# O Uso de Ontologia em Alertas de Vulnerabilidades

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

[1]Instituto de Ciências Matemáticas e de Computação – Universidade de São Paulo (ICMC-USP)
Caixa Postal 668 – São Carlos – SP – Brazil

{brandao, luciana, edson}@icmc.usp.br

***Abstract****. Computational systems have become complex and usually the use of diverse tools is necessary to ease the task of intruders' detection and maintenance of the security. However, as each tool produces alerts in a different way, the administrator has to analyze each one to make correlations. Aiming to assist this task, this paper considers the use of ontology to classify known vulnerabilities. We present basic concepts of ontologies, advantages of the use in computational security, the methodology for creation and propose a ontology of vulnerabilities implemented using the OWL language.*

***Resumo:*** *Os sistemas computacionais tornaram-se complexos e geralmente é necessário o uso de diversas ferramentas para facilitar a tarefa de detecção de intrusos e manutenção da segurança. Porém, cada ferramenta produz alertas personalizados, cabendo ao administrador a tarefa de abstrair valor semântico para correlacioná-los. Visando auxiliar essa tarefa, este artigo propõe o uso de ontologia para classificar vulnerabilidades conhecidas. São introduzidos conceitos básicos de ontologias, vantagens do uso em segurança computacional, a metodologia para criação e é apresentada uma ontologia de vulnerabilidades implementada utilizando a linguagem OWL.*

## 1. Introdução

Com a grande evolução da Internet, cada dia mais pessoas e sistemas computacionais estão conectados por meio das redes de computadores e cada vez mais as informações precisam ser protegidas. Nesse contexto, a preocupação com a segurança computacional, principalmente com a segurança da informação propriamente dita, tem se tornado mais presente nas organizações. Atualmente, a segurança computacional tem gerado grandes quantidades de informação, tornando as tarefas de manipular e gerenciar essa informação bastante custosa.

Tecnologias como Bancos de Dados, *Data Warehouse* e Mineração de Dados (*Data Mining*) têm auxiliado na execução dessas tarefas, com o intuito de facilitar a geração de conhecimento que possa ser utilizado para que administradores de sistemas tomem decisões a respeito de problemas com segurança da informação.

Em sistemas de segurança computacional, o administrador tem em mãos uma grande quantidade de dados e informações que devem ser gerenciados e analisados a fim de tomar uma decisão caso o sistema seja invadido. Esses dados podem ser de fontes diretamente relacionadas à segurança, tais como: *logs* de acesso às máquinas que compõem o sistema, *logs* do *firewall* e alertas de vulnerabilidades, e/ou de fontes de dados que podem caracterizar uma quebra de segurança, tais como: estatísticas de taxas

de utilização de processadores, estatísticas da utilização da largura de banda, sobrecarga de memória. Devido a essa grande quantidade de dados e informações disponíveis, torna-se, muitas vezes, impraticável seu processamento em tempo hábil para uma tomada de decisão antes que um incidente alcance grandes proporções.

Diferentes institutos de pesquisas têm realizado esforços no sentido de catalogar e classificar dados e informações relacionados à segurança computacional. O projeto CVE (*Common Vulnerabilities and Exposures*), desenvolvido no Instituto Mitre[1], apresenta um padrão para nomenclatura de vulnerabilidades que facilita a identificação de uma mesma vulnerabilidade em diferentes ferramentas [Mann e Christey, 1999].

O CERT®/CC (*Computer Emergency Response Team/Coordination Center*)[2], que é um centro de informações de segurança computacional fundado pela Universidade de *Carnegie Mellon* nos Estados Unidos, armazena grandes quantidades de informações relacionadas a vulnerabilidades, incidentes de segurança, alertas de vírus, entre outras.

Apesar dos avanços das iniciativas CERT® e CVE, tais esforços não agregam semântica às informações das vulnerabilidades armazenadas e divulgadas. Entende-se por semântica a capacidade de processar o significado dos conceitos e não apenas a estrutura da informação. Sem o entendimento semântico, o agente de software fica incapaz de fazer relacionamentos, seja entre alertas de diferentes fontes, seja entre vulnerabilidades divulgadas por uma mesma fonte.

Essa correlação é importante para estabelecer um vocabulário de conceitos e relacionamentos único e formal a respeito de um determinado domínio de aplicação. No sentido de estabelecer esse vocabulário único, diversos pesquisadores têm estudado e aplicado ontologias. Ontologias são representações formais dos conceitos de um domínio e suas relações [Gruber, 1995; Grüninger e Fox, 1995; Guarino e Giaretta, 1995; Uschold, 1996; Fernández López, 1996; Deepsia, 2001; Pacheco e Kern, 2001; Grüninger e Lee, 2002; Herrera *et al.*, 2002].

Este artigo apresenta a iniciativa do uso de ontologia em alertas de segurança, mais especificamente alertas de vulnerabilidades. Foi efetuada a modelagem de uma ontologia de vulnerabilidades, com a especificação de suas classes, propriedades e relacionamentos. A ontologia foi implementada utilizando-se a linguagem OWL [McGuinness e Harmelen, 2003] e utilizou-se a base de vulnerabilidades do projeto CVE como fonte de informações. Foram classificadas na ontologia 117 vulnerabilidades, com seus respectivos softwares, fabricantes e correções. Posteriormente, testes foram efetuados utilizando-se a linguagem RDQL [Seaborne, 2004] com o objetivo mostrar a viabilidade de extrair informações da ontologia resultante.

Este artigo está organizado da seguinte forma. A Seção 2 apresenta uma breve fundamentação a respeito dos conceitos de ontologia. A Seção 3 apresenta os trabalhos relacionados ao tema de catalogação e classificação de vulnerabilidades. Em seguida, a Seção 4 apresenta a metodologia de desenvolvimento utilizada e a ontologia de alertas de vulnerabilidades desenvolvida, enquanto que a Seção 5 apresenta alguns dos testes realizados com a ontologia. A Seção 6 apresenta as vantagens decorrentes do uso de

[1] http://cve.mitre.org/

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

ontologia em alertas de vulnerabilidades. Por fim, a Seção 7 apresenta as considerações finais.

## 2. Ontologias

Segundo Gruber (1995), uma ontologia é uma especificação formal e explícita de uma abstração, uma visão simplificada de um domínio de conhecimento. Uma ontologia modela uma parte do "mundo", ou seja, um domínio de conhecimento, definindo um vocabulário comum. Tal modelo pode ser utilizado tanto por humanos quanto por agentes de software, a fim de estabelecer um entendimento comum sobre os conceitos e relacionamento desse domínio de conhecimento [Schumacher, 2003].

Inúmeros são os benefícios apresentados na literatura para o uso de ontologias [Uschold e Grüninger, 1996; Menzies, 1999; Noy e McGuinness, 2001; Schumacher, 2003], entre eles:

- **Compartilhamento:** permite compreensão comum sobre um domínio de conhecimento.
- **Reúso**: o uso de definições explícitas e formais facilita a manutenção do conhecimento, permitindo o fácil entendimento por parte dos usuários e facilitando a reutilização da ontologia, ou de parte dela.
- **Estruturação da informação**: permite a captura da semântica dos dados e seu processamento automático gerando conhecimento para os humanos.
- **Interoperabilidade**: permite que diferentes sistemas computacionais possam compartilhar dados e informações.
- **Confiabilidade**: uma representação formal torna possível uma automatização consistente e mais confiável.

Independentemente do domínio representado pela ontologia e das possíveis diferenças existentes entre ontologias, alguns elementos são comuns [Chandrasekaran *et al.*, 1999]:

- Existem **objetos** do mundo real a serem representados.
- Objetos têm **propriedades** ou **atributos** que podem assumir valores variados.
- Objetos podem se relacionar com outros objetos.
- **Propriedades** e **relacionamentos** podem mudar com o tempo.
- Existem **evento**s que podem ocorrer em determinados instantes.
- Existem **processos** que podem ocorrer ao longo do tempo.
- O mundo e seus objetos podem estar em diferentes **estados**.
- Eventos podem causar outros eventos ou estados como efeitos.
- Objetos podem ser decompostos ou fazerem parte de outros objetos (agregação).

Cada um desses elementos é instanciado dependendo única e exclusivamente do domínio representado, ou seja, podem não estar presente na ontologia.

Diversas linguagens foram desenvolvidas para representação de ontologias. Alguns exemplos são XOL, SHOE, DAML, RDF/RDF(S), OIL e OWL. Gómez-Pérez e Cocho (2002) apresentam em seu artigo uma comparação entre essas linguagens. Entre essas linguagens, a OWL (*Ontology Web Language*) é a mais promissora. Foi desenvolvida baseando-se nas linguagens DAML e OIL e construída em cima da arquitetura XML e RDF. A OWL foi proposta como padrão pelo W3C[3], agregando diversos pontos positivos das linguagens anteriores, e está sendo utilizada pela WEB Semântica [Berners-Lee *et al.*, 2001].

## 3. Trabalhos Relacionados

Em sistemas de segurança computacional, o administrador tem em mãos uma grande quantidade de dados e informações que devem ser gerenciados e analisados a fim de tomar uma decisão caso o sistema seja invadido. Esses dados podem ser de fontes diretamente relacionadas à segurança e/ou de fontes que podem fornecer dados que auxiliam na elaboração de informações que podem caracterizar uma quebra de segurança. Além disso, existe uma falta de padrão para os dados utilizados pelas diferentes ferramentas e uma falta de estruturas e taxonomias que representem esses dados. Devido a essa grande quantidade de dados e informações disponíveis, muitas vezes torna-se impraticável o processamento da informação em tempo hábil para a tomada de decisão antes que um incidente alcance grandes proporções.

### 3.1 - CERT®/CC

O CERT®/CC foi criado como parte do Instituto de Engenharia de Software (SEI – *Software Engineering Institute*) da Universidade de *Carnegie Mellon* nos EUA, com o intuito de prover uma organização capaz de coordenar respostas a incidentes de segurança na Internet.

O CERT® mantém uma base de dados com as vulnerabilidades conhecidas na Internet. Essa base também possui as correções relacionadas a cada vulnerabilidade. Ao contrário do projeto CVE, as informações mantidas pelo CERT® são restritas e confidenciais, somente as partes interessadas têm acesso a essas informações. Somente as informações relacionadas às características gerais da vulnerabilidade e aos detalhes específicos de como eliminar essa vulnerabilidade são públicos. Essa política é adotada a fim de impedir um ataque a tal vulnerabilidade. A mesma política é adotada com incidentes em páginas na Internet. Somente o *website* atacado recebe todas as informações. Caso o *website* autorize, as informações se tornam públicas.

A partir do CERT®, diversas comunidades, tanto acadêmicas quanto comerciais ou governamentais, surgiram com o intuito de contribuir para a segurança dos sistemas existentes. Atualmente, alertas gerados pelo CERT® são acompanhados pelo respectivo número da vulnerabilidade no projeto CVE.

As descrições constantes nos boletins são feitas em linguagem natural. Nenhum processamento automático é possível a partir dos alertas divulgados pelo CERT. Desta maneira, não supre a demanda pela interoperabilidade de ferramentas.

[3] http://www.w3c.org

## 3.2 O projeto CVE

Segundo Martin (2001), organizações que descobrem uma vulnerabilidade agem, geralmente, como se elas fossem as únicas fontes de informação a respeito daquela vulnerabilidade e utilizam sua própria abordagem para quantificar, nomear, descrever e compartilhar a informação. Essa postura torna difícil o gerenciamento da informação existente com relação às muitas vulnerabilidades encontradas nos diferentes produtos de software.

No sentido de facilitar o gerenciamento do compartilhamento das informações relacionadas às vulnerabilidades, o Instituto Mitre, em 1999, criou o projeto CVE. Esse projeto tem como principal objetivo criar uma base de dados padronizada de alertas de vulnerabilidades. Atualmente, 71 organizações (entre elas: CERT®/CC, CERIAS, SANS, CISCO, SYMANTEC, IBM, SECURITYFOCUS) participam desse projeto e mantém a base de dados, chamada lista CVE (*CVE list*). Essa lista tem a intenção apenas de nomear as vulnerabilidades, e não de representá-las ou classificá-las.

Essa base centralizada possui as seguintes características:

- enumerar as vulnerabilidades conhecidas;
- atribuir um nome padrão e único para cada vulnerabilidade;
- ser independente das diferentes perspectivas em que a vulnerabilidade ocorre;
- ser aberta e compartilhada, sem nenhum tipo de restrição;

A manutenção da lista é resultado de uma discussão aberta e colaborativa entre o corpo editorial do CVE, que é composto por especialistas em segurança de organizações comerciais, acadêmicas e de pesquisa e coordenado por um Editor. Com o suporte do Instituto Mitre, o corpo editorial identifica quais vulnerabilidades ou exposições podem ser incluídas (candidatas CVE – *CVE candidates*) na lista CVE. Cada vulnerabilidade ou exposição candidata recebe uma identificação atribuída pelo CNA (*Candidate Numbering Authority*). Essa identificação é semelhante à identificação de uma vulnerabilidade já incluída na lista. No entanto, ao invés de CVE no nome, tem-se CNA (CNA-1999-0002). Após o fórum de discussão, se a vulnerabilidade for aprovada, ela recebe um nome comum e uma descrição.

A Tabela 1 mostra alguns exemplos de entradas da lista CVE[5]. Os nomes são concatenados pelo ano de descoberta da vulnerabilidade e ordenados de maneira seqüencial. As descrições não apresentam uma estrutura padrão. Além disso, a lista não apresenta nenhum tipo de relação que possa existir entre as vulnerabilidades. Assim, o nome CVE-1999-0002 indica somente a segunda vulnerabilidade registrada em 1999.

A simplicidade do projeto oferece diversas vantagens, tais como: criar uma base de dados compartilhada e mantida por diversas organizações, estabelecer uma compatibilidade entre diferentes ferramentas de segurança que utilizam o padrão CVE. No entanto, o projeto CVE não realiza uma correlação entre as diferentes entradas da lista CVE, impossibilitando, por exemplo, prever a relação entre CVE-2001-0012 e CVE-2001-0301. Além disso, o fato das descrições não seguirem um padrão de

[4] http://www.securityfocus.com/archive/1

[5] Versões da lista CVE podem ser encontradas em http://www.cve.mitre.org/cve/versions/.

estrutura dificulta a extração automática do valor semântico dos dados, ficando, assim, a cargo do administrador analisar essas descrições.

**Tabela 1. Exemplos de entradas da lista CVE**

| Nome | Descrição |
|---|---|
| CVE-1999-0002 | *Buffer Overflow in NFS mountd gives root access to remote attackers, mostly in Linux Systems* |
| CVE-1999-0010 | *Denial of Service vulnerability in BIND 4.9 Releases via maliciously formatted DNS messages* |
| CVE-1999-0022 | *Local user gains root privileges via buffer overlflow in rdist, via expstr() function* |
| CVE-1999-0023 | *Local user gains root privileges via buffer overlflow in rdist, via lookup() function* |
| CVE-1999-0024 | *DNS cache poisoning via BIND, by predictable query ID* |

## 4. Proposta de Uso de Ontologia

O trabalho apresentado por este artigo propôs e desenvolveu uma ontologia capaz de representar informações semânticas a respeito das vulnerabilidades atualmente conhecidas.

Um dos problemas a serem enfrentados para a criação de ontologias é a catalogação dos itens a serem inseridos. Tal problema é ainda mais crítico no caso do domínio em questão, já que vulnerabilidades são descobertas todos os dias.

Como já mencionado, o CVE apresenta catalogação das vulnerabilidades conhecidas, com rígido controle de quais vulnerabilidades são inseridas ou retiradas desse repositório. Devido a isso, optou-se por utilizar o repositório de vulnerabilidades do projeto CVE como ponto de partida para a ontologia.

A ferramenta de busca ICAT[6] também foi utilizada. Tal ferramenta oferece interface de consulta para a base de vulnerabilidades do projeto CVE. Alguns dos atributos existentes na ontologia resultante deste trabalho já estavam classificados na ferramenta ICAT e foram aproveitados, inserindo-se relações semânticas entre eles para interligá-los aos outros conceitos representados.

Para a criação da ontologia, os seguintes passos foram seguidos, baseados na metodologia proposta por Noy e McGuinness (2001):

- **Enumerar os termos do domínio**. Esta fase consistiu em criar uma lista com todos os termos da ontologia. Para isso foram analisados boletins de alertas de vulnerabilidades e os textos presentes na base de alertas do projeto CVE. Alguns exemplos: sistemas operacionais, tipos de ataques, grau de periculosidade, conseqüências, plataformas vulneráveis, etc. Os termos são extraídos das descrições das vulnerabilidades existentes no projeto CVE.
- **Definir as classes e a hierarquia de classes da ontologia**. Nesta etapa os termos resultantes da etapa anterior foram colocados em hierarquias de forma que os mais genéricos foram especializados em termos mais específicos. Para isso, foi utilizada combinação das estratégias *top-down* e *bottom-up*, de acordo com o apresentado em

[6] http://icat.nist.gov

[Noy e McGuinness, 2001]. Isto é, os principais conceitos foram definidos primeiro e foram refinados e/ou generalizados para a definição das demais classes.

- **Criação dos atributos e relacionamentos.** Cada classe possui algumas variáveis que armazenam valores descritivos de suas características. Essas variáveis são conhecidas como atributos. Por exemplo, a classe **Vulnerabilidade** tem o atributo **Descrição**. Alguns atributos têm como possíveis valores outras classes. Por exemplo, a classe **Produto** possui o atributo **possuiFornecedor**, que tem como possíveis valores as instâncias da classe **Fabricante**. Esses atributos que unem duas classes são conhecidos como relacionamentos. A partir deste ponto a ontologia já possui seu modelo conceitual e têm-se os conceitos de classes, relacionamentos, atributos e instâncias definidos.
- **Implementação da ontologia.** A partir do modelo conceitual, a ontologia foi implementada em linguagem OWL. A escolha da linguagem levou em conta o fato de que o consórcio W3C recomenda a OWL como linguagem para representação de ontologias e conteúdos da web semântica.
- **Instanciação das vulnerabilidades.** Nesta fase entradas da base CVE foram cadastradas, classificando nas respectivas classes as vulnerabilidades, fornecedores, produtos, componentes afetados, correções, descrições, pré-requisitos, etc. Atualmente a base do projeto CVE possui 6459 vulnerabilidades. Para validação da ontologia, escolheu-se cadastrar as vulnerabilidades catalogadas no projeto durante o mês de março de 2003. Isto correspondeu a 117 vulnerabilidades, que afetam 378 produtos de 101 fornecedores.

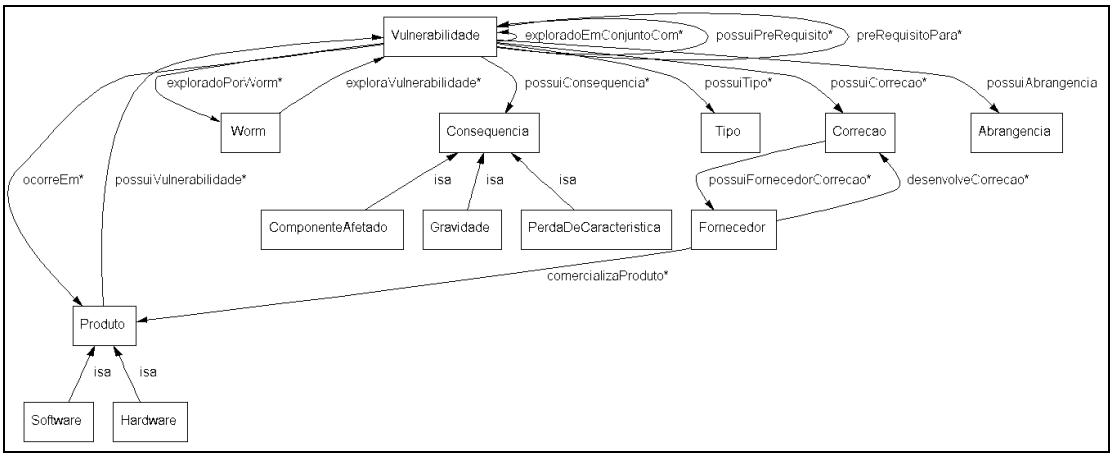


**Figura 1 – Parte do modelo conceitual da ontologia**

A Figura 1 apresenta parte do modelo conceitual da ontologia de vulnerabilidades (o modelo conceitual completo foi omitido por questões de espaço). As classes são representadas por retângulos e as relações por setas. Pode-se perceber que algumas setas (relações) são do tipo *isa*. Essas relações são do tipo superclasse-subclasse, isto é, representam classes que são subclasses daquelas para onde a relação aponta. Esse tipo de relação é também conhecido como pai-filho.

A seguir são descritas as principais classes presentes na figura:

[8] http://protege.standford.edu

- **Classe Vulnerabilidade.** Nessa classe são armazenadas as vulnerabilidades cadastradas. A partir dessa classe partem as relações que informam qual o tipo de vulnerabilidade em questão, sua abrangência (local ou remota), suas conseqüências, o software ao qual está atribuída, correções e relacionamentos com outras vulnerabilidades.
- **Classe Fornecedor.** Armazena dados referentes às empresas ou organizações desenvolvedoras de softwares e de correções. Essa classe está relacionada à classe **Produto** que, por sua vez, está relacionada à classe **Vulnerabilidade**.
- **Classes Software e Hardware.** Essas classes armazenam informações a respeito dos produtos que possuem vulnerabilidades cadastradas. Nelas são encaixados sistemas operacionais, programas servidores, bibliotecas, roteadores e demais produtos que possam apresentar vulnerabilidades
- **Classe Correcao.** Armazena informações sobre as correções (*patches*) das vulnerabilidades cadastradas.
- **Classe Abrangência.** Armazena informações sobre como a vulnerabilidade pode ser explorada e quais as condições necessárias para que isso ocorra. Essa classe possui as seguintes instâncias pré-cadastradas: **AcessoFisico**, **Interceptacao**, **LadoCliente**, **Local** e **Remoto**.
- **Classe Tipo.** Armazena os diversos tipos possíveis de vulnerabilidade. Essa classe possui as seguintes instâncias pré-definidas: **BufferOverflow**, **CondicaoDisputa**, **CrossSiteScripting**, **ErroConfiguracao**, **ErroProjeto**, **ErroTratamentoExcecoes**, **ErroValidacaoAcesso** e **SQLInjection**.
- **Classe PerdaDeCaracteristica.** Armazena informações referentes às características perdidas quando ocorre a exploração da vulnerabilidade. Possui as seguintes instâncias pré-cadastradas: **Confidencialidade**, **Disponibilidade**, **Integridade**, **AcessoNivelUsuario**, **AcessoNivelAdministrador**.
- **Classe Gravidade.** Essa classe também armazena informações referentes às características perdidas quando ocorre a exploração da vulnerabilidade. As instâncias são extraídas do projeto ICAT e utiliza-se, portanto, os mesmos conceitos para gravidade Alta, Baixa, e Média. Possuiu as seguintes instâncias pré-cadastradas: **Alta**, **Baixa** e **Média**.
- **Classe ComponenteAfetado.** Armazena dados dos possíveis componentes do sistema afetados em decorrência da exploração de uma vulnerabilidade. Instâncias pré-cadastradas da classe: **AplicacaoNaoServidora**, **AplicacaoServidora**, **ItemHardware**, **ModuloCriptografia**, **PilhaProtocoloRede**, **ProtocoloComunicacao**, **SistemaOperacional**.

Como já mencionado, a ontologia foi implementada utilizando a linguagem OWL. A **Figura 2** exemplifica a sintaxe da linguagem. Neste trecho, pode-se notar a declaração inicial da ontologia, a definição da classe **Vulnerabilidade** e da relação que indica a presença de pré-requisitos entre vulnerabilidades.

```
<?xml version="1.0" ?>
- <rdf:RDF xmlns:rss="http://purl.org/rss/1.0/" xmlns="http://intermidia.icmc.usp.br/brandao/vulontology#"
    xmlns:jms="http://jena.hpl.hp.com/2003/08/jms#" xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#" xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns:vcard="http://www.w3.org/2001/vcard-rdf/3.0#" xmlns:daml="http://www.daml.org/2001/03/daml+oil#"
    xmlns:dc="http://purl.org/dc/elements/1.1/" xml:base="http://intermidia.icmc.usp.br/brandao/vulontology">
    <owl:Ontology rdf:about="" />
  - <owl:Class rdf:ID="Vulnerabilidade">
      <owl:versionInfo>Versão 0.4</owl:versionInfo>
      <rdfs:comment>Esta classe armazena as vulnerabilidades cadastradas na ontologia</rdfs:comment>
      <rdfs:label xml:lang="en">Vulnerability</rdfs:label>
    </owl:Class>
  - <owl:ObjectProperty rdf:ID="possuiPreRequisito">
    - <owl:inverseOf>
        <owl:ObjectProperty rdf:about="#preRequisitoPara" />
      </owl:inverseOf>
      <rdfs:domain rdf:resource="#Vulnerabilidade" />
      <rdfs:comment>Esta relacao indica que a exploração de uma outra vulnerabilidade é pré-requisito para exploração desta
        vulnerabilidade</rdfs:comment>
      <rdfs:range rdf:resource="#Vulnerabilidade" />
    </owl:ObjectProperty>
```

**Figura 2 - Parte do código da ontologia implementada**

## 5 - Testes efetuados na ontologia

Para comprovação da viabilidade do uso da ontologia, testes foram efetuados. Os testes foram executados utilizando a interface de consultas RDQL e têm como resultados instâncias das classes da ontologia. De acordo com o teste, as instâncias podem ser vulnerabilidades, produtos, correções, etc.

A seguir, são apresentados dois exemplos. O primeiro (Figura 3) mostra a consulta RDQL utilizada para buscar correções da vulnerabilidade CVE-2002-0387. O resultado da execução da consulta é a listagem de instâncias da classe **Correcao** que satisfazem a relação mencionada: CAN-2002-0387-patch1 e CAN-2002-0387-patch2.

```
SELECT ?c

WHERE (<http://intermidia.icmc.usp.br/brandao/vulontology#CAN-2002-0387>,
<http://intermidia.icmc.usp.br/brandao/vulontology#possuiCorrecao>, ?c)
```

**Figura 3 - Consulta RDQL para busca por correções de uma vulnerabilidade**

O segundo exemplo (Figura 4) ilustra a busca por vulnerabilidades do tipo **BufferOverflow** cuja exploração acarreta como conseqüência a perda da disponibilidade do sistema. O resultado é uma lista de 13 instâncias da classe Vulnerabilidade. Entre as entradas retornadas estão: CAN-2003-0100, CAN-2002-1542 e CAN-2003-0099.

```
SELECT  ?vulnerabilidade
WHERE ( ?vulnerabilidade,
<http://intermidia.icmc.usp.br/brandao/vulontology#possuiConsequencia>,
<http://intermidia.icmc.usp.br/brandao/vulontology#Disponibilidade>),
( ?vulnerabilidade,
<http://intermidia.icmc.usp.br/brandao/vulontology#possuiTipo>,
<http://intermidia.icmc.usp.br/brandao/vulontology#BufferOverflow>)
```

**Figura 4 - Consulta RDQL para busca por vulnerabilidades baseando-se em seu tipo e conseqüência da exploração**

Uma das vantagens do uso de ontologias é a possibilidade de outras ontologias serem agregadas à original, aumentando a riqueza semântica do sistema. Portanto, as consultas aqui mostradas poderiam ser expandidas, conforme ontologias forem sendo desenvolvidas em linguagem OWL e agregadas à ontologia de vulnerabilidades. Por exemplo, uma ontologia de softwares poderia ser agregada para que as buscas por vulnerabilidades fossem feitas por softwares produzidos em determinada linguagem.

## 6 - Benefícios do Uso da Ontologia em Alertas de Segurança

Esta Seção visa ilustrar alguns cenários em que há benefício no uso de uma ontologia de vulnerabilidades.

### 6.1 - Auxílio na análise forense

Os relacionamentos entre vulnerabilidades auxiliam o perito durante a análise forense. Vulnerabilidades comumente exploradas em conjunto podem receber relacionamento que indique esse fato. Isso pode ser feito pelos relacionamentos **exploradoEmConjuntoCom**, **preRequisitoPara** e **possuiPreRequisito**.

Por exemplo, se a exploração de uma vulnerabilidade A é pré-requisito para a exploração da vulnerabilidade B, então ao encontrar rastros da exploração de B uma ferramenta forense poderia já indicar ao perito para que procure por rastros da exploração de A.

### 6.2 – Correlacionamento de alertas em tempo real

É comum o uso de diversas ferramentas de segurança simultaneamente e essas ferramentas geram alertas em formatos proprietários. Se as ferramentas utilizadas para análise forem compatíveis com a ontologia, então seus alertas poderão ser gerados como instâncias, já na linguagem OWL. Dessa maneira, ferramentas poderão importar alertas de outras fontes e correlacioná-los por meio dos relacionamentos da ontologia.

### 6.3 - Facilitar a aplicação de correções (*patches*)

Um dos problemas decorrentes do gerenciamento de segurança de sistemas computacionais é o controle das aplicações de correções. Além de registrar quais correções foram aplicadas em seus *hosts*, o administrador deve levar em conta quais as vulnerabilidades são solucionadas por tais correções. Isso está presente na ontologia.

Além disso, algumas correções são mais confiáveis do que outras. Por exemplo, em julho de 2002 a empresa ISS[9], especializada em Segurança da Informação, anunciou a descoberta de uma vulnerabilidade no servidor web Apache[10] [Rash, 2002]. Junto com o anúncio da vulnerabilidade foi disponibilizada correção, desenvolvida pela própria ISS, sem prévia consulta ao fabricante do software Apache. Essa correção na verdade não solucionava a vulnerabilidade e foram grandes os problemas decorrentes do caso.

O histórico das correções disponibilizadas pelo fabricante pode ser usado como aferição da qualidade da correção, antes que esta seja aplicada no sistema. A ontologia pode ajudar nessa tarefa, por meio das classes **Correcao**, **Fornecedor** e **Vulnerabilidades**, que são interligadas por meio dos relacionamentos **desenvolveCorrecao**, **possuiCorrecao**, **possuiFornecedorCorrecao**, **possuiVulnerabilidade**, **ocorreEm** e **comercializaProduto**.

### 6.4 – Documentar o conhecimento tácito em forense computacional

Uma das vantagens do uso de ontologias é a estruturação formal das informações. A partir do momento que as vulnerabilidades, suas características, correções e seus relacionamentos são colocados na ontologia, passa a ser possível cadastrar também

[9] http://www.internetsecuritysystems.com/

[10] http://www.apache.org

parte do conhecimento que, atualmente, está presente apenas como conhecimento tácito nos especialistas.

É o caso, por exemplo, das relações **possuiPreRequisito**, **preRequisitoPara** e **exploradoEmConjuntoCom**. Devido à experiência, especialidades geralmente sabem quais são os requisitos para exploração de uma vulnerabilidade, quais outras são exploradas em conjunto, etc. Isso pode ser catalogado e utilizado para ser usado por ferramentas de segurança.

## 7. Considerações Finais

Este trabalho teve como objetivo investigar e dar os primeiros passos para a utilização de ontologias para classificação de vulnerabilidades em sistemas computacionais. Foi efetuada a modelagem da ontologia para representação dos conceitos do domínio de alertas de vulnerabilidades, com a especificação de suas classes, propriedades e relacionamentos.

Utilizou-se como fonte de dados o projeto CVE e 117 vulnerabilidades foram classificadas. Após a classificação, foi possível efetuar testes que comprovaram a viabilidade do uso da ontologia para facilitar o gerenciamento de segurança em sistemas computacionais.

Este trabalho tratou do uso de ontologias em vulnerabilidades, um subitem do amplo domínio da segurança da informação. Aqui foram dados os passos iniciais e novos trabalhos podem agora serem desenvolvidos para a implementação de ferramentas de segurança capazes de interpretar a linguagem OWL e utilizar a ontologia de maneira análoga aos testes aqui mostrados.

## Referências Bibliográficas


Berners-Lee, T., Hendler, L, Lassila, O. (2001) “The Semantic WEB”. Scientific American 284, 5. pp. 34-43.

Brandão, A. J. S. (2003) “Uso de Ontologias em Sistemas de Segurança”. Exame de Qualificação de Mestrado, ICMC-USP. São Carlos-SP. 47p.

Chandrasekaran, B., Josephson, J.R., Benjamins, V. R. (1999) “What Are Ontologies, and Why Do We Need Them?” IEEE Intelligent Systems. p. 20-26.

Deepsia (2001) “Dynamic *on-line* Internet Purchasing System based on Intelligent Agents”. Disponível *on-line* em http://www.deepsia.com/br.

Fernández López, M. (1996) “CHEMICALS: Ontologia de elementos químicos”. Final-Year Project. Facultad de Informática de la Universidad Politécnica de Madrid.

Garfinkel, S., Spafford, G. (1996) “Practical UNIX and Internet Security”. 2ª Edição. O´Reilly & Associates, Inc. 971p.

Gómez-Pérez, A., Corcho, O. (2002) Ontology languages for the semantic web”. IEEE Intelligent Systems, 17(1). p.54-60 .

Guarino, N., Giaretta, P. (1995) “Ontologies and Knowledge Bases: Towards a Terminological Clarification”. Towards Very Large Knowledge Bases. Ed. IOS Press. pp. 25-32.

Gruber, T. R. (1995) “Toward principles for the design of ontologies used for knowledge sharing”. In Formal Ontology in Conceptual Analysis and Knowledge Representation. Kluwer Academic Publishers.

Grüninger, M., Fox, M. S. (1995) “Methodology for the Design and Evaluation of Ontologies”. Proceedings of the International Joint Conference on Artificial Inteligence (IJCAI95), Workshop on Basic Ontological Issues in Knowledge Sharing, April.

Grüninger, M., Lee, J. (2002) “Ontology: Applications and Design”. Communications of the ACM, February, Vol. 45, Nº 02. pp. 39-41.

Herrera, J. A. F., Martins Júnior, J., Moreira, E. S. (2002) “A Model for Data Manipulation and Ontology Navigation in DEEPSIA Project”. In First Seminar on Advanced Research in Electronic Business (EBR02). Novembro. Rio de Janeiro-RJ.

Mann, D. E., Christey, S. M. (1999) “Towards a Common Enumeration of Vulnerabilities”. Disponível *on-line* em http://cve.mitre.org. Acesso em 01/12/2003.

Martin, R. A. (2001) “Managing Vulnerabilities in Network Systems”. IEEE Computer, 2001. Vol. 34, Nº 11. p. 32-38.

McGuinness, D. L., Harmelen F. (2003) OWL Web Ontology Language Overview. W3C Candidate Recommendation 18 August 2003. Disponível *on-line* em: http://www.w3.org/TR/2003/CR-owl-features-20030818/. Acesso em 28/11/2003.

Menzies, T. (1999) “Cost Benefits of Ontologies”. ACM Press – Intelligence. p. 26-32.

Noy, N.F., McGuinness, D. L. (2001) “Ontology Development 101: A Guide to Create Your First Ontology”. Knowledge Systems Laboratory Technical Report KSL-01-05, Stanford University. 25p.

Pacheco, R.C.S., Kern, V.M. (2001) “Uma Ontologia Comum para a Integração de Bases de Informação e Conhecimento sobre Ciência e Tecnologia”. Revista Ciência da Informação, Vol.30, Nº 03, Set/Dez. Disponível *on-line* em http://www.ibict.br/cionline/300301/3030801.pdf. Acesso em 20/11/2003. pp. 56-63.

Rash, 2004. Rash, W. Apache worm highlights operational flaws. ZDNet Austrália. 11/07/2002. Disponível *on-line* em: http://www.zdnet.com.au/news/communications/0,2000061791,20266325,00.htm. Acesso em 20/02/2004.

Schumacher, M. (2003) “Security Engineering with Patterns – Origins, Theoretical Model, and New Applications”. Lecture Notes in Computer Science (LNCS 2754). p. 29-44.

Seaborne, A. (2004). “RDQL - A Query Language for RDF”. W3C Member Submission 9 January 2004. Disponível *on-line* em: http://www.w3.org/Submission/RDQL/. Acesso em 10/03/2004.

Uschold, M, Grüninger, M. (1996) “Ontologies: Principles, Methods and Applications”. Knowledge Engineering Review. Vol. 11, Nº 02. June.

Uschold, M. (1996) “Building Ontologies: Towards a Unified Methodology”. Proceeding of Expert Systems - 16TH Annual Conference of the British Computer Society Specialist Group in Expert System, December.