O uso de ontologias para o desenvolvimento seguro de software utilizando a metodologia do CLASP

Rogério Winter
Instituto Tecnológico da Aeronáutica – Praça Marechal Eduardo Gomes, 50 – Vila das Acácias. CEP 12228-900 – São José dos Campos – SP

Resumo **—** Cada vez mais os negócios dependem de informações armazenadas nos diversos sistemas corporativos. Ao mesmo tempo, a cada dia surgem novos tipos de ameaças – especialmente na Internet – que colocam em risco a segurança dessas informações.

Na proteção de um sistema computacional um administrador de segurança ou o desenvolvedor necessita ter nas mãos uma informação de boa qualidade a fim de que a tomada de decisão seja eficiente mesmo em momentos de exceção.

Hoje existem diversas metodologias para o desenvolvimento de software. Todas com pontos fortes e fragilidades que priorizam uma determinada característica em um projeto. Neste contexto, o CLASP é uma metodologia para o desenvolvimento de software com ênfase na segurança.

Palavras-chaves **— Segurança, CLASP, desenvolvimento.**

1. INTRODUÇÃO

Em um ambiente digital como a web, a falta de uma presença física e explícita da semântica dos dados tem evidenciado a necessidade por estruturas de informação que facilitem o entendimento dos usuários e a navegação no espaço conceitual. Portanto, as pesquisas nessa área retornam aos princípios básicos, usados como fundamentos na construção dessas arquiteturas, tais como: esquemas de classificação, ontologias, vocabulários controlados e tesauros. [LIMA MARQUES, 2006]

A análise de informação, para obter uma compreensão melhor dos desenvolvimentos atuais e para estimular resultados futuros, é um componente essencial da tomada de decisões. A sobrecarga de informação contribui para a sobrecarga cognitiva [THOLT, 2006]. Assim, paira a dúvida se a informação armazenada é suficiente para compreender um problema ou atingir a consciência situacional.

O objetivo deste artigo é apresentar o uso de ontologias no desenvolvimento seguro de software utilizando a metodologia proposta pelo CLASP.

O presente artigo está organizado da seguinte forma. A Seção 2.1 apresenta uma breve fundamentação a respeito dos conceitos de ontologia. A Seção 2.2 apresenta a metodologia do CLASP. Na Seção 2.3 apresenta o uso de ontologia com o CLASP. A Seção 2.4 apresenta uma revisão de literatura e na Seção 2.5 um posicionamento sobre o assunto. Encerrando o trabalho a Seção 3 apresenta a conclusão.

2. DESENVOLVIMENTO

2.1 ONTOLOGIAS

Conforme CHAUÍ (2000), a partir da classificação aristotélica, os campos da investigação filosófica são três:

1) o conhecimento da realidade última de todos os seres, ou da essência de toda a realidade. Como, em grego, **ser** se diz *on* e **os seres** se diz *ta onta*, este campo é chamado de **ontologia;**

2) o conhecimento das ações humanas ou dos valores e das finalidades da ação humana: valores morais, valores e as técnicas e as artes e seus valores;

3) o conhecimento da capacidade humana de conhecer é **a epistemologia**.

Porém, no contexto da Ciência da Computação, a Ontologia é considerada o caminho obrigatório para o desenvolvimento, na medida em que permite a solução dos problemas por ela atualmente enfrentados.

Uma das mais fortes razões para o desenvolvimento de ontologias é a possibilidade de compartilhamento e reutilização de conhecimento formalmente representado para uso em sistemas de Inteligência Artificial e de Arquitetura da Informação, o que exige a definição de um vocabulário comum para representação do conhecimento a ser aplicado. As entidades e relações constituintes da conceituação do conhecimento de um domínio, quando representadas em um formalismo declarativo, são denominadas universo de discurso. [LIMA MARQUES, 2006]

Para BREITMAN (2005), ontologias são especificações formais e explícitas de conceitualizações compartilhadas. Ontologias são modelos conceituais que capturam e explicitam o vocabulário utilizado nas aplicações semânticas. Servem como base para garantir uma comunicação livre de ambigüidades.

Uma ontologia constitui um documento ou arquivo que define formalmente as relações entre termos, sendo normalmente especificada por:

- uma taxonomia, que define classes e subclasses de objetos e as relações entre elas, muitas das quais podem ser expressas atribuindo propriedades as classes e permitindo que suas subclasses as adquiram por herança;
- um conjunto de regras de inferência que permita a um programa de computador, mesmo sem de fato compreender a informação analisada, realizar deduções e manipular os termos visando auxiliar seus usuários. (BERNERS-LEE; HENDLER; LASSILA, 2001).

2.2 CLASP (Comprehensive Lightweight Application Security Process)

A necessidade de proteger a informação estratégica para o negócio deve estar ligada a premissa de não reduzir a produtividade com um investimento compatível com o risco. Isso inclui a definição da arquitetura de segurança a ser

adotada, que deve estar de acordo com as normas, leis e regulamentos – e até mesmo com as práticas de governança da empresa. Portanto, a incorporação de requisitos de segurança ao software deve estar presente desde a sua concepção, passando pelos controles de acesso.

Open Web Application Security Project (OWASP)[1]é uma comunidade aberta dedicada a permitir que organizações desenvolvam, comprem e mantenham aplicações confiáveis. O CLASP se integra ao OWASP como um projeto de documentação focado em definir elementos do processo de desenvolvimento com ênfase na segurança.

CLASP do acrônimo em inglês que significa **C**omprehensive (completo), Lightweight (leve), Application, Security, Process, é a conseqüência do extensivo trabalho de campo, no qual os recursos de desenvolvimento de sistema, de vários ciclos de vida de software, foram metodicamente decompostos de maneira a criar um compreensivo conjunto de exigências de segurança [YANO, 2008]. As exigências resultantes formam melhores práticas do CLASP que permitem que os desenvolvedores se organizem sistematicamente e tratem as vulnerabilidades. Estas, se exploradas, podem resultar na falha de serviços básicos de segurança como, por exemplo, confidencialidade, autenticidade e controle de acesso.

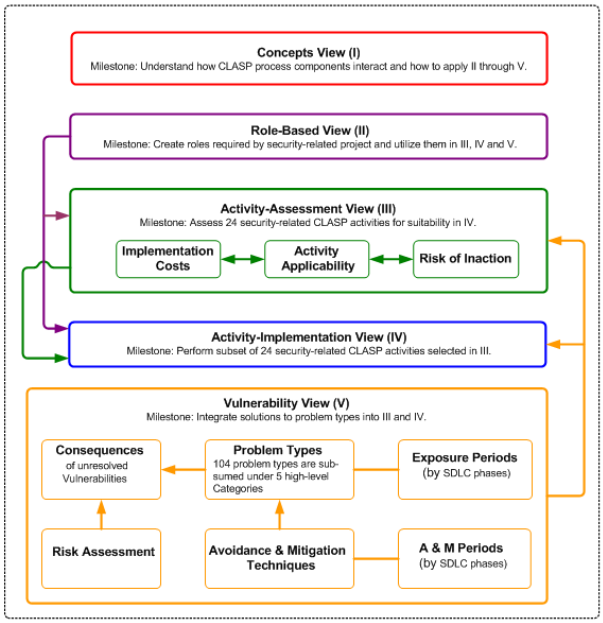

Figura 2 - Visão geral do Processo do CLASP

O CLASP está organizado em visões, recursos e casos de uso de vulnerabilidades. As visões do CLASP provêem o conhecimento da estrutura do processo e das dependências entre seus componentes. Já os recursos determinam os artefatos úteis para implementar os processos e os casos de uso de vulnerabilidades são as descrições de condições sobre as quais serviços de segurança podem se tornar vulneráveis.

O CLASP possui um léxico detalhado da vulnerabilidade que ajuda as equipes do desenvolvimento a evitar os erros do projeto. Além de possuir codificação específica que pode conduzir a exploração dos serviços de segurança. As conseqüências de vulnerabilidades exploradas provocam as falhas de um ou mais dos seguintes serviços de segurança: Autorização (controle de acesso), Confidencialidade, Autenticação (controle de identidades), disponibilidade, responsabilidade final e não repudiação.

A base do léxico é uma taxonomia altamente flexível - isto é, uma estrutura de classificação que permite as equipe de desenvolvimento encontrar rapidamente informações de várias perspectivas, por exemplo: tipos de problema; categorias dos tipos de problemas; períodos de exposição nos quais as vulnerabilidades podem ser introduzidas; conseqüências; plataformas e linguagens de programação; recursos requeridos para atacar as vulnerabilidades; avaliação de riscos das vulnerabilidades e períodos para mitigação. Esta taxonomia do CLASP identifica 104 tipos de problemas (ou causas básicas).

Como exemplo de taxionomia do CLASP tem-se:

Categoria 1: erros de tipagem e limites

Buffer overflow

Visão geral: buffer overflow condição que existe quando um programa tenta colocar mais dados no buffer do que estava dimensionado.

Conseqüências

Disponibilidade: buffer overflow geralmente pode levar a travamentos;

Controle de acesso: buffer overflow pode ser usado para executar código arbitrário;

Outros: utilizado para subverter a qualquer outro serviço de segurança

Período de Exposição

Requisitos especificação: A escolha pode ser feita para usar uma linguagem que não é sensível a estas questões.

Projeto: tecnologias de mitigação, tal como safe - string libraries e container abstractions;

Implantação: Isto pode ser exacerbado pela perda ou pelo mal uso de tecnologias

Plataforma

Linguagens: C, C + +, Fortran

Plataformas de operação: Todos

Recursos necessários

Qualquer

Gravidade Muito alta

Probabilidade de exploração Alta a muito alta Prevenção e mitigação

Uso de uma linguagem ou compilador que realiza verificação automática de limites.

Exemplos

No código, aqui é um simples, no caso previsto exemplo:

```
void example(char *s) {
    char buf[1024];
            strcpy(buf, s);
          }
          int main(int argc, char **argv) {
           example(argv[1]);
          }
```

Os problemas relacionados com

- Stack overflow
- heap overflow
- Integer overflow

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

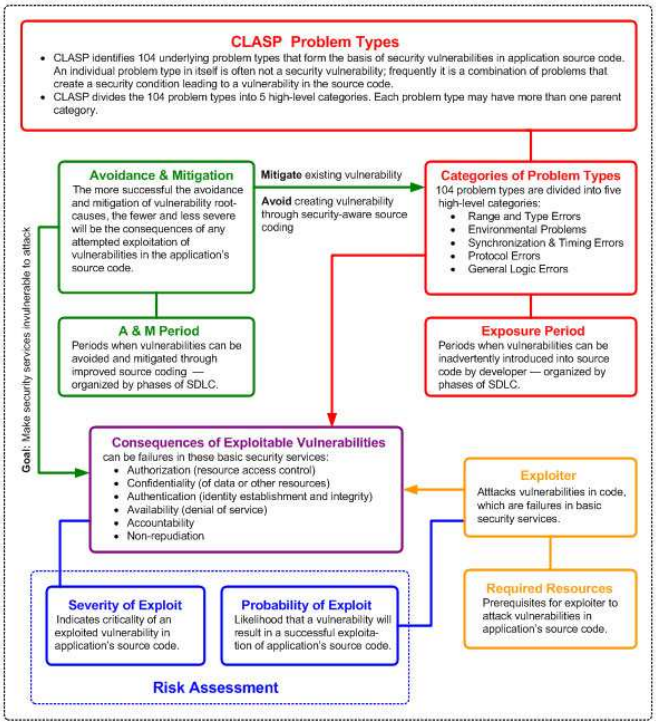

Figura 1 – taxionomia do CLASP

2.3 O USO DE ONTOLOGIA COM O CLASP

Como visto, o CLASP possui um sistema de classificação de vulnerabilidades com uma taxionomia própria. Entretanto, a taxionomia não apresenta nenhum tipo de relação que possa existir entre os diversos tipos de vulnerabilidades. Fica difícil estabelecer relações entre as diversas vulnerabilidades e os seus efeitos.

A fim de tornar ágil a tomada de decisão na execução do acompanhamento de vulnerabilidades, propõe-se a criação de uma ontologia utilizando a estrutura taxionômica do CLASP.

Desta forma o desenvolvimento de uma ontologia propiciaria:

- **Compartilhamento:** permite a compreensão comum sobre as vulnerabilidades no desenvolvimento de software.
- **Reuso**: o uso de definições explícitas e formais facilita à manutenção do conhecimento agregado as vulnerabilidades, permitindo o fácil entendimento por parte dos usuários e possibilitando 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 desenvolvedores e gerentes.
- **Interoperabilidade**: permite que diferentes sistemas computacionais possam compartilhar dados e informações.
- **Confiabilidade**: uma representação formal torna possível uma consistente e confiável automatização.

Na estruturação da ontologia desenvolvida, as perguntas abaixo deveriam ser respondidas:

- Quais são as vulnerabilidades cadastradas?
- Qual são as vulnerabilidades e a correção necessária?
- Qual é o programador com maior número de erros?
- Qual é o software com maior número de vulnerabilidades?
- Quais são as vulnerabilidades com risco alto?
- Qual é a plataforma com maior número de vulnerabilidades?
- Quais são as conseqüências da exploração de vulnerabilidades?
- A exploração da vulnerabilidade está relacionada com outros problemas?

Para a criação da ontologia, os seguintes passos devem ser seguidos, baseados na metodologia proposta por Noy e McGuinness [7]:

• Enumerar os termos do domínio. Esta fase consiste em criar uma lista com todos os termos da ontologia. Para isso, deve-se analisar o documento *Vulnerability View* do CLASP. Alguns exemplos: sistemas operacionais, tipos de ataques, grau de periculosidade, conseqüências, plataformas vulneráveis, etc.

• Definir as classes e a hierarquia de classes da ontologia. Nesta etapa os termos resultantes da etapa anterior devem colocados em hierarquias de forma que os mais genéricos foram especializados em termos mais específicos. Para isso, pode-se utilizar combinação das estratégias *top-down* e *bottom-up*.

• 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 **Fragilidade** tem o atributo **Descrição**. Alguns atributos têm como possíveis valores outras classes. Por exemplo, a classe **Software** está relacionada com o atributo **possuiFabricante**, que tem como possíveis valores as instâncias da classe **Implementador**. Esses atributos que unem duas classes são conhecidos como relacionamentos. A partir deste ponto a ontologia possui um modelo conceitual e têm-se os conceitos de classes, relacionamentos, atributos e instâncias definidos.

• Aplicação da ontologia. A partir do modelo conceitual, a ontologia pode ser implementada em uma linguagem qualquer, entretanto é de bom alvitre que se opte pela OWL. A escolha desta linguagem leva 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 CLASP serão cadastradas, classificando nas respectivas classes as vulnerabilidades, tipo de problema, período de exposição, conseqüências, correções, descrições, conseqüências, plataformas e linguagens de programação, etc. Atualmente a base do CLASP possui 104 tipos de problemas.

São as principais classes presentes na ontologia:

Classe Fragilidade. 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 **Tipos_Falhas**. Armazena dados referentes aos tipos de falhas de softwares. Os valores são: Erros de tipagem e de limites, Problemas ambientais, Erros de temporização e de sincronização, Erros de protocolo, Erros gerais de lógica.

Essa classe está relacionada à classe **Consequencia** que, por sua vez, está relacionada à classe **Plataforma**.
Classe **Correcao**. Armazena informações sobre as correções das vulnerabilidades cadastradas.

2.4 REVISÃO DA LITERATURA

A fim de facilitar o compartilhamento e melhorar o gerenciamento das informações relacionadas às vulnerabilidades, o Instituto MITRE[2], criou o projeto CVE (Common Vulnerabilities and Exposure).

Esse projeto tem como principal objetivo criar e manter uma base de dados padronizada de alertas de vulnerabilidades. Desta forma, foram criados identificadores originais, comuns para vulnerabilidades e publicamente conhecidos da segurança da informação. O CVE não é apenas uma base de dados de vulnerabilidades, mais do que isso, o seu propósito foi permitir que bases de dados de vulnerabilidades e outros potenciais serviços ligados a segurança da informação possam consultar e estabelecer comparações entre ferramentas e serviços de segurança. Organizações, tais como: CERT/CC, CERIAS, SANS, CISCO, SYMANTEC, IBM, SECURITYFOCUS participam desse projeto e mantêm a base de dados, chamada lista CVE (*CVE list*).

Essa base de informações tem a intenção de nomear as vulnerabilidades, e não de representá-las ou classificá-las. Para tanto, ela possui como 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.

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*).

Entretanto, para que uma vulnerabilidade seja aprovada ela passa por um fórum de discussão, só então, ela recebe um nome comum e uma descrição. A formação do nome é composta pelo ano de descoberta da vulnerabilidade e numerada em ordem crescente. Desta forma, uma vulnerabilidade nomeada com CVE-2005-0014 corresponde à décima quarta vulnerabilidade confirmada e registrada em 2005.

Uma das características marcantes do CVE a simplicidade oferece diversas vantagens, tais como: criar uma base de dados compartilhada e mantida por diversas organizações, além de estabelecer uma compatibilidade entre diferentes ferramentas de segurança que utilizam o padrão CVE.

2.5 POSICIONAMENTO

A administração de um projeto de software é uma tarefa difícil, em virtude do número de variáveis que acompanha a empreitada. A cada dia se descobrem novas vulnerabilidades, as quais mantêm os sistemas computacionais reféns da própria sorte. Um sistema bem administrado depende de informações de qualidade e em tempo oportuno, além do relacionamento intrínseco das vulnerabilidades com outras partes do projeto de software.

Com o emprego de ontologias e com a identificação semântica das informações, haverá uma quantidade indeterminada de possibilidades para o projeto de mecanismos de recuperação de informações.

Portanto, o uso de ontologias em sistema de alerta de vulnerabilidades é uma solução plausível e exeqüível. Entretanto, independentemente da metodologia utilizada ou do domínio de conhecimento, o desenvolvimento de uma ontologia é um processo iterativo, dinâmico e custoso. Uma ontologia evolui constantemente, principalmente para atender as mudanças ocorridas com o domínio de conhecimento que ela representa.

Para isso, o CLASP como metodologia de desenvolvimento seguro de software permite um acompanhamento completo de todas as suas fases. Além disso, o CLASP está calcado na simplicidade e em uma bem estruturada taxionomia de classificação de vulnerabilidades.

A taxionomia de vulnerabilidades do CLASP possui um cadastro bem completo com uma vocação no processo de desenvolvimento seguro de software. Entrementes, o desenvolvimento de uma ontologia utilizando a classificação do CLASP permitiria uma ligação com outras ontologias da área de segurança da informação tornando bem mais completo as buscas e inferências no domínio do conhecimento.

3. CONCLUSÃO

A chave para a compreensão dos problemas que existem na segurança de informática é um reconhecimento de que os problemas não são novos. Eles são problemas antigos, que datam do início da segurança de informática [BISHOP, 2002]. A literatura técnica é profícua em exemplos sobre falhas de segurança em software. Todo projeto de software deveria, pelo menos em tese, garantir a confidencialidade, integridade e disponibilidade dos dados armazenados no sistema informatizado.

A metodologia CLASP apresenta um processo bem estruturado de construção segura de software. Além disso, o CLASP possui uma taxionomia de vulnerabilidades abrangente com as características bem definidas, os impactos descritos e as conseqüências mapeadas. Contudo, a taxionomia não apresenta nenhum tipo de relação com as vulnerabilidades do modelo. Por conseguinte, a construção de uma ontologia que agregue os conceitos da taxionomia do CLASP e os relacione entre si permitindo a execução de inferências para a extração de conhecimento é um aspecto relevante para a melhoria da metodologia.

4. REFERÊNCIA BIBLIOGRÁFICA

[1] ANTONIOU, Grigoris; HARMELEN, Frank van. **A Semantic Web Primer**. MIT Press, Cambridge, MA, 2004.

[2] BERNERS-LEE, T; HENDLER, J; LASSILA, O. **The Semantic Web. Scientific** American, p. 5-7, mai 2001.

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

[3] BISHOP,Matt. **Computer Security: Art and Science.** Addison Wesley. USA. November 29, 2002.

[4] BREITMAN, karin Koogan. **Web Semântica - A Internet do Futuro**. LTC – Livros Técnicos Científicos Editora S.A, Rio de Janeiro, 2006.

[5] BREITMAN, Karin Koogan; CASANOVA, Marco Antonio; TRUSZKOWSKI, Walter. **Semantic Web: Concepts, Technologies and Applications**. London, Springer-Verlag, 2007.

[6] CHAUI, Marilena. **Convite à Filosofia.** PP 50. Ed. Ática, São Paulo, 2000.

[7] LIMA-MARQUES, Mamede. **Ontologias: da filosofia à representação do conhecimento**. Thesaurus. Brasília. 2006.

[8] NOY, Natalya F.; MCGUINNESS, Deborah L. **Ontology Development 101: A Guide to Creating Your First Ontology**. Disponível em: <http://protege.stanford.edu/publications/ontology_development/ontology101.html>. Acesso em: 10 Jun 2008.

[9] YANO, Edgar Toshiro. **Segurança Lógica de Software**. Notas de aula do Instituto Tecnológico de Aeronáutica. São José dos Campos. 2008.

[10] _____. **OWASP CLASP Project**. Disponível em: http://www.owasp.org/index.php/Category:OWASP_CLASP_Project. Acessado em 15 de Jun 08.

[11] THOLT, Carlos. **Decida com Inteligência.** Thesaurus, ABRAIC, Brasília, 2006.

Rogério Winter, winter@ita.br, Tel +55-12-39476897