A Project Report

On

Constrained Policy Mining in Attribute Access Control Models

BY

GUMMA VARUN

2017A7PS0165H

Under the supervision of

Dr. BARSHA MITRA

SUBMITTED IN PARTIAL FULLFILLMENT OF THE REQUIREMENTS OF

CS F376: DESIGN ORIENTED PROJECT

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE PILANI (RAJASTHAN)

HYDERABAD CAMPUS

(November 2019)

ACKNOWLEDGMENTS

I would like to thank Dr. Barsha Mitra, our guide, for helping us through this project; clearing our doubts and aiding us understand the concepts given in the reference materials.

Birla Institute of Technology and Science-Pilani,

Hyderabad Campus

Certificate

This is to certify that the project report entitled **Constrained Policy Mining in Attribute Access Control Models** submitted by Mr. GUMMA VARUN (ID No. 2017A7PS0165H) in partial fulfillment of the requirements of the course CS F376, Design Oriented Project Course, embodies the work done by him under my supervision and guidance.

Date:

Dr. BARSHA MITRA

BITS- Pilani, Hyderabad Campus

ABSTRACT

- With an ever-increasing number of roles and objects, access to documents and resources ought to be regulated. These regulations are implemented in the form of policies, a set of generalized rules that verify/grant/deny access to resources based on the features associated with the resources and requester. However, as the number of attributes (or features) associated with the user and object may run into vast numbers, we attempt to cap the *weight of the rule*, which will allow us to have an optimal execution time and space complexity.
- In this report, we attempt to analyze one such solution, and make further improvements on it by analyzing *attribute hierarchy* and *mutually exclusive attributes*, given an ACM matrix along with User-Attribute matrix, Object-Attribute matrix, User-Attribute-Assignment matrix and Object-Attribute-Assignment matrix.
- In extension, we will also have a look at a few algorithms that will help us *minimize the number of rules* in the policy along with the *weight of the whole policy*.

CONTENTS

Contents

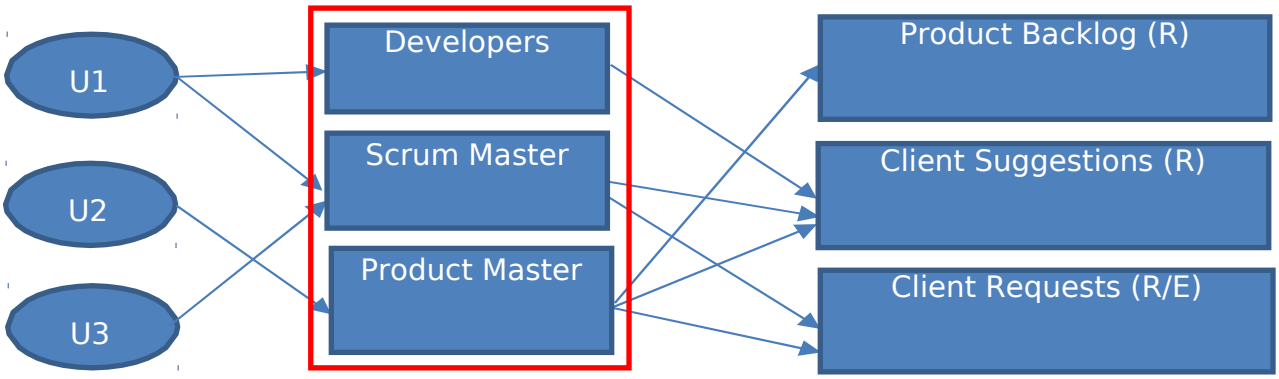
1. ROLE BASED ACCESS CONTROL MODELS

In Role Based Access Control Models, the access to resources is dictated by groups of users having same authorization. Each user is associated with a role and can access resources that reflect the same. RBACs are particularly useful while functioning in an organization, as they control decisions on the function a user is allowed to perform in the organization.

For example, in a Scrum Development system, let us consider three groups of users, *Developers*, *Scrum Master* and *Product Owner*. Now, let us consider the following rules that exist in the organization:

- Client suggestions can be viewed by anyone.
- Client requests can be viewed by the Scrum Master or Product Owner.
- Product backlog can only be viewed/edited by the Product Owner.

With the above given rules, we can easily deduce a many-to-many mapping between the users and resources, as the roles act as intermediaries in the transaction.



Most of the organizations follow a role hierarchy, which can even be implemented in ABACs, as each role is translated into an attribute. This hierarchy can be represented as a partial order (which can be extended to a total order further) or a single inheritance in Object Oriented Programming.

2. ATTRIBUTE BASED ACCESS CONTROL MODELS

In ABACs, one or more attributes associated with the user are used to determine the access to an object. Unlike roles, which need to be associated with each individual, attributes are intrinsic to each user and are very dynamic. Moving from a RBAC model to an ABAC model is relatively simpler as each role can be put down as the user's attributes. However, to implement an ABAC Model, we need have an exhaustive enumeration of all possible user and object attributes.

2. 1 Policy Mining

As attributes alone are not sufficient to express authorizations, they need to be associated with policies. By combining the attributes together, policies generate positive/negative cases. These policies are generalized set of rules that aid in authentication and oversee a minimal change with the onset of new attributes. Policy mining refers to the generation of these rules from a given ACM, UAL and OAL.

2.2 Constrained Policy Mining

As the number of rules increases significantly with onset of new object/user attributes, the space and time complexity associated with the mining algorithm proportionally increases. Hence, we try to cap the elements in the rules (weight of the rule) yet retaining all major rule information. For example, $\langle [designation = IC],\ [evaluationType = Mids],\ RWE \rangle$ has a weight 3. By minimizing the weight of all rules, we can prove that the weight of the whole policy given by $TW(P) = \sum_{r \in P} W(r)$ is reduced.

2.3 Terminologies associated with CAPM

- *ACM Matrix:* This matrix stores the permissions associated (RWE encoded as 3 bit binary numbers ranging from 000 to 111, i.e. 0-7) for the user U_i and object O_j . It is represented as a tuple $\langle U, O, p \rangle$.
- *User-attributes (U_a):* Attributes associated with a user. (department, designation)
- *Object-attributes (O_a):* Attributes associated with an object. (colour, shape, evaluation)
- *Set of User-attribute-name-value pairs (A_u):* (department = CS, designation = TA)
- *Set of Object-attribute-name-value pairs (A_o):* (evaluation = Online, testType = quiz)
- *UAV(u, a):* Denotes the value of the attribute a, for a given user v.
- *OAV(o, a):* Denotes the value of the attribute a, for a given object o.

2.4 Attribute hierarchy in ABAC

As all roles can be implemented as attributes, even attributes follow a hierarchical model or are mutually exclusive. This hierarchy can now be represented as partial order or by extension a total order. For instance, FDTA ≤ HDTA ≤ Instructor ≤ IC. Some mutual exclusion dictated by this partial order is:

- A FDTA/HDTA cannot be a IC/Instructor.
- A FDTA cannot be a HDTA.
- An IC cannot be FDTA or HDTA.

We need to note that we cannot conclude that an Instructor cannot be a IC or vice-versa. Many a times, this hierarchy in the attributes is dictated by the organizational setup and must be specified before along with the attributes. During the construction of the rules, many rules which can be deemed *semantically incorrect* can be neglected which reduces the computational overhead for further steps.

2.5 Extension to the CAPM Algorithm (Mayank Gautam,2017)

Let us consider the following scenario:

User: (3 users: U0, U1, U2)

- *attributes & values:*
  - designation: [IC, Instructor, HOD, TA]
  - department: [CS, EEE, ECE, Civil]

Object: (3 objects: O1, O2, O3)

- *attributes & values:*
  - assignments: [Offline, Online]
  - answer_scripts: [Quiz, Mids, Compre]

Permissions:

- [Read, Write, Evaluate]

In this case, the ACM will look like, in which O1 refers to a CS course, O2 refers to a ECE course and O3 refers to a EEE course.

	O0	**O1**	**O2**
U0	1	7	2
U1	7	1	0
U2	1	0	0

Here each permission is the decimal representation of binary RWE (000 – 111). Now the User- Attribute name value pairs and Object-Attribute name value pairs are given as:

	IC	**Instructor**	**HOD**	**TA**	**CS**	**ECE**	**EEE**	**Civil**
U0	1	1	1	0	0	1	0	0
U1	0	1	0	0	1	0	0	0
U2	0	0	0	1	1	0	0	0

The following partial orders hold the User-Attribute list:

- TA ≤ IC
- TA ≤ Instructor
- TA ≤ HOD

	offline	online	quiz	mids	Compre
O0	1	0	1	0	0
O1	0	1	0	0	1
O2	0	0	0	1	0

	IC	Instr	HOD	TA	CS	ECE	EEE	Civil	off	on	quiz	mids	Compre	P
R0	1	0	0	0	0	0	0	0	1	1	0	1	1	2
R1	1	1	1	1	1	1	0	0	0	0	0	1	0	4
R2	1	1	1	0	1	0	0	0	1	0	0	0	1	7
R3	1	1	1	0	1	0	0	0	1	0	1	0	0	7
														..

In the above table:

- Offline, online are mutually exclusive.
- All types of exams (quiz, mids, compre) are mutually exclusive
- quiz ≤ mids ≤ compre

In this case, $A_u \; U \; A_o \; U \; p$ (TotalAttrSet) has 16 (8 + 5 + 3) entries. $\boldsymbol{p}$ can be expanded to 3 rows, RWE. The number of rows in the above table with no mutual exclusion or attribute hierarchy would be $2^{16} = 65536$. The number of columns in the above table will be 16.

The time complexity after the modification still remains $\Theta(2^N)$, where N = (no. of columns of User-Attribute name-value pairs + no. of columns of Object-Attribute name-value pairs + no. of permissions)

Now, using mutual exclusion and attribute hierarchy:
$EffectiveRules \leftarrow \; union \; of \; all \; ValidRules$

R0: Semantically incorrect/doesn't represent a correct rule (an exam cannot be offline and online at the same time)
R1: Does not represent a correct rule (a user cannot be IC and TA at the same time)
R2, R3: Represent proper rules as per ACM (ValidRules)

We notice that even after applying all constraints and mutual exclusions on the set of all possible rules, the time complexity still remains to be $\Theta(2^N)$, as we have to initially run through all possibilities, while eliminating rules with greater weights and semantically incorrect rules. This statement is also in accordance with the fact that CAPM is a *NP-Hard* problem. However, the space complexity may drastically reduce from $(N \times 2^N)$ with the advent of a stiffer partial ordering in the attributes.

3. ADDITIONAL ALGORITHMS FOR CAPM:

3.1 Naïve Elimination Algorithm:

In this method, we try to minimize the weight of each rule and by extension, the total weight of all the rules by eliminating those attributes from rules *(setting them to zero)* which do not contribute to the permission level associated to that rule. For that, in the given rule, we set each non-zero bit to zero and check if the permission level still holds, i.e. find a similar rule in the ACM with same or higher permission. This approach will not reduce the weight of each rule by much, as it requires similar rules to be present in the ACM, which might not always be feasible.

3.2 Rule Elimination Algorithm:

In this procedure, the following steps are applied sequentially. A rule is formed by fusing an entry of the UAV and OAV and appending the appropriate permission as per the ACM. Hence, initially we have (N × M) rules. This method allows us to reduce the weight of the whole policy by removing unnecessary/redundant rules:

- Firstly, the rules are divided into clusters or blocks, with each block representing all the available options/values for an attribute. For example, consider the rule 100011110101011000010010 where the first 6 bits represent the values the first attribute (designation) can take, while the next 4 bits represent the different values the second attribute (department) can take (CS, ECE, EEE, Civil).
- Secondly, any duplicate rules are removed.
- Next, if two rules in the list have same permission, but differ by exactly one bit in the attribute values, we can consider that condition to be a *don't-care* and remove the rule with the higher weight. This completes the preprocessing.
- In the next step, we try to merge the rules that differ in *exactly one block*. For this, we examine every pair of rules, and if the necessary condition is satisfied, we can **OR** both the rules to generate the new merged rule and delete the two. This goes well in hand with the hierarchy and mutual exclusion mentioned above. For instance, the *ordered* rule has the information in the following fashion:
  HOD or **Ph.D TA** can **Read** the **Quiz** papers of **their department**.
  It is necessary that the two rules to be merged have the exactly same permission.

3.2 Weight Minimization Algorithm:

This is a very rudimentary approach to weight minimization as suggested *Mayank Gautam et al.* In this procedure, we try to discard a few rules that have weight greater than a certain threshold percent. Of course, this might lead to removal to rules that represent outliers or multiple merged rules. One way to minimize this error margin is to cap the number of times a rule can undergo merging, as multiple merging increase the weight of a rule. This cap on the *merge count* can be fine-tuned by checking for outliers while minimizing the average rule weight and number of tuples that cannot be covered by the policy.

4. RESULTS AND OBSERVATIONS:

The time complexity of the major functions associated with this algorithm are as follows:

1. Pre-processing: $O(N^4 \times M^4)$
2. Merging: $O(N^2 \times M^2 \times rule\_size)$
3. Minimizing: $O(N \times M \times rule\_size)$

From the above three, can conclude that the worst-case time complexity for this whole algorithm will be $O(N^4 \times M^4)$.

Sr.no	users	UA	objects	OA	Step-1: total rules	Step-1: total weight of rules	Step-1: avg. weight of rules	Step-2: total rules	Step-2: total weight of rules	Step-2: avg. weight of rules	Time
1	200	27	40	25	7997	168302	21.0456	7997	168302	21.0456	38s
2	350	27	75	25	26233	618023	23.559	26233	618023	23.559	508s
3	450	27	100	25	44960	1020563	22.6994	44960	1020563	22.6994	1365s
4	150	27	35	25	5247	108479	20.6745	5247	108479	20.6745	16s
5	500	27	75	25	37465	786615	20.996	37465	786615	20.996	1156s

This table depicts the number of rules generated after step-1 and step-2 of processing. We observe that a few rules are eliminated during the pre-processing step, but no rules are eliminated in the merging step, as step-2 requires a one-block variation between two rules, which was not found in our dataset. A side effect that springs up after step-2 is that the average weight of a rule in the policy increases. This is because there is a reduction in the number of rules, but the total weight of all the rules *(total number of ones in the policy)* almost remain the same. The weight minimization algorithm could not be applied to the dataset as no merging is observed. Better results using the weight minimization algorithm can be obtained when multiple rule merging take place. With these algorithms, we see that, the conditions for CAPM is satisfied and satisfactory results are obtained.

5. CONCLUSION

Until now, we have been able to propose a rudimentary modification and extension to the algorithm given by *Mayank Guatam et al*, which allows us to reduce the number of entries in the final policy matrix before merging the rules, thus reducing its complexity. Even after the modification, we observe that the *Constrained Role Mining Problem* is still a *NP-Hard* problem. As the partial ordering in the attribute list increases, each role can be viewed more atomically and merging them would be easier in the next step.

All the above algorithms were coded in in C++, with optimal use of data structures like Vectors and Pairs. The results obtained in our case are limited by time and datasets (which were generated by using another C++ program and randomly assigning bits and values), but are promising as the results do show the intended output of policy size and weight reduction. Better results can be obtained by using this program on real life datasets of organizations, which have hierarchical setup.

6. REFERENCES

- Mayank Gautam, Sadhana Jha, Shamik Sural, Jaideep Vaidya, Vijayalakshmi Atluri Poster: *Constrained Policy Mining in Attribute Based Access Control (2017)*
- David F. Ferraiolo, D Richard Kuhn: *Role-Based Access Controls (1992)*
- Ed Coyne, Timothy R. Weil: ABAC and RBAC: *Scalable, Flexible and Auditable Access Management (2013)*
- Zhongyuan Xu, Scott D. Stoller: *Algorithms for Mining Meaningful Roles (2012)*
- Zhongyuan Xu, Scott D. Stoller: *Mining Attribute-Based Access Control Policies (2015)*
- David Brossad, Gerry Gebel, Mark Berg: *A Systematic Approach to Implementing ABAC*
- Zhongyuan Xu, Scott D. Stoller: *Mining Attribute-Based Access Control Policies from Logs (2014)*
- Yan Zhu, Dijiang Haung, Chang-Jyun Hu, and Xin Wang: *From RBAC to ABAC: Constructing Flexible Data Access Control for Cloud Storage Services (2015)*
- Tanay Talukdar, Gunjan Batra, Jaideep Vaidya, Vijayalakshmi Atluri, and Shamik Sural: *Efficient bottom-up Mining of Attribute Based Access Control Policies (2017)*