Using ontologies to assist security management

Luciana A. F. Martimiano[1], **Edson dos Santos Moreira**[1]

[1] Departamento de Ciências de Computação - SCC
Instituto de Ciências Matemáticas e de Computação - ICMC
Universidade de São Paulo - Campus São Carlos
Caixa Postal 668, CEP 13560-970, São Carlos, SP

{luciana, edson}@icmc.usp.br

Abstract. *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 and contributions of the security incident ontology developed to model, using a unique format, the concepts of the security incident domain.*

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, security administrators face out the difficulty in generating knowledge about security incidents[1] to make decisions and to solve security problems efficiently. Moreover, security administrators (or software agents) are unable to automatically make important and implicit correlations among security incidents. They have to use their tacit knowledge, based on their experience, to perform these tasks.

To ease and make it possible to automatically correlate security incidents from different sources, and also to assist the security management, we propose to use ontologies, defining a unique vocabulary of concepts and relations related to security incidents. An ontology *is an explicit specification of a conceptualization* (Gruber, 1993). It defines a common vocabulary for people and for software agents that need to share information and need to have a common understanding about a domain knowledge.

2. The Security Incident Ontology (OntoSec)

Figure 1 presents the main concepts and relations of ONTOSEC[2] (Martimiano and Moreira, 2005), 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*

[1] In the context of this work, a security incident is "*the act of violating an explicit or implied security policy*", which is the CERT/CC definition.

[2] The ONTOSEC was developed using OWL (Web Ontology Language) and Protégé 3.1. OWL is a W3C standard language to represent ontologies and Protégé is an ontology editor and knowledge-base framework. More information: http://www.w3.org/2004/OWL/ and http://protege.stanford.edu/.

to a **Vulnerability**. A **Consequence** can be *related* to an **Asset** and a **Security Incident** can *precede* or/and *proceed* another one[3]. An **Asset** can *have* a **Vulnerability**. A **Vulnerability** *has* a **Correction**, which *is developed by* a **Supplier**, has a **Type** and a **Range**. The concepts related to the vulnerability were imported from the Vulnerability Ontology (Brandão, 2004), and are represented with the label `ontovul`.

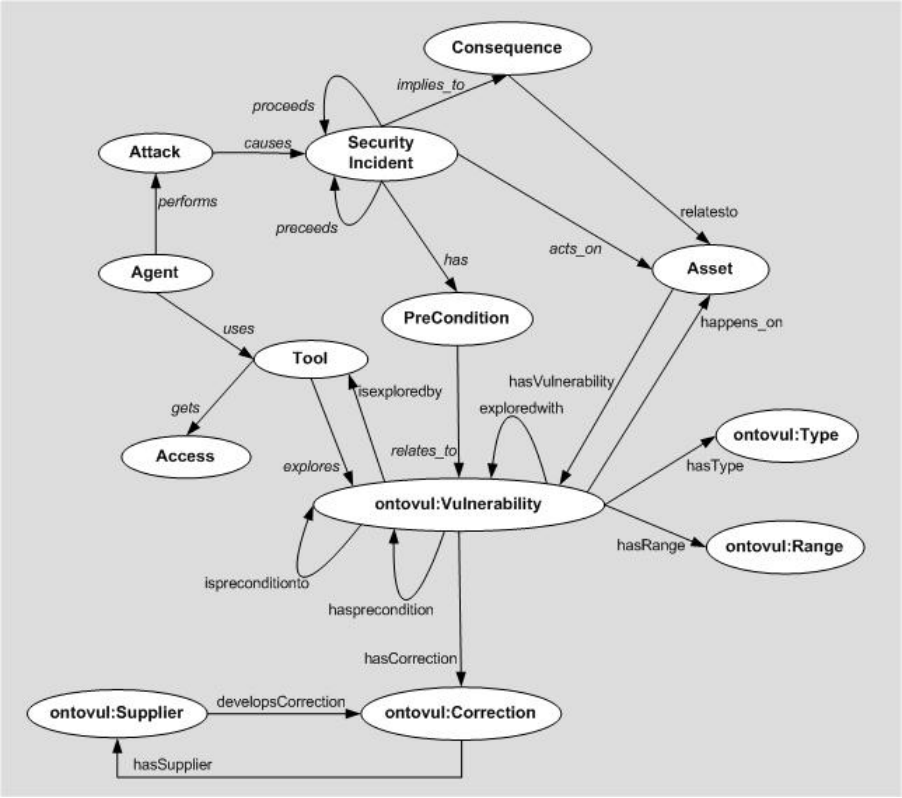

Figure 1. Main concepts and relations of ONTOSEC.

ONTOSEC has four levels of classes (or concepts), the main level (0) has 13 classes, which are presented in Figure 1, the second level (1) has 18 classes, which are the subclasses of the level 0, the third level (2) has 20 classes, which are the subclasses of the level 1, and the fourth level (3) has 11 classes, which are the subclasses of the level (2), summing up 62 classes. Besides these classes, the ONTOSEC has 36 relations and 55 attributes, summing up 91 properties.

Some classes represent a hierarchy of classes (taxonomy). For instance, the class **Asset** has as sub-classes: **Software**, **Hardware**, **User Account**, **Root Account**, **Protocol**, **Port** and **Resource**. Other classes have only attributes, such as the class **Attack**: ***hasattack_type*** attribute, which can have the values *Passive* or *Active*, and ***hasattack_location*** attribute, which can have the values *Remote* or *Local*.

A validation procedure was carried out to show that ONTOSEC represents correctly important information about security incidents. Besides that, it is also important to show that ONTOSEC can help the security administrators to deal with security incidents more efficiently, querying the ontology about them. For instance, they can ask the ontology (i) which security incidents have happened because such a vulnerability? (ii)

[3]The relations *precedes* and *proceeds* are inverse and reflexive.

which are the most common type of security incident? (iii) which are the assets that have more security problems and must to receive more attention concerning patches? (iv) which types of security incident precede another type of security incident (looking for correlation)?

The validation procedure has been carried out in two phases: (i) Mapping security incident data into ONTOSEC, which have been completed, and (ii) Developing a querying and reasoning application, which is under development.

2.1. Mapping the Security Incident Data

Security incident data generated by Snort were used to validate the ONTOSEC. Snort generates intrusion alerts based on rules according to signatures of attacks. Based on the Snort alerts, the mapping into ONTOSEC was done, creating a security knowledge base that is used in the next phase of the validation process. This knowledge base is a RDF[4] file. As RDF has a simple data model that is easy for applications to process and manipulate and RDF's generality offers greater value from sharing, it is a common practice to use RDF to store ontology data instead of storing the data inside the ontology itself.

The mapping process consists of two parts: (1) searching and struturing the Snort alerts, and (2) mapping the alerts into ONTOSEC. As the mapping 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 alerts using a standard (text files), it was easy to get the security data. A security alert file of 600Kbytes was used. This alert file generated a security incident knowledge base of 3Mbytes. 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, priority, time, protocol, source and destination IP, source and destination port. But others, as the incident type, could not be because Snort uses its standard types of incidents (or classification). In this case, a previous manual work was done in which a type of incident in Snort was mapping to a type of incident in ONTOSEC. For instance, the types *Attempted Information Leak* and *Detection of a Network Scan* in Snort were mapped as type *Scanning*, the types *A Network Trojan was detected*, *A suspicious string was detected* and *Executable code was detected* were mapped as *Malicious Code*. Some Snort 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 of such a security incident.

All these information are modeled in ONTOSEC by the **SecurityIncident** class. Besides these information, ONTOSEC is also modeling the consequences of a such 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* (*Distributed Denial of Service*) or a *Buffer Overflow* can compromise the availability of the system. Or a *Malicious Code* can allow a remote execution or can compromise data confidentiality and integrity.

[4]Resource Description Framework.

2.2. Querying and Reasoning Application

Once the alerts have been stored in RDF, 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[5] and a specific RDF querying language, SPARQL[6] have been used. SPARQL is a SQL-like[7] language that uses the basic clause **SELECT FROM WHERE**.

3. Final Remarks and Future Works

We can point out the following advantages of using ontologies to assist 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 ontology can facilitate the interoperability among different security tools, creating a unique way to represent security data and allowing that security data from any security tool is mapped into an ontology.
- Other ontologies about security domain can 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 reasoning process can help the security administrators to be more confident of the decisions made, because the ontology developed is a knowledge base about security incidents. The ontology allows the security administrators to learn from previous security problems, assisting them in solving and preventing new problems.
- The continuous improvement of the ontology (maintenance process) through the introduction of new rules (concepts, relations and restrictions) about security incidents, when the knowledge base is not enough to help the security administrators to make the most suitable solution, can reduce as much as possible the security problems.

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. This system is under development and it will integrate the mapping tool, the querying and reasoning application, and will automatically store security incidents using RDF.

References

Brandão, A. J. S. (2004). Using ontology to classify vulnerabilities in computational systems. Master's thesis, Instituto de Ciências Matemáticas e de Computação - ICMC, Universidade de São Paulo - USP, São Carlos - São Paulo. In Portuguese.

Gruber, T. R. (1993). Towards principles for the design of ontologies used for knowledge sharing. In *Formal Ontology in Conceptual Analysis and Knowledge Representation*. Kluwer Academic Publishers.

Martimiano, L. A. F. and Moreira, E. S. (2005). An owl-based security incident ontology. In *Eighth International Protégé Conference*, pages 43–44.

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

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

[7]Structured Query Language.