IGI PUBLISHING

ITJ3908

701 E. Chocolate Avenue, Suite 200, Hershey PA 17033-1240, USA
Tel: 717/533-8845; Fax 717/533-8661; URL-http://www.igi-pub.com

This paper appears in the publication, International Journal of Information Security and Privacy, Volume 1, Issue 4
edited by **Hamid R. Nemati © 2007, IGI Global**

An Ontology of Information Security

Almut Herzog, Linköpings Universitet, Sweden

Nahid Shahmehri, Linköpings Universitet, Sweden

Claudiu Duma, Linköpings Universitet, Sweden

ABSTRACT

We present a publicly available, OWL-based ontology of information security which models assets, threats, vulnerabilities, countermeasures and their relations. The ontology can be used as a general vocabulary, roadmap, and extensible dictionary of the domain of information security. With its help, users can agree on a common language and definition of terms and relationships. In addition to browsing for information, the ontology is also useful for reasoning about relationships between its entities, for example, threats and countermeasures. The ontology helps answer questions like: Which countermeasures detect or prevent the violation of integrity of data? Which assets are protected by SSH? Which countermeasures thwart buffer overflow attacks? At the moment, the ontology comprises 88 threat classes, 79 asset classes, 133 countermeasure classes and 34 relations between those classes. We provide the means for extending the ontology, and provide examples of the extendibility with the countermeasure classes 'memory protection' and 'source code analysis'. This article describes the content of the ontology as well as its usages, potential for extension, technical implementation and tools for working with it.

Keywords: *computer science; information systems; is security; ontologies; software*

INTRODUCTION

Agreeing on the meaning of concepts and their relations is useful in all domains because the consequences of a misunderstanding can be time-consuming and costly. In the domain of information security many concepts are vaguely defined, even for security professionals. Is a password "a unique character string held by each user, a copy of which is stored within the system" (Oxford University Press, 2004) or "an example of an authentication mechanism based on what people know" (Bishop, 2003, p. 310)? Such ambiguities could be mitigated by a common repository of domain knowledge for the security domain. In this article, we present such a repository by means of an ontology. An ontology "defines the basic terms and relations comprising the vocabulary of a topic area, as well as the rules for combining terms and relations to define extensions to the vocabulary" (Neches, Fikes, Finin, Gruber, Patil, Senator, & Swartout, 1991).

The need for an ontology of information security has also been clearly verbalised by Donner (2003):

What the field needs is an ontology—a set of descriptions of the most important concepts and the relationship among them. ... Maybe we [the community of security professionals] can set the example by building our ontology in a machine-usable form in using XML and developing it collaboratively.

Previous work, such as Schumacher (2003); Kim, Luo, and Kang (2005); Jutla and Bodorik (2005); Squicciarini, Bertino, Ferrari, and Ray (2006); Nejdl, Olmedilla, Winslett, and Zhang (2005); Undercoffer, Joshi, Finin, and Pinkston (2004); Tsoumas, Dritsas, and Gritzalis (2005); Takahashi, Abiko, Negishi, Itabashi, Kato, Takahashi, and Shiratori (2005), has only partly addressed these needs, and, so far, an ontology of information security that provides general and specific concepts, is machine-usable, and can be developed collaboratively is still missing.

In this article we present an ontology that (1) provides a general overview over the domain of information security, (2) contains detailed domain vocabulary and is thus capable of answering queries about specific, technical security problems and solutions, and (3) supports machine reasoning. As a step towards an ontology that is collaboratively developed and acceptable by the security and ontology community, we have designed our ontology according to established ontology design principles (Gruber, 1995) and best practices (obofoundry.org1) and make our ontology available online. Consequently, users can browse the ontology online. They can extend it either by downloading and modifying it or by importing the ontology from the Web and extending it with new concepts.

Our security ontology builds upon the classic components of risk analysis (Whitman & Mattord, 2005, p. 110ff): assets, threats, vulnerabilities and countermeasures. By modeling these four basic building blocks of information security and their relations, and refining each block with technical concepts, we arrive at an ontology that provides the "big picture" of the domain of information security as well as a classification and definition of specific domain vocabulary.

Our ontology provides natural language definitions for general terms such as 'asset', as well as domain-specific, technical terms, such as 'SSH'. By implementing high-level relations for specific, technical concepts, one can also find answers to questions such as "What and how does SSH protect?". Other examples of questions that our ontology helps answer are: Which threats threaten user authentication? Which countermeasures protect the confidentiality of data? Which vulnerabilities enable a buffer overflow attack? Which countermeasures protect against buffer overflow attacks? Which countermeasures use encryption?

Users may find our ontology useful, (1) as a reference book or hypertext learning material on information security, (2) as a template for classifying and comparing security products, security attacks or security vulnerabilities, (3) as a framework for plugging in new or existing detailed security taxonomies and (4) as a knowledge base for reasoning with semantic Web applications. We have implemented our ontology in OWL (Web Ontology Language) (Bechhofer, van Harmelen, Hendler, Horrocks, McGuinness, Patel-Schneider, & Stein, 2004), a markup language based on RDF/XML (Resource Description Framework/Extensible Markup Language) (Powers, 2003), specifically devised for creating extensible ontologies for the semantic Web. Thus, our ontology uses a commonly accepted notation for describing ontologies and supports querying and acquisition of new knowledge through inference and rule-based reasoning using OWL reasoners and OWL query languages.

The remainder of the article is structured as follows. An overview of our ontology is given in the following section. Then we present refinements of the core concepts. Afterward, we provide examples that demonstrate the power of inference and querying. We also describe useful tools for creating and working with ontologies.

Towards the end, we present related work, critically discuss our ontology, address future work and conclude the article. An appendix explains the concepts of OWL, needed for understanding ontology details in earlier sections.

ONTOLOGY OVERVIEW

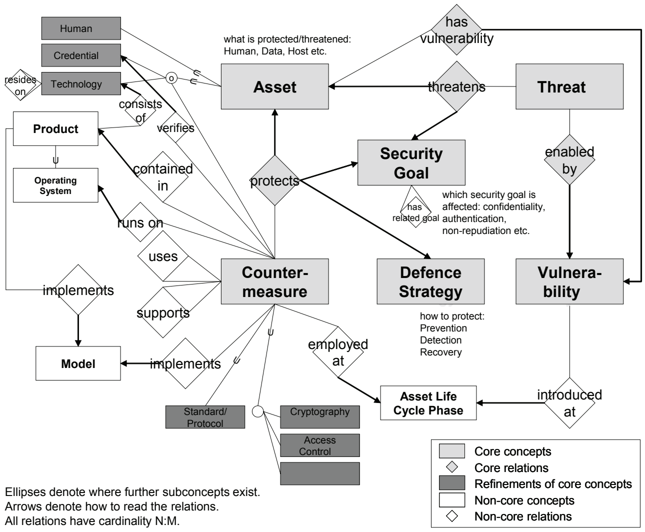
An ontology may be domain-oriented and address a very specific part of a domain; it may be task-oriented so that it achieves a specific task; or it may be generic with a focus on high-level concepts (Stevens, Goble, & Bechhofer, 2000). An ontology may be used as an application-neutral knowledge base, as a basis for software development, as common information access for humans and applications and as support for information search (Lambrix, 2004; Stevens et al., 2000). An ontology can range in complexity from a controlled vocabulary that consists of a flat list of concepts to logic-based knowledge bases that contain concepts, instances, relations and axioms and provide reasoning services (Lambrix, Tan, Jakonien˙e, & Strömbäck, 2007).

The ontology that we have created is a generic knowledge base with reasoning services, mainly intended as a source for common information access, containing key concepts, instances, relations and axioms from the domain of information security. Our security ontology builds upon the classic components of risk analysis—assets, threats, vulnerabilities, countermeasures—and their relations to each other. These components are core concepts in our ontology and, together with their relations, provide an overview of the domain of information security. While the concepts are taken from literature, for example, Whitman and Mattord (2005), relations between the concepts are only mentioned implicitly in the literature. It is our contribution to state them explicitly.

Figure 1 presents a simplified overview of our security ontology using an adapted notation of extended entity-relation (EER) diagrams (Chen, 1976). In the text, relations and core and noncore concepts are denoted in *italics*, at least upon their first occurrence and appear with the same name in Figure 1. Refinements are denoted with 'single quotes'.

An *asset* is connected to the concept *vulnerability* through its *has vulnerability*-relation. An asset is *threatened* by *threats* that also denote which *security goal* they threaten. An asset is *protected* by *countermeasures*; a countermeasure is also an asset. A *countermeasure protects* a

Figure 1. Overview of the security ontology



security goal and an asset with a *defence strategy*. For example: The countermeasure 'backup' *protects* the integrity and availability (security goals) of the asset 'data' through recovery (defence strategy). Instances of the concept *defence strategy* are prevention, detection, recovery, correction, deterrence and deflection. Instances of the concept *security goal* are confidentiality, integrity, availability, authentication, accountability, anonymity, authenticity, authorisation, correctness, identification, nonrepudiation, policy compliance, privacy, secrecy and trust. Security goals may be related. For example, privacy *has the related goals* confidentiality, anonymity and secrecy.

In addition to its *protects*-relation, a countermeasure can make use of additional countermeasures. For example: The countermeasure 'SSL' *uses* 'symmetric encryption' and 'public-key encryption' (two countermeasure subconcepts). A countermeasure can also support other countermeasures. 'Key management' (a countermeasure) *supports*, for example, 'encryption' (a countermeasure).

A *threat* (e.g. 'eavesdropping') *threatens* a security goal (e.g. confidentiality) and an asset (e.g. 'network packet'). The countermeasure against a threat is found by matching assets and security goals of the *threatens*- and *protects*-relation. For example: 'Eavesdropping' *threatens* the confidentiality of 'data in transit'; 'VPN' (virtual private network), 'SSH' (secure shell), 'SSL' (secure socket layer), and the like *protect* the confidentiality of 'data in transit'. A threat is *enabled by* one or more *vulnerabilities*.

The core ontology gives an overview of the general concepts and their relations in information security. However, to answer specific questions such as "What is SSH?" and "What and how does SSH protect?", the ontology must also be populated with specific concepts that refine the core concepts and implement the core relations. A few core refinements are shown in Figure 1; they are further described and illustrated in the following section.

To answer questions, such as "Which are the products that contain SSH?" or "For which operating systems is SSH available?" we also need to model concepts such as *product* and *operating system*. These concepts are not central to information security and should therefore be imported from external ontologies. In the ontology overview of Figure 1, we denote these concepts as noncore. The noncore concepts that we have identified, and which are shown in Figure 1, are useful for comparing security products and countermeasures. For example, the *product* concept denotes which countermeasure is contained in which product; for example, 'access control' *is contained in* 'Microsoft Word'. The product concept could be further refined by an external taxonomy of software, hardware and system products; the *contained in*-relation, however, must be populated by the person that plugs in the product taxonomy. A product *consists of* one or more technological assets. For example, a 'database management system' *consists of* 'database files', 'executable program files' and 'configuration files'. Other noncore concepts can be added to the ontology depending on the queries that have to be supported by the ontology.

REFINEMENT OF THE CORE ONTOLOGY

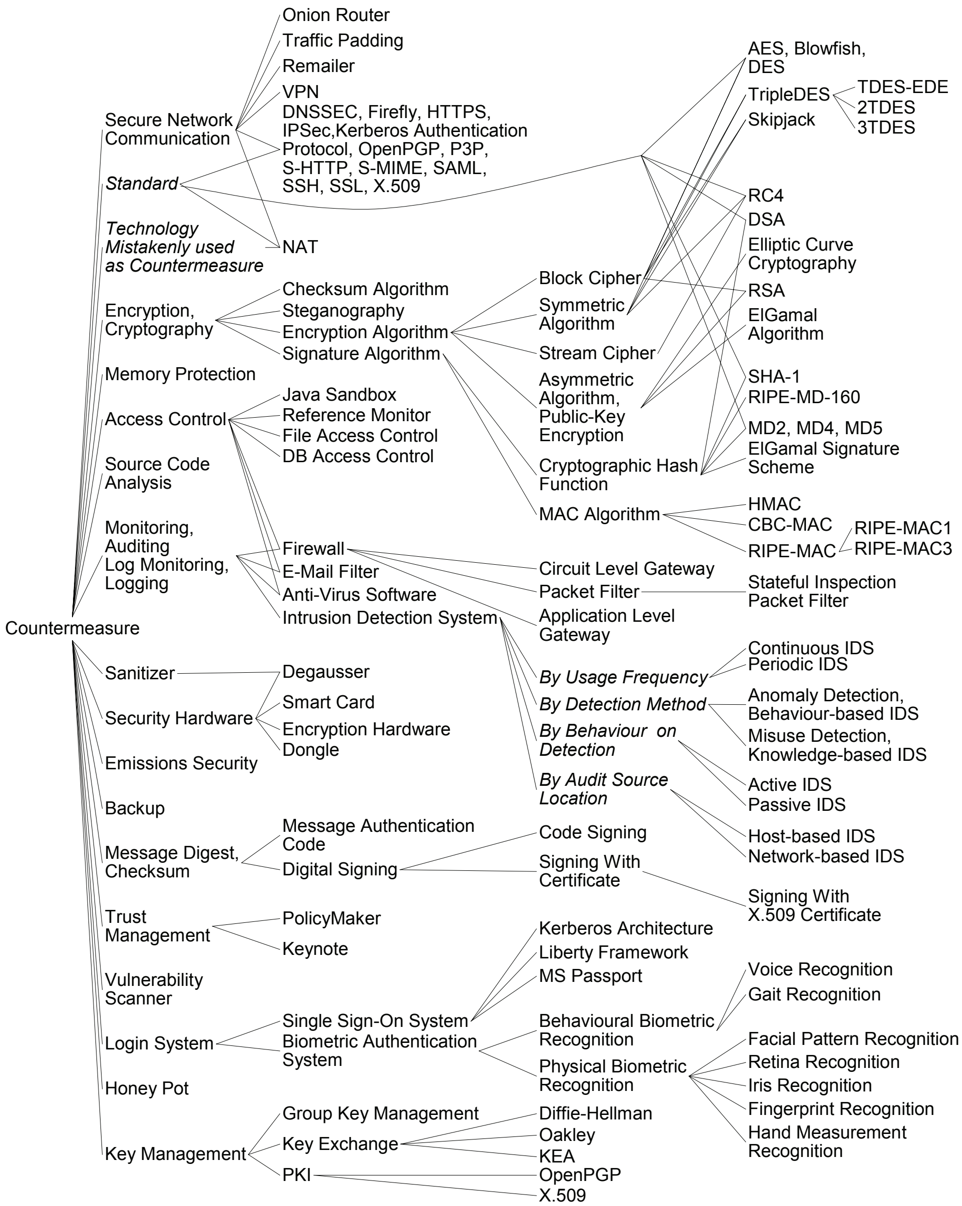
The refinements or subconcepts for countermeasures, assets, threats and vulnerabilities consist of specific, technical domain vocabulary and are described in the following subsections. The vocabulary was collected from literature and from security taxonomies (see the Related Work section, this article). The actual ontologies are accessible with browsable documentation and as OWL files from http://www.ida.liu.se/~iislab/projects/secont.

Our ontology is implemented in OWL, therefore, in this section, we use some OWL terminology to talk about the ontology. In OWL, concepts are implemented as classes, relations are implemented as properties and axioms are implemented with restrictions. An introduction to OWL is given in the appendix.

Countermeasures

In the asserted ontology, the first level of subconcepts of the countermeasure concept are 'access control', 'cryptography' or 'encryption', 'secure network communication'. These and more subconcepts are shown in Figure 2 where lines denote generalisation-specialisation-relations. Typically, countermeasures that are subconcepts of 'secure network communication' are also subconcepts of 'standard or protocol', for example, 'SSL', 'IPSec' (Internet Protocol Security) and 'DNSSEC' (Domain Name System Security Extensions). The concept 'technology mistakenly used as countermeasure' is a container for technologies that are not designed to

Figure 2. Countermeasure classification



be countermeasures but are commonly used as such. 'NAT' (network address translation) is such a technology. It was designed to remedy the shortcoming of the IPv4 address space but is sometimes used instead of a firewall to hide a network.

The *protects*-relation of Figure 1 denotes which assets and security goals the countermeasure protects through which defence strategy. Each countermeasure contains constraining axioms in OWL called restriction as to what it protects. An example is the countermeasure 'backup' whose OWL representation is shown in Box 1.

The OWL code expresses that backup is a countermeasure. A backup must protect at least (denoted by the someValuesFrom-restrictions) availability, integrity and data through recovery and not more (as denoted by the allValuesFrom-restrictions). More details and explanations of the OWL syntax are shown in the appendix.

If one wants to know all details about a countermeasure, one can browse to it in the OWL or html representation of the ontology and examine its definitions and restrictions. If a user wonders "What is SSH and what does it protect?", the ontology provides answers. SSH is textually described in the ontology as "a network protocol designed for logging into and executing commands on a networked computer, replacing the insecure ftp, telnet and rlogin protocol". Its class definition (see Box 2) states that SSH is a countermeasure that belongs to the subconcepts of 'secure network communication' and 'standard'. It is a preventive measure, protects the confidentiality and integrity of data in transit and provides host authentication. The allValuesFrom-restriction on the *protects*-property contains the intersection of the expressions in the someValuesFrom-restrictions, meaning that SSH does not protect more than what is stated as someValuesFrom-restrictions. SSH uses both symmetric and public-key encryption. Restrictions like the ones described for backup and SSH are given for all 133 countermeasures, which make up one major strength of our ontology, as these restrictions are used for inference as shown later.

Assets

Figure 3 shows the asset subconcepts in our ontology. The direct subconcepts are 'human', 'technology', 'credential' and also 'countermeasure'. All countermeasures are assets; they deserve protection, and they have a value. A credential is an asset that is typically verified by 'login system' countermeasures. This is implemented by the *credentialVerifiedBy*-relation between credential and countermeasure. Credentials are grouped into biometric, physical and electronic tokens (Kim et al., 2005). The biometric tokens that are not fully shown in Figure 3 contain 'gait' and 'voice' (behavioural) as well as 'facial pattern', 'fingerprint', 'hand measurement', 'iris' and 'retina' (physical biometric credential).

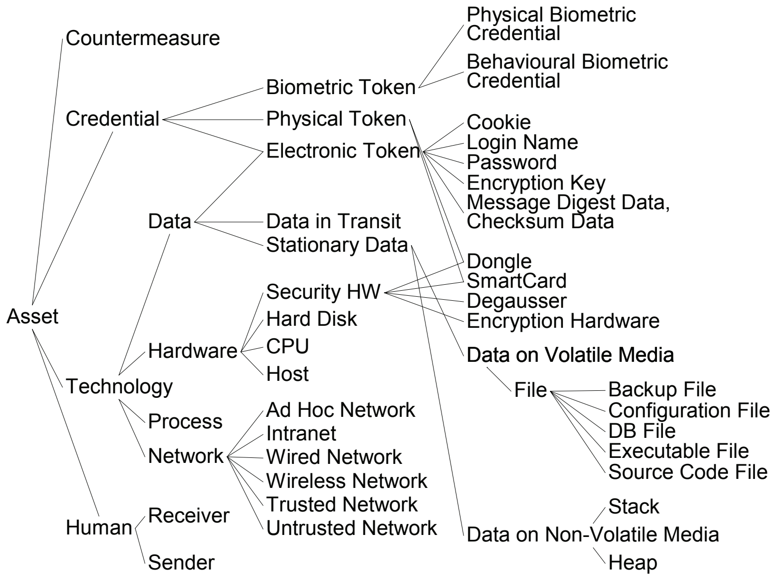
Box 1.

```
Class (Backup partial
    Countermeasure
    restriction(protects allValuesFrom(intersectionOf(
            unionOf(_Availability _Integrity) _Data _Recovery)))
    restriction(protects someValuesFrom(_Availability))
    restriction(protects someValuesFrom(_Data))
    restriction(protects someValuesFrom(_Recovery))
    restriction(protects someValuesFrom(_Integrity)))
)
```

Box 2.

Class (SSH **partial**
SecureNetworkCommunication
Standard
restriction(protects **allValuesFrom**(**intersectionOf**(
_Prevention
unionOf(
intersectionOf(_Host _Authentication)
intersectionOf(**unionOf**(_Confidentiality _Integrity) _DataInTransit)
)
)))
restriction(protects **someValuesFrom**(_Prevention))
restriction(protects **someValuesFrom**(
intersectionOf(_Confidentiality _DataInTransit)))
restriction(protects **someValuesFrom**(
intersectionOf(_Integrity _DataInTransit)))
restriction(protects **someValuesFrom**(
intersectionOf(_Host _Authentication)))
restriction(uses **someValuesFrom**(SymmetricEncryption))
restriction(uses **someValuesFrom**(Public-KeyEncryption))
)

Figure 3. Some levels of assets. Ellipses show where further subconcepts exist.



The asset 'technology' is further developed as 'data', 'hardware', 'process' and 'network'. 'Network' is further split up into, for example, 'ad-hoc network', 'wireless network', 'Intranet' and so forth. 'Hardware' is, among others, refined by 'host' that is further developed into 'bastion host', 'router', 'wireless access point', 'local host' and more. 'Data' is specialised by many file types, but also by 'data in transit' that is further refined as 'application layer packets' such as 'e-mail' or 'http traffic', as 'transport layer packets' and 'network layer packets' with the additional subconcepts of 'TCP', 'UDP' and 'IP packet'.

The asset 'human' is refined by 'sender' and 'receiver' which are used in certain network communication countermeasures to denote whether the privacy of sender or receiver is protected. The interdependency of assets is described by the *resides on*-relation for technical assets. It expresses that, for example, the asset 'data' resides on a 'hard disk' which resides on a 'host' which resides on a 'network'. Thus, a countermeasure that protects a network may also be useful in protecting data on a networked host.

Threats or Attacks

In this work, we use the words 'threat' and 'attack' interchangeably. Some publications, notably Whitman and Mattord (2005), use the word 'threat' for high-level threats such as 'act of human error or failure' or 'compromises to intellectual property' and the word 'attack' for more specific threats such as 'virus' or 'Web site defacement'. The distinction is not clear as, for example, one threat is called a 'deliberate software attack' (Whitman & Mattord, 2005). At the moment, our ontology focuses more on specific attacks than on high-level threats.

The attacks or threats as well as their hierarchical classification in the ontology have been taken from other books or articles, mentioned in the Related Work section in this article. The sources are also documented in the ontology, either as comments or as annotations that point out the actual publication. The top level classification is *passive* and *active* attacks (Ince, 2001; Neumann, 1995) (see Figure 4). A passive attack is an attack that does not modify the attacked system but violates the confidentiality of the system. Typical passive attacks are eavesdropping, statistical attacks on databases, scavenging data from object residue, system mapping and side channel attacks. Typical active threats are unauthorised system modification, spoofing, denial of service attacks and more.

Sometimes a countermeasure can also be a threat. A vulnerability scanner in the hands of a system administrator is a countermeasure; in the hands of a malicious user, it may prepare an active attack and is thus a threat. Our ontology allows such modeling. The vulnerability scanner appears both as a countermeasure and as a threat. Each threat threatens a security goal and an asset, usually expressed together as in the example 'confidentiality (security goal) of data (asset)'. Each threat concept is modeled with axioms that indicate what it threatens. For example, the threat posed by spyware is implemented (see Box 3) which reads: Spyware is malicious code. Minimally, it threatens the privacy of humans, the availability of the host—because it consumes resources—and the integrity of the host—because it was installed without the user's consent. The threat of stack overflow shows how further properties for threats are used (see Box 4).

This reads: Stack overflow is a kind of buffer overflow. It threatens the integrity of the stack. If successful, the threat may lead to the additional threats of malicious code or usurpation—using the *ifSuccessfulLeadsToThreat*-relation. Stack overflow is at least enabled by the use of a vulnerable programming language and missing input validation (*enabledByVulnerability*). Similar axioms are provided for all 88 threats in the ontology and are used for finding countermeasures against threats as shown in the Inference section.

Vulnerabilities

The vulnerability concept is the least developed concept of the core ontology, and its refinement is future work. At the moment, we only model 13 vulnerabilities. A vulnerability participates

Box 3.

```
Class (Spyware partial
      MaliciousCode
      restriction(threatens someValuesFrom(
            intersectionOf (_Privacy _Human)))
      restriction(threatens someValuesFrom(
            intersectionOf (_Availability _Host)))
restriction(threatens someValuesFrom(
            intersectionOf (_Integrity _Host)))
)
```

Box 4.

```
Class (StackOverflow partial
      BufferOverflow
      restriction(threatens someValuesFrom(
            intersectionOf(_Integrity _Stack)))
      restriction(threatens allValuesFrom(
            intersectionOf(_Integrity _Stack)))
      restriction(ifSuccessfulLeadsToThreat allValuesFrom(
            unionOf(MaliciousCode Usurpation)))
      restriction(enabledByVulnerability someValuesFrom(
            UseOfVulnerableProgrammingLanguage))
      restriction(enabledByVulnerability someValuesFrom(
            MissingInputValidation))
)
```

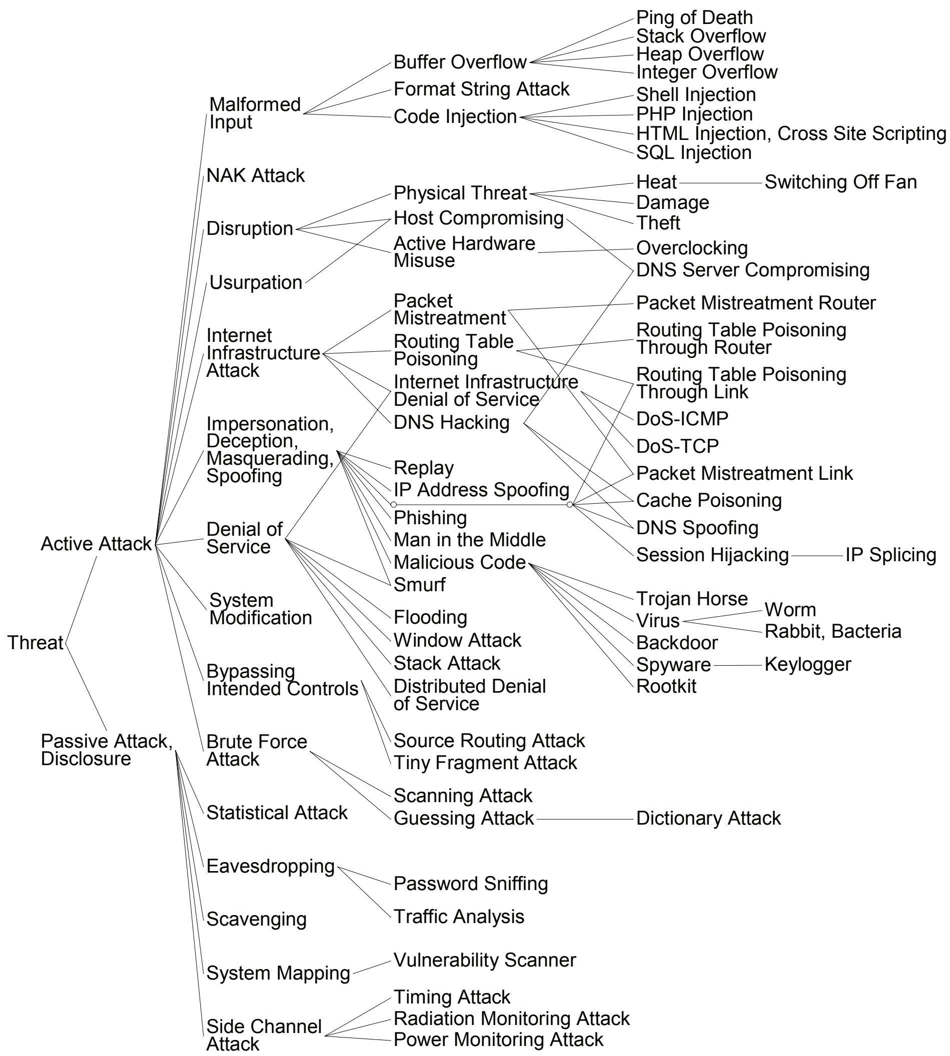
in the *enablesThreat*-relation and the *existOnAsset*-relation. For example: The vulnerability 'missing input validation' enables the threat 'malformed input', a superclass of, among other things, buffer overflows, and exists on the asset 'program source code file'.

Further Refinement of Two Countermeasure Concepts

To demonstrate how users can use our security ontology for comparing tools or products, we refined the countermeasure concepts 'memory protection' and 'source code analysis' with additional subconcepts representing tools used in the respective areas. For the implementation of these two subontologies, we imported our general security ontology into new OWL files and then implemented the new concepts. Thus the two in-depth ontologies are updated automatically with the latest general security ontology when they are opened. It also means that the new ontologies are very small in size because only the additional classes and restrictions reside in the new OWL files. All the basic structure comes from the general security ontology.

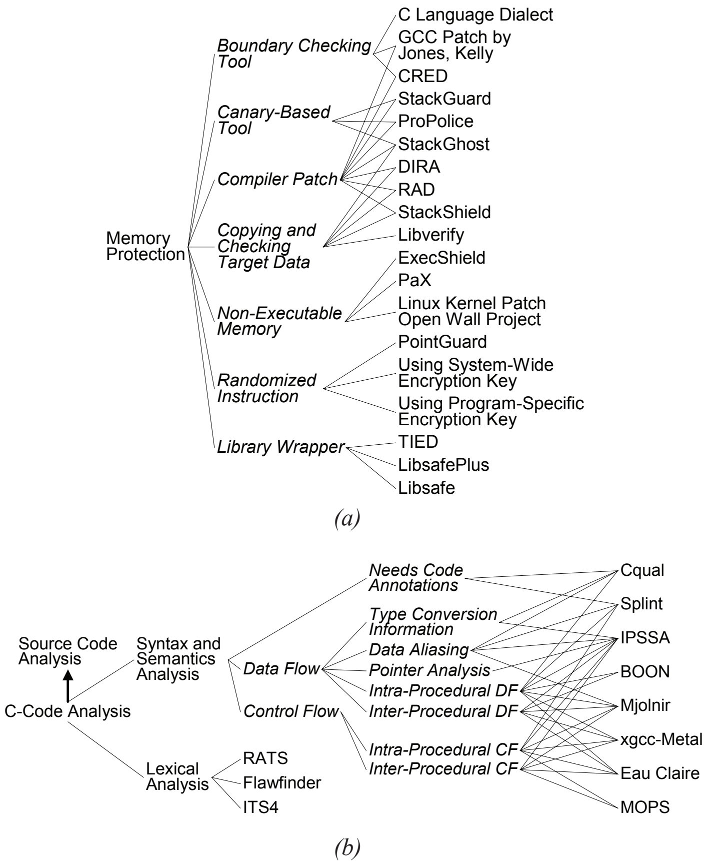
The source for the facts in the two ontologies are two studies (Wilander, 2005; Wilander

Figure 4. Threat classification



& Kamkar, 2003) that contain an overview of tools that are useful for providing memory protection (28 concepts) and C source code analysis (25 concepts) respectively. The resulting two subontologies are basically machine- and human-readable versions of sentences, references and a number of tables that exist in the source literature. The classification of the tools according to certain criteria is now available to both human users and reasoning applications. Memory protection, in Figure 5(a), contains, for example, subconcepts for stating that a tool is either a C library wrapper, a compiler patch or works by using non-executable memory. Again, all concepts are defined with descriptions from Wilander (2005), and all tools are referred with a link to their documentation or publication. C-source code analysis describes tools for lexical analysis and syntax or (partially) semantic analysis. The latter group is further subdivided

Figure 5. Subontology of tools for (a) memory protection, (b) C source code analysis. Text in italics denotes the sorting criteria.



according to certain criteria from Wilander and Kamkar (2003) as shown in Figure 5(b). The tools make up the leaf nodes.

ADVANCED USES OF THE ONTOLOGY

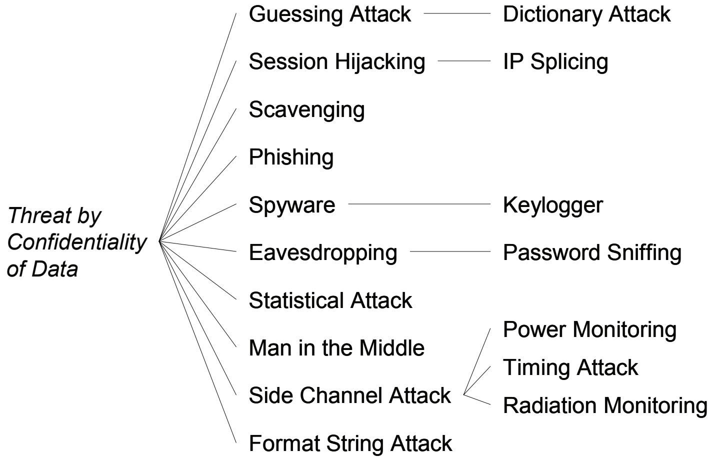
So far, we have described the hierarchy of the ontology and the information it provides for specific concepts. Now we use this hierarchy and information to categorise, for example, threats or countermeasures according to their security goal, asset or defence strategy. A query language offers additional possibilities to find and process information in an ontology.

Inference

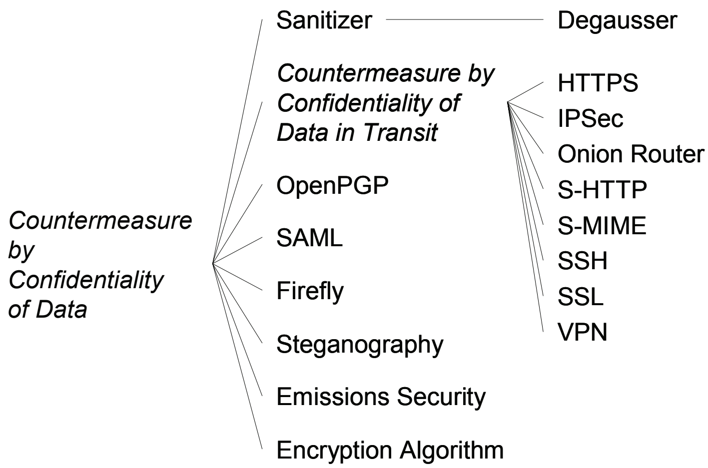
This section describes the contents of the ontology called SecurityViews.owl, accessible at http://www.ida.liu.se/~iislab/projects/secont. This ontology imports the general security ontology of Figure 1 and allows the grouping of countermeasure and threat concepts according to certain criteria described below. The grouping is achieved by defining view concepts and by letting a reasoner such as FaCT, Racer, Pellet, and so forth (see www.w3.org/2001/sw/WebOnt/impls) infer subclasses for these view concepts.

We are, for example, interested in finding all threats that threaten the confidentiality of

Figure 6. Inference results: (a) threats that violate the confidentiality of data, (b) countermeasures that protect the confidentiality of data



(a)



(b)

data. To achieve this, we define a categorising view class *ThreatByConfidentialityOfData* which is defined as a threat that threatens the confidentiality of data (see Box 5).

After inference, this class contains all threats that threaten the confidentiality of data. The result is seen in Figure 6(a). Countermeasures that thwart the above threats are found in a similar way. One creates a class *CountermeasureByConfidentialityOfData* which is defined as a countermeasure that protects confidentiality of data (see Box 6).

The result of this inference is shown in Figure 6(b). The inference mechanism supports even more detailed formulations. After inference, the following class contains the inferred countermeasures that detect or prevent violation of integrity of data (see Box 7).

The inferred subconcepts are all block ciphers, all signature algorithms, memory pro-

Box 5.

```
Class (ThreatByConfidentialityOfData complete
        intersectionOf(
            Threat
            restriction(threatens someValuesFrom(
                intersectionOf(_Confidentiality _Data)))
        )
)
```

Box 6.

```
Class (CountermeasureByConfidentialityOfData complete
        intersectionOf(
            Countermeasure
            restriction(protects someValuesFrom(
                    intersectionOf(_Confidentiality _Data)))
        )
)
```

Box 7.

```
ClassSS(CountermeasureByDetPrevIntData complete
        intersectionOf(
            Countermeasure
            restriction(protects
                someValuesFrom( intersectionOf(
                        _Data _Integrity
                        unionOf(_Prevention _Detection)
        ))))
)
```

tection, message digest and checksum creation systems and some of the secure network communication concepts such as SAML (Security Assertion Markup Language), OpenPGP (Open Pretty Good Privacy), S-MIME (Secure/Multipurpose Internet Mail Extensions), SSL, IPSec, Firefly, SSH, SHTTP (Secure Hypertext Transfer Protocol), DNSSEC and HTTPS (Hypertext Transfer Protocol over SSL).

Inference is also useful for finding countermeasures against specific threats. We assume that one wishes to find countermeasures against the specific threat of a rootkit. The rootkit threat is a subconcept of malicious code and threatens the integrity of a host (see Box 8).

We propose a three-step procedure for finding countermeasures against specific threats.

Box 8.

```
Class (Rootkit partial
    MaliciousCode
    restriction(threatens someValuesFrom(
        intersectionOf(_Integrity _Host)))
    ...
))
```

1. One must find those countermeasures that protect what the rootkit threatens, namely integrity of host. This is done by the view class (see Box 9) that infers all countermeasures that protect the integrity of a host or subclasses of 'host' such as 'networked host' or 'router'. The user should then browse the result and decide with the help of the countermeasure documentation which countermeasure is most suitable.
2. There may be additional suitable countermeasures that protect against rootkits, and they can be found by inferring those countermeasures that protect the integrity of any asset (see Box 10). The inference result for this view class is imprecise with regard to the goal of finding countermeasures against rootkits and retrieves a lot of goal-irrelevant countermeasures such as digital signature algorithms (which protect the integrity of data, with data being a—for this query irrelevant—sibling concept of host). This can be remedied with a ranking algorithm that matches results from step two to the asset that is used in step one. Countermeasures that protect assets that are direct superclasses of the asset of step one (for the rootkit: host) are more likely to thwart the given threat of a rootkit than countermeasures that protect assets which are sibling concepts of host (such as CPU) or sibling concepts of direct superclasses of host (such as data, network or human).

Box 9.

```
Class(CountermeasureByIntegrityOfHost complete
    intersectionOf(
    Countermeasure
    restriction(protects someValuesFrom(
    intersectionOf(_Integrity _Host)))))
```

Box 10.

```
Class(CountermeasureByIntegrity complete
    intersectionOf(
    Countermeasure
    restriction(protects someValuesFrom(_Integrity))))
```

3. There may be countermeasures that protect security goals that are related to integrity, such as correctness and policy compliance. These countermeasures could be potential countermeasures against rootkits, too. They are found by defining a view class that makes use of the relation named *has related goal*, which denotes which security goals are similar to each other.

In this ontology, inference is primarily used for sorting and categorising threats and countermeasures according to security goals, assets and defence strategies. Inference also assists in finding countermeasures against a given threat. In future work, we will implement an algorithm that suggests and ranks countermeasures against a given threat following the three steps described above so as to relieve the user from performing these steps manually.

Querying with SPARQL

We showed before that inference with view classes can find, for examples, threats that can compromise the confidentiality of data and thus provide a means of categorising concepts in the ontology. The standard query language SPARQL (SPARQL Protocol and RDF Query Language) (Prud'hommeaux & Seaborne, 2006) allows additional degrees of freedom in retrieving data from an OWL ontology. SPARQL can, for example, be used to do some postprocessing of results as shown in Box 11.

Box 11.

```
PREFIX ns: <http://www.ida.liu.se/~iislab/projects/secont/SecurityViews.owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT DISTINCT ?c
WHERE {
        ?c rdfs:subClassOf ns:CountermeasureByConfidentialityOfData
        FILTER ( ?c != owl:Nothing)
}
ORDER BY (?c)
```

Box 12.

```
PREFIX ...
SELECT DISTINCT ?c
WHERE {
        ?c rdfs:subClassOf ns:Countermeasure .
        OPTIONAL {
           ?c2 rdfs:subClassOf ?c .
           FILTER (?c2 != owl:Nothing && ?c2 != ?c)
        }
        FILTER (!bound(?c2))
}
ORDER BY (?c)
```

This query returns all inferred countermeasures that protect the confidentiality of data using the previously described view class *CountermeasureByConfidentialityOfData* and orders the result alphabetically.

SPARQL also supports yes/no-questions, sorting, filtering, string matching and so forth. The following advanced SPARQL code, which makes use of sorting and filtering, finds all leaf classes of the countermeasure hierarchy (see Box 12).

IMPLEMENTATION

Users that are not familiar with OWL files and OWL-specific tools can still use the ontology for reference by browsing through it. Browsing is possible with the ontologies' html-representations at http://www.ida.liu.se/~iislab/projects/secont. The html representation shows an OWL ontology complete with properties, individuals, restrictions and pointers to where each concept is used, all in javadoc style. Together with the overview provided in Figure 1, the ontology should be useful for human users "as is".

A convenient but less detailed overview of the tree of concepts without comments, definitions and restrictions is the dumpont service: http://www.daml.org/cgi-bin/dumpont?http://www.ida.liu.se/~iislab/projects/secont/Security.owl.

Our ontologies are created using the Protege OWL tool (protege.stanford.edu) with the Pellet reasoner (www.mindswap.org/2003/pellet). The editor Swoop (www.mindswap.org/2004/SWOOP) was useful for finding the sources of inconsistencies in the knowledge base. Images from the ontology are produced using Protege with the Jambalaya plugin (www.cs.uvic.ca/~chisel/projects/jambalaya/jambalaya.html) and the OWLViz plugin (www.co-ode.org/downloads/owlviz/). Certain programming, for example, for creating the helper classes and retrieving countermeasures for certain threats, was done using the Jena API (jena.sourceforge.net). For querying OWL files, we used the SPARQL language of the ARQ implementation (jena.sourceforge.net/ARQ/), Jena and Pellet. A convenient Web interface for both the Pellet reasoner and SPARQL queries exists on www.mindswap.org/2003/pellet/demo.shtml. Our ontology imports the general Dublin Core ontology (www.dublincore.org) for making annotations such as describing a class, citing a source and so forth.

RELATED WORK

Schumacher (2003) describes a core security ontology for maintaining a knowledge base of security patterns. The ontology consists of the concepts asset, threat, attack, vulnerability, attacker, risk, countermeasure, stakeholder and security objective (confidentiality, integrity, availability, etc.) and their relations. However, the ontology has the following problems: (1) Countermeasures are not directly related to security objectives or assets but only to threats. This makes it unclear what a countermeasure protects. (2) If an attack is described as the realisation of a threat, it is difficult to distinguish between the concepts 'threat' and 'attack'. (3) A risk, being a probability, should not be modeled as a concept but as a property of a threat. Refinements of the core ontology are not available. Thus, there is no technical domain terminology such as specific threats or security countermeasures, and the ontology can consequently not be used for queries.

Kim et al. (2005) have put up a number of small ontologies, which they use for matching security services. These ontologies are online at http://chacs.nrl. navy.mil/projects/4SEA/ontology.html and show quite a level of detail, for example, in the areas of encryption algorithms and credentials. However, a core that shows the connections between concepts is missing. Assets, threats and vulnerabilities are not modeled, and the countermeasure branch is less refined than our work.

Other ontologies in the domain of information security are used for more specific purposes, for example, reasoning about privacy settings or negotiations (Jutla & Bodorik, 2005; Squicciarini et al., 2006), policy settings (Nejdl et al., 2005), automatic intrusion classification (Undercoffer et al., 2004), risk assessment of organisations (Tsoumas et al., 2005), learning about encryption in network security (Takahashi et al., 2005) and rarely are the actual ontologies made available. Especially the work of Tsoumas et al. (2005) may be based on an interesting core ontology, but only a few concepts are exposed in the publication and the actual ontology is not made available. One strength of our ontology is that it is publicly available, general and thus of use for a broad audience. Domain knowledge is also collected in taxonomies—non-overlapping concept classifications with a single top-level concept and no relations between concepts. These can refine general concepts of the core ontology and are therefore of interest.

Threat taxonomies (Álvarez & Petrovic, 2003; Chakrabarti & Manimaran, 2002; DeLooze, 2004; Lindqvist & Jonsson, 1997; Simmonds, Sandilands, & van Ekert, 2004; Welch & Lathrop, 2003) are rich and well-developed. There is even a rudimentary threat ontology, but it is not available online anymore (Undercoffer et al., 2004). Also, textbooks (Amoroso, 1994; Bishop, 2003; Stallings, 2006; Ször, 2005; Whitman & Mattord, 2005) are usually good in grouping or classifying threats. Thus for the threat branch of our ontology, we could harvest from many sources. The same sources also supply useful input for the vulnerability branch. However, countermeasure taxonomies are less well-developed.

The security technology taxonomy of Venter and Eloff (2003) puts high-level concepts like 'access control', 'biometrics' or 'cryptography' on the same level as technical concepts like 'VPN' (virtual private network), 'digital signature' or 'digital certificate'. The six concepts above and 10 more are grouped into proactive and reactive technologies as well as by their level of interaction: network, host or application level. In contradiction with the authors' own definition, access control and passwords are classified as reactive measures.

Irvine and Levin (1999) show a countermeasure taxonomy and use it for determining the cost of network security services. The taxonomy starts out by grouping security technologies by security goals like CIA (confidentiality, integrity, availability) but does not remain consistent. Both 'data confidentiality' and 'audit and intrusion detection' figure as grouping criteria. The former is a security goal, the latter however are two security technologies.

Wang and Wang (2003) put up a countermeasure taxonomy of four concepts: 'standards and policies', 'library and tools', 'administration and system management' and 'physical tools'. However, the important concept of encryption is missing, and it is unclear where it should be added. Also, the authors mix between more general concepts like 'PKI' (public-key infrastructure) and 'biometric authentication' and products such as 'Secure SQLnet' and 'Tripwire' in a list of only 19 concepts. Better-developed taxonomies can be found in the area of intrusion detection (Axelsson, 2000; Carver & Pooch, 2000; Debar, Dacier, & Wespi, 1999), but these naturally do not cover other security technologies.

DISCUSSION AND FUTURE WORK

In the introduction, we put up goals for our ontology. We set out to achieve an ontology that provides a general *overview*, contains detailed *domain vocabulary*, allows *queries*, supports *machine reasoning* and may be used *collaboratively*. The core ontology provides the *overview* over the domain by focusing on the four pillars of risk assessment—assets, threats, vulnerabilities and countermeasures—and their relations. The core ontology is refined by a great number of subconcepts that make up the *domain vocabulary*. These refinements of the core provide details and allow *queries* for specific problems and solutions in the domain of information security.

Our work contains definitions and explanations of concepts and relations in natural language, which make it easy for human users to understand them. The ontology is implemented in a standard language that supports *machine reasoning*. We have made first steps towards *collaborative use* of the ontology by the choice of the ontology implementation language and by making the ontology available online. Users can download our ontologies and edit them to their liking; or they can integrate our work with any existing ontology through aligning and merging. Tools for this are readily available, both in Protege and in the research community (Lambrix & Tan, 2005, 2007). But as we have shown with our extensions—memory protection, source code analysis, security views— it is also possible to import the general ontology from the Web and extend it with new concepts. Hopefully, these possibilities for extension will make our work interesting for others and can lead to an ontology of information security that is accepted by the community.

An issue for discussion is *concept naming* and *concept classification*. We had to make choices that may not be acceptable to everyone. An example is that the concept of 'credential'—for example, 'password' or 'smart card'—is sometimes used as a pars-pro-toto (part of something is used as name for the whole) to denote a system that verifies the credential. Some use the concept 'password' as a countermeasure. However, we model a password as a credential, which is a subconcept of asset, not a countermeasure. A system that verifies credentials is called 'login system' (a countermeasure) in our ontology. An object property denotes that login systems *verify credentials* such as a 'password'. At the moment, there are no subconcepts to login systems because they would duplicate the credential hierarchy and confuse more than help. When the need arises, subconcepts of 'login system' can be created that can then show that, for example, a 'password system' verifies 'passwords'. If someone insists on using 'password' as the name for a system that administrates and verifies passwords, a different name must be chosen because OWL does not allow use of the same name for two different concepts. Easier to resolve are issues of synonyms that can easily be declared in OWL, using equivalent classes.

Security technologies, threats and vulnerabilities are quickly changing. Thus, an important part in the life-cycle of an ontology like ours is *maintenance*. In the future, we plan to increase the ontologies' availability to the public, for example, by using a wiki to make them editable by contributors. So far, we provide the raw OWL files, illustrations and html-documentation of the ontology, but we would also like to offer a *Web interface* for finding threats given a countermeasure, or finding countermeasures given a threat, or finding threats and countermeasures given an asset.

Extensions and refinements on all levels can be envisioned. General concepts that may make interesting extensions are 'attacker', 'stakeholder' and 'impact' of a threat. Technical details in all branches can be found; the ontology is far from complete. We envision refinements for the concepts of vulnerability, threat and the countermeasure subconcepts of firewall, backup, intrusion detection system and more. Also procedures and written policies—being countermeasures (and thus also assets)—may need to be integrated.

CONCLUSION

This article shows how the need for a general and specific machine-usable and extensible ontology for the security community can be met. We have described an OWL-based ontology with its core concepts *asset, threat, vulnerability, countermeasure, security goal* and *defence strategy*. All the core concepts are subclassed or instantiated to provide the domain vocabulary of information security. Relations connect concepts. Axioms, implemented as OWL restrictions, model constraints on relations and are used to express, for example, which countermeasure protects which asset and which security goal. Inference and the query language SPARQL allow additional views on the ontology. They can show countermeasures that protect the confidentiality of data, countermeasures that detect integrity violations, threats that can compromise the availability of a host and so forth. Inference also assists in finding countermeasures for a given threat.

Our work can be used as online learning material for human users, as a framework for comparing security products, security attacks or security vulnerabilities, as a publicly available knowledge base for rule-based reasoning with semantic Web applications and as a starting point and framework for further extensions and refinements. We hope that our ontology will be a trigger for discussions leading to even more detailed and acceptable ontologies in the area of information security.

ACKNOWLEDGMENT

We would like to thank Christoph Schuba and Patrick Lambrix for fruitful discussions on the content, structure and implementation of the ontology. The developers of the Jena API and Pellet reasoner –especially Chris Dollin, Dave Reynolds, Andy Seaborne, Bijan Parsia, Evren Sirin—helped clarify implementation issues.

REFERENCES

Álvarez, G., & Petrovic, S. (2003). A new taxonomy of Web attacks suitable for efficient encoding. *Computers & Security, 22*(5), 435-449.

Amoroso, E. (1994). *Fundamentals of computer security technology*. Prentice Hall.

Axelsson, S. (2000). *Intrusion detection systems: A survey and taxonomy* (Tech. Rep. No. 99-15). Department of Computer Engineering, Chalmers University of Technology.

Bechhofer, S., van Harmelen, F., Hendler, J., Horrocks, I., McGuinness, D. L., Patel-Schneider, P. F., & Stein, L. A. (2004). OWL Web ontology language reference. Retrieved June 28, 2007, from *http://www.w3.org/TR/owl-ref/*

Bishop, M. (2003). *Computer security: Art and science*. Addison-Wesley.

Carver, C. A., & Pooch, U. W. (2000). An intrusion response taxonomy and its role in automatic intrusion response. In *Proceedings of the IEEE Workshop on Information Assurance*. IEEE.

Chakrabarti, A., & Manimaran, G. (2002). Internet infrastructure security: A taxonomy. *IEEE Internet Computing, 16*(6), 13-21.

Chen, P. P.-S. (1976). The entity-relationship model: Toward a unified view of data. *ACM Transactions on Database Systems (TODS), 1*(1), 9-36.

Debar, H., Dacier, M., & Wespi, A. (1999). Towards a taxonomy of intrusiondetection systems. *Computer Networks, 31*, 805-822.

DeLooze, L. L. (2004). Classification of computer attack using a selforganizing map. In *Proceedings of the IEEE Workshop on Information Assurance* (pp. 365-369). IEEE.

Donner, M. (2003). Toward a security ontology. *IEEE Security and Privacy, 1*(3), 6-7.

Gruber, T. R. (1995). Toward principles for the design of ontologies used for knowledge sharing. *International Journal of Human-Computer Studies, 43*(5-6), 907-928.

Ince, D. (2001). *A dictionary of the Internet*. Oxford University Press.

Irvine, C., & Levin, T. (1999). Toward a taxonomy and costing method for security services. In *Proceedings of the 15th Annual Computer Security Applications Conference (ACSAC'99)* (pp. 183-188). IEEE.

Jutla, D. N., & Bodorik, P. (2005). Sociotechnical architecture for online privacy. *IEEE Security and Privacy, 3*(2), 29-39.

Kim, A., Luo, J., & Kang, M. (2005). Security ontology for annotating resources. In *Proceedings of the On the Move to Meaningful Internet Systems: CoopIS, DOA, and ODBASE* (pp. 1483-1499). Springer-Verlag. Lecture Notes in Computer Science 3761.

Lambrix, P. (2004). Ontologies in bioinformatics and systems biology. In W. Dubitzky & F. Azuaje (Eds.), *Artificial intelligence methods and tools for systems biology* (pp. 129-146). Springer-Verlag.

Lambrix, P., & Tan, H. (2005). A framework for aligning ontologies. In *Proceedings of the 3rd Workshop on Principles and Practice of Semantic Web Reasoning* (pp. 17-31). Springer-Verlag. Lecture Notes in Computer Science 3703.

Lambrix, P., & Tan, H. (2007). Ontology alignment and merging. In A. Burger, D. Davidson, & R. Baldock (Eds.), *Anatomy ontologies for bioinformatics: Principles and practice*. Springer-Verlag.

Lambrix, P., Tan, H., Jakonien˙e, V., & Strömbäck, L. (2007). Biological ontologies. In C. J. Baker & K.-H. Cheung (Eds.), *Semantic Web: Revolutionizing knowledge discovery in the life sciences* (pp. 85-99). Springer-Verlag.

Lindqvist, U., & Jonsson, E. (1997). How to systemactically classify computer security intrusions. In *Proceedings of the IEEE Symposium on Security and Privacy (S&P'97)* (pp. 154-163). IEEE.

Neches, R., Fikes, R., Finin, T. W., Gruber, T. R., Patil, R., Senator, T. E., & Swartout, W. R. (1991). Enabling technology for knowledge sharing. *AI Magazine, 12*(3), 36-56.

Nejdl, W., Olmedilla, D., Winslett, M., & Zhang, C. C. (2005). Ontologybased policy specification and management. In *Proceedings of the 2nd European Semantic Web Conference (ESCW'05)* (pp. 290-302). Springer-Verlag. Lecture Notes in Computer Science 3532.

Neumann, P. G. (1995). *Computer-related risks*. Addison-Wesley.

Noy, N., & Rector, A. (2006). Defining N-ary relations on the semantic Web. Retrieved June 28, 2007, from *http://www.w3.org/TR/swbp-n-aryRelations/*

Oxford University Press. (2004). *A dictionary of computing* (Oxford Reference Online). Author.

Patel-Schneider, P. F., Hayes, P., & Horrocks, I. (2004). OWL Web ontology language semantics and abstract syntax. Retrieved June 28, 2007, from *http://www.w3.org/TR/ owl-absyn/*

Powers, S. (2003). *Practical RDF*. O'Reilly.

Prud'hommeaux, E., & Seaborne, A. (2006). *SPARQL query language for RDF*. World Wide Web Consortium (W3C).

Rector, A., Drummond, N., Horridge, M., Rogers, J., Knublauch, H., Stevens, R., Wang, H., & Wroe, C. (2004). OWL pizzas: Practical experience of teaching OWL-DL: Common errors and common patterns. In *Proceedings of the 14th International Conference of Engineering Knowledge in the Age of the SemanticWeb (EKAW'04)* (pp. 63-81). Springer-Verlag. Lecture Notes in Computer Science 3257.

Schumacher, M. (2003). *Security engineering with patterns: Origins, theoretical model, and new applications*. Springer-Verlag. Lecture Notes in Computer Science 2754.

Simmonds, A., Sandilands, P., & van Ekert, L. (2004). An ontology for network security attacks. In *Proceedings of the 2nd Asian Applied Computing Conference (AACC'04)* (pp. 317-323). Springer-Verlag. Lecture Notes in Computer Science 3285.

Squicciarini, A. C., Bertino, E., Ferrari, E., & Ray, I. (2006). Achieving privacy in trust negotiations with an ontology-based approach. *IEEE Transactions on Dependable and Secure Computing, 3*(1), 13-30.

Stallings, W. (2006). *Cryptography and network security: Principles and practices* (4th ed.). Prentice Hall.

Stevens, R., Goble, C. A., & Bechhofer, S. (2000). Ontology-based knowledge representation for bioinformatics. *Briefings in Bioinformatics, 1*(4), 398-414.

Ször, P. (2005). *The art of computer virus research and defense*. Addison-Wesley.

Takahashi, Y., Abiko, T., Negishi, E., Itabashi, G., Kato, Y., Takahashi, K., & Shiratori, N. (2005). An ontology-based e-learning system for network security. *Proceedings of the 19th International Conference on Advanced Information Networking and Applications (AINA'05), 1*, 197-202. IEEE.

Tsoumas, B., Dritsas, S., & Gritzalis, D. (2005). An ontology-based approach to information systems security management. In Gorodetsky, V., Kotenko, I., & Skormin, V. (Eds.), *Computer Network Security: Third International Workshop on Mathematical Methods, Models, and Architectures for Computer Network Security (MMM-ACNS'05)* (pp. 151-164). Springer-Verlag. Lecture Notes in Computer Science 3685.

Undercoffer, J., Joshi, A., Finin, T., & Pinkston, J. (2004). Using DAML+OIL to classify intrusive behaviors. *The Knowledge Engineering Review*, pp. 221-241.

Venter, H. S., & Eloff, J. H. P. (2003). A taxonomy for information security technologies. *Computers & Security, 22*(4), 299-307.

Wang, H., & Wang, C. (2003). Taxonomy of security considerations and software quality. *Communications of the ACM, 46*(6), 75-78.

Welch, D., & Lathrop, S. (2003). Wireless security threat taxonomy. In *Proceedings of the IEEE Workshop on Information Assurance* (pp. 76-83). IEEE.

Whitman, M. E., & Mattord, H. J. (2005). *Principles of information security* (2nd ed.). Thomson Course Technology.

Wilander, J. (2005). Modeling and visualizing security properties of code using dependence graphs. In *Proceedings of the 5th Conference on Software Engineering Research and Practice in Sweden* (pp. 65-74). Retrieved June 28, 2007, from http: //www.idt.mdh.se/serps-05/SERPS05.pdf

Wilander, J., & Kamkar, M. (2003). A comparison of publicly available tools for dynamic buffer overflow prevention. In *Proceedings of the 10th Network and Distributed System Security Symposium (NDSS'03)* (pp. 149-162). Internet Society.

APPENDIX: INTRODUCTION TO OWL

An OWL ontology typically consists of the concepts that are to be modeled. These concepts are called *classes*, which can have *subclasses*. *Object properties* describe the relation between classes. *Datatype properties* denote attributes of a class. *Individuals* are instances of a class; and *restrictions* on properties model axioms that, for example, constrain the values of that property.

OWL is a description-logic based language based on RDF. RDF can be used to model the world using subject-predicate-object statements called *triples*, for example, THREAT *isEnabledBy* VULNERABILITY. N-ary relations like COUNTERMEASURE *Protects* SECURITYGOAL *and* ASSET *through* DEFENCESTRATEGY cannot be expressed directly and need to be resolved with helper patterns (Noy & Rector, 2006).

The actual OWL code, being a derivative of XML, is verbose. The following example defines a class called 'Data' which is a subclass of a class called 'Technology', which is defined elsewhere in the local ontology.

```
<owl:Class rdf:ID="Data">
    <rdfs:subClassOf>
        <owl:Class rdf:about="#Technology"/>
    </rdfs:subClassOf>
</owl:Class>
```

Humans prefer the more dense, abstract OWL syntax (Patel-Schneider, Hayes, & Horrocks, 2004)

Class (Data **partial** Technology)

which is what we will use when we want to show OWL code. Keywords of OWL are provided in bold face.

Object properties, as shown in the code example below, describe relations between classes. The object property *existsOnAsset*, for example, is a property between *vulnerability* (the domain) and *asset* (the range). A vulnerability exists on an asset. The inverse property between asset and vulnerability is named *assetHasVulnerability*.

```
ObjectProperty(existsOnAsset
    inverseOf(assetHasVulnerability)
    domain(Vulnerability)
    range(Asset))
```

Class definitions can contain restrictions that define constraints on the use of properties. Our ontology makes heavy use of these restrictions, therefore we explain these in detail. The countermeasure class 'vulnerability scanner', with its OWL code below, is our guiding example.

```
Class (VulnerabilityScanner partial
        Countermeasure
        restriction(protects someValuesFrom(_Correctness))
        restriction(protects someValuesFrom(_Host))
        restriction(protects someValuesFrom(_Detection))
        restriction(protects someValuesFrom(_Integrity))
        restriction(protects allValuesFrom(intersectionOf(
          unionOf(_Correctness _Integrity)
          _Host
          unionOf(_Detection _Correction)
        )))
)
```

The class 'vulnerability scanner' is a subclass of *countermeasure*. Every vulnerability scanner protects *at least* the correctness and integrity of a host through detection. This statement is implemented with the someValuesFrom-restrictions on the *protects*-property. Some vulnerability scanners also allow certain correction of found vulnerabilities. This possibility is expressed in the closure axiom, in the allValuesFrom-restriction, which expresses that a vulnerability scanner *at best* protects the correctness or integrity of a host using detection or correction. The syntax of the allValuesFrom-restriction combines intersection and union of the three components—security goal, asset, defence strategy—of the protects-property. In concise format, the restriction is:

$$\forall protects.((\_Correctness \cup \_Integrity) \cap \_Host \cap (\_Correction \cup \_Detection))$$

If the closure of the allValuesFrom-restriction were not given, it would be possible for a reasoner or human to erroneously believe that a vulnerability scanner protects, for example, the privacy of a human, because nothing contradicts this. If the someValuesFrom-restrictions were not given, and only the allValuesFrom-restriction existed, a reasoner would not find the vulnerability scanner as a countermeasure that protects a host because the knowledge base would only express that a vulnerability scanner *at best* could protect a host, but is not required to do so. It could actually protect nothing at all. This kind of reasoning is called the *open world assumption*, which is typical for reasoning on the semantic Web.

In the example, the classes starting with an underscore (e.g. *_Integrity*) are helper classes that denote yet another restriction, namely a restriction on a property of the class that implements the quaternary *protects*-relation. The pattern of helper classes is described in Noy and Rector (2006) and not further explained here.

Restrictions are not only useful as descriptions of constraints. They are a natural starting point for applying inference: Given restrictions on the *protects*-property, it is straightforward for a reasoner to find all countermeasures that protect a certain asset, that protect confidentiality and that use detection or a combination of these. How this is done is shown in section 4.

Synonyms, such as ‘encryption’ and ‘cryptography’, can be handled by declaring two or more classes as equivalent. Thus, all subclasses of ‘encryption’ are also subclasses of ‘cryptography’ and vice versa.

For more about OWL, we recommend the tutorial in Rector, Drummond, Horridge, Rogers, Knublauch, Stevens et al. (2004). The concise language definition is in Bechhofer et al. (2004). Tools for editing, viewing or reasoning about OWL files are presented in the main body of this article.

Almut Herzog received her MSc in Medical Informatics from Heidelberg University, Germany (1994). Between 1994 and 1999 she worked as a software developer in the area of medical information systems. She is currently finishing her PhD studies at the Department of Computer and Information Science, Linköpings universitet, Sweden. Her research interest is in information security, and specifically Java security and usable security.

Nahid Shahmehri received her PhD in 1991, in the area of programming environments, and has been a full professor in computer science at Linköping University since 1998. Since 1994 her research activities have been concerned with various aspects of engineering advanced information systems, her main research interests being software security, security and usability, and trust/policy management. Professor Shahmehri chairs the Swedish Section of the IEEE Chapter for Computer/Software and the steering committee for the IEEE International Conference on Peer-to-Peer Computing. She is one of the founders of the iTRUST Center for Information Security.

Claudiu Duma received his MSc in computer science from the Polytechnic University of Timisoara (1998), and his PhD in Computer Science from Linköping University (2005). Dr. Duma has performed research in the area of secure group communication, with focus on the security-efficiency tradeoff in key management, and trust and reputation management, with focus on resilient reputation systems and integrated frameworks for trust management. His latest research interests are identity management and software security. Dr. Duma acts as a reviewer, and is part of the program committee, for several conferences and workshops on security and peer-to-peer computing.