Graduate Program in Information Science and Telecommunications and Networking
School of Computing and Information
University of Pittsburgh

Lab: Unsupervised Learning Techniques for Extracting ABAC policies of a Healthcare Environment

Version 1.1

Last Edited: 3/1/2019

This lab is designed by Leila Karimi and is part of lab development module of SAHI Project

Goal

With the rapid advances in computing and information technologies, traditional access control models have become inadequate in terms of capturing fine-grained, expressive security requirements of newly emerging applications. An attribute-based access control (ABAC) model provides a more flexible approach for addressing access control needs of complex and dynamic systems. While organizations are interested in employing such newer access control approaches, the challenge of migrating to such AC approaches pose as a significant obstacle.

In this lab, the students will learn how to automatically extract ABAC policy rules from access logs of the system. The students will employ an unsupervised learning-based algorithm for detecting patterns in access logs and extracting ABAC authorization rules from these patterns.

Tools

To apply an unsupervised machine learning algorithm on the given dataset, the students are allowed to use a programming language and a machine learning tool of their choice. However, this lab is designed based on Java programming language and Weka library.

Setup

Before you start this lab, you need to setup the Java programming environment. You can install eclipse as your IDE. You also need to integrate Weka API with eclipse. Here are several links which provide the instructions of their installation and setup.

- Official website of Java
    - https://www.java.com/en/
- Official website of Weka
    - https://www.cs.waikato.ac.nz/ml/weka/
- Eclipse Tutorial:
    - https://www.tutorialspoint.com/eclipse/
- Weka with Java (Eclipse), Getting Started
    - https://ianma.wordpress.com/2010/01/16/weka-with-java-eclipse-getting-started/

Read the following guidelines before working in the lab

General Guidelines

During this lab, you will work on a data set of access control tuples which is provided to you. Each record of this data set contains an information related to one access request and the decision of the system about this access request. You can assume that the original authorization policy of the system is based on an RBAC model and you want to extract an ABAC policy corresponding to the original policy.

We separated access records based on the decision of the system to two files "permitted.txt" and "denied.txt". The file permitted.txt includes the access requests that resulted in "permit" decision based on the original authorization policy of the system and the file denied.txt includes the access request that resulted in "deny" decision.

The following set shows all attributes in our healthcare environment, and each line in the access request files contains an attribute value corresponding to these attributes for an access request.

Attributes = {action, role, type, oward, uward, team, treatingTeam, patient, author, topic, specialty, agentFor, user}

So a line in the permitted.txt file can be as follows:

{read, nurse, hrItem, carWard, carWard, oncteam2, oncteam1, carpat1, carnurse1, oncology, oncology, carpat1, carnurse1}.

Here, "read" is the attribute value corresponding to attribute "action" and "nurse" is an attribute value of attribute "role".

The possible attribute values for attributes of the system are as follows:

Action: {additem, addnote, read}

Role: {nurse, doctor}

Type: {hritem, hr}

Oward: {oncward, carward}

Uward: {oncward, carward}

Team: {oncteam1, oncteam2, carteam1, carteam2}

TreatingTeam: {oncteam1, oncteam2, carteam1, carteam2}

Patient: {oncpat1, oncpat2, carpat1, carpat2}

Author: {oncnurse1, oncnurse2, carnurse1, carnurse2, oncdoc1, oncdoc2}

User: {oncnurse1, oncnurse2, carnurse1, carnurse2, oncdoc1, oncdoc2, oncpat1, oncpat2, carpat1, carpat2}
Topic: {note, nursing, oncology, cardiology}
Specialty: {note, nursing, oncology, cardiology}
Agentfor: {oncpat1, oncpat2, carpat1, carpat2}

The original authorization policy of the system can be represented through following rules:
Rule1: <action=additem; role=nurse; type=hr; oward=oncward; uward=oncward >
Rule2: <action=additem; type=hr; team= oncteam1; treatingTeam=oncteam1>
Rule3: <action=addNote; type=hr; patient=oncpat2; user=oncpat2>
Rule4: <action=addNote; type=hr; patient=oncpat1; uagentfor=oncpat1>
Rule5: <action=read; type=hritem; author= carnurse1; user=carnurse1>
Rule6: <topic=nursing; action=read; role=nurse; type=hritem; oward=carward; uward=carward>
Rule7: <action=read; type=hritem; team=carteam1; treatingTeam=carteam1; topic=cardiology; specialty=cardiology>
Rule8: <topic=note; action=read; type=hritem; team=carteam2; treatingteam=carteam2; patient=carpat1; user= carpat1>
Rule9: <topic=note; action=read; type=hritem; patient=oncpat2; agentfor=oncpat2>

Please read the following paper thoroughly before starting the lab as it will help you understand the approach better:
Karimi, Leila, and James Joshi. "An Unsupervised Learning Based Approach for Mining Attribute Based Access Control Policies." *2018 IEEE International Conference on Big Data (Big Data)*. IEEE, 2018.

Part I: Preliminary Questions

Q1-1: Answer the following questions:

a) In your own word, explain what Role Based Access Control (RBAC) is?

b) In your own word, explain what Attribute Based Access Control (ABAC) is?

c) What are some limitations of RBAC?

d) Explain how ABAC helps with overcoming the limitations of RBAC?

e) Explain different components of an ABAC architecture.

f) In your own word, explain what an unsupervised learning method is?

g) What are the common unsupervised learning methods?

h) What are the applications of unsupervised learning approaches?

i) In your own word, explain what are the definitions of false positive, true positive, false negative, and true negative rates?

____

____

j) In your own word, explain what are the definitions of recall, precision and accuracy?

____

____

Part II: Pre-processing the Data

Before applying Machine Learning techniques to the problem, we need to pre-process the data. In this step, you handle missing attribute values of the dataset.

Q2-1What are the common approaches for handling missing values in ML methods?

Later on in this lab, you will extract the attribute expressions of authorization rules based on the frequency of attribute values in the dataset. Hence, the actual frequency of each attribute value is important. As a result, the common approaches for handling missing values are not applicable to this problem. We suggest to replace all missing values of attributes with a phrase that's not part of the dataset. For example, you can use "UNK" (i.e. an abbreviation of *unknown*) as a replacement for all missing values.

Q2-2 Write a piece of code to traverse all records in the dataset (both files) and replace missing values with "UNK".

Part III: Clustering the Data

Choice of Clustering Algorithm

All the attributes in the given dataset are categorical attributes. Knowing this, answer the following questions:

Q3-1 Is k-mean applicable to the given data set? Explain your answer.

Q3-2 What other clustering algorithm can be used for the given dataset?

Note: The K-mean algorithm of Weka library are able to handle categorical attributes as well as the numerical one.

Tuning the Parameters

One of the main challenges in applying unsupervised learning algorithms specifically the clustering methods is determining the number of clusters, *k*. (In the following questions, consider "permitted.txt" file as your learning data)

Q3-3 What are the common approaches for finding the best number of k in clustering methods? Explain three methods for finding the best k.

Q3-4 Write a piece of code to cluster the given data varying the number of clusters from k = 1, …, 15 (Consider your choice of algorithm in Q3-2).

Q3-5 For each k in Q3-4, calculate the total within-cluster sum of square (wss). Draw the Elbow Method graph showing wss based on number of clusters (k) for the given dataset.

Q3-6 Does Elbow Method clearly show the optimal k for the given dataset?

Q3-7 For each k in Q3-4, calculate the average silhouette of observations. Plot the curve of Average Silhouette according to the number of clusters k for the given dataset.

Q3-8 Does the curve of Average Silhouette reveal the optimal number of k?

Q3-9 Based on your answers to Q3-5 to Q3-8, what do you suggest as the optimal number of cluster for the given dataset?

Q3-10 Assume k=13 is the optimal k for the given dataset. Cluster the data using an appropriate clustering algorithm and report the following values:

a) the total within-cluster sum of square

b) the average silhouette of observations

c) the size of each cluster

d) ***the attribute values of centroid of each cluster***

Part IV: Extracting the Authorization Rules

The records in each of the clusters you built in the previous step correspond to one authorization rule in the system. In this step, you will extract the attribute expressions of such rules considering the definition of *Effective Attributes*:

Definition 1 *(**Effective Attribute**). Let c_i be the centroid of cluster i and $S_{c_i} = \{< a, v >\}$ be the set containing all possible user, object, and subject attributes in the system and their corresponding values for c_i; we define a_j as an* effective attribute *of ρ_i corresponding to cluster i, where $(a_j, v_j) \in S_{c_i}$, and the difference between the frequency of v_j in all points of cluster i and its distribution in the original data set is higher than a threshold $\mathcal{T}$. v_j is an effective value of a_j. The attribute expression $< a_j, v_j >$ will be added to the extracted rule ρ_i corresponding to this cluster.*

Q4-1 Write a piece of code to extract the attribute expressions for each cluster found in previous step.

Note: Set $\mathbf{T} = \mathbf{0.35}$ as your threshold. So if the relative frequency of an attribute value in one cluster is 35 % higher than its relative frequency in the original data set, this attribute value is effective and will be added as attribute expression to the corresponding rule.

Q4-2 What are your final set of rules? Write them in the following format:

$$rule1{:} < role = doctor; type = hr; action = read;\ specialty = oncology; topic = oncology >$$

Part V: Evaluating the Extracted Authorization Rules

In this step you will compare the extracted authorization rules with the original one and check how well the extracted policy represents the original access control policy of the system. For this purpose, you calculate the accuracy of the extracted policy based on the decisions of the original one.

Q5-1 Write a piece of code to determine the decision of a policy (set of AC rules) for a given access request.

Note: The decision of an authorization policy for a given access request is permit if at least the decision of one of its rule for the given access request is permit, otherwise the decision is deny.

Note: The decision of an AC rule for a given access request is permit if the access request includes all the attribute expressions of the AC rule.

Q5-2 Considering your extracted access control rules, determine the decision of them on the records of the permitted file and store the records that resulted in "deny" based on your extracted policy in a separate file.

Q5-3 Are these records considered as FP records or FN records? Explain your choice.

Q5-4 Considering your extracted access control rules, determine the decision of them on the records of the denied file and store the records that resulted in "permit" based on your extracted policy in a separate file.

Q5-5 Are these records considered as FP records or FN records? Explain your choice.

Q5-6 Based on the TP, FP, TN, and FN rates that you found in the previous steps, calculate the accuracy of the extracted policy according to the original authorization policy.

Another metric for evaluating the extracted policy is measuring its complexity. The complexity of an access control policy is measured through Weighted Structural Complexity (WSC) which is a generalization of policy size and can be calculated as follows:

$$WSC(\pi) = WSC(R)$$

$$WSC(R) = \sum_{\rho \in R} WSC(\rho)$$

$$WSC(\rho = < uaf, oaf, saf, op, d >) = w_1 WSC(uaf) + \\ w_2 WSC(oaf) + w_3 WSC(saf)$$

$$\forall f \in \{uaf, oaf, saf\} : WSC(f) = \sum |f|$$

Q5-7 Using the above formula, calculate the WSC of the original policy as well as the extracted policy.

Q5-8 Do you notice an opportunity to improve the complexity of the mined policy without compromising much of its accuracy?

Part VI: Pruning the Rules

During the learning process, it is possible to have two clusters that correspond to the same rule. In other words, the rules extracted from two clusters are similar to each other or one of the rules is a subset of the other one. A subset relation between two rules is defined as follows:

Definition 9: (**Rule Subset Relationship**) Rule ρ_1 is a subset of rule ρ_2, written as $\rho_1 \subseteq \rho_2$, iff:

$$d_{\rho 1} = d_{\rho 2} \wedge op_{\rho 1} = op_{\rho 1} \wedge$$
$$uaf_{\rho 1} \subseteq uaf_{\rho 2} \wedge oaf_{\rho 1} \subseteq oaf_{\rho 2} \wedge saf_{\rho 1} \subseteq saf_{\rho 2}$$

So, having both ρ_1 and ρ_2 in an AC system is redundant, as ρ_2 will be ineffective. This will result in having more FPs which we want to minimize. To exclude possible FPs, you have to prune the extracted rule set by eliminating the more relaxed rule (i.e.ρ_1).
As an example, consider the following two rules:

$$\rho_1 =< \{(position, nurse)\},\{(type, hritem)\}, \{\}, \{write\}, permit >$$
$$\rho_2 =< \{(position, nurse),(team, Onc1)\}, \{(type, hritem)\},\{\}, \{write\}, permit >$$

Here ρ1 is a subset of ρ2. ρ2 is more restricted as it imposes more conditions on the user attributes. To avoid possible FPs, we prune ρ1from the extracted rule set.

Q6-1 Look at the set of rules you've extracted in previous steps. Is there any rule that can be pruned? Explain it.

____

____

____

Q6-2 What are your final set of rules after rule pruning?

____

____

____

____

Q6-3 What is the complexity of your policy (WSC) after rule pruning?

Part VII: Refining the Rules

As you see in the previous steps, the extracted policy may not completely match the original policy of the system. This results in sets of false positive and false negatives. In this step, you use these FPs and FNs to refine your policy. For this purpose, you need to extract the rules corresponding to FPs and FNs. The steps are similar to Part IV.

Q7-1 Considering the false positive records you stored in Part V, cluster them into two clusters and extract the rule for each cluster. We refer to these rules as false positive rules.

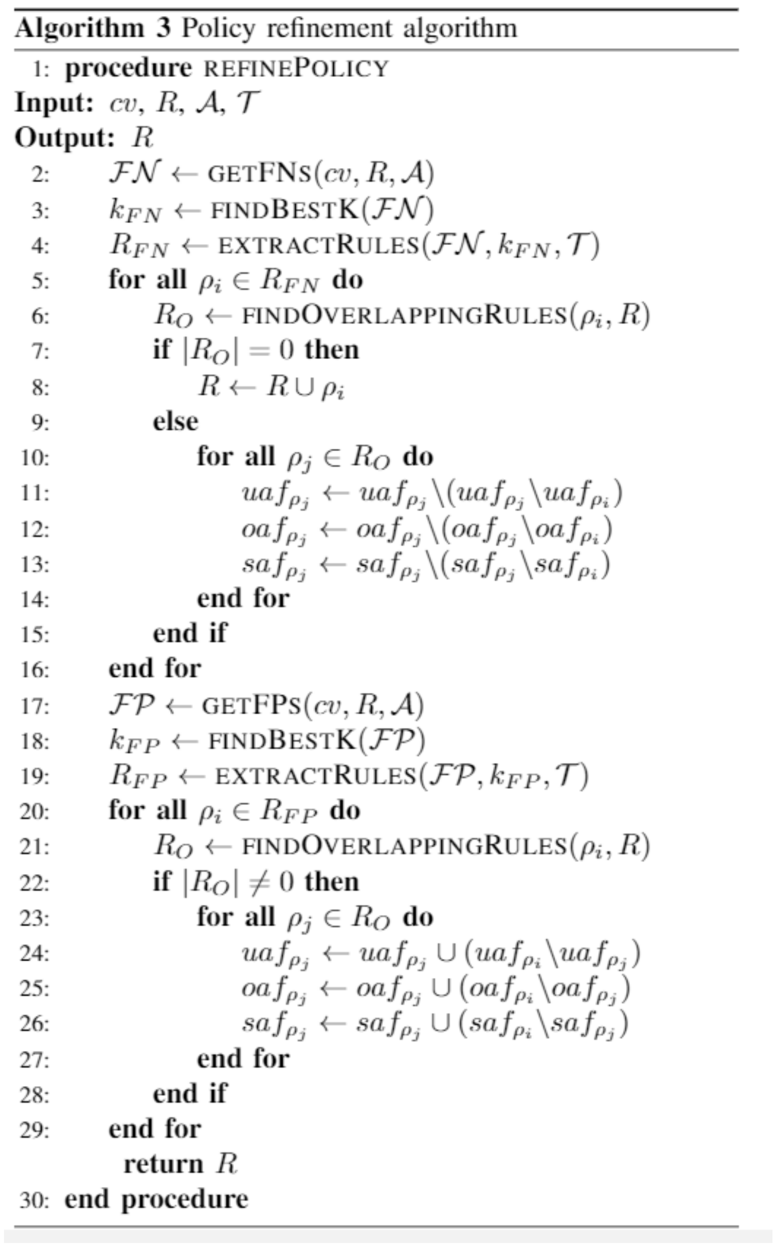
Q7-2 Considering the false negative records you stored in Part V, cluster them into two clusters and extract the rule for each cluster. We refer to these rules as false negative rules.

Q7-3 Refine your mined policy sets based on false positive and false negative rules according to Algorithm 3.

Q7-4 What are your final set of rules after rule refinement?

Q7-6 What is the accuracy of your policy after rule refinement? Calculate it according to the original policy as you've done in Part V.

Q7-6 What is the complexity of your policy (WSC) after rule refinement?



Algorithm 3 Policy refinement algorithm

```
procedure REFINEPOLICY
Input: $cv$, $R$, $\mathcal{A}$, $\mathcal{T}$
Output: $R$
    $\mathcal{FN} \leftarrow$ GETFNS$(cv, R, \mathcal{A})$
    $k_{FN} \leftarrow$ FINDBESTK$(\mathcal{FN})$
    $R_{FN} \leftarrow$ EXTRACTRULES$(\mathcal{FN}, k_{FN}, \mathcal{T})$
    for all $\rho_i \in R_{FN}$ do
        $R_O \leftarrow$ FINDOVERLAPPINGRULES$(\rho_i, R)$
        if $|R_O| = 0$ then
            $R \leftarrow R \cup \rho_i$
        else
            for all $\rho_j \in R_O$ do
                $uaf_{\rho_j} \leftarrow uaf_{\rho_j} \backslash (uaf_{\rho_j} \backslash uaf_{\rho_i})$
                $oaf_{\rho_j} \leftarrow oaf_{\rho_j} \backslash (oaf_{\rho_j} \backslash oaf_{\rho_i})$
                $saf_{\rho_j} \leftarrow saf_{\rho_j} \backslash (saf_{\rho_j} \backslash saf_{\rho_i})$
            end for
        end if
    end for
    $\mathcal{FP} \leftarrow$ GETFPS$(cv, R, \mathcal{A})$
    $k_{FP} \leftarrow$ FINDBESTK$(\mathcal{FP})$
    $R_{FP} \leftarrow$ EXTRACTRULES$(\mathcal{FP}, k_{FP}, \mathcal{T})$
    for all $\rho_i \in R_{FP}$ do
        $R_O \leftarrow$ FINDOVERLAPPINGRULES$(\rho_i, R)$
        if $|R_O| \neq 0$ then
            for all $\rho_j \in R_O$ do
                $uaf_{\rho_j} \leftarrow uaf_{\rho_j} \cup (uaf_{\rho_i} \backslash uaf_{\rho_j})$
                $oaf_{\rho_j} \leftarrow oaf_{\rho_j} \cup (oaf_{\rho_i} \backslash oaf_{\rho_j})$
                $saf_{\rho_j} \leftarrow saf_{\rho_j} \cup (saf_{\rho_i} \backslash saf_{\rho_j})$
            end for
        end if
    end for
     return $R$
end procedure
```