Access Control Policy Extraction from Unconstrained Natural Language Text

John Slankas and Laurie Williams
Department of Computer Science
North Carolina State University
Raleigh, North Carolina, USA
[john.slankas,laurie_williams]@ncsu.edu

Abstract—**While access control mechanisms have existed in computer systems since the 1960s, modern system developers often fail to ensure appropriate mechanisms are implemented within particular systems. Such failures allow for individuals, both benign and malicious, to view and manipulate information that they should not otherwise be able to access. *The goal of our research is to help developers improve security by extracting the access control policies implicitly and explicitly defined in natural language project artifacts.* Developers can then verify and implement the extracted access control policies within a system. We propose a machine-learning based process to parse existing, unaltered natural language documents, such as requirement or technical specifications to extract the relevant subjects, actions, and resources for an access control policy. To evaluate our approach, we analyzed a public requirements specification. We had a precision of 0.87 with a recall of 0.91 in classifying sentences as access control or not. Through a bootstrapping process utilizing dependency graphs, we correctly identified the subjects, actions, and objects elements of the access control policies with a precision of 0.46 and a recall of 0.54.**

Keywords—access control; documentation; machine learning; natural language processing; relation extraction; security

I. INTRODUCTION

Despite significant remediation efforts over the past decade, such as those due to information technology controls required for Sarbanes-Oxley [1], and the highlighting of access control errors in lists such lists as the CWE/SANS Top 25 Most Dangerous Software Errors [2], access control remains a significant issue. In the 2013 Verizon Data Breach Investigations Report [3], 61% of the incidents included some form of access control abuse. To mitigate data security issues, organizations must properly implement access control across all system components. Access control is a critical application mechanism to ensure confidentiality and integrity [4]. While various access control models exist (discretionary, mandatory, role-based, attribute, etc.), most models contain a tuple *(subject, resource, action)* to represent a policy as to whether or not the *subject* (a user) can perform the requested *action* on the specified *resource* (object) within the system. Access control policies are often expressed, implicitly or explicitly, within nature language. For example, "The system shall allow hospital administrators to inactive patients" explicitly grants users who are hospital administrators the ability to inactivate patients. However, creating and defining the correct access control policies can be a tedious, time-consuming, and error-prone endeavor. Developers must extract access control policies from existing documentation, application code, and database implementations.

The goal of our research is to help developers improve security by extracting the access control policies implicitly and explicitly defined in natural language project artifacts.

We propose a process, which we call Access Control Relation Extraction (ACRE), which allows organizations to utilize existing, unconstrained natural language text to extract access control policies. ACRE analyzes requirements or other natural language statements to obtain their subject, action, and resource elements. The process utilizes a combination of natural language processing (NLP), information extraction (IE), and machine learning (ML) techniques. A critical component to our process is the generation of appropriate dependency graph patterns based upon an initial set of seeded patterns and then expand the set of patterns through extracting new patterns where combinations of any discovered access control elements (subjects, actions, resources) can be located in the current document sets under investigation. This approach of learning new patterns from an initial set of seed patterns is termed "bootstrapping" [5]. Due to the ambiguity and multitude of different ways of representing concepts within natural language, our process allows for humans to enter undiscovered patterns or correct patterns misidentified in the bootstrapping approach.

To evaluate our process, we developed the following research questions:

RQ1: How effectively can we identify access control policies in natural language text in terms of precision and recall?
RQ2: What common patterns exist in sentences expressing access control policies?
RQ3: What is an appropriate set of seeded graphs to effectively bootstrap the process to extract the access control elements?

We evaluated our process and tool against an open source educational testbed, the iTrust Electronic Health Records System [6].

978-0-7695-5137-1/13 $26.00 © 2013 IEEE
DOI 10.1109/SocialCom.2013.68

II. Related Work

A. Natural Language and Access Control

Other researchers have explored using natural language to generate access control policies from natural language. He and Antón [7] proposed an approach based upon available project documents, database design, and existing policies. Utilizing a series of heuristics, humans would analyze the documents to find additional access control policies. In addition to heuristics to find the elements within the typical access control tuple (subject, resource, action), they created heuristics to identify policy constraints (temporal, location, relationship, privacy, etc.) and obligations. More recently, Xaio et al. [8] present an approach, Text2Policy, where they parsed use cases to create eXtensible Access Control Markup Language 1 (XACML) policies. Their approach was specific to use case-based requirement specifications and relied upon matching four specific sentence patterns to deduce the necessary information to populate an access control method.

B. Controlled Natural Language

Other researchers have resolved converting natural language to and from policies by utilizing a controlled natural language (CNL). Schwitter [9] defines CNLs as "engineered subsets of natural languages whose grammar and vocabulary have been restricted in a systematic way in order to reduce both ambiguity and complexity of full natural languages." While CNLs provide consistent, semantic interpretations, CNLs limit authors and typically require language specific tools to stay within the constraints of the language. Project documents previously created cannot be used as inputs without processing the documents manually into the tools. Policies authored outside of tools must confirm to strict limited grammars to be automatically parsed as well. Brodie et al. [10] used this approach in the SPARCLE Policy Workbench. By using their own natural language parser and a controlled grammar, they were effectively able to translate from controlled natural language into formal policy. Recently, Shi and Chadwick [11] presented their results of an application to author access control policies using a CNL. While they showed the improved usability of CNL interface, they were limited in the complexity of the policies that could be created as the interface did not support conditions or obligations. Our approach utilizes project artifacts without change to the vocabulary or structure of the sentences.

C. Relation Extraction

A number of different ways exist to identify semantic relations within text. Initial solutions employed hand-written patterns to detect hyponym ("is-a") relationships among words [12]. While hand-written patterns usually have high precision, they tend to suffer low recall by missing relation occurrences. Snow et al. [13] used dependency paths and grammatical relationships within a sentence to discover additional relationship patterns. Akbik and Broß [14] used a similar process to extract a diverse range of sematic relations with the goal to extract arbitrary relations for semantic search. Dependency parse tree patterns have been used to extracting relations between genes and proteins [15]. We utilize dependency graphs, but extract the minimum graph pattern that overlays the lowest common ancestor of all of the included elements of the dependency graph. These elements include the subject, action, the relation, and any contextual information necessary to disambiguate the necessary permissions for the access control policy. As with other works [13], we include capability to bootstrap the possible graph patterns and a naïve Bayes classifier to judge whether such derived patterns should be include in the evaluation set.

[1] http://www.oasis-open.org/committees/xacml

III. Access Control Relation Extraction

A. Access Control and Natural Language

Within natural language texts, access control elements are explicitly and implicitly stated. For example, "The system shall allow patients to view their own medical records" explicitly grants users who are patients the right to read their medical records. Other sentences such as "A nurse can order a lab procedure for a patient" implies two access control policies. First, the nurse has some form of create or write permission for lab procedures. Secondly, since the lab procedure is for a patient, another access control policy implicitly exists to grant the nurse read access to patients. In many situations, the verb that typically represents the action within the sentence implies the necessary permissions to be granted. However, in some cases, the permissions are not necessarily so straightforward. As with the second example, the verb "order" combined with the prepositional phrase, "for the patient", implies read access to a patient.

B. ACRE Access Control Policy Representation

Internally, we represent sentences with a dependency graph as depicted in Fig. 1 for the sentence "a nurse can order a lab procedure for a patient." (In the next section, we discuss how these graphs are produced.) Each vertex represents a word from the sentence along with the word's part of speech. In the figure, "NN" represents a noun, "VB" represents a verb, and "MD" represents a modal verb. Edges represent the grammatical relationship between two words. For instance, "nurse" functions as the nominal subject (nsubj) for "order" and "dobj" is the object to be ordered. Dependency graphs can be considered trees in most situations and are typically rooted by the sentence's main verb.

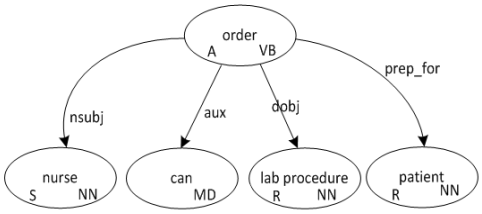

Figure 1: ACRE Sentence Representation

To represent an access control policy, we utilize the following pattern, termed an "access control pattern":

$$A(\{s\},\{a\},\{r\},[n],[l],\{c\},H,p) \quad (1)$$

A defines the overall access control policy. s contains an order set of vertices that compose the *subject* of a policy. Similarly, a and r represent the *action* and *resource*, respectively. n contains the vertex representing negativity if

required for the policy. If the policy should be limited to a particular subject s, l contains the indicating vertex. c contains any additional vertices required to provide context to given action for a set of permissions. H represents the subgraph of a sentence's dependency graph that contains the vertices and necessary edges to connect all of the vertices listed in s, a, r, n, l, c. p represents the permissions typically associated with an action. We limit permissions to have the values of "create", "retrieve", "update", and "delete" as we are primarily concerned with controlling the ability to view and manipulate data in systems.

Situations exist in which not all of the access control policy elements may be present within a single sentence. These access control policies may be identified with missing elements. Developers would be directed to the surrounding sentences to finish defining the access control policy.

C. Access Control Relation Extraction Process

The ACRE process consists of five primary steps:

1. Parse text document
2. Parse natural language
3. Classify sentence as access control or not
4. Extract access control elements
5. Validate access control policies

For input, the process takes any natural language document converted into a text only format. For the output, the extracted access control policies are listed along with any issues discovered for completeness and consistency. To assist developers with this process, we have developed a corresponding tool to automate most tasks, provide the ability for developers to correct classification issues, and to provide manual identification of access control policies which the process has been unable to find.

Step 1: Parse Text Document

The process first reads the entire text into the system. We then separate the input into tokens by either new lines or by periods at the end of sentences. Next, we apply a concise document grammar (Fig. 2) to label each token to a specific type:

- *title*: Lines which follow capitalization rules for titles. We separate them from other lines in our process because titles never indicate an access control related requirement.
- *list start*: These lines represent the header or description of a list that follows.
- *list element*: These lines represent individual items contained within an ordered or unordered list. These lines are combined with the start of the list when sent to the parser. Combining the two provides additional context to both human analysts and machine classifiers.
- *normal sentence*: These lines represent statements that are not considered titles, list starts, or list elements.

Step 2: Process Natural Language

After identifying the different sentence types, the process parses each line (sentence) with the Stanford Natural Language Parser (NLP) and outputs a graph in the Stanford Type Dependency Representation (STDR) [16]. While the parser has several output formats available, we choose the STDR because it incorporates the sentence's syntactic information in a concise and usable format.

document	→	*line*
line	→	*listID* title *line* | title *line* | *sentence line* | λ
sentence	→	normalSentence | listStart (":" | "-") *listElement*
listElement	→	*listID sentence listElement* | λ
listID	→	*listParanID* | *listDotID* | number
listParanID	→	"(" *id* ")" *listParanID* | *id* ")" *listParanID* | λ
listDotID	→	*id* "." *listDotID* | λ
id	→	letter | romanNumeral | number

Figure 2. Document Grammar

To replace shorthand or remove text that the parsing would not recognize, the process allows for a series of regular expressions to be applied to the text. Specifically in our work, we use this mechanism to replace 'w/' with 'with' and '/' with 'or.' Additionally, As the Stanford Parser processes sentences, it tags each word with a part of speech. Due to differences in text used to train the parser versus text used by our process, the parts of speech may be incorrect. To overcome this issue, we inserted a custom method into the parsing pipeline to override the part of speech tags if they are incorrect. For instance, we discovered that the parser always tagged "displays" as a plural noun whereas in most sentences in our text "displays" is a verb. Both overrides are configurations established at the time the tool starts and are applied without user intervention to the entire text.

Fig. 3 demonstrates the produced STDR for the sentence "a nurse can order a lab procedure for a patient." Each vertex contains a word from the sentence along with that word's part of speech. In the figure, "DT" represents a determiner, "MD" a modal verb, "NN" a noun, and "VB" indicates a verb. Edges represent the grammatical relationship between two words. For instance, "nurse" functions as the nominal subject (nsubj) for "order" and "dobj" is the object to be ordered.

From the STDR generated by the parser, we create our sentence representation (SR) as ACRE needs to track additional attributes for the sentence and for each word. Additionally, some words in the original sentence are not required for our purposes and, hence, removed from the SR.

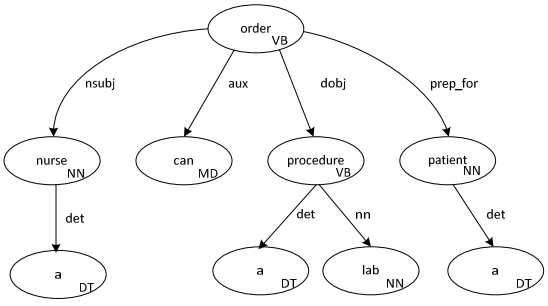

Figure 3: Stanford Collapsed Type Dependency Graph

Fig. 1 shows our corresponding SR for the same sentence as in Fig 3. The primary differences between the two graphs are the number of vertices and how each word is represented

within a vertex. Within our SR, vertices correspond to words in the sentence and contain the word, the word's lemma, part of speech, domain flag, and access control policy indicators. The indicators correspond to the subject("S"), action("A"), resource("R") typically defined within an access control tuple.

Using a pre-order traversal, the process creates the SR from the Stanford graph. As each vertex is created, we make two changes to the nodes. First, to avoid multiple versions of the same word, we use the lemma of the original word. Second, to avoid differences in the part of speech, we collapse the parts of speeches for all nouns and verbs to their base category. For example, we treat all plural nouns and proper nouns as just nouns. Similarly, verbs with different tenses are treated collectively as a single group. We use a very small stop word list to remove common determiners [2] from the SR. Additionally, we check if it is feasible to collapse adjective and noun modifiers into parent noun nodes. Fig. 1 demonstrates this collapsing as we combined "lab" and "procedure". By removing extraneous nodes from the SR, we reduces the overall size of each graph, which in turn, provides fewer irrelevant attributes to a machine learning algorithm and provides for more concise patterns to be used in extracting access control policies.

Step 3: Classify Sentence as Access Control or not

After Step 1 and Step 2 are both completed, a machine-learning algorithm classifies a sentence as access control or not. If the sentence does not express an access control policy, we perform no further analysis on it.

The process uses a combination of a k-NN classifier, naïve Bayes, and an SVM classifier. In prior work [17], we found that if we used a similarity threshold for the nearest neighbor(s) to determine whether or not to provide a classification answer, the k-NN classifier F_1 performance would be 1.0 (no misclassifications), although not all of the sentences would be classified. As such, we decide to utilize multiple machine learning algorithms to produce the final classification result. If the k-NN classifier's threshold is below a certain ratio (0.6) based upon the computed distance to the nearest neighbor(s) compared to the length of the sentence, we return the k-NN classifier's answer. Otherwise, we return a majority vote of the k-NN, naïve Bayes, and SVM classifiers. We term this classifier as "Combined SL."

Step 4: Extract Access Control Elements

Next, we need to extract the subject, action, and resource elements from the SR. We utilize a relation extraction approach for the identification of access control elements and subsequent extraction of the process. The approach follows a well-known bootstrapping technique [5], but has been adapted specifically for access control policy extraction.

To initialize the process (presented in Fig. 5), we seed a set of ten patterns with each pattern consisting of just three nodes. Each pattern is the same, except a different verb[3] is utilized for "Specific Action". The subjects and resources are marked with wildcards such that the pattern can match any nouns in sentences. We initially choose the words "create", "retrieve", "update", and "delete" because the words are commonly associated with viewing and manipulating data. We then examined the frequency of all verbs within the document and chose to add six more verbs associated with data and appearing with high frequencies within the document. From these patterns, we match all occurrences of the subjects and resources within the document along with their associate frequency counts. From the counts, we computed the mean values for the subjects and resources. We then assume any word that occurs more than the mean legitimately belongs to the application domain. Without a threshold, the potential for misidentified subjects and resources is much greater as any word matching the pattern would be accepted.

The subjects and resources are then stored in a listing of known subjects and resources. From this listing, we then search the document where any subject exists along with any resource. For each sentence that does match the condition, we extract the dependency pattern between subject and resource vertices. We then assume any verbs existing in that pattern are the actions. If more than one verb exists in the shortest path from the subject to the object, we combine the verbs. In the sentence, "the administrator chooses to create a new patient", we combine "choose" and "create" to "choose create" for the action. The subject would be "administrator" and the object would be "patient". We derive permissions for each pattern based finding the closest synonym in WordNet [4] where permissions have already been defined in the process for an action.

Once we have extracted the pattern, we also apply a series of transformations to extract additional patterns that may locate additional access control policies. Specifically, we transform patterns that have an active voice into passive voice and vice versa. We also transform the patterns to assume conjunctions may exist for two or more subjects, two or more actions, and two or more resources. From the pattern set, we then search the documents for any sentences matching one or more patterns. Once we find any match, we check to see if one of the other patterns matches the same sentence. If more than one pattern does match and one pattern can be considered a "sub-pattern" of another pattern, we discard the "sub-pattern" match from the list of results as the other graph has provided a more specific match. Additionally, we check the matched sentences for any children nodes of the matched pattern that imply negativity or subject limitation (i.e., we look to see if there is relevant indicator just outside of the match subgraph).

The extracted access policy is then stored in a list for validation and output to the user. Any new subjects or resources are then added to the list of known subjects and resources. If newly discovered subjects or resources exist, then the bootstrapping process can repeat until no new items or patterns are discovered. Once the process has stopped, the user may manually identify access control patterns. The information from these patterns is feed into the process to search for additional extracted elements.

[2] a, an, the

[3] create, retrieve, update, delete, edit, view, modify, enter, choose, select

[4] http://wordnet.princeton.edu/

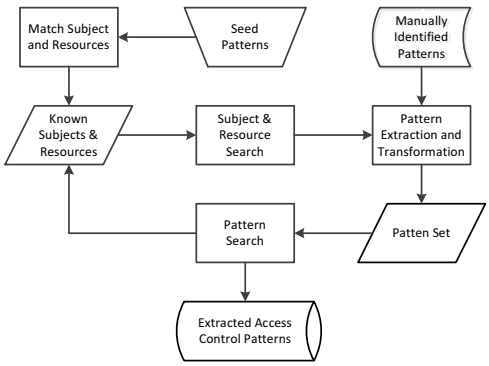

Figure 5: Access Control Extraction Overview

Step 5: Validate Access Control

In this step, the tool checks for coverage and conflicts within the extracted access control policies. Coverage is reported as measure for each subject as to the number of identified resources that it has access control rules identified. As we assume a default of no-access, 100% coverage is not required. However, low coverage values may indicate a need for further access control policies. Conflicts occur within our process when a specific subject has been both granted permission to a specific resource and restricted for the same permission on the same resource. Such conflicts may arise due to policy extraction in multiple locations or the use of a limiter to restrict access to a specific subject.

IV. Evaluation Methodology

A. Application: iTrust

To evaluate the procedure, we used iTrust as our test system. The requirements consist of 40 use cases plus additional non-functional requirements, constraints, and a glossary. The version we used contained 1159 sentences with 409 (36.7%) of those sentences determined to contain one or more access control policies.

B. Study Oracle

First we created our oracle in which we manually classified each statement in the iTrust Requirements Specification. We first converted the document into a text-only format. Next, we opened the document in the ACRE Tool to classify each sentence. The first authors classified the 1,159 sentences (or lines) in seven hours.

After the initial classification was completed, we validated the classification through several approaches. First, we used a *k*-medoids clustering algorithm to compute clusters of related sentences. We then compared the classifications within each cluster. and investigated further those sentences that did not have the same classification as other sentences in the group. Additionally, as we classified each sentence, we had access to the neighbors contained within the *k*-NN classifier. This approach allowed for more rapid manual classification by suggesting initial classification that we could then verify or correct as deemed necessary. Additionally, any discrepancies in the predicted classification could be easily traced back to the source sentences.

Next, the first author spent twelve hours to manually identify the subjects, actions, and resources for the access control policies in the 409 sentences.

C. Study Procedure

Once the oracle has been created, we executed five classifiers (the k -NN classifier, a TF-IDF classifier, the "Combined SL" classifier, a multinomial naïve Bayes classifier and a SVM - sequential minimal optimization classifier) on the requirements document. For each classifier considered, we tested using a stratified *n*-fold cross-validation and computed the precision, recall, and F_1 measure. We follow Han et al.'s recommendation [18] and use 10 as the value for *n* as this produces relatively low bias and variance. The cross-validation ensures that all sentences are used for training and that each sentence is tested just once.

In the final phase of the study, we examined seeding the process with different sets of initial actions (verbs). From the patterns generated, we extracted the access control policies from the requirements document and compared the extracted to the manually identified policies.

V. Expiremental Results

RQ1: How effectively can we identify access control policies in natural language text in terms of precision and recall?

Table 1 presents the results of executing each classifier against the entire document set using a ten-fold cross validation. We executed each test three times and present the average.

Table 1. Stratified Ten-Fold Cross Validation

Classifier	Precision	Recall	F_1 Measure
Naïve Bayes	.743	.940	.830
SMO	.845	.830	.837
TF-IDF	.588	.995	.739
k-NN (*k*=1)	.851	.830	.840
Combined SL	.873	.908	.890

Creating the "Combined SL" classifier did produce some performance gains from using individual classifiers as the F_1 Measure was .05 higher than the next best performer (*k*-NN, *k*=1). For the *k*-NN classifier, we did experiment with various values for *k* and found one produced the best performance.

RQ2: What common patterns exist in sentences expressing access control policies?

By examining the most frequently occurring patterns from our manual identification, we found that the seed pattern occurred in 25% of the sentences marked for access control. This occurrence doesn't require a small sentence, but rather somewhere in the sentence we found three nodes and two edges of that pattern. Another frequently occurring pattern (8%) occurs with sentences start with "The *subject* [chooses|selects] to *perform action."* A wide range of patterns existed due to differences in prepositions represented on edges.

RQ3: What is an appropriate set of seeded graphs to effectively bootstrap the process to extract the access control elements?

To evaluate this question, we started the process with different sets of base words and then compared the quantity of patterns generated and the performance of those patterns to extract the correct access control statements from the sentence. The best performance came with the set of 10 action verbs defined for the seed with a precision of .463 and a recall of .536.

VI. Limitations

Several limitations exist within this work. As ACRE utilizes NLP techniques, the process and associated tool cannot extract information contained in images. With regards to access control policies, our bootstrapping approach does not take into account the presence of contextual information or conditions that may affect the generated access control. The user can manually enter such information, though. We also assume all necessary information for an access control policy is contained within the same sentence. It is feasible for elements either to exist in surrounding sentences. We also have not handled resolution issues at this time. These issues occur when a pronoun or generic term such as "system" or "data" is used in place of a descriptive term. Our work has a significant external validity threat as we examined only one document for one system in a specific problem domain. While the process does not have any specific problem domain constraints, additional evaluation needs to occur across multiple domains and applications.

An internal validity threat may exist as the first author performed all of the sentence classifications and access control policies. To check the accuracy of the first author's classifications, we had five software developers classify a representative sample of 30 sentences. Utilizing Randolph's Online Kappa Calculator [19], we calculate a free-marginal kappa of 0.86 (indicating significant inter-rater agreement) by comparing the first authors classification against the majority vote of the other raters.

VII. Conclusion and Future Work

For future work, we plan to continue work on the tool to resolve the resolution issues presented in the previous section. We also plan to develop a much larger corpus of text documents for multiple systems in two or three domains. Utilizing the corpus, we can more effectively measure how the process performs and would generalize to other systems and problem domains.

In this paper, we present a new process, ACRE, and tool that assist developers in automatically extracting access control policies from natural language text. The tool provides a mechanism for developers to quickly generate an initial set of access control policies with traceability back to the originating set. Developers can utilize the process to detect conflicts in generated policies as well as evaluate the coverage of generated policies to the identified subjects and resources. We demonstrated how effective a bootstrapping process can extract policies from a very small initial set of patterns. We also presented a grammar that can be applied when parsing text documents to provide additional context information for specific elements within the document.

As we utilized a cross-fold validation on a single document, our combined classifier showed very effective performance with a F_1 Measure of .89. We also showed improved performance through combine the results of multiple classifiers. However, our performance in extracting access control patterns was substantially less with a precision of .463 and a recall of .536.

Acknowledgment

This work was supported by the U.S. Army Research Office (ARO) under grant W911NF-08-1-0105 managed by NCSU Secure Open Systems Initiative (SOSI).

References

[1] J. Bedard, R. Hoitash, U. Hoitash, and K. Westermann, "Material Weakness Remediation and Earnings Quality: A Detailed Examination by Type of Control Deficiency," *Auditing: A Journal of Practice & Theory*, 2012.

[2] "2011 CWE/SANS Top 25 Most Dangerous Software Errors," 2011. [Online]. Available: http://cwe.mitre.org/top25/. [Accessed: 14-Nov-2011].

[3] Verizon RISK Team, "2013 Data Breach Investigations Report," 2013.

[4] P. Samarati and S. di Vimercati, "Access Control: Policies , Models , and Mechanisims," in in *Foundations of Security Analysis and Design*, Springer Berlin / Heidelberg, 2001, pp. 137–196.

[5] D. Jurafsky and J. Martin, *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*, Second. Pearson, 2009, p. 988.

[6] A. Meneely, B. Smith, and L. Williams, "iTrust Electronic Health Care System: A Case Study," in in *Software System Traceability*, 2011.

[7] Q. He and A. I. Antón, "Requirements-based Access Control Analysis and Policy Specification (ReCAPS)," *Information and Software Technology*, vol. 51, no. 6, pp. 993–1009, Jun. 2009.

[8] X. Xiao, A. Paradkar, S. Thummalapenta, and T. Xie, "Automated Extraction of Security Policies from Natural-Language Software Documents," in *International Symposium on the Foundations of Software Engineering (FSE)*, 2012.

[9] R. Schwitter, "Controlled Natural Languages for Knowledge Representation," in *Proceedings of the 23rd International Conference on Computational Linguistics*, 2010, pp. 1113–1121.

[10] C. a. Brodie, C.-M. Karat, and J. Karat, "An Empirical Study of Natural Language Parsing of Privacy Policy Rules Using the SPARCLE Policy Workbench," *Proceedings of the second symposium on Usable privacy and security - SOUPS '06*, p. 8, 2006.

[11] L. Shi and D. Chadwick, "A Controlled Natural Language Interface for Authoring Access Control Policies," in *Proceedings of the 2011 ACM Symposium on Applied Computing*, 2011, pp. 1524–1530.

[12] M. Hearst, "Automatic acquisition of hyponyms from large text corpora," in *Proceedings of the 14th conference on Computational Linguistics*, 1992, pp. 539–545.

[13] R. Snow, D. Jurafsky, and A. Y. Ng, "Learning Syntactic Patterns for Automatic Hypernym Discovery," in *Advances in Neural Information Processing Systems 17*, 2004, vol. 17, pp. 1297–1304.

[14] A. Akbik and J. Broß, "Wanderlust: Extracting semantic relations from natural language text using dependency grammar patterns," in *Workshop on Semantic Search*, 2009, vol. 491.

[15] K. Fundel, R. Küffner, and R. Zimmer, "RelEx--relation extraction using dependency parse trees.," *Bioinformatics (Oxford, England)*, vol. 23, no. 3, pp. 365–71, Feb. 2007.

[16] M.-C. de Marneffe, B. MacCartney, and C. Manning, "Generating Typed Dependency Parses from Phrase Structure Parses," *Proceedings of Language Resources and Evaluation*, pp. 449–454, 2006.

[17] J. Slankas and L. Williams, "Classifying Natural Language Sentences for Policy," *2012 IEEE International Symposium on Policies for Distributed Systems and Networks*, pp. 33–36, Jul. 2012.

[18] J. Han, M. Kamber, and J. Pei, *Data Mining: Concepts and Techniques*, 3rd ed. Morgan Kaufmann, 2011, p. 744.

[19] J. J. Randolph, "Online Kappa Calculator," 2008. [Online]. Available: http://justusrandolph.net/kappa/. [Accessed: 02-Apr-2013].