An ontology for secure e-government applications

M. Karyda[1], T. Balopoulos[1], S. Dritsas[2], L. Gymnopoulos[1], S. Kokolakis[1], C. Lambrinoudakis[1], S. Gritzalis[1]*

[1]Laboratory of Information and Communication Systems Security (Info-Sec-Lab), Department of Information and Communication Systems Engineering, University of the Aegean, Samos, GR-83200, Greece, {mka, tbalopoulos lazaros.gymnopoulos, sak, clam, sgritz}@aegean.gr

[2]Department of Informatics, Athens University of Economics and Business, GR-10434, Greece, sdritsas@aueb.gr

*Corresponding author

ABSTRACT

This paper addresses the issue of accommodating security requirements in application development. It proposes the use of ontologies for capturing and depicting the security experts' knowledge. In this way developers can exploit security expertise in order to make design choices that will help them fulfill security requirements more effectively. We have developed a security ontology for two different application scenarios to illustrate its use. To validate the ontology we have used queries.

KEYWORDS

security ontology, application development, e-government

1. INTRODUCTION

The increasing need to employ secure applications has driven researchers to explore possible solutions for incorporating security features to applications as early as possible in the design and implementation processes. However, acquaintance and selection of the required security features and mechanisms often pose severe difficulties both to application developers and system owners.

This paper presents a relatively new approach to developing secure applications. This approach entails the use of a security ontology that can facilitate the communication between security experts, users and developers. The rest of the paper is structured as follows: Section two elaborates on security ontologies, and section three describes two different application contexts where security ontologies can be employed. Section four uses the two instantiations of the security ontology to present a set of exemplary cases of its use. Finally, the last section summarizes the conclusions, the major limitations, and the potential of this approach.

2. SECURITY ONTOLOGIES

In the domain of knowledge sharing, an ontology is an "[*e*]*xplicit specification of a conceptualization*" [1]. Thus, an ontology is the attempt to express an exhaustive conceptual scheme within a given domain, typically a hierarchical data structure containing all the relevant entities, their relations and the rules within that domain.

In the context of this paper, the domain of a *security ontology* comprises of security-related issues. The purpose of the security ontology is to facilitate software developers to better understand the domain of their software, so that they can be able to address its security requirements early in the software development process, and make informed choices as far as security solutions and mechanisms are concerned.

2.1 Related Work

To the best of our knowledge, there is no related work that aims to develop a security ontology to be used as a common base for the development of secure applications. Other related work focuses only on access control issues [2]. KAON [3] focuses mostly on the managing infrastructure of generic ontologies and metadata, whereas in [4] authors present a policy-ontology. Raskin et al. describe an ontology-driven approach to information security [5] in order to organize security attacks and support the reaction to these attacks by relating certain controls with specific attack characteristics, as well as attack prediction. The KAoS Policy and Domain Services is another approach based on ontologies for the representation of security related concepts [6].

2.2 Development Method

To develop the security ontology used in this paper, we followed the steps provided in [7]; emphasis was given in the iterative process proposed therein. Each cycle in this procedure has roughly four phases: determining competency questions, enumerating important terms, defining classes and class hierarchy, and instantiating.

Competency questions are loosely structured questions that a knowledge base built on the ontology should be able to answer. Setting and elaborating on competency questions is an efficient way to identify and then focus on the desired area. Enumerating important terms within the scope set by competency questions and the respective answers is a prerequisite for defining ontology classes.

We gathered a large number of related terms. The most important of them, formed ontology classes; others formed properties of classes and some were not used at all.

In the next phase, classes and the class hierarchy were developed. There are three different approaches in developing a class hierarchy: the top-down approach, the bottom-up approach, and a combination of the two. To develop a class hierarchy following the top-down approach, one starts with the definition of the most general concepts in the domain and proceeds with the specialization of the concepts. Bottom-up development, on the other hand, starts with the definition of the most specific classes that constitute the leaves of the hierarchy, while grouping of these classes into more general concepts follows.

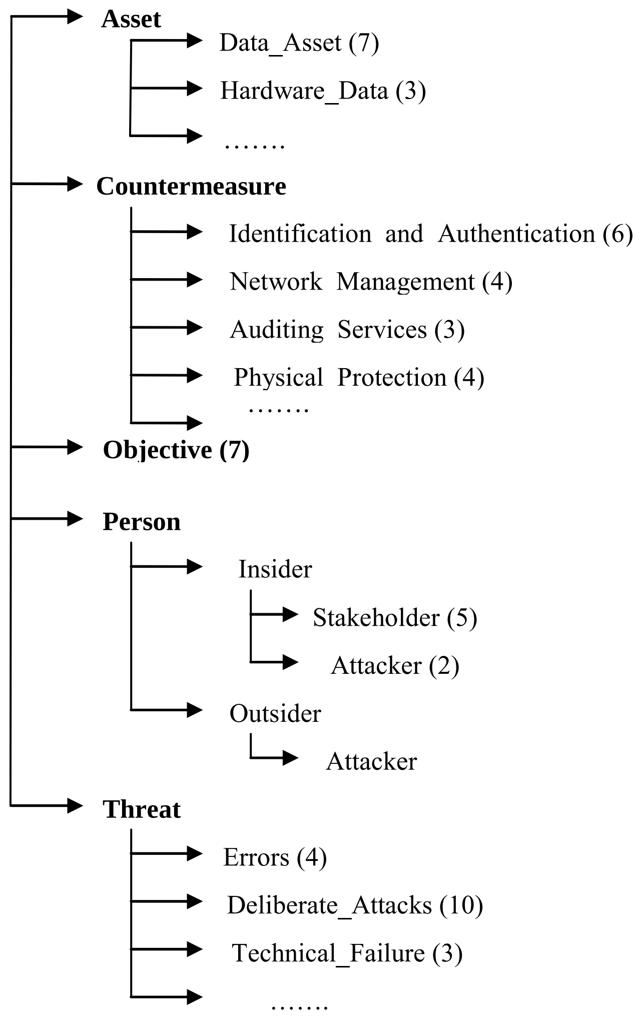

Figure 1: The ontology hierararchy

To develop the ontology presented in this paper, we found that using a combination of the two approaches described above was the most effective approach: our rich set of competency questions fitted well with the top-down approach and resulted in a class hierarchy close the final. Then the bottom-up approach was employed to fit in the remaining concepts.

After developing the class hierarchy, the next step was to come up with the slots (class properties) and their facets (slot properties, such as value type, allowed values and cardinality). Some of the question that were used in this phase, are included in Section 4. The last phase in each cycle was that of instantiation; each class was given specific instances. Two e-government domains were chosen: an e-tax and an e-vote domain (see Section 3). This gave rise to two instantiated ontologies sharing the same schema.

The four phases described above were repeated several times. The ontology was validated after each iteration (see Section 4), and iterations ended only when the results obtained were considered satisfactory.

Figure 1 depicts a part of the developed ontology hierarchy; numbers in parenthesis give the number of instances for each leaf-level class. The classes of the depicted ontology, i.e. Asset, Countermeasure, Objective, Person etc. and their corresponding subclasses cover the basic concepts that describe the context of a secure application. For example, the class Objective has been instantiated with seven different types (or values), which are not depicted; these values include, among others the terms Availability, Data_Confidentiality, Data_Integrity and other security related objectives, or requirements, that secure applications typically need to fulfil.

3. APPLICATION SCENARIOS

Most applications nowadays need to exhibit some fundamental security features, e.g. authorization/ authentication. In some cases, however, security is of critical importance, not only for the functionality of the application, but also to provide a trust environment for end users, who would otherwise refrain from using these applications. Such cases include, for example, applications used to provide electronic government services. The next paragraphs describe two such application scenarios, that are encountered increasingly often by developers and users.

3.1 Developing e-tax applications

E-government aims to provide citizen-oriented services, and to address the demands of citizens and businesses for accessibility, responsiveness, simplicity and transparency of public services.

Applications that provide e-government services need to have increased security and privacy features. A typical case in this domain is, for example, the provision of on-line tax paying and managing services to citizens. The latter will be able to make, from their home, declarations of income and manage their tax obligations. The tax collecting authority should therefore be in position to verify income declarations, to make cross-checks and to ask additional information from taxpayers. All this communication should be protected in terms of its confidentiality, privacy and integrity.

The process involved in such a scenario can be the following:

- Users manage their personal information.
- Users view the declarations they have made in previous years.
- Users process their income tax declaration.
- The system notifies the users that additional information is needed.
- Users update their tax declaration with the needed information.

In this scenario, the major security requirements that e-taxing applications need to meet are authentication, authorization, privacy, auditing and integrity.

3.2 Developing e-voting applications

Regional administration strives to use e-government applications, so as to enhance citizens' participation and accessibility to public services. Participating in local elections from their home or workplace has been made possible for many citizens in numerous countries, including France, Switzerland, Estonia, the US, Brazil and Italy. In this way, participation in the election process is facilitated for many people, especially those facing difficulties to leave their home or work.

A possible application scenario would include the following processes:

- Local officials set up the election process.
- Independent bodies can audit/supervise the election process.
- The set of eligible voters is established.
- Eligible voters have access to the electronic election process.
- Eligible voters cast their votes within the specified election time framework, following the designated procedures.
- Vote tally can be verified.

The security challenges associated with this case are several: Firstly, voters need to trust the electronic voting system. Secondly, the election organizers need to be assured that the voting application is trustworthy and does not perform any unwanted functionality. Thirdly, voters need to be assured that their votes remain secret and that they have been calculated in the right way. Fourthly, organizers need to make sure that only eligible voters have voted, that they have voted in the way they are supposed to, and that the votes have been tallied correctly. Finally, all stakeholders want to be sure that no unauthorized entities can have access to the electronic voting application.

According to this scenario, the major security requirements that e-voting applications need to meet are integrity, auditing, confidentiality, traceability, authentication and authorization.

4. USING THE SECURITY ONTOLOGY

This section illustrates how the security ontology for the application scenarios described in the previous section can facilitate the development of secure applications. It also illustrates how the ontology was validated during its development.

The tools used for developing and querying the security ontologies were Protégé and Racer. The Protégé Ontology Editor [8] has a modular design and itself provides only basic functionality; numerous plug-ins exist depending on the task in hand. For our work, we chose the Protégé plug-in, which targets OWL [9] and RDF [10] ontologies. Racer [11] is an inference engine that can be used for query answering over RDF documents. Racer was used to statically check our ontologies for inconsistencies, and for submitting queries in order to verify their validity. The queries were expressed in the new Racer Query Language (nRQL).

nRQL is a description logic query language for retrieving individuals from an A-box (a set of assertions about individuals) according to specific conditions. It allows the use of variables within queries which are bound against those A-box individuals that satisfy these conditions. A description of nRQL's syntax is beyond the scope of this paper, but the interested reader is referred to [12]. The communication between Protégé and Racer was made possible through the RQL Tab plug-in.

An indicative set of nRQL queries with their answers is provided in the following, illustrating the use of the security ontology.

4.1 nRQL Queries and Results

For an ontology to be considered useful, it must give consistent answers to real-world questions. This section lists a number of questions a developer involved in an e-tax or an e-vote application development project is likely to come up with. These questions should not be regarded as exhaustive, but as indicative of what the ontologies can deal with and reason about. Each of the questions is firstly expressed formally as an nRQL query, then the result of executing this query is presented, and furthermore, where appropriate, the result is commented upon, or justified. The questions which guided the nRQL queries were used in the iterative development of the hierarchy ontology, while the nRQL queries were used for validating the ontology.

4.1.1 Example Queries: e-Tax Environment

Q1. Which are the typical objectives of an e-tax system?

```
nRQL     (retrieve (?obj) (?obj |Objective|))
Query:
nRQL     (((?OBJ |Data_Confidentiality|))
Result:   ((?OBJ |Availability|))
          ((?OBJ |Data_Integrity|))
          ((?OBJ |User_Eligibility|))
          ((?OBJ |User_Accountability|))
          ((?OBJ |User_Non_Repudiation|))
          ((?OBJ |Accuracy|)))
```

Q2. Which assets are confidential in an e-tax system?

```
nRQL     (retrieve (?asset) (and
Query:   (|Confidentiality| ?threat
         |is_threatened_by|) (?asset ?threat
         |damaged_by|)))
nRQL     (((?ASSET |Tax_Data|))
Result:   ((?ASSET |Personal_Data|))
          ((?ASSET |Cryptographic_Keys|)))
```

To answer this question, we first find the possible threats to the confidentiality objective and then list the assets that may be damaged by these threats. For example, confidentiality is threatened by user errors; and a user error may disclose the user's cryptographic keys.

Q3. Which countermeasures protect the personal data of a tax-paying citizen?

```
nRQL     (retrieve (?cm)
Query:    (|Voter_Data| ?cm |protected_by|))
nRQL     (((?CM |Encryption|))
Result:   ((?CM |Access_Control|))
          ((?CM |Certificates|))
          ((?CM |Intrusion_Detection|))
          ((?CM |Malicious_SW_Detection|)))
```

Q4. Which countermeasures address threats that are realized by vandals?

```
nRQL     (retrieve (?cm) (and (?cm ?threat
Query:   |address|) (|Vandal| ?threat
         |realizes|)))
nRQL     (((?CM |Access_Control|))
Result:   ((?CM |Firewall|))
          ((?CM |Backup_Policy|))
          ((?CM |OS_Security_Updates|))
          ((?CM |Intrusion_Detection|)))
```

4.1.2 Example Queries: e-Vote Environment

Q5. Which are the typical objectives of an e-vote system?

```
nRQL     (retrieve (?obj) (?obj |Objective|))
Query:
nRQL     (((?OBJ |Voter_Anonymity|))
Result:   ((?OBJ |Vote_Confidentiality|))
          ((?OBJ |Availability|))
          ((?OBJ |Vote_Integrity|))
          ((?OBJ |Voter_Eligibility|))
          ((?OBJ |Voter_Accountability|))
          ((?OBJ |Accuracy|)))
```

Q6. Which assets are confidential in an e-vote system?

```
nRQL     (retrieve (?asset) (and
Query:   (|Vote_Confidentiality| ?threat
         |is_threatened_by|) (?asset ?threat
         |damaged_by|)))
nRQL     (((?ASSET |Voter_List|))
Result:   ((?ASSET |Voter_Data|))
          ((?ASSET |Voter_Credentials|))
          ((?ASSET |Vote|))
          ((?ASSET |Cryptographic_Keys|)))
```

Q7. Which threats might compromise the anonymity of a voter?

```
nRQL     (retrieve (?threat)
Query:   (|Voter_Anonymity| ?threat
         is_threatened_by|))
nRQL     (((?THREAT |Impersonation|))
Result:   ((?THREAT |Malicious_Code|))
          ((?THREAT |User_Error|))
          ((?THREAT |OS_Bugs|))
          ((?THREAT |Application_Bugs|))
          ((?THREAT |Terminal_Highjack|)))
```

Q8. Which countermeasures can protect against vote replay?

```
nRQL     (retrieve (?cm)
Query:    (?cm |Vote_Replay| |address|))
nRQL     (((?CM |Identification|))
Result:   ((?CM |Authentication|))
          ((?CM |Auditing|)))
```

Using identification and authentication we can audit the persons that have voted. To prevent them from voting again, we need to check the audit before accepting any vote.

Q9. Which threats are present in a mixnet-based voting scheme?

```
nRQL     (retrieve (?threat) (|Mixnet|
Query:   ?threat |damaged_by|))
nRQL     (((?THREAT |Vote_Selling|))
Result:   ((?THREAT |DoS_Attack|)))
```

In mixnet schemes [13], when the domain of the possible votes is sufficiently large, a voter may effectively uniquify his/her vote (e.g. by altering the vote's low-significance bits) and sell it to a buyer who had pre-chosen it. Furthermore, as mixnet schemes operate, they necessarily perform a massive amount of communication between the different parties. This makes them much more vulnerable to a denial-of-service attach than other schemes.

This section demonstrated how the security ontology can be exploited in the development of complex security critical applications. Application development can be thus be facilitated through the use of ontologies. Developers can create hierarchy ontologies for different application contexts, as this paper has shown, by collaborating with experts in each domain.

5. CONCLUSIONS

Ontologies provide an effective mechanism to capture, describe and exploit knowledge and practice in the wide and rapidly evolving area of security. This paper presents the use of a security ontology, instantiated in two different application contexts in the area of electronic government, for developing security critical applications.

One drawback of the approach presented in this paper is that, developing and maintaining ontologies adds workload to the process of application development. However, once a security ontology has been developed, it can be instantiated and applied in different contexts, as this paper has showed. Moreover, it can assist developers to save time and make better choices in applying security features, since it allows them to exploit security expertise and accumulated knowledge.

ACKNOWLEDGMENTS

This work was co-funded by 75% from the European Union and 25% from the Greek Government, under the framework of the "EPEAEK: Education and Initial Vocational Training Program—Pythagoras".

REFERENCES

1. Gruber T. Toward principles for the design of ontologies used for knowledge sharing, in Formal Ontology in Conceptual Analysis and Knowledge Representation, Kluwer (1993).
2. Denker, G., Access Control and Data Integrity for DAML+OIL and DAML-S, SRI International, USA, (2002)
3. Bozsak, E., Ehrig, M., Handschub, S., Hotho: KAON - Towards a Large Scale Semantic Web. In: Bauknecht, K.; Min Tjoa, A.; Quirchmayr, G. (Eds.): Proc. of the 3rd International Conference on E-Commerce and Web Technologies, (2002), pp. 304-313
4. Kagal, L., Finin, T., Joshi, A.: "A policy language for a pervasive computing environment". In IEEE 4th International Workshop on Policies for Dis¬tributed Systems and Networks, (2003)
5. Raskin, V., Hempelmann, C., Triezenberg, K., and Nirenburg, S.: Ontology in Information Security: A Useful Theoretical Foundation and Methodological Tool. In Viktor Raskin and Christian F. Hempelmann, editors, Proceedings of the New Security Paradigms Workshop, New York. ACM, (2001)
6. Uszok, A., Bradshaw, J., Jeffers, R., Suri, N., Hayes, P., Breedy, M., Bunch, L., Johnson, M., Kulkarni, S. and Lott, J. KAoS Policy and Domain Services: Toward a Description-Logic Approach to Policy Representation, Deconfliction and Enforcement. In Proceedings of the IEEE Workshop on Policy (2003)
7. Noy, N.F. and Mc Guinness, D.L. "Ontology Development 101: A Guide to Creating Your First Ontology", Stanford Knowledge Systems Laboratory Technical Report KSL-01-05. (2001)
8. Protégé, http://protege.stanford.edu/
9. Web Ontology Language (OWL), http://www.w3.org/2001/sw/WebOnt/
10. O. Lassila, Ralph Swick (eds).: Resource Description Framework (RDF) Model and Syntax Specification, W3C Recommendation 22 February 1999, http://www.w3.org/TR/REC-rdf-syntax/
11. Racer Inference Engine, http://www.sts.tu-harburg.de/~r.f.moeller/racer/
12. The New Racer Query Language, http://www.cs.concordia.ca/~haarslev/racer/racer-queries.pdf
13. Smith, W.: Cryptography Meets Voting, http://www.math.temple.edu/~wds/homepage/cryptovot.pdf