framework in mHealth systems.

In this paper, we also device a model for representing access control policies that allows fine-grained release of patient records, to granularity of individual items. For example, if a patient's record consists of more than one items like cardiovascular data and mental-health related data, our model enables authorized release of individual items. Finally, we implement our PM framework on a graph database. This allows us to execute graph search algorithms natively on persistently stored access control policies.

E. Paper Outline

The rest of the paper is organized as follows. In section II we present, in details, the background knowledge required to clearly comprehend the work done in this paper. This includes a brief overview of the NIST policy machine [1] framework, the Neo4j graph database and the CM algorithm [12]. Section III presents the salient features of the solution architecture presented in this paper. We start by describing our solution architecture, followed by a through description of the PM graph generation process and algorithms used in answering access queries. In section IV we present a thorough analysis of our algorithms. Finally in V we summarize the contributions made in this paper followed by a concise description of the tasks we aim to accomplish in the future.

II. Background

A. Policy Machine and Next Generation Access Control

Policy Machine, in general, is termed as a "Logical Machine " which consist of fixed set of relations and functions between the policy elements. It forms the basis for the development of the Next Generation Access Control or NGAC. NGAC utilizes the policy machine constructs to model diverse access control policies including combination of multiple policies. The NGAC consist of i) set of data elements and relationships, ii) set of operations and iii) set of functions. The data elements and relationships are structured carefully to express a rich variety of access control policies. The operation set represents the action a subject need to perform on resources or the administrator needs to perform in order to properly manage the policy elements. Finally, the functions are there to assist implementation of access decision computation and enforcement of policy for every action.

1) Policy Machine Constructs: The policy machine is constructed of basic policy elements and the fixed set of relationships.

a) Policy Elements:

- Users (U) The individuals authenticated by the system are called Users. They are identified by unique identifier in the system.
- Processes (P) Processes are system entities with memory and operates on behalf of user. A user can submit access request only through processes. A process can issue access request and can reach out to their own memory exclusively but none to other processes. While communicating and exchanging data with other processes, they do it through physical medium known as clipboard or sockets.
  In PM, users and processes are viewed as totally independent entities. A user at a time may have association with one or more process but a process is always governed by one user at a time.
- Objects (O) Objects can be defined as system entities that are governed by one or more defined policies. They have unique identifier in the system. In groups, they reflect environment-specific entities and need protection, e.g. files, ports, email messages, records and fields, clipboards. Objects also comprise of policy elements and relations that need protection.
- Operations (OP) Operations denote actions that can be performed on the contents of objects that represent resources or on PM data elements and relations that represent policy. The entire set of operation comprises of common resource operations such as read , write(other related opertions) and the administrative operations which are limited to PM data elements and relation.
- Access Right (AR) For an user to carry out any operation, s/he needs to have proper access rights. Access right can also be categorized as administrative and non- administrative access rights.
- Policy Classes (PC) Policy class is used to organize and distinguish between distinct types of policy being expressed and enforced. A policy class can be thought of as a container for policy elements and relationships that pertain to a specific policy.
- User Attribute (UA) User attribute also plays similar role i.e. a container that assists to organize and distinguish between distinct classes of users.
- Ojbect Attribute (OA) The role of user and object attribute is same except for object attribute helps to distinguish objects.
  Every object serve as an Object attribute in PM model. i.e. object is subset of object attribute. However, the case is not true for the user and user attributes.

b) Relationships: Realtionship is PM model are categorized into two groups.

- Assignment (ASSIGN) It is used as a means to express a relatinonship between users and user attributes, objects and object attributes,user (object) attributes and user (object) attributes, and user (object) attributes and policy classes. The assignment relationship should be irreflexive (if x ASSIGN $y \Rightarrow x \neq y$), acylic. The acylic nature of assignment results in a hierarchically ordered relaitonship. Also, a sequence of assignments must exist from every element in U , UA and OA to some element in PC.
  A user may be assigned to one or more user attributes. The assignment U ASSIGN ua means that the user u is assigned to or contained by the user attribute ua. It also denotes that user u takes on or inherits the properties held or represented by the attribute ua. The properties of a user attribute are defined as the capabilites for and prohibitions against acessing certain types of objects. Similarly, an object may be assigned to one or more object attributes through one or more object-to-attribute assignments. The assignment o ASSIGN oa means similar to u ASSIGN ua. The properties of an object attribute are defined as the capabilites and prohibitions allotted to users , which govern access to contained objects.
  Unlike user and object attributes , the policy class cannot be assigned to any other policy class.
- Association (ASSOC) Association define relationship that involve the authorization of access rights between policy elements. They are the policy settings that govern which users are authorized to access which objects and exercise which access right. Association are represented as ternary relation (UA x ARs x OA). Within one policy class, an association ua-ars-oa specifies that all users contained by ua possess the authority denoted by ars over all objects contained by oa. The associations affecting a user's access rights over objects can occur at various levels within an attribute hierarchy. Similary, associations that affect an objects's accessibility by user can also occur at various levels.

PM also consist of a notion of prohibitions and obligations which further assist in making the policies granular. However, we do not consider these constructs in our implementation. Also, an authorization graph can consist of more than one policy class. For the purpose of this paper, we utilize only one policy class.

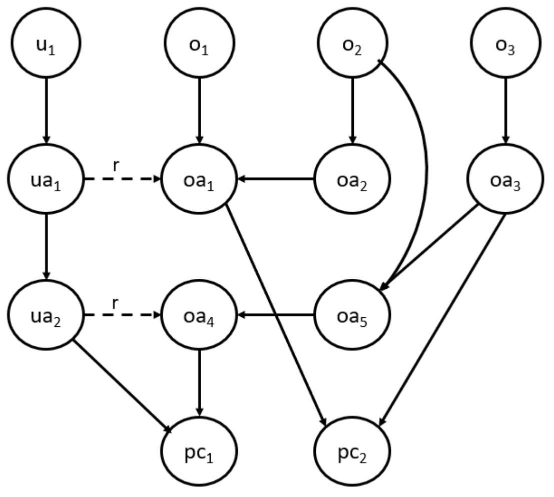

Fig. 2: Example PM Instantiation

2) Evaluating User Privileges: We make use of the sample policy shown in Fig. 2 for the process of evaluating user privileges. Policy machine [1] defines a user privilege as the triple (u_1, r, o_2), where $u_1 \in U, r \in OP$ and $o_2 \in O$. However, Mell et al. provide a more intuitive graph theory oriented definition in [2]. According to the authors, a user u_1 can perform an operation r on an object o_2 if there exists an edge labeled r, the tail of which can reach u_1 and the head of which can reach o_2. Mell et al. also mandate that privilege is considered granted if the head nodes can belong to a superset of the policy classes ($\{pc_1\}$) that the tail nodes can reach ($\{pc_1\}$). However, since in our preliminary stages we only use 1 policy class, our evaluation does not incorporate this aspect.

B. Graph Database: Neo4j

Neo4j is a NoSQL native (graph specific storage and processing engine) graph data base that implements property graph model to the storage label. In addition it provides the facilities like ACID properties.

A property graph model is comprised of nodes, relationship, properties and labels. Nodes are main data elements connect to each other via relationships. The nodes as well as connecting relationship can have properties of their own. In order to categorize the nodes into the related groups, labels are used. A node can have multiple labels in the same time. All the labels are indexed in order to accelerate finding nodes in the graph. Neo4j makes use of Cypher Query Language for its operation. Cypher is a declarative graph query languange , inspired by SQL with pattern matching. Its natural pattern matching removes the necessity to debug JOINs.

Finally, Neo4j allows computation of a records location in O(1) time . It makes this possible by storing the files records into multiple node store files. The record stored are fixed sized (9 bytes). Therefore, if we have a node with id 100 then we know that the record begins 900 bytes into the file. Also, the user of relationship instead of indexes further aids into fast traversal of the graph. It is this feature of Neo4j that prompted us to select it as our primary choice for storing and processing policies.

C. Common predecessor Algorithm

Common predecessor algorithm is a graph traversal algorithm which allows us to find common predecessor given a node in a directed acyclic graph. It utilizes the basic Depth First Search to do so. The following paragraph describe the algorithm in a nutshell.

Let us consider a graph G(V, E) where V is the vertexes and E is the edges. G^t would represent the transpose of the graph G were all the directed edges of graph G are inversed. The vertex having zero incoming edges are the source nodes. Similarly, a vertex v is said to be a successor of a vertex w if there is a direct path from w and v. A vertex v is said to be a predecessor of a vertex w if there is a direct path from v to w. The problem this algorithm solves is given two vertices u,v determine whether u and v have a common predecessor. To solve the problem, a very naive way would be to run a depth first search from u in G^t marking all the visited vertices and then performing depth first search from v in G^t, determining whether any marked vertices are encountered. Due to the time complexity of the algorithm which is O(m) might increase to quadratic with increase in number of queries. Therefore, an indexed version of common predecessor in use which runs depth first search from each source nodes labeling all its successors with the source nodes. In case where there are multiple sources in same node they are appended to the back. If the labels to each vertices are at most p than the time complexity of the labeling would be O(pm) and the query takes O(p) to determine if the queried vertices share a common label.

III. Proposed Solution

A. Solution Architecture

Fig. 3 shows the architecture of our reference implementation. As mentioned earlier in section II-A, the policy machine model includes user (u), user attribute (ua), object (o) and object attribute (oa) nodes. We obtain information pertaining to o and oa nodes from the incoming traffic as received from the MN nodes (I-A). For the purpose of instantiating the model, we make use of a Fitbit dataset as obtained from [13]. The dataset includes attributes such as 'Calories' (amount of calories burned by the wearer), 'Steps' (number of steps taken), 'Date-Time' (date and time of the observation), 'Distance' (distance traversed), 'PID' (patient ID), 'elevation' (elevation at which the wearer is) etc. For the sake of simplicity, in this work, we only consider the features 'PID', 'Date-Time', 'Calories' and 'Steps'. This can be seen in Fig. 3 [1]. Each measurement (like calorie and step) is envisioned as a data object $o \in O$.

The architecture also consists of a *Registration Portal* via which we collect subject information. Subjects are those entities which request for access to the data from HRS. Subjects can include, but are not restricted to, physicians, insurance companies, lawyers and patients. Subject attributes

[1]To be more application specific, we rename the 'PID' attribute to 'ownerid'.

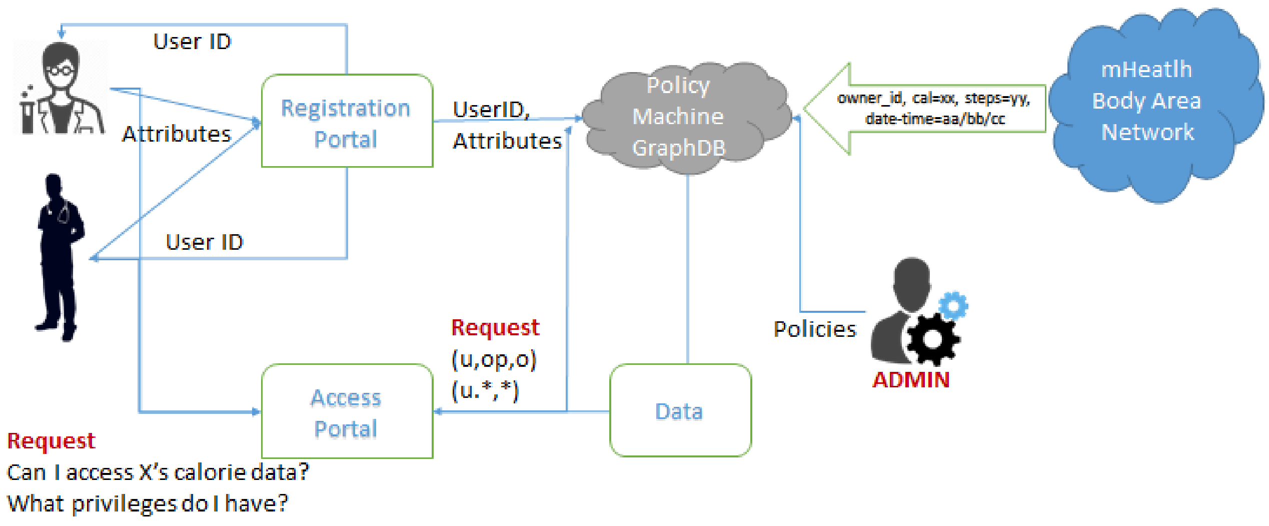

Fig. 3: Proposed Solution Architecture

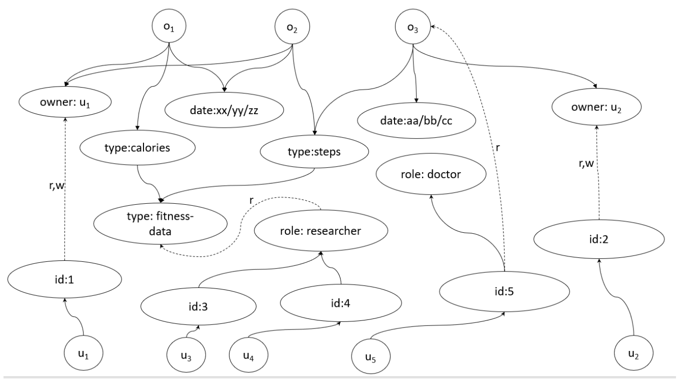

Fig. 4: Example NGAC Policy

can be *role*, *name* etc. Once a subject is registered, they are assigned and provided with a system-wide unique identifier. At the time of data retrieval subjects query via the *Access portal* and their queries are converted into PM formalized access requests. These requests are then evaluated using algorithms demonstrated later in the section III-D. If granted access, authorized objects (or set of objects) are returned to the user.

Although, it is recommended to have user-centric policies in an mHealth system [14], in this work we assume this responsibility to be fulfilled by the system administrator. We however, plan to address this issue in future.

B. An Example NGAC Policy for the mHealth Domain

A healthcare institute which provides mHealth services has its policy defined as follows. In this scenario, we consider a total of five users in the system, who identify as an Owner, and also with the roles Doctor and Researcher. Users u1 and u2 are Owners, i.e., they are the patients in the mHealth system, and possess read and write privileges over the type fitness-data, which in turn consists of the features Calories and Steps. This institutes database consists of each object node having three attributes - date (the date on which the observation was made), owner (the patient with whom the data is associated to) and type (the type of data being recorded). The patient is allowed to possess read and write privileges over Calories and Steps in this case, since the institute does not consider knowledge of these features to be in any way a detriment to the patient, and also believes that incorrect alteration of this data (by the patient) would not result in any significant harm. Users u3 and u4 identify as Researchers, and are allowed read access to all fitness-data features of all the patients, by the institute. User u5 identifies as a doctor, and according to the policies of this institute, is only allowed read access to his own patient u2s Steps on the date aa/bb/cc. The PM for this institutes policy is represented in Fig. 4.

C. Constructing the PM Graph

To construct the authorization graph for Policy Machine, we need to figure out the users, object, user attributes, object attributes and permissions. Since, we are having only one policy class we will be assigning every elements in the system to a single policy class.

1) Object and Object attributes: For construction of object DAG we will be processing the sample data we have about the mHealth. Each row is the information association with a user and each columns represents what information about the user. Therefore, attributes will be designed on the basis of rows index and column index whereas each cell would be an individual object in our system. The processing will be done in the row basis. From each row, one of the object attribute wil be the userid to which the record belongs to if not originally present in the system. Other object attribute will be the features data if not originally present in the system. For example, if a record belongs to user with id 'user54' and the record is displaying the time the record was taken , pulse rate at the time, and cholesterol label at the time, the object attribute would be user54, pulse data , cholesterol and time. Each of these attributes will be represented by a separate node. On each of pulse, cholesterol , time would be assigned the node (representing object) with value of the cell. Each of these will also be assigned to the node representing userid and time at which the data is taken. In this way, we have a graphical representation that certain data categorized into certain groups belongs to a user and were taken in a specific time. Each row will be processed accordingly and a new attribute will be added to the system if and only if it doesnot exist in the system already. for example, if a row represents record for a new user "user60" then an attribute user60 will be created and attribute time will be created (assuming there are no records in this specific time for any of the user in the system which is highly possible time being continuous). But nodes for attributes cholestrol, pulse rate will not be created again.

2) User and User Attribute: For User and User Attribute we will have two sources available. When processing for object and object attributes we will have certain information about the user to be included in the system. Therefore, we will be creating user node for each of the user existing in the system, however, these users will not have any attrbutes yet except patient since we know that these records belong to patients.. To gather the information of users who would require to use the system, we will have a web portal available. Through the web portal they will be able to register themselves to the system with their respective attributes. Each user will be assigned to respective nodes representing respective attributes. Again, attibutes will be created only if they are not initially present in the system.

3) Policies: The assignment of policies are the crucial part of our system. Therefore, we are considering the existence of trustworthy admins for this purpose. The admins will recieve the policies that needs to be created through some media (messages). The admin is reponsible for making analysis of the incoming policies and verifying for the eligibility. After the analysis the admin is also responsible for changing the graph to accomodate the policies. One of the major change we have made in our system is the representaion of the polices in the authorization graph. Originally, according to the PM construct and association relationship represents the assignment of polices from user attribute to object attribute. We have changed the association relationship to be reprsentationg by a node which containg the privelege to be assigned connected by and incoming edges from user attribute and the object attribute. This actually makes it possible to find the required privelege through common predecessor algorithm.

4) Constructing the Index: During the indexing process the authorization graph is indexed by two different kind of indexes depending on the nature of node being indexed. The first kind of index includes the list of common operation nodes and the second kind of index includes the list of reacable sourc list of type user and object.

Algorithm 1

```
1:  for all the source list [user and object]  do
2:      if source is a user then
3:          user,userlist[]=list of all the nodes visited during DFS traversal from user
4:      elseIf soure is object
5:          object,objectlist[]=list of all the nodes visited during DFS traversal from object
6:      end if
7:  end for
8:  for  all the userlists do
9:      for  all the objectlists do
10:         Find common nodes for user list and object list
11:         Index the user and object node with the list
12:         if not done for this object then
13:             Index all the nodes in the list with the object except the object itself
14:         end if
15:     end for
16:     Index all the nodes in the list with the user except the user itself
17: end for
```

The changes we have made to our algorith is that we are traversing the non transposed graph and not marking all the visited nodes in the way. Instead we are listing all the visited nodes and further processing to the list the generate two kind of indexes for each source nodes and non source nodes. All the source nodes indexed with its common predecessor if the DFS was done in Trasposed graph and all the non source nodes indexed with source nodes through which it is reachable. Also, a point to note would be all the indexes contain list of node ids and all of the lists are sorted in an ascending order.

D. Evaluation of Access Request

For evaluation of the access request we perform a preliminary indexing into the graph using a modified version of common predecessor algorithm. Our future plan is to be able to do the indexing during the time of graph creation. For the very first approach our assumption is that the graph is created first and indexing is done in the whole graph and the queries will be performed in the indexed graph.

1) Evaluation of the access request: After indexing the graph is ready to be queried. There are two types of queries the graph will be able to answer which are described in following subsections.

2) Queries of the form $< u, op, o >$: The query of the form $< u, op, o >$ is represents whether a user u has permission to perform op in the object o. It tends to answer a single access request to the resource in the form of grant or denial. Following algorithm is used to process the query.

Algorithm 2

```
1: cplist_for_u ← get cplist property of node u
2: cplist_for_o ← get cplist property of node 0
3: for intersection between cplist_for_u and cplist_for_o do
4:     if op exits in any of the intersection node then
5:         Access granted
6:     else
7:         Access denied
8:     end if
9: end for
```

3) Queries of the form $< u, *, * >$: The query of this form represents what objects would a user have permission to perform given operation or what user would have permission to perform given operation on the object. This might be useful to see the capabilites of a given user or to see the exploiters of a given object. Following algorithm is used to process the query.

Algorithm 3

```
1: for all the action nodes present in the system do
2:     if action node contains op then
3:         if query is for user then
4:             if user is in user source list of the action node then
5:                 return all the nodes present in object source list
6:             end if
7:         elseIf query is for object
8:             if object is in object sorce list of the action node then
9:                 return all the nodes present in the user source list
10:            end if
11:        end if
12:    end if
13: end for
```

The search process for the existence of the object in a object list and a user in a user list is done by using Binary Search which reduces the complexity of the algorithm.

IV. Analysis and Discussion

A. Complexity Evaluation

Let us consider, U represents the number of user nodes, O represents the number of object nodes, OP represents the number of operation nodes, n is the total number of nodes in the authorization graph and m is the total number of edges. Also, u is a user , op is an operation and o is an object.

1) Indexing: As mentioned in the algorithm we are performing depth first search through out the graph. Therefore, the complexity of this process is equal to the complexity of depth first search which is O(n+m). In the worst case,the complexity can be upto $O(n^2)$. The complexity of indexing is quadratic but since it is done before the access request is presented we do not consider this in our evaluations.

2) Evaluating $< u, op, o >$: For this query request, we pull the property of two nodes, which is O(1) for Neo4j. The property being extracted is the sorted list of operation nodes or common predecessor. An intersection is performed in these list which would have a less complex than a regular intersection. The complexity at the worst case would be $O(\mid OP \mid)$.

3) Evaluating $< u, *, * >$: The process is started with getting all the operation nodes in the system. Each operation nodes are indexed with list of object nodes and list of user nodes through which it is reachable. The list are distinct and sorted. We perform a binary search on the user list and return the object list if, the node we are searching for is present in the list. The time complexity for the search process is $O(log(\mid U \mid))$ and since this is done for each OP node in the graph, the total complexity is $O(|OP| log(|U|))$

V. Conclusion and Future Work

References

[1] D. Ferraiolo, V. Atluri, and S. Gavrila, "The policy machine: A novel architecture and framework for access control policy specification and enforcement," *Journal of Systems Architecture*, vol. 57, no. 4, pp. 412–424, 2011.

[2] P. Mell, J. M. Shook, and S. Gavrila, "Restricting insider access through efficient implementation of multi-policy access control systems," in *Proceedings of the 8th ACM CCS International Workshop on Managing Insider Security Threats*. ACM, 2016, pp. 13–22.

[3] J. Fagerberg, L. Kurkinen, and Berg Insight. mHealth and Home Monitoring. Accessed: 2018-01-02. [Online]. Available: http://www.berginsight.com/reportpdf/productsheet/bi-mhealth5-ps.pdf

[4] S. Adibi, *Mobile Health: A Technology Road Map*. Springer Publishing Company, Incorporated, 2015.

[5] S. Avancha, A. Baxi, and D. Kotz, "Privacy in mobile technology for personal healthcare," *ACM Comput. Surv.*, vol. 45, no. 1, pp. 3:1–3:54, Dec. 2012.

[6] D. Kotz, "A threat taxonomy for mhealth privacy," in *Proceedings of the 3rd International Conference on Communication Systems and Networks*. IEEE, Jan 2011, pp. 1–6.

[7] P. Kulkarni and Y. Ozturk, "mphasis: Mobile patient healthcare and sensor information system," *Journal of Network and Computer Applications*, vol. 34, no. 1, pp. 402–417, 2011.

[8] A. Karp, H. Haury, and M. Davis, "From abac to zbac: The evolution of access control models," in *Proceedings of the 5th International Conference on Information Warfare and Security 2010*. Academic Conferences Ltd, 2010, pp. 202–211.

[9] S. Mukherjee, I. Ray, I. Ray, H. Shirazi, T. Ong, and M. G. Kahn, "Attribute based access control for healthcare resources," in *Proceedings of the 2Nd ACM Workshop on Attribute-Based Access Control*, ser. ABAC '17. ACM, 2017, pp. 29–40.

[10] R. Lu, X. Lin, and X. Shen, "Spoc: A secure and privacy-preserving opportunistic computing framework for mobile-healthcare emergency," *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, no. 3, pp. 614–624, March 2013.

[11] V. C. Hu, D. Ferraiolo, R. Kuhn, A. R. Friedman, A. J. Lang, M. M. Cogdell, A. Schnitzer, K. Sandlin, R. Miller, K. Scarfone *et al.*, "Guide to attribute based access control (abac) definition and considerations (draft)," *NIST special publication*, vol. 800, no. 162, 2013.

[12] M. Toahchoodee, I. Ray, and R. M. McConnell, "Using graph theory to represent a spatio-temporal role-based access control model," *International Journal of Next-Generation Computing*, vol. 1, no. 2, 2010.

[13] R. D. Furberg, "Self-generated Fitbit dataset 10.22.2011-09.20.2014," Feb. 2015. [Online]. Available: https://doi.org/10.5281/zenodo.14996

[14] B. Martínez-Pérez, I. De La Torre-Díez, and M. López-Coronado, "Privacy and security in mobile health apps: a review and recommendations," *Journal of medical systems*, vol. 39, no. 1, p. 181, 2015.