The Evaluation Process of a Computer Security Incident Ontology

Luciana A. F. Martimiano and Edson Moreira

Departamento de Ciências de Computação,
Instituto de Ciências Matemáticas e de Computação,
Universidade de São Paulo - Campus São Carlos,
Caixa Postal 668, CEP 13560-970, São Carlos, SP, Brazil
luciana@icmc.usp.br, edson@icmc.usp.br

Abstract. Ontologies have been developed and used by several researchers in different knowledge domains aiming to ease the structuring and management of knowledge, and to create a unique standard to represent concepts of such a knowledge domain. Considering the computer security domain, several tools can be used to manage and store security information. These tools generate a great amount of security alerts, which are stored in different formats. This lack of standard and the amount of data make the tasks of the security administrators even harder, because they have to understand, using their tacit knowledge, different security alerts to make correlation and solve security problems. Aiming to assist the administrators in executing these tasks efficiently, this paper presents the main features of the computer security incident ontology developed to model, using a unique standard, the concepts of the security incident domain, and how the ontology has been evaluated.

1 Introduction

Security data can be generated by different sources, such as access systems logs, firewall logs, vulnerabilities alerts, and statistics of processors or memory use. Due to this great volume of information, the security administrators face out a difficult problem, which is efficiently generate knowledge about security to make decisions and solve security incidents.

Besides this volume of information, the security tools generate data in different formats, making the security management even harder. Some research institutes, such as CVE Project (Common Vulnerabilities and Exposures) [1] and CERT/CC (Computer Emergency Response Team/Coordination Center)[1] have made efforts to classify and store security data using a unique standard. However, these institutes do not attribute semantic to the data stored and published. Without the semantic meaning, the software agents or the administrators are unable to automatically make implicit correlations among security incidents.

To ease and make it possible to automatically correlate different security data from different sources, and also to ease information management about security

[1] http://www.cert.org/.

incidents, we propose to use ontologies, defining a unique vocabulary of concepts and relations related to security incidents[2].

According to Gruber [2], ontology is explicit formal specifications of the concepts in a domain and the relations among them. Ontology defines a common vocabulary for a community of people, such as researchers and security administrators, and for software agents that need to share information and need to have a common understanding about a domain knowledge. Ontologies consist of a set of concepts, relations, and axioms that formalize a field of interest, with detail and structure that enable computers to process its content.

Some reasons to develop and to use ontologies are [3, 4]:

- **Sharing common understanding of the structured information among people or software agents**. For instance, suppose that several Web sites contain information about vulnerabilities. If these Web sites share and publish the same ontology of the concepts and relations they use, software agents can extract and aggregate information from these different sites, using them to answer queries about vulnerabilities or as input data to other applications.
- **Reusing knowledge**. If one group of researchers develops an ontology, others can simply reuse it, saving efforts. For instance, the Computer Security Incident Ontology (OntoSec) [5, 6] developed and described in this paper reuses some concepts and relations of the Vulnerability Ontology developed by Brandão [7]. Or, a network management ontology could reuse and extend the class **Asset** defined by OntoSec to represent the assets that can be attacked in a computational system.
- **Interoperability**. Different applications can use the concepts and the relations defined by an Ontology (e.g. OntoSec) to extract and infer useful information to deal with vulnerabilities, easing the interoperability that allows sharing data among different applications.

The remainder of this paper is organized as follows. Section 2 presents some security issues. Section 3 briefly presents the related works. Section 4 presents the Security Incident Ontology developed and its main features. The ontology validation process is presented in Section 5. Section 6 presents the main benefits of using ontology to help security management. And, finally, Section 7 presents final remarks and future works.

2 Computational Security Issues

According to Gollmann [8], computational security deals with the prevention and the detection of unauthorized actions done in a computational system, its information and resources. A secure system must satisfy some important features: confidentiality, integrity, availability and authentication. *Confidentiality* assures

[2] A security incident is "*the act of violating an explicit or implied security policy*", which is a CERT/CC definition.

that only authorized people or systems access the information, and it is related to the *Authentication*, which ensures that the user is really who he/she says. *Integrity* ensures that the information is not modified accidentally or maliciously. *Availability* ensures that the system works with no degradation and provides resources whenever authorized users need. The satisfaction grade is defined by a security policy, which creates the rules of what is and what is not allowed in the system.

Along the years, the volume of security data that the security administrators have to manage has grown exponentially. Unfortunately, managing these data and information has became more and more difficult, because different security tools generate alerts in a different standard and with different kinds of data, and the administrators, using their tacit knowledge, have to correlate and analyze these data.

In this sense, it is important to define a way to generate the security data in a unique standard, including a unique vocabulary of concepts and their relations. This unique vocabulary allows the security administrators to manage and correlate different security incidents automatically, and also make it possible to implement countermeasures to solve security incidents more efficiently and to ensure the important features already mentioned, namely confidentiality, integrity, availability and authentication.

3 Related Works

Howard and Longstaff [9] developed a common language for security incidents. This common language is a high-level taxonomy of security incidents concepts. The taxonomy was based primarily on security incident theory, but the experience in incidents classification of CERT was used to refine and expand the taxonomy. Basically, the incident taxonomy defines concepts as **attacker**, **tool**, **vulnerability** and **security incident**. OntoSec reuses some of the concepts defined in the security incident taxonomy.

Two relevant works are the ones conducted by Raskin et al. [10] and Schumacher [11]. Raskin et al. proposed the use of natural language to define, in an unique way, the meaning of the main concepts about security incidents. Basically, the ontology is composed by two parts: a set of high-level concepts and a method of classifying them as a taxonomy. Schumacher proposed the *Core Security Ontology* to represent a conceptual mapping of security policies. The ontology represents the top-level security concepts and how they related themselves. Depending on the organization and its security policies, these concepts can be specialized and new ones can be created. Both works propose an abstract and high level vision of security management.

OntoSec, on the other hand, represents not only the concepts but also the relations among the concepts of the security incident domain. It also proposes a more concrete vision of security management, because it specializes these main concepts.

4 The Security Incident Ontology

To develop the OntoSec, two methodologies were used: the **Methontology**, developed by Fernández et al. [12], and the methodology developed by Noy and McGuinness [3]. The **Methontology** was used to guide all the ontology development process, because it was developed taking IEEE standard for developing Software Life Cycle Processes (IEEE 1074-1995) [13] as a starting document, and it has its roots in a methodology for developing Knowledge-Based Systems (IDEAL [14]) [15]. The methodology developed by Noy and McGuinness was used to guide the conceptualization phase because it defines clearly what should be done, and how it should be done.

Most of the concepts in the OntoSec were compiled using security incident glossaries and taxonomies [16, 9, 17]. Besides these sources, the CSIRT/USP (*Computer Security Incident Response Team/Universidade de São Paulo*)[3] was also used.

After defining the concepts and the relations among them, the OntoSec was formalized using the W3C[4] standard language for modeling ontologies - OWL (Web Ontology Language) [18] and the Protégé tool, which is a free, open-source ontology editor and knowledge-base framework[5].

OntoSec has four levels of classes (or concepts). These four levels represent the main concepts related to the security incident domain. The first level has 13 classes, which are presented in Figure 1, the second one has 11 classes, which are the subclasses of the main level, the third one has 14 classes, which are the subclasses of the second level, and the fourth one has 11 classes, which are the subclasses of the third level, summing up 49 classes. Besides these classes, the OntoSec has 36 non-hierarchical relations and 58 attributes, summing up 94 properties. The first level represents the ***"core"*** of the OntoSec.

Figure 1 presents the main concepts and relations, which are: an **Agent** *performs* an **Attack** that can *cause* a **Security Incident**. To perform an **Attack**, an **Agent** can *use* a **Tool**, which can *explore* a **Vulnerability**, *to get* **Access**. The **Security Incident** *implies to* a **Consequence** and *acts on* an **Asset**. The **Security Incident** also can *have* a **PreCondition**, and this **PreCondition** can be *related to* a **Vulnerability**. A **Consequence** can be *related* to an **Asset** and a **Security Incident** can *precede* or/and *proceed* another one[6]. A **Vulnerability** can be *explored with* other ones, it can has another vulnerability as *precondition*, or it can be *precondition to* other vulnerabilities. It also *has* a **Correction**, which is *developed by* a **Supplier**, a **Type** and a **Range**.

The vulnerabilities are represented by the Vulnerability Ontology. The Vulnerability Ontology models the main concepts and relations about the vulnerability domain. These concepts and relations were defined based on CVE Project

[3] `http://www.security.usp.br/`.
[4] World Wide Web Consortium.
[5] `http://protege.stanford.edu/`.
[6] The relations *precedes* and *proceeds* are inverse, transitive, symmetric and reflexive.

and the NVD (National Vulnerability Database)[7]. And, some of these concepts and relations are imported by OntoSec[8]. The main concepts imported are: **Vulnerability**, **Type**, **Correction**, **Range** and **Supplier**.

Some classes represent a taxonomy, with *isA* relations, which are specialization and can be also called hyponymy relation. For instance, the class **Asset** has as sub-classes: **Data**, **Hardware**, **Port**, **Resource**, **RootAccount**, **Software** and **UserAccount**. The *isA* relations are represented using two constructors, one in OWL and the other in RDF Schema (Resource Description Framework Schema)[19]: `owl:class` and `rdfs:subClassOf`, respectively.

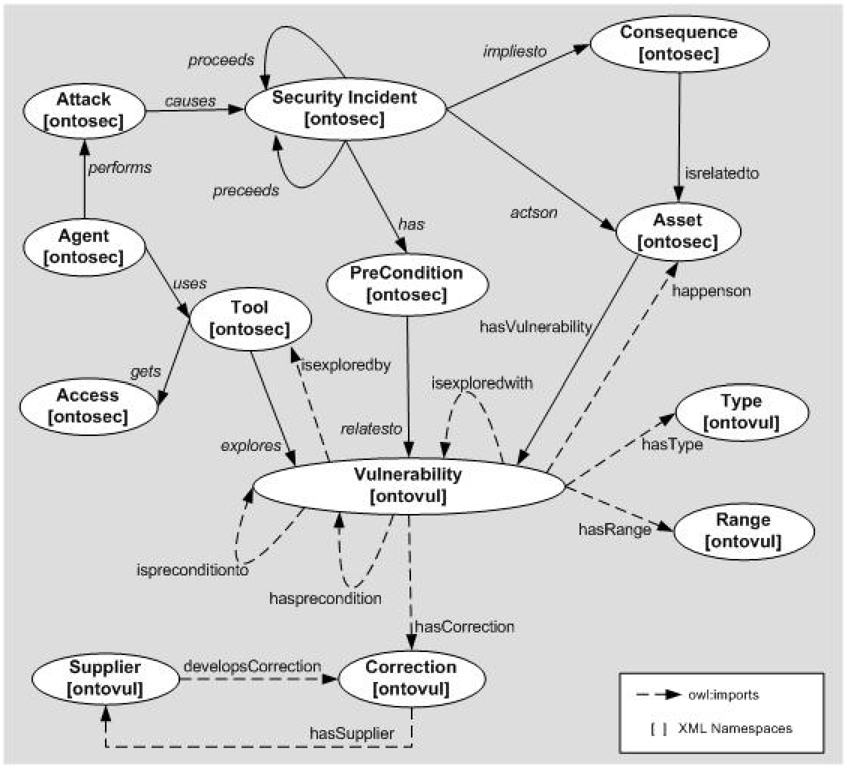

Fig. 1. Main concepts and relations of the OntoSec

To identify a security incident, some attributes were defined: ***hasip_source*** (computers from where a security incident begins), ***hasip_destination*** (computer being attacked), ***hassecurity_incident_type*** (27 security incidents types are represented in OntoSec), ***has_date*** (the date the security incident oc-

[7] NVD is a searchable index of information on computer vulnerabilities and was known as ICAT Metabase. `http://nvd.nist.gov/`.

[8] The label *ontovul* denotes this importing.

curred), ***has_time*** (the time the security incident occurred), ***has_description*** (describe the security incident), ***has_reference*** (any Web site with information about the security incident), ***has_weekday*** (week day in which the security incident occurred), ***has_severity*** (how serious is the security incident). This last attribute can have the following predefined values: *Low*, *Medium* or *High*.

All attributes are represented in OWL as a `Datatype Property`. This property also defines which are the `Domain Resource` and the `Range Resource`. In this case, the `Domain Resource` is the class which has the attribute and the `Range Resource` is the type of attribute. Similarly to the `Object Property`, the `Datatype Property` also can have a restriction as cardinality, representing how many instances the property can have. It is also possible to predefine instances. For example, the attribute ***has_severity*** can have only one of the following instances: *Low*, *Medium* or *High*.

5 OntoSec Evaluation Process

The ontology evaluation process is the problem of assessing the quality of a given ontology, either to aid in the selection of an ontology for the needs of a particular task or organization, or to evaluate or guide an ontology constructing effort [20].

Several approaches have been developed to evaluate an ontology, depending on what kind of ontologies are being evaluated and for what purpose. Some approaches are: to use the ontology in an application [21], to compare the ontology to a "golden standard" of the domain [22], to compare the ontology with a source of data or documents about the domain that is represented by the ontology - the "data driven" approach [23, 24], and to assess how well the ontology meets a set of predefined criteria, standards and requirements, etc. [25].

To validate the OntoSec, the approach defined by [23, 24] was used, because this approach defines a way to check if the ontology is representing the concepts of such a domain knowledge. In this sense, security alerts from SNORT was used to check how well OntoSec cover the security incident domain. SNORT is an open-source network intrusion prevention and detection system based on a rule-driven language[9].

5.1 Validation Process

The main aim of the validation process is to show that OntoSec represents important information about security incidents correctly. Besides that, it is also important to show that OntoSec can help the security administrators to deal with security incidents problems more efficiently, querying the ontology about security incidents. The validation process has been carried out in two phases:

1. Mapping security incident data into OntoSec, which has been done.
2. Developing an application to query and infer security incident information from OntoSec, which is under development.

[9] `http://www.snort.org/`.

Mapping Security Incident Data. Security alerts generated by SNORT were used to validate the OntoSec. SNORT generates intrusion alerts based on rules according to signatures of attacks. The following example shows an alert generated by SNORT. Line 1 presents the alert description and line 2 presents its classification and priority (severity) according to SNORT configuration. Line 3 presents when (date and time) this alert occurred and which are the source IP, source port, destination IP and destination port used. Lines 4 to 8 present URLs[10] where it is possible to find information about the alert.

```
1 [**] [1:1421:11] SNMP AgentX/tcp request [**]
2 [Classification: Attempted Information Leak] [Priority: 2]
3 01/20-11:11:38.672417 192.168.0.253:111 -> 192.168.0.18:705
4 [Xref => http://cve.mitre.org/cgi-bin/cvename.cgi?name=2002-0013]
5 [Xref => http://cve.mitre.org/cgi-bin/cvename.cgi?name=2002-0012]
6 [Xref => http://www.securityfocus.com/bid/4132]
7 [Xref => http://www.securityfocus.com/bid/4089]
8 [Xref => http://www.securityfocus.com/bid/4088]
```

To support OntoSec validation process, a tool was developed using Java and Jena[11]. This tool does the mapping of the security alerts generated by SNORT into OntoSec, creating a security knowledge base that is used in the next phase of the validation process (query and inference). As RDF has a simple data model that is easy for applications to process and manipulate and RDFt's generality offers greater value from sharing, it is used to store the security incident knowledge base instead of storing the base inside the ontology itself.

The following RDF code represents the information about the *"SNMP AgentX/tcp request"* alert. For each `rdf:description`, RDF creates a `nodeID`. The `nodeIDs` are also used to relate a security incident to the logical ports (`rdf:nodeID=A5151`) and to the protocols (`rdf:nodeID=A3989`) used during the attack.

```
<rdf:Description rdf:nodeID="A5149">
   <rdf:type rdf:resource="http://a.com/SecurityIncidentOntology#SecurityIncident"/>
    <j.0:has_description>SNMP AgentX/tcp request</j.0:has_description>
    <j.0:hassecurity_incident_type>Scanning</j.0:hassecurity_incident_type>
    <j.0:has_weekday>Friday</j.0:has_weekday>
    <j.0:has_time>11:11:38</j.0:has_time>
    <j.0:has_date>2006-01-20</j.0:has_date>
    <j.0:hasip_source>192.168.0.253</j.0:hasip_source>
    <j.0:hasip_destination>192.168.0.18</j.0:hasip_destination>
    <j.0:has_severity>Medium</j.0:has_severity>
    <j.0:has_reference>http://cve.mitre.org/cgi-bin/cvename.cgi?name=2002-0013
       </j.0:has_reference>
    <j.0:has_reference>http://cve.mitre.org/cgi-bin/cvename.cgi?name=2002-0012
       </j.0:has_reference>
    <j.0:has_reference>http://www.securityfocus.com/bid/4132</j.0:has_reference>
    <j.0:has_reference>http://www.securityfocus.com/bid/4089</j.0:has_reference>
    <j.0:has_reference>http://www.securityfocus.com/bid/4088</j.0:has_reference>
    <j.0:uses_Port rdf:nodeID="A5151"/>
    <j.0:uses_Protocol rdf:nodeID="A3989"/>

<rdf:Description rdf:nodeID="A5151">
     <rdf:type rdf:resource="http://a.com/SecurityIncidentOntology#Port"/>
```

[10] Universal Resource Locator.

[11] Jena is an open-source Java framework for building Semantic Web applications. `http://jena.sourceforge.net/`.

```
     <j.0:has_sourceportnumber>111</j.0:has_sourceportnumber>
     <j.0:has_destinyportnumber>705</j.0:has_destinyportnumber>
</rdf:Description>

<rdf:Description rdf:nodeID="A3989">
     <rdf:type rdf:resource="http://a.com/SecurityIncidentOntology#Protocol"/>
     <j.0:hasprotocol_name>TCP</j.0:hasprotocol_name>
   </rdf:Description>

</rdf:Description>
```

The tool is composed by two parts: one responsible to collect and structure the SNORT alerts, and another to do the mapping into the ontology. As the tool was developed using a modular structure, the task of integrating new security tools is easier, only the interface between the mapping tool and the security tool has to be modified. As SNORT generates security alerts using a standard (text files), to get the security data was an easy task to be done. A security alert file of 600KBytes were use. This alert file generated a security incident knowledge base of 20MBytes. This knowledge base is stored in a MySQL[12] database[13]. Jena framework was used to store the RDF file in the database. Herein, it is important to point out that not necessarily the security alerts generated by SNORT are really security incidents, many of these alerts are false-positives. But the security alerts were used to know how well the OntoSec models security incidents.

Some data could be directly mapped into OntoSec, such as description, date, time, source and destination IP, source and destination port. But others, as the incident type, could not be because SNORT uses its own standard types of incidents. In this case, a previous task was done in which a type of incident in SNORT was mapping into a type of incident in OntoSec. For instance, the types *Attempted Information Leak* and *Detection of a Network Scan* were mapping as type *Scanning* in OntoSec, the types *A Network Trojan was detected*, *A suspicious string was detected* and *Executable code was detected* were mapped as *Malicious Code*.

Some attributes, such as source and destination port, and protocol, are mapped as relations between **SecurityIncident** and **Asset** classes, because in OntoSec a protocol and a port are modeled as potential targets (assets) of such a security incident.

Table 1 shows an example of how some attributes were mapped into the attributes of the **SecurityIncident** class, the **Port** and the **Protocol** classes in OntoSec. From the date, the day of the week is calculated and the ***has_weekday*** attribute in OntoSec is used to represent it.

It is possible to notice that OntoSec models important information about security incident domain. On the other hand, it is also possible to see that the ontology is modeling information that a security tool does not represent, such as the consequences of a security incident. In this sense, according to the type of the security incident, the ontology can be used to infer which are the consequences. For instance, a *Dos* (*Denial of Service*), *DDos* (*Distribuited Denial of Service*) or

[12] http://www.mysql.com/.

[13] To perform previous tests with the knowledge base, it was stored firstly in a text file

Table 1. Mapping SNORT attributes into the OntoSec attributes

SNORT Attributes	**OntoSec Attributes**	**Class**
`description`	has_description	SecurityIncident
`classification`	hassecurity_incident_type	SecurityIncident
`date`	has_date	SecurityIncident
`hour`	has_hour	SecurityIncident
`source IP`	hasip_source	SecurityIncident
`destination IP`	hasip_destination	SecurityIncident
`reference`	has_reference	SecurityIncident
`priority`	has_severity	SecurityIncident
`source port`	has_sourceportnumber	Port
`destination port`	has_destinyportnumber	Port
`protocol`	hasprotocol_name	Protocol

a *Buffer Overflow* can compromise the availability of the system. Or a *Malicious Code* can allow a remote execution or can compromise data confidentiality.

Querying and Inference Application. Once the security alerts have been stored in RDF using a MySQL database, it is possible to ask questions to the ontology. This task allows to check how well the ontology can answer important questions about security incidents. To perform this task, Jena and a specific RDF querying language, SPARQL[14], have been used. SPARQL is a SQL-like[15] language that uses the basic clause **SELECT FROM WHERE**.

Using the querying and inference application, the security administrators can ask questions like:

- How many security incidents have happened because of such a vulnerability?
- Which are the most common types of security incidents?
- Which are the assets that have more security problems and must to receive more attention concerning patches?
- Which security incidents have happened after a previous security incident (looking for correlation)?
- Which are the most common ports used to perform a security incident?

The following two examples show how a query is done using SPARQL.

(i) Which security incidents happened on January 15, 2006?

```
Prefix ab: <http://a.com/SecurityIncidentOntology#>
Select Distinct ?incident ?type
Where { ?x ab:has_date "2006-01-15";
ab:has_description ?incident;
ab:hassecurity_incident_type ?type} Order by ?x
```

[14] Recursively, SPARQL Protocol and RDF Query Language. `http://www.w3.org/TR/rdf-sparql-query/`.

[15] Structured Query Language.

(ii) Which security incidents were caused by a malicious code?

```
Prefix ab: <http://a.com/SecurityIncidentOntology#>
Select distinct ?incident ?date
Where { ?a ab:hassecurity_incident_type "Malicious Code";
ab:has_description ?incident;
ab:has_date ?date}
```

The results of these queries allow the administrators to know which incidents happened in a specific day and which incidents were caused by a malicious code, respectivelly. If many incidents happen because of malicious codes, such as viruses or worms, the administrators can update the computational systems more frequently. Combining both queries, it is also possible to know in which date security incidents caused by malicious codes happen. Using the results, the administrators can prevent attacks that happen on specific periods.

6 Benefits of OntoSec in Information Security Management

We can point out the following advantages of using ontologies to assist the information security management:

– The development of ontologies creates a conceptual model that makes it possible to the organization to know better its security incidents domain.
– The ontologies can facilitate the interoperability among different security tools, creating a unique way to represent security data and, for instance, allowing that security alerts from any security tool is mapped into an ontology.
– The Security Incident Ontology imports the Vulnerability Ontology, allowing the reuse of knowledge and information. Other ontologies about security domain could be imported, such as a Virus Ontology or a Worm Ontology. The same reuse can be scaled up in such a way that security information can be treated in a more abstract level.
– The querying and inference process helps the security administrators to be more confident of the decisions made about the security information management, because the ontology developed is knowledge bases about security incidents. The ontology allows the security administrators to learn from previous security problems, assisting them in solving and preventing new problems.

7 Final Remarks and Future Works

Not only ontologies can fundamentally change the way in which systems are constructed, as stated by Swartout and Tate [26], but they also will change the way people and systems can communicate to each other about a domain knowledge.

Using a security incident ontology, the organization can improve its ability to manage and to control security incidents problems. Besides that, the administrators can learn more about security and can prevent the computational systems from previous problems efficiently.

The mapping process shows that the ontology models important information about security incidents. The querying and inference process helps the security administrators to be more confident of the decisions made about the security management, because OntoSec allows to create a knowledge base about security incidents.

The next steps of the validation process are: (i) to use real security incidents (security incidents from the CSIRT/USP will be used), (ii) to map security alerts from other security tools, and (iii) to develop a security incident management system based on the ontology, making it easier the interoperability among different security tools.

References

1. Mann, D.E., Christey, S.M.: Towards a common enumeration of vulnerabilities (1999) http://cve.mitre.org.
2. Gruber, T.R.: Toward principles for the design of ontologies used for knowledge sharing. International Journal of Human-Computer Studies **43**(5/6) (1995) 907–928
3. Noy, N.F., McGuiness, D.L.: Ontology development 101: A guide to create your first ontology. Technical report, Knowledge Systems Laboratory - Stanford University (2001) TR KSL-01-05.
4. Uschold, M., Gruninger, M.: Ontologies: principles, methods and applications. Knowledge Engineering Review **11**(2) (1996) 93–155
5. Martimiano, A.F.M., Brandão, A.J.S., Moreira, E.S.: Towards a security network incident ontology to ease the security knowledge management. In: Proceedings of the Third International Information and Telecommunication Techniques Symposium (I2TS2004). (2004) 88–95
6. Martimiano, A.F.M., Moreira, E.S.: An owl-based security incident ontology. In: Proceedings of the Eighth International Protégé Conference. (2005) 43–44 Poster.
7. Brandão, A.J.S.: Uso de ontologia para classificação de vulnerabilidades em sistemas computacionais. Master's thesis, Instituto de Ciências Matemáticas e de Computação - ICMC, Universidade de São Paulo - USP, São Carlos - São Paulo (2004)
8. Gollmann, D.: Computer Security. John Wiley & Sons (1999)
9. Howard, J.D., Longstaff, T.A.: A common language for computer security incidents. Technical report, Sandia National Laboratories (1998)
10. Raskin, V., Hempelmann, C.F., Triezenberg, K.E., Nirenburg, S.: Ontology in information security: A useful theoretical foundation and methodology tool. In: Proceedings of the Workshop on New Security Paradigms. (2001)
11. Schumacher, M.: Toward a security core ontology. In: Security Engineering with Patterns - Origins, Theoretical Model and New Applications. Springer Verlag (2003) 87–96 Lectures Notes in Computer Science (LNCS 2754).
12. Fernández, M.A., Gómez-Pérez, A., Juristo, N.: Methontology: From ontological art towards ontological engineering. In: Proceedings of the AAAI Spring Symposium Series. (1997) 33–40

13. IEEE: IEEE standard for developing software life cycle processes (1996) IEEE Computing Society.
14. Gómez-Pérez, A., Juristo, N., Montes, C., Pazos, J.: Ingeniería del conocimiento: Diseño y construcción de sistemas expertos (1997) Ceura, Madri, Spain.
15. López, M.F.: Overview of methodologies for building ontologies. In: Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI99) - Workshop on Ontologies and Problem-Solving Methods: Lessons Learned and Future Trends. (1999)
16. NCSC: Glossary of computer security terms (1988) National Computer Security Center - Trusted Network (NCSC-TG-004).
17. Shirey, R.: RFC 2828: Internet security glossary (2000) GTE-BBN Technologies.
18. Bechhofer, S., Harmelen, F.v., Hendler, J., Horrocks, I., McGuinness, D.L., Patel-Schneider, P.F., Stein, L.A.: OWL web ontology language reference (2004) http://www.w3.org/TR/owl-ref/.
19. Brickley, D., Guha, R.V.: RDF vocabulary description language 1.0: RDFSchema (2004) http://www.w3.org/TR/rdf-schema/.
20. Brank, J., Grobelnik, M., Mladeniæ, D.: A survey of ontology evaluation techniques. In: Proceedings of the Conference on Data Mining and Data Warehouses (SiKDD05). (2005)
21. Porzel, R., Malaka, R.: A task-based approach for ontology evaluation. In: Proceedings of the Workshop on Ontology Learning and Population (ECAI2004) - Sixteenth European Conference on Artificial Intelligence. (2004) 9–16
22. Maedche, A., Staab, S.: Measuring similarity between ontologies. In: Proceeding of the Thirteenth European Conference on Knowledge Acquisition and Management (EKAW02). (2002) 251–263
23. Brewster, C., Alani, H., Dasmahapatra, S., Wilks, Y.: Data driven ontology evaluation. In: Proceedings of the International Conference on Language Resources and Evaluation. (2004)
24. Chintan, P., Kaustubh, S., Yugyung, L., Park, E.K.: Ontokhoj: A semantic web portal for ontology searching, ranking and classification. In: Proceedings of the Fifth ACM International Workshop on Web Information and Data Management. (2004) 58–61
25. Lozano-Tello, A., Gómez-Pérez, A.: Ontometric: A method to choose the appropriate ontology. Journal of Database Management **15**(2) (2004) 1–18
26. Swartout, W., Tate, A.: Ontologies. IEEE Intelligent Systems **14**(1) (1999) 18–19