Employing Ontologies for the Development of Security Critical Applications

The Secure e-Poll Paradigm

Stelios Dritsas, *Lazaros Gymnopoulos*,
Maria Karyda, Theodoros Balopoulos,
Spyros Kokolakis, Costas Lambrinoudakis,
Stefanos Gritzalis

Funding: European Union (75%) and the Greek Government (25%), Education and Initial Vocational Training Programme (EPEAEK) – Pythagoras

General project information

- *"Pythagoras"* Programme for the Support of Research Units in Greek Universities
  - **Specific project title:** "Embedding Security Patterns in Software Development"
  - **Project duration:** 2004-2006
  - **Project papers – more to follow:**
    - Incorporating Security Requirements into the Software Development Process - 4th ECIW conference (2005)
    - Employing Ontologies for the Development of Security Critical Applications – 5th I3E Conference (2005)

- Funding: European Union (75%) and the Greek Government (25%)

Structure of the Presentation

- Introduction
- Ontology and Developing Secure Software
  - Ontology
  - Secure software development
  - Security ontology
- Research Methodology
  - Building a secure application ontology
  - Methods and tools
- Case Study: The Secure e-Poll Paradigm
  - Remote internet voting: The e-poll application environment
  - The e-poll application
  - Secure e-poll ontology
  - nRQL queries and results
- Conclusions and Further Research

Introduction

- **Problem:** Need for secure applications
- **Proposed solution:** Use of ***security ontologies*** during the software development process

- **Implementation** in:
  - e-Government
    - e-Voting
      - ***Remote internet voting*** (security sensitive application environment)

Structure of the Presentation

- Introduction
- Ontology and Developing Secure Software
  - Ontology
  - Secure software development
  - Security ontology
- Research Methodology
  - Building a secure application ontology
  - Methods and tools
- Case Study: The Secure e-Poll Paradigm
  - Remote internet voting: The e-poll application environment
  - The e-poll application
  - Secure e-poll ontology
  - nRQL queries and results
- Conclusions and Further Research

Ontology

"an ontology is the attempt to express an exhaustive conceptual scheme within a given domain"

- In computer science: ontology is used as a means for modelling information

Structure of the Presentation

- Introduction
- Ontology and Developing Secure Software
  - Ontology
  - Secure software development
  - Security ontology
- Research Methodology
  - Building a secure application ontology
  - Methods and tools
- Case Study: The Secure e-Poll Paradigm
  - Remote internet voting: The e-poll application environment
  - The e-poll application
  - Secure e-poll ontology
  - nRQL queries and results
- Conclusions and Further Research

Secure software development

- It is generally accepted that security should be "*built-in*" rather than "*added-on*"
- A number of methodologies exist that try to handle security issues in the design level but they
  - make no reference as to how security requirements can be translated into system components
  - do not provide a generic model of security and thus can be applied only in specific application environments
  - are rather technical

Structure of the Presentation

- Introduction
- Ontology and Developing Secure Software
  - Ontology
  - Secure software development
  - Security ontology
- Research Methodology
  - Building a secure application ontology
  - Methods and tools
- Case Study: The Secure e-Poll Paradigm
  - Remote internet voting: The e-poll application environment
  - The e-poll application
  - Secure e-poll ontology
  - nRQL queries and results
- Conclusions and Further Research

Security ontology

- A security ontology can facilitate secure applications development through
  - the provision of a common vocabulary for application developers and security experts
  - expression of security concepts and realization of their relationships and thus the provision of a generic security model
  - being implementation agnostic
- To the best of our knowledge no such ontology exists*

* See paper section 2.2 for related work details

Structure of the Presentation

- Introduction
- Ontology and Developing Secure Software
  - Ontology
  - Secure software development
  - Security ontology
- Research Methodology
  - Building a secure application ontology
  - Methods and tools
- Case Study: The Secure e-Poll Paradigm
  - Remote internet voting: The e-poll application environment
  - The e-poll application
  - Secure e-poll ontology
  - nRQL queries and results
- Conclusions and Further Research

Building a secure application ontology

- No generally accepted, robust methodology exists for developing an ontology
- Our approach:
  - Step one:
    - Determining ontology domain and scope
    - Found and used existing material
      - other ontologies
      - CRAMM database of countermeasures, etc.
  - Step two (iterative):
    - Determining competency questions
    - Enumerating important domain terms
    - Defining classes and class hierarchy
    - Instantiating defined classes
    - Querying the ontology

Structure of the Presentation

- Introduction
- Ontology and Developing Secure Software
  - Ontology
  - Secure software development
  - Security ontology
- Research Methodology
  - Building a secure application ontology
  - Methods and tools
- Case Study: The Secure e-Poll Paradigm
  - Remote internet voting: The e-poll application environment
  - The e-poll application
  - Secure e-poll ontology
  - nRQL queries and results
- Conclusions and Further Research

Methods

- ***Competency questions*** are loosely structured questions that a knowledge base based on the ontology under production, should be able to answer
  - Example
    - Q: Are voters stakeholders of the system?
    - A: Yes
- Approximately 100 terms were ***enumerated***
  - Some formed classes, others formed properties, some were not used at all
- Classes and relations between them, the class hierarchy, and class slots along with their domain and range were ***defined***
- ***Instantiation*** was based on the CRAMM countermeasure database

Tools

We used

- **Protégé** to construct the ontology
  - It is a software tool for constructing ontologies
  - Used along with its OWL and RQL Tab plug-ins
- **Racer** to detect inconsistencies & submit queries
  - It is an inference engine for query answering over RDF documents
  - new Racer Query Language (nRQL) language used
- The RQL Tab plug-in allows the OWL plug-in to send queries to Racer and receive the answers (results)

Structure of the Presentation

- Introduction
- Ontology and Developing Secure Software
  - Ontology
  - Secure software development
  - Security ontology
- Research Methodology
  - Building a secure application ontology
  - Methods and tools
- Case Study: The Secure e-Poll Paradigm
  - Remote internet voting: The e-poll application environment
  - The e-poll application
  - Secure e-poll ontology
  - nRQL queries and results
- Conclusions and Further Research

Remote internet voting: The e-poll application environment

Domain characteristics

- Voter authentication is a mandatory requirement
- There is a specific list of authorized voters
- Voters are not allowed to vote more than once
- Voters can vote from any computer connected to the internet
- Voters are presented with a predefined set of choices and/or with alternative ways of expressing opinion

Structure of the Presentation

- Introduction
- Ontology and Developing Secure Software
  - Ontology
  - Secure software development
  - Security ontology
- Research Methodology
  - Building a secure application ontology
  - Methods and tools
- Case Study: The Secure e-Poll Paradigm
  - Remote internet voting: The e-poll application environment
  - The e-poll application
  - Secure e-poll ontology
  - nRQL queries and results
- Conclusions and Further Research

The e-poll application

- It supports remote internet voting for organizations as well as for other bodies (e.g. local authorities)
- It is a distributed application
- Voters
  - must have internet access
  - visit the e-poll web site to vote
- Organizers
  - use the back office application to manage
    - voter registration
    - vote tallying
    - ballot design etc.

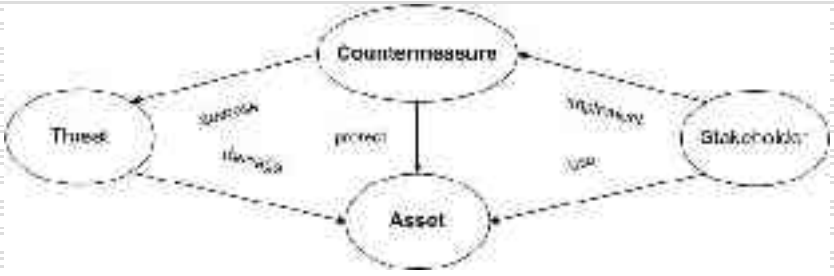
Structure of the Presentation

- Introduction
- Ontology and Developing Secure Software
  - Ontology
  - Secure software development
  - Security ontology
- Research Methodology
  - Building a secure application ontology
  - Methods and tools
- Case Study: The Secure e-Poll Paradigm
  - Remote internet voting: The e-poll application environment
  - The e-poll application
  - Secure e-poll ontology
  - nRQL queries and results
- Conclusions and Further Research

Secure e-poll ontology (1/2)

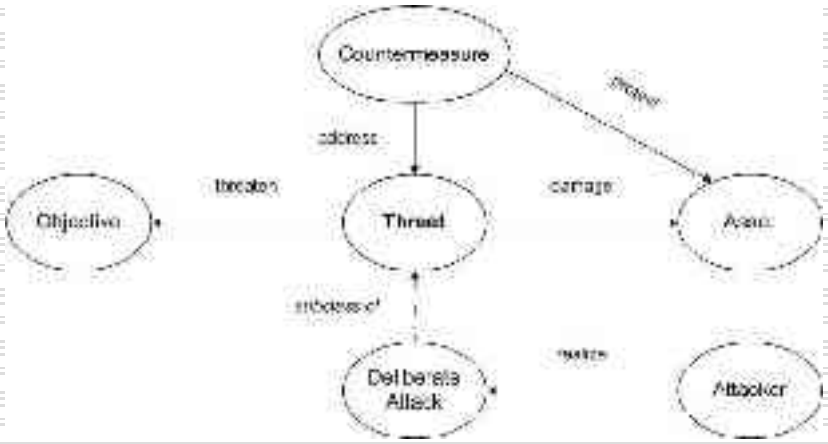
- ***Objectives*** are the desired properties of the system (e.g. vote anonymity)
- ***Stakeholders*** are the people that place value on the system (e.g. voter)
- A ***threat*** is a potential for a damage of an asset (e.g. fire)
- Objectives are ***defined*** by Stakeholders, and they are ***threatened*** by threats

- A ***countermeasure*** is an action taken to protect an asset against threats (e.g. investigation of incidents)
- ***Assets*** are pieces of information or resources upon which stakeholders place value (e.g. e-poll application server)
- A ***threat*** is a potential for a damage of an asset (e.g. fire)
- ***Stakeholders*** are the people that place value on the system (e.g. voter)
- Stakeholders ***implement*** Countermeasures while they ***use*** Assets, and Countermeasures ***address*** Threats while Threats ***damage*** Assets

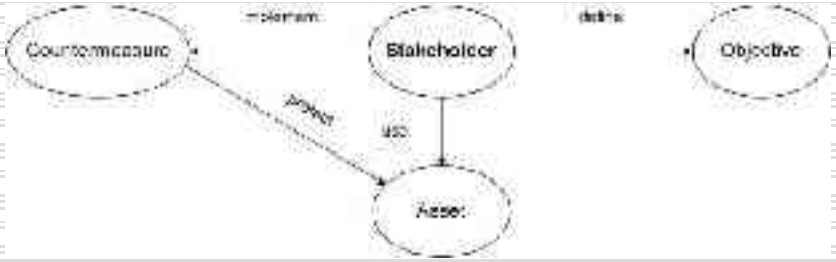


Secure e-poll ontology (2/2)

- A ***threat*** is a potential for a damage of an asset (e.g. fire)
- ***Objectives*** are the desired properties of the system (e.g. vote anonymity)
- A ***countermeasure*** is an action taken to protect an asset against threats (e.g. investigation of incidents)
- ***Assets*** are pieces of information or resources upon which stakeholders place value (e.g. e-poll application server)
- ***Deliberate Attack*** is a deliberate human action that damages an asset (e.g. vote corruption)
- An ***attacker*** is a person that deliberately damages an asset (e.g. hacker)
- Countermeasures ***address*** Threats and thus ***protect*** Assets while Threats ***threaten*** Objectives and ***damage*** Assets
- A Deliberate Attack is a ***subclass of*** the Threat class and is ***realized*** by an Attacker



- ***Stakeholders*** are the people that place value on the system (e.g. voter)
- A ***countermeasure*** is an action taken to protect an asset against threats (e.g. investigation of incidents)
- ***Objectives*** are the desired properties of the system (e.g. vote anonymity)
- ***Assets*** are pieces of information or resources upon which stakeholders place value (e.g. e-poll application server)
- A Stakeholder ***defines*** Objectives, ***implements*** Countermeasures and ***uses*** Assets (e.g. a voter uses the e-poll system to vote etc.)



Structure of the Presentation

- Introduction
- Ontology and Developing Secure Software
  - Ontology
  - Secure software development
  - Security ontology
- Research Methodology
  - Building a secure application ontology
  - Methods and tools
- Case Study: The Secure e-Poll Paradigm
  - Remote internet voting: The e-poll application environment
  - The e-poll application
  - Secure e-poll ontology
  - nRQL queries and results
- Conclusions and Further Research

nRQL queries

- An ontology gains practical value when it is able to give consistent answers to real world questions
- We constructed eight indicative, as to what the presented ontology can deal with and reason, questions
  - Questions a software developer is likely to come up when faced with an e-voting software project
- Here we will present the three most illustrative

nRQL results (1/3)

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

nRQL Query:	(retrieve (?obj) (?obj |Objective|))
nRQL Result:	(((?OBJ |Vote_Anonymity|)) ((?OBJ | Confidentiality|)) ((?OBJ |Availability|)) ((?OBJ |Integrity|)) ((?OBJ | Voter_Eligibility|)) ((?OBJ |Accountability|)) ((?OBJ |Accuracy|)))

nRQL results (2/3)

- **Q4.** Which countermeasures can prevent vote replay?

- Rational
  - We can audit the persons that have voted by using identification and authentication
  - To prevent them from voting again, we need to check the audit before accepting any vote

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

nRQL results (3/3)

- **Q7.** Which countermeasures can prevent a hacker but not a vandal?
- Rational: Operating System (OS) permissions are likely to be more effective against a hacker's objectives than against a vandal's, since the vandal's main intention is to irrevocably destroy the system than to just alter its functionality

nRQL Query:	(retrieve (?cm) (and (and (|Hacker| ?threat | realizes|) (not (|Vandal| ?threat |realizes|))) (?cm ?threat |address|)))
nRQL Result:	(((?CM |OS_Permissions|)))

nRQL results (4/4)

- Q8. Which threats are not present in an homomorphic encryption voting scheme, but are present in other voting schemes?

- Rational:
  - In mixnet schemes, when the domain of the possible votes is sufficiently large, a voter may effectively uniquify his vote (e.g. by altering the vote's low-significance bits) and sell it to a buyer who had pre-chosen it. This is much harder to do in homomorphic encryption, as only an aggregate (sum) of the votes is disclosed and not the votes themselves.
  - 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.
  - Election schemes based on secret sharing among several mutually distrustful election authorities suffer from the vulnerability that, if a sufficient number of these authorities cooperate, they can link votes to voters. The security of the other schemes is not based on trust among authorities, and hence this vulnerability does not apply to them.

nRQL Query:	(retrieve (?threat) (and (?schemes | Voting_Schemes|) (and (?schemes ?threat | damaged_by|) (not (|Homomorphic_Encryption| ? threat |damaged_by|)))))
nRQL Result:	(((?THREAT |Vote_Selling|)) ((?THREAT | DoS_Attack|)) ((?THREAT | Compromise_Anonymity|)))

Structure of the Presentation

- Introduction
- Ontology and Developing Secure Software
  - Ontology
  - Secure software development
  - Security ontology
- Research Methodology
  - Building a secure application ontology
  - Methods and tools
- Case Study: The Secure e-Poll Paradigm
  - Remote internet voting: The e-poll application environment
  - The e-poll application
  - Secure e-poll ontology
  - nRQL queries and results
- Conclusions and Further Research

Conclusions and Further Research

- Questions like the ones presented previously are likely to come up in any software design or development process concerning e-voting or e-government in general
- Designers and developers should then make critical decisions for security related issues

- We believe that the presented solution can substantially aid in making those decisions
- We should note though, that in order to address the issue of secure applications effectively and thoroughly, fully developed specialized ontologies are needed
  - Thus the presented ontology should be further developed and enhanced

- We intend to further investigate the possibilities offered by employing security ontologies in this and other security critical contexts

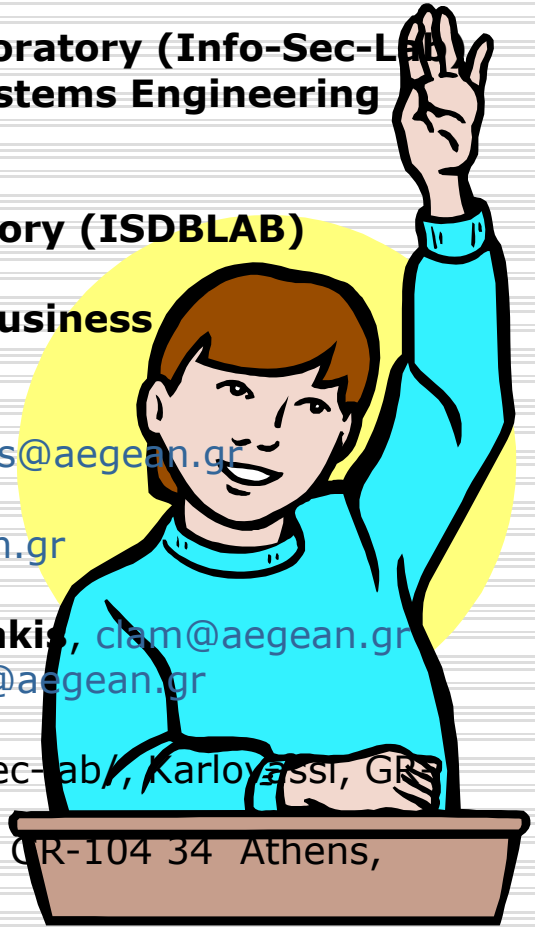
Questions – Author Contact Info.

Information & Communication Systems Security Laboratory (Info-Sec-Lab)
Department of Information & Communication Systems Engineering
University of the Aegean

Information Security and Data Bases Laboratory (ISDBLAB)
Department of Informatics
Athens University of Economics and Business

- **Mr. Stelios Dritsas**, sdritsas@aueb.gr
- **Mr. Lazaros Gymnopoulos**, lazaros.gymnopoulos@aegean.gr
- **Dr. Maria Karyda**, mka@aegean.gr
- **Mr. Theodoros Balopoulos**, tbalopoulos@aegean.gr
- **Lecturer Dr. Spyros Kokolakis**, sak@aegean.gr
- **Assistant Professor Costantinos Lambrinoudakis**, clam@aegean.gr
- **Associate Professor Stefanos Gritzalis**, sgritz@aegean.gr

- **Info-Sec-Lab:** http://www.icsd.aegean.gr/info-sec-lab/, Karlovassi, GR-832 00 Samos, Greece
- **ISDBLAB:** http://www.aueb.gr/, 74 Patission St., GR-104 34 Athens, Greece



Structure of the Presentation

- Introduction
- Ontology and Developing Secure Software
  - Ontology
  - Secure software development
  - Security ontology
- Research Methodology
  - Building a secure application ontology
  - Methods and tools
- Case Study: The Secure e-Poll Paradigm
  - Remote internet voting: The e-poll application environment
  - The e-poll application
  - Secure e-poll ontology
  - nRQL queries and results
- Conclusions and Further Research
- References

References (1/4)

- Noy, N.F. and Musen, M.A. (2000), *"PROMPT: Algorithm and Tool for Automated Ontology Merging and Alignment"*, in Proceedings of AAAI'00, Austin.
- CCTA, CRAMM User Manuals, version 5.0, United Kingdom, 2002.
- IETF and W3C XML Signature Working Group. http://www.w3.org/Signature/
- OASIS Security Service TC. Security Assertion Mark-up Language (SAML).
- http://www.oasis-open.org/committees/security/ (accessed August 2004).
- Bozsak, E., Ehrig, M., Handschub, S., *Hotho: KAON - Towards a Large Scale Semantic Web*. In Bauknecht, K., Min Tjoa, A., Quirch-ma-yr, G. (Eds.): Proc. of the 3rd International Conference on E-Commerce and Web Technologies, 2002, pp. 304-313.
- Kagal, L., Finin, T. and Joshi, A., "*A policy language for a pervasive computing en-vironment*". In IEEE 4th International Workshop on Policies for Dis-tributed Systems and Networks, 2003.

References (2/4)

- Viktor Raskin, Christian F. Hempelmann, Katrina E. Triezenberg, and Sergej Nirenburg. *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.
- Chung, L.: *Dealing with Security Requirements during the development of Information Systems*. CaiSE '93. The 5th Int. Conf of Advanced Info. Systems Engineering. Paris, France, (1993).
- Mylopoulos, J., Chung L., Nixon, B.: *Representing and Using Non-Functional Requirements A Process Oriented Approach*. IEEE Trans. Soft Eng., vol. 18. (1992) 483-497.
- Mouratidis, H., Giorgini, P., Manson, G.: *An Ontology for Modelling Security: The Tropos Project*. Proceedings of the KES 2003 Invited Session Ontology and Multi-Agent Systems Design (OMASD'03), United Kingdom, University of Oxford, (2003)
- Liu, L., Yu, E., Mylopoulos, J.: *Analyzing Security Requirements as Relationships among Strategic Actors*. (SREIS'02), Raleigh, North Carolina, Oct 15-16, (2002)

References (3/4)

- He, Q., Antȯn, I., A.: *A Framework for modelling Privacy Requirements in Role Engineering*. Int'l Workshop on Requirements Engineering for Software Quality (REFSQ) Austria Klagenfurt / Velden (2003)
- Moffett, D., J., Nuseibeh, A., B.: *A Framework for Security Requirements Engineering*. Report YCS 368, Department of Computer Science, University of York, (2003)
- Antȯn, I., A.: *Goal-Based Requirements Analysis*. ICRE '96 IEEE Colorado Springs Colorado USA (1996) 136-144
- Antȯn, I., A., Earp, B., J.: *Strategies for Developing Policies and Requirements for Secure Electronic Commerce Systems*. 1st ACM Workshop on Security and Privacy in E-Commerce (2000)
- Uszok, A., Bradshaw, J., Jeffers, R., Suri, N., Hayes, P., Breedy, M., Bunch, L., Johnson, M., Kulkarni, S. and Lott, J. (2003). *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, IEEE Press.

References (4/4)

- Uszok, A., Bradshaw, J., Jeffers, R. (2004). KAoS: A Policy and Domain Services Framework for Grid Computing and Semantic Web Services. In Proceedings of the Second International Conference on Trust Management (iTrust 2004), Springer-Verlag.
- http://ontology.ihmc.us/ontology.html
- Warren D. Smith, Cryptography meets voting, http://www.math.temple.edu/~wds/homepage/cryptc
- Protégé, http://protege.stanford.edu/
- Racer Inference Engine, http://www.sts.tu-harburg.de/~r.f.moeller/racer/
- The New Racer Query Language, http://www.cs.concordia.ca/~haarslev/racer/racer-qu