THESIS

AUTOMATED EXTRACTION OF ACCESS CONTROL POLICY FROM NATURAL LANGUAGE DOCUMENTS

Submitted by

Saja Alqurashi

Department of Computer Science

In partial fulfillment of the requirements

For the Degree of Master of Science

Colorado State University

Fort Collins, Colorado

Fall 2023

Master's Committee:

Advisor: Indrakshi Ray

Indrajit Ray
Yashwant Malaiya
Steve Simske

ABSTRACT

AUTOMATED EXTRACTION OF ACCESS CONTROL POLICY FROM NATURAL LANGUAGE DOCUMENTS

Data security and privacy are fundamental requirements in information systems. The first step to providing data security and privacy for organizations is defining access control policies (ACPs). Security requirements are often expressed in natural languages, and ACPs are embedded in the security requirements. However, ACPs in natural language are unstructured and ambiguous, so manually extracting ACPs from security requirements and translating them into enforceable policies is tedious, complex, expensive, labor-intensive, and error-prone. Thus, the automated ACPs specification process is crucial. In this thesis, we consider the Next Generation Access Control (NGAC) model as our reference formal access control model to study the automation process. This thesis addresses the research question: **How do we automatically translate access control policies (ACPs) from natural language expression to the NGAC formal specification?**

Answering this research question entails building an automated extraction framework. The proposed framework aims to translate natural language ACPs into NGAC specifications automatically. The primary contributions of this research are developing models to construct ACPs in NGAC specification from natural language automatically and generating a realistic synthetic dataset of access control policies sentences to evaluate the proposed framework. Our experimental results are promising as we achieved, on average, an F1-score of 93 % when identifying ACPs sentences, an F1-score of 96 % when extracting NGAC relations between attributes, and an F1-score of 96% when extracting user attribute and 89% for object attribute from natural language access control policies.

ACKNOWLEDGEMENTS

I would like to thank my advisor, Prof.Indrakshi Ray, for her guidance in accomplishing this project. I also would like to thank my committee members, Prof. Yashwant Malaiya, Prof. Indrajit Ray, and Prof. Steve Simske, for generously offering their time and guidance. I am grateful to Dr. Hossein Shirazi for his support and feedback. I also would like to thank the team member Videep Venkatesha, for his effort in this project.

Furthermore, I want to thank the funding agencies for supporting this work: this work was supported in part by funding from NSF under Award Numbers DMS 2123761, CNS 1822118, and from AFRL, ARL, Statnett, AMI, NewPush, and Cyber Risk Research and from NIST under Award Number 60NANB23D152.

DEDICATION

I would like to dedicate this thesis to my parents.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

Protecting sensitive data from unauthorized access or unauthorized modifications while at the same time ensuring their availability to legitimate users is a fundamental security requirement in any information system. Access control is one of the vital factors in information systems to ensure secure access of resources, systems and data in an organization. Access control is the process of monitoring the flow of information to prevent unauthorized access to ensure that each request of resources or data is controlled by a system able to make an appropriate decision to either grant or deny the request [31]. Thus, an access control system is the core of information systems that provide security and privacy for resources and data. Access control system involves various policies and policy models to address sophisticated security problems such as data integrity and data privacy. A complete access control system should provide the following functions [31]:

- **Authentication**: the process to ensure the identity of a user.
- **Authorization**: the process to specify access privileges to resources.
- **Accountability**: the process to trace activities on the protected system.

Developing an access control system that provides the previous functionality consists of two main components [31]:

- **Access Control Policy (ACP)** is a formal access request that includes rules associated with authorized users to manage access rights to pre-determined resources for tasks such as reading, writing, modifying, executing, deleting, and others.
- **Model** is a formal representation of access control policies.

Many studies proposed access control mechanisms, such as Discretionary Access Control (DAC), Mandatory Access Control (MAC), Role-Based Access Control (RBAC), and Attribute-Based Access Control Model (ABAC). However, those traditional models have shown inadequacy

in withstanding the increasing complexity of today's business requirements, such as dynamic environment requirements in the Internet of Things technologies (IoT). Next Generation Access Control (NGAC) is a flexible access control framework proposed by the National Institute of Standards and Technology (NIST). The NGAC is a fundamental reworking of traditional access control into a form suited to the needs of the modern and distributed interconnected enterprise [13]. Thus, we consider the NGAC model as our reference formal access control model to study the automation process.

The main obstacle to deploying NGAC is the precise development of NGAC policies. Because of the large number of business processes, users, and permissions in the system, manually developing access control policies is tedious, time-consuming and error-prone. Therefore, it is necessary to reduce these manual operations and automate the construction of policies.

Most organizations describe their security policy in natural language access control policy documents (NLACPs). These documents include security policy requirements in natural languages such as English. To implement access control systems, ACPs should be derived from NLACPs and then converted from natural language to formal specifications. There are two primary reasons behind incorrect ACPs specifications [41]. First, there are many attributes and implicit information in ACPs expression. Second, NLACP contains large numbers of sentences where a portion of these sentences are ACPs. Moreover, NLACPs contain unstructured, ambiguous, and implicit information. Thus, manual ACP extraction is tedious, complex, expensive, labor-intensive, and error-prone.

The following is an example of a security requirement obtained from a university.

- *"DoIT has the authority, at its discretion, to scan any computers connected to the University's network without explicit permission from the computer's owner, operator or system administrator. DoIT shall use reasonable and prudent measures to inform subnet managers of the scope and nature of scans that are to be done. Departmental IT staff may develop policies and procedures for scanning their own systems. Except as noted above, no one is*

authorized to scan systems they do not own or administer without prior, written approval from departmental officials at an appropriate level [36]."

Given this requirement, a security architect should extract sentences that are access control policies (ACP). Then, the security architect extracts elements of access control policies such as user, resource, and operation. This is tedious, complex, expensive, labor-intensive, and error-prone. This motivates us to automate the process of ACPs extraction, parsing, and formalization.

To bridge the gap between natural expression and formal expression of access control policies, we proposed an automated framework to extract elements in access control policies (ACPs) from natural language policies and generate the ACPs in the NGAC specification [41] [35] [26].

1.1 Underlying Problem

Identifying ACPs is the first step to developing access control systems. ACPs should be expressed correctly, because improper policy expression introduces security vulnerabilities. For example, unauthorized users or malicious users can exploit those vulnerabilities to access the protected data and resources.

ACPs can be formulated in several formats such as eXtensible Access Control Markup Language (XACML), mathematical, or natural languages [38], [21]. Most organizations describe their security policy in natural language access control policies (NLACPs) documents [9]. These documents include security requirements in natural languages such as English or other languages.

ACPs should be derived from NLACPs and converted to machine-executable instructions. Usually, ACPs are extracted manually [6], [5]. NLACPs contain unstructured, ambiguous, and implicit information. Thus, manual extraction is tedious, complex, expensive, labor-intensive, and error-prone. Automatic extraction helps system developers to extract ACPs easily with high accuracy and low cost in terms of labor, time, and complexity. That motivates us toward automated policy extraction from security requirements documents in natural language and generating enforceable policies.

Our research focuses on providing an access control framework that is suitable for secure application development. The framework will include automated policy generation from natural language specification that conforms to a standardized access control model.

In this thesis, we use NGAC as the underlying security model of our proposed approach. Many factors motivate us to use NGAC. First, NGAC supports ABAC, which allows us to express fine-grained policies that can be used for applications spanning multiple domains. Second, the policies can be updated while they are deployed, which makes them suitable for situation-monitoring applications. Third, policy management and auditing are greatly simplified [13]. On the other hand, the NGAC has many different types of entities and relations that make extracting NGAC policy from natural language statements challenging. The main question in this reseach is **how do we automatically translate ACPs from natural language expression to the NGAC formal specification?**

Thus, this thesis focuses on **automated of NGAC access control policies extraction.** In this thesis, we aim to tackle the manual ACPs extraction issues and automate the process using NLP techniques.

1.2 The Proposed Approach

To answer our research question, we aim to build an automated ACPs extraction framework that translates ACPs in natural language (NL) expression into formal NGAC specifications. To build this framework, we need to extract the NGAC elements, including entities and relations between those entities. To identify NGAC elements, we first must identify which sentence is an ACP. We utilize Bidirectional Encoder Representations from Transformer (BERT) as a binary classifier to classify NL sentences as either ACPs or not-ACPs. Second, we use the extracted ACPs to identify the NGAC elements. To achieve this, we employ BERT to extract NGAC relations and entities. To identify the NGAC relation, we utilize BERT as a multi-label classifier. Thus, ACPs can be identified as one or more NGAC relations, including association, assignment, prohibition, and obligation. To identify NGAC entities, we utilize BERT as a token classifier. BERT classifies each

token in each ACPs as one of the NGAC entities, including user (U), user attribute (UA), object (O), object attribute (OA), and action (A). For the implementation, we use PyTorch [1] , and the Transformers library.[2]

1.3 Contributions

The key contributions of this thesis are as follows:

- An automated framework that generates NGAC policies from security requirement documents in natural language.
- An automated approach to identify ACP sentences in natural language documents.
- An automated approach to extract the NGAC entities in each ACP sentence and their relations.
- An automated graph testing for NGAC ACPs using Neo4j.
- Providing a set of realistic synthetic natural language access control policies (ACPs) sentences to evaluate the proposed framework.

1.4 Thesis Organization

The rest of the thesis is organized as follows: Chapter 2 provides an overview of access control models. Chapter 3 describes related work on automated ACPs extraction approaches and discusses their limitations. Chapter 4 overviews our proposed framework. Chapter 5 elaborates on the structure and statistics of datasets, including the three phases: collection, annotation, and augmentation. Chapter 6 demonstrates the proposed framework for extracting ACPs from natural language security requirements and evaluating the experiment's performance. Finally, Chapter 7 concludes the thesis and outlines the directions for future work.

[1]https://pytorch.org/ pytorch.org

[2]https://huggingface.co/transformers/ huggingface.com/transformers

Chapter 2

Prerequisite

In the 1970s, the first access control systems were created using the Bell–LaPadula model (BLP) and the Biba model [3], [4]. BLP model was used for military information security. The BLP model aims to solve confidential information access control problems with a hierarchical structure of the military clearance levels. On the other hand, the Biba model [4] provides an integrity policy by defining formal access rules to protect data from improper modification.

In 1987, the Clark–Wilson model [34] is proposed as a transition system to control and audit the subject's policy parameters. The Clark–Wilson model provides complete integrity protection.

In the 1980s, many studies proposed flexible access control mechanisms such as Discretionary Access Control (DAC), Mandatory Access Control (MAC), Role-Based Access Control (RBAC), and Attribute Access Control Model (ABAC) known as traditional Access Control Systems. The following section describes each model.

2.1 Traditional Access Control Systems

The main entities that are used to describe traditional access control systems are:

- **Subject** represents the users that interact with the system.
- **Object** represents the resources that the system wants to protect.
- **Attributes** represents the characteristics of a subject or an object.
- **Policy** represents the logical form of the access permissions.

2.1.1 Discretionary Access Control (DAC)

In DAC, the administrator determines an access policy that prohibits or grants access to specific objects. DAC is considered decentralized because it depends on the user's authorization and the management of access authorities. An example of DAC model implementation is Windows's New Technology File System (NTFS). However, resource and authorization management in the DAC model is manual, so it is unsuitable for distributed environments due to the high complexity of the management work [22].

2.1.2 Mandatory Access Control (MAC)

MAC classifies the system's objects and subjects to a clearance level that determines the confidentiality level of the requested information. According to NIST, MAC is an access control policy enforced across all subjects and objects within a classification system. In other words, access is granted based on a clearance level, so each object is labeled with a classification level such as top secret and secret, and each subject has a clearance level to access an object. SELinux is an example of MAC model implementation.

MAC model is secure, but it is not flexible. However, distributed environments need a flexible access control model. Thus, MAC is not suitable for distributed environments due to the complexity of resource management via centralized authority. The advantage of MAC is solving the problem of decentralized resource management by centralized management [22].

2.1.3 Role-Based Access Control (RBAC)

RBAC sets permissions and privileges to enable access for authorized users. In other words, it is used for controlling user access to resources based on the user's role. This model ensures that only authorized users can access information and perform operations such as read, write, and execute [14].

There are three essential rules for the RBAC model as follows:

- Role assignment: a subject can exercise permission after a role is assigned to a subject
- Role authorization: a subject's role should be authorized for the subject
- Permission authorization: a subject can exercise permission only if the permission is authorized for the subject's active role.

This model is the most widely used among organizations because it is flexible in specifying access permissions and privileges based on the employee's job role. RBAC is more flexible compared with MAC and DAC. RBAC can facilitate security policy management in large organizations by enforcing security policies. However, the RBAC model is inappropriate for security in complex and ambiguous scenarios [27].

2.1.4 Attribute-Based Access Control Model (ABAC)

The core idea of this model is that access requests to perform actions on objects based on assigned attributes of the subject, assigned attributes of the object, and assigned attributes of the environment, so policies are specified in terms of those attributes [32] [19]. According to NIST documents in attribute-based access control model, ABAC is an access model in which the authorization is based on the evaluation of the attributes associated with the subject, object, requested operations, and environmental conditions [33], [18].

2.2 Next Generation Access Control (NGAC)

According to NIST, NGAC is a fundamental reworking of traditional access control into a form suited to the needs of the modern and distributed interconnected enterprise.

NGAC models the access decision as a Directed acyclic graph (DAC) as shown in Figure 2.1

According to the NIST document, the core policy entities that are used to describe NGAC model are as follows:

- **Users (U):** represents the actors that interact with the system.

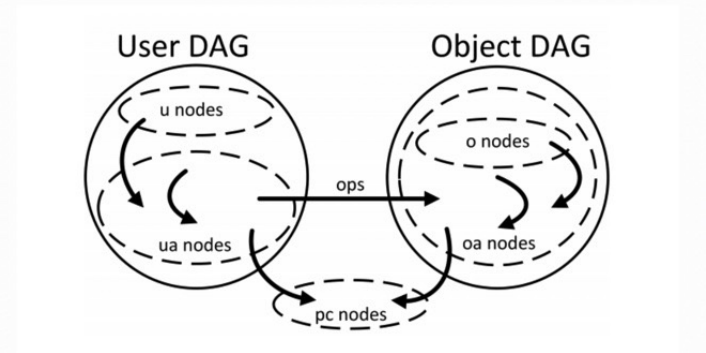

Figure 2.1: NGAC model components as DAC graph [32]

- **Objects (O):** represents the resources you want to protect.
- **User attributes (UA)** represents the characteristics of a user.
- **Object attributes (OA)** represents the characteristics of an object.
- **Access rights (AR)** represents the appropriate rights required for users and objects. AR is divided into two types, Administrative access rights(AAR) and Resource access rights(RAR).
- **Operation (OP)** refers to the actions that can be performed on the object. OP is divided into two types, Administration operation (AOP) and resources operation (ROP).
- **Policy classes (PC)** which organize and distinguish between classes of users and objects policies.

NGAC presents a system as a DAG graph to show resource model and user model in a system. The NGAC model is based on the relationship between the resource model and the user model and policy that specifies the permissions. [12], [20].

Chapter 3

Literature Review

The aim of this work is to provide an automated framework that deals with the access control policies specification. ACPs specification is the process of defining the ACPs elements and the relations among them and expressing the policy in formal language. There was prior work addressing the automatic ACPs specifications problem. Therefore, in this section, we present the related works that aim to automate ACPs specification and we analyze and compare their approaches with our proposed approach.

3.1 ACPs Extraction from Natural Language documents

Xiao *et al.* [41] propose an automated framework called Text2Policy to extract ACPs from natural language documents. The proposed framework is based on the XACML format. Text2Policy depends on linguistic analysis. Semantic pattern matching detects whether a sentence is ACP, along with a sentence with a negative meaning. The Text2Policy is limited to a fixed set of predefined access control patterns and cannot automatically learn new ACP patterns. It also produces incomplete ACPs when conditions exist in the document specifications.

Slankas *et al.* [35] present an Access Control Rule Extraction (ACRE) approach. ACRE represents natural language sentences as a dependency parse graph illustrating words as vertices and relationships between words as edges. ACRE uses a machine learning algorithm to search for a basic pattern of a noun as a subject, a verb as an action, and a noun as a direct object. ACRE constructs a naïve Bayes classifier to minimize pattern incorrectness to accept or reject patterns based on domain-independent features. ACRE automatically incorporates information extraction and machine learning techniques to learn new ACR patterns. However, ACRE is incapable of detecting ACPs that contain conditions. Our transformer-based approach detects ACPs that contain conditions such as obligations.

Narouei *et al.* [26] introduce a method to detect ACPs in natural language (NL) text and extract the main attributes in ACPs, including subject, action, and object. To detect ACPs, the authors use a K-NN classifier to identify if a sentence is ACP. Then, semantic role labeling (SRL) is utilized to identify predicate-argument structure in ACP sentences automatically. The approach effectively extracts the subject, action, and resource elements of ACPs from ACP sentences with a 75% F1-score. Nevertheless, each ACP has many attributes related to the subject, action, and object. The SRL approach is incapable of detecting the attributes associated with those elements. In our work, we bridge this gap by introducing a transformer-based approach that detects the attributes associated with the subject, action, and object.

Abdelgawad *et al.* [1] introduces a set of algorithms that extract policy elements and relations from ACP statements to form the NGAC security model. They used *spaCy* to extract entities in the form of (*subject, predicate, object*) from ACP statements and identify the relation between entities based on the syntactic dependency. They then used *Neo4j*, a graph database, to express the extracted elements and relations into NGAC form. The approach is applied to various ACP datasets to analyze the approach's correctness. The results indicate that unclean ACP datasets influence the extraction, causing incompleteness, redundancy, and inconsistency of the extracted security model. The *spaCy* is based on the convolutional neural network (CCN), a deep learning model. CNN manipulates tokens sequentially, while NLP transformers manipulate tokens in parallel. This parallelization makes our approach much faster. Moreover, NLP transformers have an attention layer that computes the importance score between different parts in the sequences. This feature makes NLP transformers much more potent than regular deep learning algorithms.

Alohaly *et al.* [2] propose a deep learning-based automated approach to extract ABAC attributes from natural language policy sentences. The authors developed a framework that leverages natural language processing (NLP), relation extraction (RE), and Convolutional Neural Networks (CNN) to automate attribute extraction. The framework first locates the modifiers of each policy element by analyzing the NL requirements, extracts the attribute value and corresponding policy element, and identifies the attribute's category, short name, and data type. Authors use CNN

to capture subject-attribute, object-attribute, and element-specific relations. The proposed framework resulted in an average F1 score of 85% for extracting the subject's attribute values and 71% for extracting object attribute values. The proposed framework can only detect attributes' values (attribute extraction) in ABAC policies without hierarchical relation among subject or object attributes. Our proposed approach detects all attributes and their relations to define the hierarchical relation between subject attributes and object attributes.

Abu Jabal *et al.* [29] propose a framework called Polisma to learn ABAC policies from different resources such as a log of access requests and corresponding access control decisions and context information provided by external resources such as Lightweight Directory Access Protocol (LDAP) directories. The approach uses data mining and machine learning techniques to generate the ABAC rules. The strength of the Polisma approach is learning ABAC rules from multiple resources. The approach accurately generates ABAC rules with 80% F1-score on existing datasets. Based on the NLP transformer's capabilities, we developed a framework that outperforms Polisma's accuracy with a 93% F1-score.

Xia *et al.* [40] present a BERT-based Semantic role labeling (SRL) system to identify the ACPs and ABAC attribute values. The authors use SRL forms of ARG0, ARG1, and VERB to identify the ABAC rule structure as subject, operation, and object. The authors evaluated their work on datasets in [41], [26], [37]. The ACPs identification accuracy reached 86.56% F-score while 72% F1-score for attribute identification. Using SRL can not detect attributes related to a subject or an object. In contrast, our approach detects the attributes associated with a subject and an object.

3.2 Research Gap

According to the literature review, we have identified the following research gaps:

- To the best of our knowledge, no work studied NGAC policy extraction.
- There are areas of improvement in ACP identification in NL sentences using natural language process algorithms.

- There are areas of improvement in attribute identification in ACPs sentences using natural language process algorithms.
- There is a lack of datasets representing ACPs to study the related problems of access control systems.

Chapter 4

The Proposed Framework

In this work, we investigate the use of NLP transformers, specifically Bidirectional Encoder Representations from Transformers (BERT), to generate NGAC policy. We propose an automated framework for ACPs specification. In this framework, we adapt NLP transformers to solve the manual ACPs specification issue since NLP transformer architectures prove their efficiency in extracting elements in natural language text. This thesis proposes an automated framework to extract access control policies from Natural Language Access Control Policy (NLACP) documents and convert them to an NGAC specification. We have chosen the NGAC model as our reference model because it shows inadequacy in withstanding the increasing complexity of today's business requirements, such as dynamic environment requirements in the Internet of Things technologies (IoT). NGAC is suitable for expressing ACPs of various applications, including those spanning multiple distributed, interconnected enterprises and situational monitoring applications where policies may be changed while deployed. Our proposed automated ACPs extraction framework is shown in 4.1.

We divide the framework into multiple tasks. The following bullets describe each task.

Task 1: Access Control Policies Identification, this task aims to extract the ACPs from NL documents. To do this task, we proposed a transformer-based model to classify a sentence as ACPs or not-ACPs. We fine-tune the BERT based on a sequences classification problem. Section 6.1 analyzes this task in detail.

Task 2: NGAC Relation Types Identification, this task aims to extract the relation type between the user attributes and object attributes in access control policies sentences. To do this task, we proposed a transformer-based model able to classify ACPs sentences as one or more of the NGAC relation types, including association, assignment, prohibition, and obligation. We fine-tune the BERT based on a multi-label classification problem. Section 6.2 deliberates this task in detail.

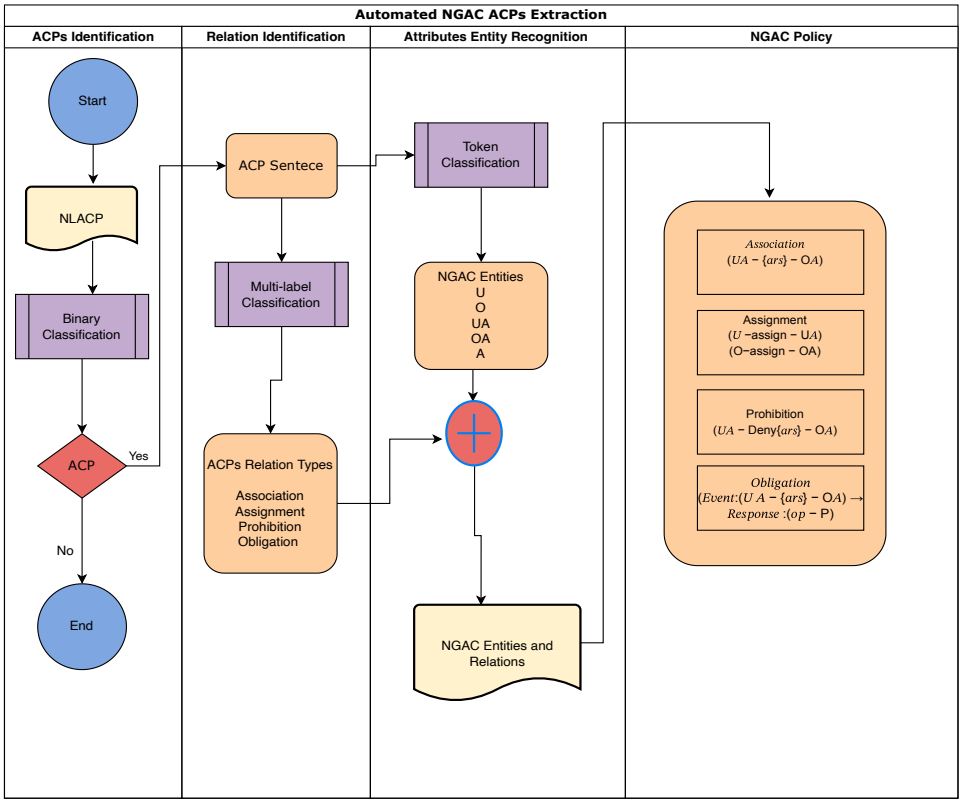

Figure 4.1: The Proposed Framework

Task 3: NGAC Attribute Identification, this task aims to extract NGAC attributes in each ACPs sentence. We proposed an Attribute Entity Recognition (AER) model to do this task. AER is based on the BERT model to solve a token classification. We fine-tune BERT to classify each token in a sequence as NGAC attributes user, user attributes, operation or action, object, and objects attributes. For example: *Professor Joe can add, change, or modify CS500 grades.* Where user: Joe, user attribute: professor, object: CS500, object attribute: grades. Section 6.3 elaborates on this task in detail.

Chapter 5

Datasets

Our proposed framework is based on NLP transformer downstream tasks. To build the framework, we need ground-truth datasets to build models to achieve the goals of each task. Thus, we aim to build large and worthy ground truth datasets for each model.

We collected data and then annotated it. However, we do not only use the original data we collected; we also use the augmented version. The following sections explain data collection, annotation, and utilization in this project.

5.1 Data Collection

Table 5.1: Four access control policy sentences represent different types of NGAC relations between user attributes and object attributes represented as (A) Association; (B)Assignment;(C) Prohibition; (D) Obligation.

(A)	$\{$Only administrators change the security label of resources.$\}_{\text{Association}}$
(B)	$\{$The administrator is allowed to set the length of this period.$\}_{\text{Assignment}}$
(C)	$\{$Instructors cannot change the course they teach after registration closes for the current semester.$\}_{\text{Prohibition}}$
(D)	$\{$If the requested appointment time does not conflict with existing LHCP appointments, the request is saved.$\}_{\text{Obligation}}$

Prior research on the identification of ACPs sentences provides several noteworthy datasets to be leveraged for supervised learning in our approach. These datasets are iTrust-v1 (DS1), iTrust-v2 (DS2), IBM (DS3), CyberChair (DS4), and Collected-ACPs (DS5). These datasets include ACPs sentences and not-ACPs sentences. Some examples are shown in Table 5.1.

In addition to these datasets, we have created a corpus containing about a thousand ACP sentences pertaining to access control in information technology from US universities' websites. We named our dataset DS6. To create this dataset, we created a web crawler to collect all text in IT access control policies on the university's websites. Our crawler collects the text in a paragraph.

We segment the paragraph into sentences and associate it with an ID that provides a reference of the paragraph that contains the sentence to preserve the context. Table 5.2 shows the data collected from the universities used in DS6.

Table 5.2: Security Sentences Collected from the Universities Websites

University Name	Number of ACPs
Colorado State University	66
University of Colorado Denver	165
University of Colorado Boulder	35
University of Northern Colorado	350
Harvard University	165
Georgia State University	25
University of Arizona	36
New York University	139
Rochester Institute of Technology	27
University of California	26
University of Massachusetts	23
Tennessee State University	34

We use six corpora under the monikers DS1, DS2, DS3, DS4, DS5, and DS6:

DS1: It is collected from iTrust [25], an open-source healthcare application that includes many features such as patient's medical history, primary caregivers identification, recording communications with doctors, and sharing the results. This dataset is used in [41].

DS2: It is the largest version of DS1, which contains more than a thousand sentences. This dataset is used in the ACRE approach [26].

DS3: IBM dataset collected by IBM course registration system. This is a dataset in the education domain that used [35] in their research.

DS4: Cyberchair dataset is in the conference management domain. This dataset was used in many different conferences and workshops [37].

DS5: This dataset is a combination of 114 ACP sentences collected from 18 sources, including published papers, public websites, etc.

DS6: The dataset we collected from US universities' information technology access control policies.

5.2 Data Annotation

We did the following three types of annotations: (i) use a binary label to indicate if a sentence is *ACPs* or *not-ACPs*, (ii) annotate a sentence as one or more of NGAC relation types *association*, *assignment*, *prohibition*, or *obligation*, and (iii) annotate each word in each ACPs as *user*, *action*, *object* , *user attributes*, or *object attributes*.

We did all the above three types of manual annotation for DS6. We manually annotate DS1, DS2, DS3, DS4, and DS5 for the last two annotations as the first one is already done by the authors when they published those data [41], [26], [37].

The annotation task is formulated by posing three questions, shown in Table 5.3, along with their labels. The first question requires binary yes/no labels, while the second question requires one or more of four labels:(*association*, *assignment*, *prohibition*, *obligation*,) and the third question requires one of five labels: (*user*, *action*, *object*, *user attribute*, *object attribute*).

Table 5.3: The three questions answered by annotators, and the corresponding labels: Yes/No, or Association [A], Assignment [AA], Prohibition [P], Obligation [O], or User [U], Object [O], Action [A], User attribute[UA], Object attribute [OA]

Question	Total	Label
Does the sentence contain an access control policy specification?	3482	Yes/No
What are the relation types between subject and object in ACPs sentence?	3482	[A],[AA][P],[O]
Where is the User (U),Object(O),Action (A)and User Attributes(SA),Object Attributes (OA)in ACPs sentence?	38223	[U],[UA],[A],[O],[OA]

We used the annotated data described above and also some augmented data described in following section.

5.3 Data Augmentation

Data augmentation is an effective technique to reduce overfitting by creating a different modified data version. We used various augmentation techniques, including back translation, combining different datasets, and the GPT3 model.

Back Translation

In ACPs identification task in section 6.1, we used the back translation augmentation technique that consists of the three steps described below.

1. *Temporary translation:* translate each of the original training labeled data into a different language. We translated from English to a random language.

2. *Back translation:* translate back each of those translated sentences into the original language, meaning a translation from the random language to English. We repeated this step ten times for each sentence.

3. *Duplicate removal:* At the end of the process, we will have the corresponding back-translation for each original text data. The goal here is to keep only one occurrence of each ACP.

Table 5.4 shows the number of ACPs and not-ACPs included in each dataset after back translation augmentation. These datasets are used in ACP identification task in section 6.1.

Table 5.4: Size of Datasets Before and After Augmentation

DS_n	**ACPs**	**Not-ACPs**	**Aug. ACPs**	**Aug. Not-ACPs**
DS1	419	52	4034	4538
DS2	528	643	5526	10116
DS3	163	239	1700	2977
DS4	124	179	2805	1454
DS5	115	27	1126	1385
DS6	828	109	4770	8440

Combined Datasets

In NGAC relation type identification task in section 6.2, to build a model, we need a dataset containing all four NGAC relation types (association, assignment, prohibition, and obligation). However, not all datasets contain all four NGAC relations as shown in 5.5. Thus, we combined all original datasets, so we have all NGAC relations types in the training dataset that be used in the NGAC relation identification task in section 6.2.

Table 5.5: Number of NGAC Relations in Each Dataset

DS_n	**Association**	**Assignment**	**Prohibition**	**Obligation**
DS1	408	269	6	0
DS2	484	317	4	12
DS3	124	105	1	1
DS4	177	51	0	28
DS5	89	76	8	10
DS6	474	68	28	47
Combined DS_n	1756	886	47	98

GPT3 Model Generator

In NGAC attribute identification task in section 6.3, we used the GPT3 model to generate new ACPs samples similar to the approach [42]. This approach is based on the GPT-3 model to produce synthetic sentences with hyper-realistic text samples from a mixture of actual samples utilizing GPT-3.

The procedure for generating augmented ACPs is as follows: extracting a few ACPs sample sentences from the training data, embedding these samples in the prompt, and generating an augmented sentence influenced by the sample sentences. As a result, we generate a thousand of ACPs sentences.

Table 5.6 shows examples of synthetic ACPs generated by GPT-3. This data is used in the attribute entity recognition task explained in section 6.3.

Table 5.6: Example of Five Synthetic ACPs Sentences Generated by GPT3

(A) Only administrators, not data owners, make changes to the security label of a resource.
(B) LHCP cannot cancel old appointments.
(C) Inactive patients cannot be changed or logged into the system and can only be reactivated by the administrator.
(D) The professor can change a student's grade at any time.
(E) If a teacher is signed up to teach the course in the current stream, the system will not delete the course.

Chapter 6

Automated ACPs Specification Framework

In this chapter, we describe the details of the proposed approach. As shown in Figure 6.1, the approach comprises three main tasks. Each task solves a specific problem to extract ACPs from Natural language documents. The following sections explain each task in detail.

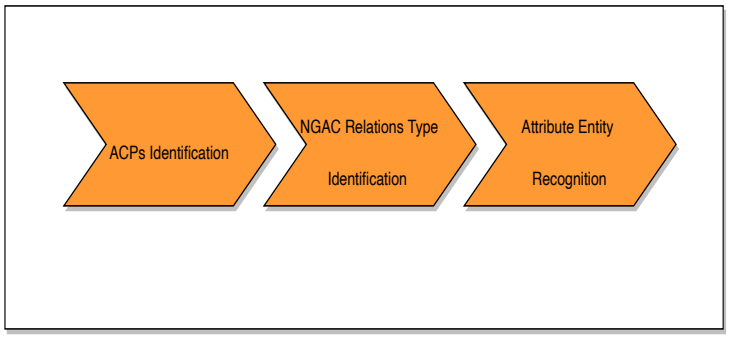

Figure 6.1: ACPs Extraction Approach Pipeline

6.1 Task 1: ACPs Identification

In NLP, text classification assigns text to different categories. A large pre-training language model has been shown to be sufficient for learning natural language representations by leveraging copious unlabeled data. The most distinguished examples of those large pre-training language models are Bidirectional Encoder Representations from Transformers (BERT) [10], Embeddings from Language Model(ELMo) [28], Generative Pre-trained Transformer(GPT) [24], and Universal Language Model Fine-tuning (ULMFiT) [17]. These large language models are trained on text data using unsupervised approaches.

We used the BERT model because it performs well in text classification tasks due to its language comprehension capabilities. BERT is based on a multi-layer bidirectional transformer and is trained on a large amount of plain text for the following two main tasks: masked word prediction task and next-sentence prediction task. To apply a pre-trained model to a specific task, we must fine-tune these models using task-specific training data. Thus, we need to design additional task-specific layers after the pre-training model.

For the text classification task, BERT adds a simple layer after the pre-trained model to fine-tune the model on a particular data and creates state-of-the-art models for text classification tasks.

In our ACPs identification task, we aim to classify the given sentence as ACPs or not-ACPs, as a binary classification problem. Thus, we leverage BERT for the ACPs identification task, as shown in the following section.

6.1.1 Fine-Tune Bidirectional Encoder Representations from Transformer (BERT) for ACPs Classification

Access Control Policies (ACPs) identification is done as follows. First, segmentation is dividing a string of written language into its component sentences. So we denoted the sentences as $S = \{s_1, ..., s_n\}$ where s_n a sequence of words of length n. Second, we must transform this sentence into a sequence of tokens $S_n[SEP]$ where [SEP] represents a separator token .Third, reformat that sequence of tokens by adding [CLS] and [PAD] tokens before using them as input to our BERT model, where [CLS] represents a single vector representing the whole input sentence used to feed to a classifier, and [PAD] represents the end paddings to the sentence.

So, inputs are designed as { [CLS],$S_n[SEP]$,[PAD] } then fed into the BERT encoder. Fourth, the BERT encoder will output an embedding vector of size 768 in each token. Finally, we used these vectors as input for binary text classification. The embedding vector from [CLS] token is used as an input to the binary classifier, which will then output a vector of size of the number of classes in our classification task ACPs, not-ACPs.

6.1.2 Challenges

The main challenge in the ACPs identification task is the imbalance data. We have few ACPs sentences compared with not-ACPs sentences. This imbalance makes the model training overfitting on a specific class, which leads to a biased model. To avoid bias, we used an augmentation technique to balance the data. In this task, we used the back translation technique to achieve our goal as described in section 5.

6.1.3 Experimental Results

Table 6.1 shows the average precision, recall, and F1-score across the resultant of BERT text classification that represents ACPs sentences in test datasets. In the majority of cases, the performance is promising. However, we found that the model's performance after data augmentation is better, proving the importance of data size and quality in NLP models. Furthermore, this thesis examines other BERT-based models like DistillBERT, RoBERTa, ELECTRA, and XLNET. Our goal is to compare the performance of different BERT models on the ACPs identification task as shown in Table 6.1. In most cases, we found XLNET model performs better than other models.

6.1.4 Discussion of the Experimental Results

Our experimental results are shown in Table 6.1. We display them in two sections, with the second showing the performance of various models on the test set of datasets. Overall, the XLNET shows significantly better performance in terms of precision, recall, and F1 score. Moreover, our proposed ACPs identification model overall outperforms the prior work Tex2Policy [41], ACRE [35], and SENNA [26] for ACPs identification in terms of F1-score as shown in the Table 6.2.

6.2 Task 2: NGAC Relations Types Identification

In a multi-label classification problem, the training set is composed of instances, and each one can be assigned multiple categories represented as a set of target labels. Thus, the task is to predict the label set of test data.

Table 6.1: The performance measures of training BERT,DistillBERT, RoBERTa, ,ELECTRA ,and XLNET on various datasets

Dataset	BERT			DistilBERT			RoBERTa			ELECTRA			XLNET		
	P	**R**	F_1	**P**	**R**	F_1	**P**	**R**	F_1	**P**	**R**	F_1	**P**	**R**	F_1
DS1	0.92	0.91	0.88	0.78	0.88	0.83	0.78	0.88	0.83	0.78	0.88	0.83	0.93	0.92	0.91
DS1-Aug	0.98	0.98	0.97	0.98	0.98	0.97	0.98	0.98	0.98	0.96	0.96	0.95	0.98	0.98	0.98
DS2	0.63	0.79	0.70	0.63	0.79	0.70	0.63	0.79	0.70	0.63	0.79	0.70	0.63	0.79	0.70
DS2-Aug	0.93	0.80	0.79	0.78	0.80	0.74	0.63	0.79	0.70	0.63	0.79	0.70	0.87	0.87	0.87
DS3	0.93	0.91	0.91	0.88	0.86	0.85	0.78	0.66	0.58	0.88	0.88	0.88	0.93	0.92	0.92
DS3-Aug	0.98	0.98	0.98	0.97	0.97	0.97	0.99	0.99	0.99	0.94	0.93	0.93	0.99	0.99	0.99
DS4	0.75	0.75	0.75	0.83	0.79	0.77	0.31	0.56	0.40	0.62	0.58	0.48	0.72	0.72	0.72
DS4-Aug	0.86	0.85	0.85	0.87	0.86	0.86	0.87	0.86	0.86	0.81	0.88	0.80	0.88	0.88	0.88
DS5	0.92	0.92	0.92	0.90	0.90	0.90	0.90	0.90	0.90	0.84	0.83	0.83	0.95	0.95	0.95
DS5-Aug	0.96	0.96	0.96	0.95	0.95	0.75	0.96	0.95	0.95	0.94	0.94	0.94	0.95	0.95	0.95
DS6	0.77	0.88	0.82	0.77	0.88	0.82	0.77	0.88	0.82	0.77	0.88	0.82	0.89	0.89	0.89
DS6-Aug	0.89	0.90	0.90	0.89	0.90	0.89	0.89	0.90	0.89	0.90	0.91	0.89	0.88	0.90	0.89

Table 6.2: Comparison with Prior Work

Dataset	Text2Policy [41]			ACRE [35]			SENNA [26]			Our Approach		
	P	**R**	F_1	**P**	**R**	F_1	**P**	**R**	F_1	**P**	**R**	F_1
DS1	0.86	0.88	0.87	0.96	0.99	0.98	0.75	0.88	0.80	0.98	0.98	0.98
DS2	–	–	–	0.90	0.86	0.88	0.54	0.87	0.58	0.87	0.87	0.87
DS3	1	0.96	0.97	0.83	0.92	0.87	0.46	0.84	0.59	0.99	0.99	0.99
DS4	–	–	–	0.63	0.64	0.64	0.79	0.86	0.82	0.90	0.88	0.88
DS5	–	–	–	0.83	0.96	0.89	0.64	0.86	0.70	0.95	0.95	0.95

This thesis aims to construct a model that can predict multiple types of NGAC relations in an ACP sentence. We propose an approach based on the BERT transformer to solve multi-label classification problems [7].

6.2.1 FineTune BERT for ACPs Multilabel Classification

In the NGAC model, ACPs can fall into multiple categories of relations. For example, the admin can add a student's record. This ACP sentence contains two categories: an association and

an assignment simultaneously. The assignment is the relation between the user and user attributes $\{User \leftarrow (role : Admin)\}$, and the association whereas admin associated with student's record by read and write operation:{admin$\rightarrow$ (read, write) $\rightarrow$ record}. In this work, we built a model to predict a multi-type of NGAC relations to obtain the relation between user and object attributes in ACPs. NGAC model includes association, assignment, prohibition, and obligation.

After we extract ACPs sentences in Task 6.1, we aim to identify the NGAC relation types to define the relation between user attributes and object attributes in each ACPs sentence. To accomplish this task, we need to classify the relation in each APCs as a multi-label classification problem.

BERT can be easily applied to downstream NLP tasks in a fine-tuned manner after obtaining the sentence vectors. We fine-tuned the BERT model to be able to classify ACPs for one or more types of NGAC relations (association, assignment, prohibition, and obligation). In this task, we fine-tune a BERT-base-uncased model [10]. We load the pre-trained model and then train the last layer for the classification task. Then, we add an extra dense layer with a plain Sigmoid activation function followed by a binary cross-entropy (BCE) Loss by combining the operations into one layer [30]. Further, we use two loss functions (BCE and local loss) to minimize the errors during our model training [23]. The model's output is the probabilities of all classes. We set a threshold value to pick up a list of the most relevant classes to input ACPs as the final output.

6.2.2 Challenges

To predict all NGAC relation types, we need to train the model on a dataset that contains all NGAC relation types. Since not all datasets in section 5 contain all types of relations, we combined all datasets. As a result, we have a dataset that contains all NGAC relation types, as shown in Table 5.5. There are unbalanced classes between the four types, which is the nature of access control policies. In general, we find ACPs associate users with resources more than prohibit users from resources. For example, faculty and teacher assistants can modify the grades, which means that no one else can modify the grades.

6.2.3 Experimental Results

Our experimental results are shown in Table 6.3. We obtained 96% F1- score, 99% recall, and 93% precision for association relation types identification. For assignment relation type, we obtained 87%,85%, 88% for F1-score, recall, and precision sequentially. These results are promising for the most dominant relation types: association and assignment. However, we obtained 43% and 65% F1-scores for prohibition and obligation, which are less visible in NLACP documents. Overall, we obtained 91% accuracy for the NGAC relation identification model. Our approach shows promising performance that can be used as a baseline for further improvement.

Table 6.3: NGAC Relation Types Identification Module Performance

Relation	**P**	**R**	F_1
Association	0.93	0.99	0.96
Assignment	0.88	0.85	0.87
Prohibition	0.1	0.27	0.43
Obligation	0.79	0.55	0.65

6.2.4 A Discussion of the Experimental Results

To demonstrate the capabilities of our model, we evaluate the experiments on precision, recall, and F1-score across the resultant of multi- label classification that represents NGAC relations types in our test data. Overall, our approach shows significantly promising performance. Our experimental results are shown in Table 6.3 shows that the performance with an overall average of 91% accuracy on the test set of ACPs sentences that contains different types of NGAC relations. However, the low performance for the prohibition is because of a low training sample because of the nature of access control policy specification. To illustrate, if an access control policy specifies what is allowed for a specific person, it implicitly disallows others. Thus, NLACP documents contain association relations more than prohibition. To our knowledge, no previous work investigated

how to automatically identify the relation types between user and object attributes in access control policies.

6.3 Task 3: NGAC Attributes Entity Recognition (AER)

A sequence labeling task is defined where a sequence of tokens is given, and a model should assign labels or classes to this token. The objective of a model is to learn the corresponding word labels in a set of labels or tags to identify and classify patterns, which can then be used to infer labels for new sequences of tokens [11]. Well-known examples of sequence labeling tasks are Named Entity Recognition (NER) and Part-of-Speech (PoS) tagging [39] [15].

6.3.1 Attributes Entity Recognition (AER)

The NGAC model relies on the relations between data elements to create, manage and enforce access control policies [8]. ACP specification in NGAC model has multiple entities, including user, user attribute, object, object attribute, and operation. While NGAC does not express policies through rules but instead through configurations of relations of four types: assignments (define membership in containers), associations (derive privileges), prohibitions (specify privilege exceptions), and obligations (dynamically alter access state), we aim to extract the main NGAC entities to construct ACPs in NGAC specification format.

We proposed Attribute Entity Recognition (AER) model. This model aims to solve a sequence labeling classification problem in ACPs specification. Since the BERT model has a promising performance on token classification, we fine-tune the BERT model to classify each token in a sequence as a user, user attribute, object, object attributes, and operation. We aim to predict the following attributes *U*ser(U), *U*ser Attributes (UA), *O*bject (O), *O*bject Attribute (OA), and *A*ction (A) as illustrated in Algorithm 1.

The architecture of BERT for the token classification task is as follows: the input of BERT represents a text sentence. Next, BERT tokenization, which converts a sentence into tokens (words) and then embeds every token of the text sentence. Thus, every sequence starts with a special token.

([CLS]),and ends with ([PAD]). In our model, we use BERT-based-cased; the number of layers L, the hidden size H, and the number of self- attention heads A as the default setting as follows: L=12, H=768, A=12, Total Parameters=110M Sequences MAX LEN = 75 batch size = 32

6.3.2 Challenges

Neural architectures and NLP transformers are very sensitive to the size of training data and tend to over-fit on small datasets. The training data is the main challenge to building the AER. We need various sentences that have all different attributes. Thus, we aim to increase the training data by augmenting our data. In this task, we generated thousands of different ACPs in the English language using GPT3 model as shown in section 5

Algorithm 1: Attribute Entity Recognition (AER) Algorithm

Input : $S = \{s_1 \dots s_n\} \in ACPs$
Output: $NGACAttributes = \{U, UA, A, O, OA\}$
Tokenization
$\{W_i, , \dots W_i\} \leftarrow WordTokenization(s_i)$
{Sent_ID, $Word_i$,Tag_i} $\leftarrow Get.Sentence$ (s_i)
$Sent(words, tags) \leftarrow GroupBYSentID(words, Tags)$
$TokenizeSent \leftarrow Sent.Tokenization(Sent)$
input_id $\leftarrow TokenToId(Tokenized.Sent)$
Fine-Tune BERT
(Tr_Inputs, Val_Inputs, Tr_Tags, Val_Tags) $\leftarrow TrainTest.Split$ (input_id, tags)
AER.Model $\leftarrow Token Classification$ (Tr_inputs, Val_inputs, Tr_tags, Val_tags)
NGAC Attributes $\leftarrow AER.Model$ (Test_data)

6.3.3 Experimental Results

To demonstrate the capabilities of our model, we evaluate the experiments on precision, recall, and F1-score across the resultant of token classification that represents NGAC attributes. The proposed AER shows promising results to predict each token in an ACP sentence as shown in Table 6.4.

Table 6.4: The Proposed AER Model Accuracy Report

Token	P	R	F_1
User(U)	1	0.53	0.70
User Attribute (UA)	0.93	0.99	0.96
Object(O)	0.67	0.57	0.62
Object Attribute (OA)	0.91	0.87	0.89
Action(A)	0.83	0.80	0.81
Other	0.96	0.83	0.89

Table 6.5: Comparison with Prior Work

Access Control Attributes	Alohaly [2]			AER		
	P	R	F_1	P	R	F_1
Subject Attributes	0.91	0.80	0.85	0.93	0.99	0.96
Object Attribute	0.75	0.69	0.71	0.91	0.87	0.89

6.3.4 A Discussion of the Experimental Results

AER model significantly outperforms the module proposed in [2] by obtaining a higher accuracy in terms of precision, recall, and f1-score for user attributes and object attributes identification as shown in Table 6.5. Moreover, AER model outperforms the module in [16] that aims to identify actor, action, and resource in user stories, which obtained 87% accuracy while our AER model obtained 93%.

6.4 NGAC Policy Extraction Example

This section presents an example of extracting entities and relations of ACPs in NGAC format. As shown in Table 6.6, we used a set of ACPs in NL. We applied our models to extract NGAC entities, and the relations among them as (association, assignment, prohibition, and obligation). Then, we synthesized the NGAC policy based on our models results.

To illustrate, we extract user and object attributes and their relations. Also, we assign users and objects for user attributes and object attributes. The security requirement document does not show

the exact names of users and objects. Then, we assigned names to the users and objects. Once we have all NGAC entities and relations, we generate the NGAC policy as shown in Table 6.6, including UA, OA, U, O, and the NGAC policy.

Table 6.6: ACPs in the NGAC Format

ACPs	
1	A student can not modify a course grade.
2	A teaching assistant can add a course grade.
3	A professor can change a course grade .
$UserAttribute(UA)$	
4	User is a student.
5	User is a teaching assistant.
6	User is a professor.
$ObjectAttribute(OA)$	
7	course grade
$Relations$	
7	prohibition, and assignment (ACP 1).
8	association and assignment (ACP 2).
8	association and assignment (ACP 3).
$User(U)$	
9	Sara is a student.
10	Mike is a teaching assistant.
11	Joe is a professor.
$Object(O)$	
12	CS 550 is an object.
$NGACPolicy$	
13	$(Sara \rightarrow student \xrightarrow{deny[modify]} course\ grade \rightarrow CS550)$
14	$(Mike) \rightarrow teaching\ assistant \xrightarrow{[add]} course\ grade \rightarrow CS550)$
15	$(Joe \rightarrow professor \xrightarrow{[change]} course\ grade \rightarrow CS550)$

Chapter 7

Conclusion

The access control system is the backbone of IT infrastructures to secure data and resources. The first step in establishing an access control system is defining the ACPs. While ACPs are embedded in security requirements, often written in natural language. Currently, such access control statements are manually extracted, and security mechanisms enforce these access control policies. Manual ACPs extraction is tedious, complex, expensive, labor-intensive, and error-prone. Incorrect access control policies have severe consequences and can introduce security vulnerabilities controllable by malicious users.

The main contribution of this thesis is proposing an automated framework to translate access control policy sentences into formal specifications. In this thesis, we construct NGAC ACPs from natural language. We develop a model-based framework to extract ACPs from security documents in natural language and automatically identify the NGAC entities and relations needed to form NGAC policies. Furthermore, we contributed by providing a realistic dataset containing access control policies in the education domain, allowing further natural language analysis in the access control models domain.

Our proposed framework can identify ACPs with a 93% F1-score, 96% F1-score for association identification, the dominant relation type in NLACPs, and average F1-score of 96% for the user attribute and 89 % for object attribute identification. The framework is promising for extracting the NGAC access control policies. However, more data collection effort is required to improve the performance of the proposed models. Future work will consider static and dynamic NGAC model analysis.

References

[1] Abdelgawad, M., Ray, I., Alqurashi, S., Venkatesha, V., Shirazi, H.: Synthesizing and analyzing attribute-based access control model generated from natural language policy statements. In: SACMAT (2023)

[2] Alohaly, M., Takabi, H., Blanco, E.: Towards an automated extraction of ABAC constraints from natural language policies. In: ICT Systems Security and Privacy Protection. pp. 105–119. Springer (2019)

[3] Bell, D.E., LaPadula, L.J.: Secure computer systems: Mathematical foundations. Tech. rep., MITRE CORP BEDFORD MA (1973)

[4] Biba, K.J.: Integrity considerations for secure computer systems. Tech. rep., Mite Corp Bedford MA (1977)

[5] Bijon, K.Z., Krishnan, R., Sandhu, R.: Towards an attribute based constraints specification language. In: 2013 International Conference on Social Computing. pp. 108–113. IEEE (2013)

[6] Breaux, T.D., Antón, A.I.: Deriving semantic models from privacy policies. In: Sixth IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY'05). pp. 67–76. IEEE (2005)

[7] Chalkidis, I., Fergadiotis, M., Malakasiotis, P., Androutsopoulos, I.: Large-scale multi-label text classification on EU legislation. arXiv preprint arXiv:1906.02192 (Jul 2019)

[8] Chiquito, A., Bodin, U., Schelén, O.: Access control model for time series databases using NGAC. In: ETFA. vol. 1, pp. 1001–1004. IEEE (2020)

[9] Damianou, N., Dulay, N., Lupu, E., Sloman, M.: The ponder policy specification language. In: Policies for Distributed Systems and Networks: International Workshop, POLICY 2001 Bristol, UK, January 29–31, 2001 Proceedings. pp. 18–38. Springer (2001)

[10] Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: BERT: Pre-training of deep bidirectional transformers for language understanding. arXiv (Oct 2018)

[11] Ehrmann, M., Hamdi, A., Pontes, E.L., Romanello, M., Doucet, A.: Named entity recognition and classification on historical documents: A survey. arXiv preprint arXiv:2109.11406 (Sep 2021)

[12] Ferraiolo, D., Chandramouli, R., Kuhn, R., Hu, V.: Extensible Access Control Markup Language (XACML) and Next Generation Access Control (NGAC). In: Proceedings of the 2016 ACM International Workshop on Attribute Based Access Control. pp. 13–24 (2016)

[13] Ferraiolo, D., Gavrila, S., Jansen, W.: Policy machine: Features, architecture, and specification (2015-10-27 2015)

[14] Ferraiolo, D.F., Chandramouli, R., Ahn, G.J., Gavrila, S.I.: The role control center: features and case studies. In: Proceedings of the 8th ACM symposium on Access control models and technologies. pp. 12–20 (2003)

[15] Fritzler, A., Logacheva, V., Kretov, M.: Few-shot classification in named entity recognition task. In: Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing. pp. 993–1000 (2019)

[16] Heaps, J., Krishnan, R., Huang, Y., Niu, J., Sandhu, R.: Access control policy generation from user stories using machine learning. In: DBSec. Springer (2021)

[17] Howard, J., Ruder, S.: Universal language model fine-tuning for text classification. arXiv preprint arXiv:1801.06146 (Jan 2018)

[18] Hu, V.C., Ferraiolo, D., Kuhn, R., Friedman, A.R., Lang, A.J., Cogdell, M.M., Schnitzer, A., Sandlin, K., Miller, R., Scarfone, K., et al.: Guide to attribute based access control ABAC definition and considerations (draft). NIST special publication **800**(162), 1–54 (2013)

[19] Hu, V.C., Kuhn, D.R., Ferraiolo, D.F., Voas, J.: Attribute-based access control. Computer **48**(2), 85–88 (2015). https://doi.org/10.1109/MC.2015.33

[20] Huh, J.H., Bobba, R.B., Markham, T., Nicol, D.M., Hull, J., Chernoguzov, A., Khurana, H., Staggs, K., Huang, J.: Next-generation access control for distributed control systems. IEEE Internet Computing **20**(5), 28–37 (2016)

[21] Jajodia, S., Samarati, P., Subrahmanian, V.: A logical language for expressing authorizations. In: Proceedings. 1997 IEEE Symposium on Security and Privacy (Cat. No. 97CB36097). pp. 31–42. IEEE (1997)

[22] Jordan, C.S.: Guide to Understanding Discretionary Access Control in Trusted Systems. Diane (1987)

[23] Lin, T.Y., Goyal, P., Girshick, R., He, K., Dollár, P.: Focal loss for dense object detection. In: IEEE on computer vision. pp. 2980–2988 (2017)

[24] Liu, X., Zheng, Y., Du, Z., Ding, M., Qian, Y., Yang, Z., Tang, J.: GPT understands, too. arXiv preprint arXiv:2103.10385 (Mar 2021)

[25] Meneely, A., Smith, B., Williams, L.: Appendix b: itrust electronic health care system case study. Software and Systems Traceability p. 425 (Feb 2012)

[26] Narouei, M., Takabi, H., Nielsen, R.: Automatic extraction of access control policies from natural language documents. IEEE Dependable and Secure Computing (2018)

[27] Ouaddah, A., Mousannif, H., Abou Elkalam, A., Ouahman, A.A.: Access control in the internet of things: big challenges and new opportunities. Computer Networks **112**, 237–262 (2017)

[28] Peng, Y., Yan, S., Lu, Z.: Transfer learning in biomedical natural language processing: an evaluation of BERT and ELMo on ten benchmarking datasets. arXiv preprint arXiv:1906.05474 pp. 58–65 (Aug 2019)

[29] Pereira, H.G., Fong, P.W.: SEPD: An access control model for resource sharing in an iot environment. In: ESORICS. pp. 195–216. Springer (2019)

[30] PyTorch: Bce with logits loss (2023), https://pytorch.org/docs/stable/generated/torch.nn.BCEWithLogitsLoss.html#torch.nn.BCEWithLogitsLoss

[31] Samarati, P., de Vimercati, S.C.: Access control: policies, models, and mechanisms. In: FOSAD. pp. 137–196 (2000)

[32] Sandhu, R., Ferraiolo, D., Kuhn, R., et al.: The NIST model for role-based access control: towards a unified standard. In: ACM workshop on Role-based access control. vol. 10 (2000)

[33] Servos, D., Osborn, S.L.: Current research and open problems in attribute-based access control. ACM Computing Surveys (CSUR) **49**, 1–45 (2017)

[34] Shockley, W.R.: A9 implementing the Clark and Wilson integrity policy using current technology. Computer Science and Technology **1**(11), 1 (1989)

[35] Slankas, J., Xiao, X., Williams, L., Xie, T.: Relation extraction for inferring access control rules from natural language artifacts. In: ACSAC. pp. 366–375 (2014)

[36] University, C.S.: CSU POLICY: INFORMATION TECHNOLOGY SECURITY (2023), https://policylibrary.colostate.edu/policy.aspx?id=492

[37] Van De Stadt, R.: Cyberchair: A web-based groupware application to facilitate the paper reviewing process. arXiv preprint arXiv:1206.1833 (June 2012)

[38] Wijesekera, D., Jajodia, S.: Policy algebras for access control the predicate case. In: Proceedings of the 9th ACM conference on Computer and Communications Security. pp. 171–180 (2002)

[39] Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., et al.: Transformers: State-of-the-art natural language processing.

In: Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations. pp. 38–45 (2020)

[40] Xia, Y., Zhai, S., Wang, Q., Hou, H., Wu, Z., Shen, Q.: Automated extraction of ABAC policies from natural-language documents in healthcare systems. In: 2022 IEEE(BIBM). pp. 1289–1296. IEEE (2022)

[41] Xiao, X., Paradkar, A., Thummalapenta, S., Xie, T.: Automated extraction of security policies from natural-language software documents. In: Proceedings of the ACM SIGSOFT 20th. pp. 1–11 (2012)

[42] Yoo, K.M., Park, D., Kang, J., Lee, S.W., Park, W.: GPT3Mix: Leveraging large-scale language models for text augmentation. arXiv (2021)