CS 495 Tyler Baylson
Capstone Proposal Andrew Sexton

Converting an English Sentence to a Format Understandable by Computers

1 Introduction

What is the problem?
A benefit of file systems, whether locally housed or "in the cloud", is allowing for the increased work efficiency of offering different users in the network access to the same resources. But given the growing desire for companies to give their employees this level of efficient access to data, that data is at risk anytime security isn't easy to establish for its users. And, in the worst cases, where something like a credential file may itself be accessed by a user who should not have access, a security failure cascade may be initiated that could lead to very serious problems for the company and its users.
Why is it important?
Given this, it is extremely important that users can easily and efficiently control access to their data, starting from simple and straightforward privacy purposes all the way to fully customized schedules of access to highly confidential data.
How do you plan to solve it?
In order to meet this goal, easy and effective means to protect information and resources must be made available to those responsible for its security. Being that natural language is the most natural way to input a command, we will allow for the natural input of rules both to build effective access policies for a given resource and for the input of request inquiries to establish whether or not the requesting entity has authorized access to that resource. We will allow for policies and inquiries to be converted to an easy-to-use logical format, enhancing time efficiency of evaluating inputs. Our system will use a well-formed notation both for specifying access control policies and for comparing access when inquiries against data are made. In this way, the definition of policies becomes simpler and more efficient, which will allow users to protect their assets under their own terms, and will ensure that requesters can feed access requests to the system simply as well.

2 Problem Definition

Briefly describe the data that you used (or plan to use).
All types of data in a file system are at risk when access is not defined or if that access is defined in an unclear or incomplete manner. Our system will take user input as input data and determine the intent of a given security rule or access request for the extant data on a network or filesystem which implements our access control methods.
What do people usually try to do with the data?
While typical access for a given file may be legitimate, when the owner of data is not able to continuously monitor access (especially given the complexity of certain situations where data may be shared in different ways to different people under different conditions), it's easy for security requirements to relax.
Why is it difficult to do what they want to do?
Natural language requests such as "Bob can view my documents on weekdays" do not translate well into bash scripts or other forms of security management. Though research has

been performed (as discussed below), no specific approach has been quite flexible enough for general use within social networks, especially not based on natural language in a way that is directly accessible to someone who has not been able to invest in security knowledge or practices.
What specific data did you use (or plan to use)? What is the source?
In a given Network, which can be interpreted as a directed labeled graph (vertices connected by edges)

1. N is a finite set of subjects (S) and objects (O), that may include an environmental condition.

2. Σ is a finite set of relationship types (labels for connections).

3. W is a number attached to a relationship between two subjects. This is a percentage of trust between two users.

4. $E \subseteq N \times N \times \Sigma \times W$ is the edges of existing connections

Users will define policies and make inquiries in English via an access control interface, such that a well-formed format will transform user defined policies into system rules and user defined inquiries will be used as system queries. The source of our data will be English sentences typed in a natural-sounding way.
What are the specific user requirements?
Our system must process natural user command sentences in a given structure within a basic UI, resulting in appending data to policy file in the user's directory in the format to be demonstrated below. This will require a simple UI for access control input, which, though ultimately intended for web-use, could well be presented within a desktop application. It will be flexible and responsive, as companies may wish to have more granular control over the presentation of their access application. In the event of an error in input or discovery, depending on certain circumstances, the program may ask for further clarification. A program already exists which will manage the basic security functionality (reading the policy files and acting on their instruction when queried), and it also handles conflicts (giving priority to most recent addition).
List a company and organization that may use the features of this program.
Facebook would use some features Team-oriented group work Users for whom it is not time-efficient to learn the details of things like 'chmod' would benefit greatly from this sort of setup, allowing them full control for very specific users in a way that has not been managed before, to our knowledge.

3 Solution

List/explain the alternate solutions available. Why did you not use other solutions?
Numerous solutions to access control have been theorized. Fong et al. defined an access control model that employed authorization decisions in terms of the relationships between the resource owner and a resource requester [1]. We decided that we wouldn't use this solution because it seems only appropriate for simple policies. Aktoudianakis et al. defined

Table 1: MoSCoW Board

Must Have Should Have Could Have Won't Have
PBAAC model implementation Basic GUI with function calls for business embedding and customization
UI
Appending distinct well-formed policies to a text file

a syntax for specifying a social network and the associated access control policies. Generating policies with this method is simple, however, this syntax does not provide flexible capabilities directly associated to users to define effective policies. [2] Some solutions include an XML-based format. Extensible Access Control Markup Language (XACML) is one such access control policy language. This language uses XML to generate both an access control policy language and a request/response language. We decided not to use XACML due to the complexity of its XML tags.
Why do you plan on using the solution that you chose?
The aforementioned methods can be cumbersome and/or complicated to use. Our solution will solve these issues. The policy-based attribute access control model will allow its users to use a natural language to communicate policies. Using the system will be as fluid and as simple as normal conversation.
Describe your proposed solution thoroughly. Include a data flow diagram, interaction-flow diagram, and MoSCoW analysis.
Our solution stands as a faithful implementation of Dr. Morovat's own Policy Based Language for Access Control [3]. Given sentences will be in the format: Subject verb object [conditional(time/date)] Example: "Alice is allowed to access photos" Access: Read / write / modify (including negation / not) Objects may be referenced by filetype: photos = jpg, gif, etc / doc = txt, doc, etc. Differentiate between types of entities in network as modeled above: Target users, which get policies protecting them from unauthorized access Target resources, which get policies protecting them from unmanaged actions Inquiries have two groups: Attempting to access target users by tagging in a photo or poking a target user Do an operation against resources like updating or deleting resource Template needed to process natural language inputs into easy-to-use and well-formed format Words in policy/ inquiry appear with order: subject, verb, object Words in rule/request appear with order: subject, action, object System will have a simplified UI which may be used as a framework for accessing resources in various file system configurations and social networks
Describe and show a storyboard that illustrates how your system will be used.
Users will start our program via command line or some graphical application. Users may then type access control commands into the program. These commands may be written as natural-sounding sentences. The program will then parse the sentence into individual words. These individual words will be used with WordNet's lexical database to ascertain

Table 2: Project Time Table

Date	Deliverable Description
2/13	Proposal with Timetable
2/27	Input via interface, accept and validate, handle errors, test on frameworks
3/12	Sorting words into parts of speech, recognize access actions (“read” / “write” / “access”) and their
3/26	Accept specific object references (“house.png”), Accept generalized object references and groups by
4/09	GUI implementation / function calls, Generate well-formed format, Append the formatted rule to u
4/12	Poster due
4/19	Poster demonstration
4/23	Confirm full functionality with policy reader program, Capstone I scope complete

their grammatical and syntactical significance. The program will use the grammatical and syntactical data to generate a well-formed policy rule for access control based on the original natural-sounding sentence. The newly generated policy will be cross examined with an existing policy text file to ensure the same policy does not already exist. If the newly generated policy does not exist in the policy text file, it will be appended to the policy text file.
Establish which part(s) of the problem will not be covered by this solution.
Everything listed in the ‘Will Not Have’ part of MoSCoW Dr. Morovat’s program handles the security activities themselves.

4 Plan

How will you test the system?
We will utilize the pytest framework to help streamline the development process. Additionally, we will use Dr. Morovat’s policy reader to ensure our program generates the well-formed format. What is the proposed timeline for the semester?

Finally, in reviewing how specific coursework we have completed will reflect in this project, we provide this list:

1. CS 150 & 151- Intro to Problem-Solving I & II: Intro to python and working with data structures
2. CS 263- Software Eng.: Breaking down the project into tasks, preparing proposal, specifics like the MoSCoW analysis, etc.
3. CS 351- Data Structs. & Algors.: Working with graph data structures
4. CS 352- Prog. Lang.: Separating input for parsing (lexemes, etc.)
5. Math 270- Statistics: Any analytics we perform are likely to build on the principles of 270
6. Math 310- Discrete Structs.: Learning about graph structures and, more specifically, markov chains.

7. On our own, we will be learning learn PyTest and approaches to implement a theoretical model in programming practice (PBAAC).

References

[1] Philip Fong and Ida Siahaan. Relationship-based access control policies and their policy languages. In *SACMAT '11: Proceedings of the 16th ACM Symposium on Access Control Models and Technologies*, pages 51–60, New York, NY, USA, 01 2011. Association for Computing Machinery.

[2] Evangelos Aktoudianakis, Jason Crampton, Steve A. Schneider, Helen Treharne, and Adrian Waller. Policy templates for relationship-based access control. *2013 Eleventh Annual Conference on Privacy, Security and Trust*, pages 221–228, 2013.

[3] Katanosh Morovat and Brajendra Panda. Policy language for access control in social network cloud. *2016 IEEE International Conference on Smart Cloud (SmartCloud)*, pages 313–318, 2016.