Taxonomias de Vulnerabilidades: Situação Atual

André Ricardo Abed Grégio[1], Luiz Gustavo C. Barbato[1,2], Luiz Otávio Duarte[1], Antonio Montes[1,2], Cristine Hoepers[1,3], Klaus Steding-Jessen[1,3]

[1]Laboratório Associado de Computação Aplicada
Instituto Nacional de Pesquisas Espaciais

[2]Centro de Pesquisas Renato Archer
Ministério da Ciência e Tecnologia

[3]Centro de Estudos, Resposta e Tratamento de Incidentes de Segurança no Brasil
Comitê Gestor da Internet no Brasil

{andre.gregio,lgbarbato,duarte}@lac.inpe.br,
antonio.montes@cenpra.gov.br, {cristine,jessen}@cert.br

Abstract. *There is a large number of software vulnerabilities being discovered every year, but there is no accepted classification scheme or standard format to store information about vulnerabilities. This paper presents a survey of several initiatives in the area of vulnerabilities' taxonomies and classification, including the early proposals and more recent work.*

Resumo. *É grande o número de vulnerabilidades em softwares que têm sido descobertas a cada ano, porém não existe nenhum tipo de classificação ou padronização das informações que devem ser mantidas a respeito dessas vulnerabilidades. Este artigo apresenta uma revisão dos trabalhos nas áreas de taxonomias e classificações de vulnerabilidades, discutindo desde as primeiras propostas até os trabalhos mais atuais.*

1. Introdução

Ao longo dos últimos anos tem sido encontrado um grande número de vulnerabilidades em softwares. O CERT Coordination Center, por exemplo, tem recebido em média 10 notificações de novas vulnerabilidades por dia nos últimos quatro anos [1]. Devido ao grande número de vulnerabilidades existentes em softwares em utilização tem sido necessário um grande esforço para compreender estas vulnerabilidades e sugerir possíveis correções. Além disso, existem poucos mecanismos amplamente difundidos para classificar estas vulnerabilidades.

Segundo Seacord e Householder, compreender as vulnerabilidades é crucial para entender as ameaças que elas representam. A classificação de vulnerabilidades permite coletar dados sobre a freqüência; análise de tendências de vulnerabilidades; correlação com incidentes, *exploits* e artefatos; e avaliação da efetividade das contramedidas [2].

Neste artigo é apresentada uma revisão das taxonomias de vulnerabilidades e dos métodos de classificação existentes, visando discutir a sua aplicabilidade na área de segurança de software. Com isso espera-se contribuir para a área de análise de vulnerabilidades.

O artigo está dividido como segue: a Seção 2 define o conceito de taxonomia e suas características desejáveis. Na Seção 3 é discutida a dificuldade em definir uma terminologia para segurança de software e são definidos diversos termos usados ao longo do artigo. Uma discussão de alguns esquemas para a classificação de vulnerabilidades, notadamente aqueles que serviram de base para os modelos mais atuais e também os que tiveram uma maior visibilidade na comunidade de segurança, é apresentada na Seção 4. Já na Seção 5 são descritas algumas classificações em uso atualmente, enquanto na Seção 6 são apresentadas iniciativas recentes para melhor descrever e classificar vulnerabilidades assim como para utilizar esses dados na definição de estratégias de recuperação. As conclusões são colocadas na Seção 7.

2. Taxonomias

Uma taxonomia é o processo científico (ou um sistema particular) de categorizar entidades, ou seja, organizá-las em grupos. Um sistema de taxonômico deve, além de ser claro e consistente, cumprir os requisitos de ser flexível, extensível e prático.

As bases para o desenvolvimento de uma boa taxonomia são as características taxonômicas. Elas são as propriedades ou características dos objetos que serão categorizados. As características taxonômicas são comumente chamadas de atributos e devem satisfazer as seguintes propriedades [3, 4]:

- Objetividade: a característica deve ser identificada a partir do conhecimento objetivo e não do conhecimento subjetivo. O atributo mensurado deve ser claramente observado.
- Determinismo: deve haver um processo claro que pode ser seguido para se extrair a característica.
- Repetibilidade: várias pessoas, independentemente, extraindo a mesma característica do objeto devem concordar com o valor observado.
- Mutuamente exclusiva: a categorização em um grupo exclui a categorização em qualquer outro grupo.
- Exaustiva: juntos, os grupos incluem todas as possibilidades.
- Aceitável: lógica e intuitiva, de forma que as categorias possam ser aceitas pela comunidade.
- Útil: pode ser utilizada para a obtenção de conhecimento no campo de pesquisa.

Caso qualquer uma destas propriedades não seja mantida, a categorização não poderá ser repetida, podendo haver contradições ou equívocos. Um exemplo clássico de taxonomia que atende às propriedades taxonômicas desejadas é a dos seres vivos, onde cada qual é categorizado conforme o Reino, Filo, Classe, Ordem, Família, Gênero e Espécie.

3. Terminologia de Segurança de Software

A área da computação continua em expansão. Novos termos estão sendo criados e novos significados estão sendo adotados para esses termos. Para entender, classificar corretamente e, principalmente, criar uma taxonomia de vulnerabilidades, é importante compreender alguns termos utilizados na área de segurança de software. Mas esse processo não é tão trivial, pois vários autores utilizam suas próprias definições com diferentes

conotações. Um outro problema enfrentado é relacionado a tradução dos termos do idioma inglês para o português.

Artigos, páginas na Internet e livros descrevem, traduzem e utilizam de forma própria esses termos, e com isso a dificuldade de se criar uma taxonomia aumenta. Em algumas taxonomias duas ou mais pessoas conseguem classificar uma mesma vulnerabilidade de formas diferentes, porque elas utilizam conceitos diferentes.

Para mostrar a dificuldade, quatro termos em inglês serão retirados de um livro [5], de um artigo [6] e de um dicionário de termos [7] e posteriormente serão discutidos e traduzidos. São eles: *bug*, *error*, *fault* e *flaw*.

Em Hoglund [5], *bug* é um problema de software que existe no código, mas que não foi introduzido na fase de projeto, e pode nunca vir a ser executado. Já *flaw* é também um problema de software, porém em um nível mais aprofundado que *bug*, podendo aparecer na fase de projeto. Portanto, são termos que possuem diferentes empregos.

Para Landwehr [6], *bug* tem a mesma definição de Hoglund, mas para ele *flaw* é um sinônimo de *bug*, exceto por incluir problemas de software inseridos intencionalmente. De modo geral, para Landwehr *flaw*, *bug*, *error* e *fault* possuem o mesmo significado.

Por sua vez, o documento *IEEE Standard Glossary of Software Engineering Terminology* [7] define *bug* como sinônimo de *error* e de *fault*, porém *error* é mais utilizado para denominar a diferença entre um valor computado e o real, e *fault* é utilizado, por exemplo, para designar uma instrução incorreta em um programa de computador.

Diferentes empregos destes termos podem ser encontrados em diversas outras fontes, assim como outros termos mais específicos, como por exemplo, *buffer overflow*. Há lugares onde esse termo é utilizado para designar um ataque, outros utilizam como ameaça, vulnerabilidade, problema de codificação, etc. Uma breve investigação nos mecanismos de busca na Web pode retornar variados empregos deste termo.

O primeiro passo para se criar uma taxonomia é padronizar os termos de forma que sejam universalmente aceitos. Há várias iniciativas para isso como as RFCs [8], os dicionários da IEEE e outras fontes para termos de segurança, engenharia de software e computação em geral. Porém, não existe um padrão de terminologia universalmente aceito para segurança de software.

Uma tentativa de padronização seria utilizar a terminologia adotada pela IEEE [7] e acrescentar alguns outros termos que não aparecem neste glossário. Esse projeto da IEEE tem a finalidade de identificar termos atualmente em uso na área computacional e estabelecer definições padrões para eles. Alguns termos padronizados pela IEEE e suas respectivas traduções para a língua portuguesa são:

bug É sinônimo de *error* e *fault* e não possui tradução.

error É a diferença entre um valor ou uma condição computada e o valor ou condição teoricamente correta. A tradução para esse termo é erro.

fault Um passo, um processo ou uma definição incorreta. O termo utilizado em português é falha.

mistake Ação humana que produz um resultado incorreto. O termo utilizado em português é “erro humano”.

failure Um resultado incorreto. Não existe uma palavra única na língua portuguesa para traduzir esse termo e em alguns casos pode-se usar "resultado de uma falha".

Um outro termo bastante utilizado e que não se encontrada neste glossário é *flaw*. Alguns autores o traduzem por falha, mas falha já está sendo utilizada para traduzir *fault*, e essa tradução já é aceita e vem da disciplina de sistemas de tolerância a falhas. De acordo com o dicionário da língua portuguesa Houaiss, um sinônimo para falha é defeito e essa tradução é a que será utilizada para o termo *flaw*. Desta forma, adotando a descrição mencionada por Hoglund, "um defeito é um problema de software em um nível mais aprofundado que *bug* e que pode aparecer no projeto . . . ", e completando com Landwehr, ". . . porém não pode ser introduzido intencionalmente em um sistema".

Neste artigo serão utilizadas as definições dos termos padronizados pela IEEE e suas respectivas traduções discutidas acima. Além dos termos mencionados, outros também serão utilizados:

vulnerabilidade Um conjunto de condições que podem levar à violação de uma política de segurança explícita ou implícita [2].

worm Um programa que se auto-propaga através de uma rede, explorando vulnerabilidades de segurança ou de políticas em serviços amplamente utilizados [9].

ataque Uma série de passos intencionais seguidos por um atacante para violar uma política de segurança [4].

exploit Um software ou uma técnica que tira vantagem de uma vulnerabilidade para violar uma política de segurança [2].

4. Taxonomias de Vulnerabilidades

Nesta seção serão apresentadas taxonomias propostas para a área de segurança como as de Landwehr [6], de Aslam [10] e de Krsul [3], que têm como característica uma proposta ampla, de incluir todas as possíveis categorias de vulnerabilidades e ataques. Mas, primeiramente serão discutidos dois estudos [11, 12] que serviram de base para a definição destas taxonomias. Posteriormente serão apresentadas algumas taxonomias mais recentes, que cobrem um subconjunto mais específico de vulnerabilidades e ataques.

Em 1976, foi divulgado um estudo denominado *Research Into Secure Operating Systems* (RISOS) [11], cujo objetivo era o de auxiliar administradores de sistemas a compreender aspectos de segurança dos sistemas operacionais e aumentar sua segurança. Vários sistemas operacionais foram examinados para a elaboração de tal estudo, como MULTICS, GECOS, IBM OS, entre outros sistemas utilizados na época.

As falhas foram agrupadas em sete classes:

1. Validação incompleta de parâmetros;
2. Validação inconsistente de parâmetros;
3. Compartilhamento implícito de privilégios ou dados confidenciais;
4. Validação assíncrona ou serialização inadequada;
5. Autorização ou autenticação ou identificação inadequadas;
6. Violação de proibição ou limite;
7. Erro explorável de lógica.

Através deste estudo foi possível derivar muitas informações valiosas a respeito da natureza das falhas. Além disso, RISOS serviu de base para os outros estudos na área de vulnerabilidades em sistemas computacionais, como discutido abaixo.

Um outro estudo importante, datado de 1978, foi o *Protection Analysis* (PA) [12], cuja proposta é a de segmentar o problema da proteção de sistemas operacionais em pedaços menores e melhor gerenciáveis. O esquema de classificação de vulnerabilidades derivado da PA separa as falhas em classes, como mostrado a seguir:

1. Reforço e inicialização de domínio da proteção;
    - domínio;
    - representações expostas;
    - consistência dos dados através do tempo;
    - nomeação;
    - resíduos;
2. Validação de operandos / dependências no gerenciamento de filas;
3. Sincronização imprópria;
    - operações atômicas interrompidas;
    - serialização;
4. Erros de seleção de operadores críticos.

Tal classificação proposta foi uma que influenciou o estudo de vulnerabilidades e a pesquisa na área de segurança de computadores, pois embora seus métodos não tenham sido amplamente utilizados, a idéia por trás deles é eficiente [13].

Após um longo período de latência, a área voltou a ter visibilidade com o desenvolvimento da taxonomia de Landwehr [6], em 1992, baseada no estudo RISOS.

A taxonomia de Landwehr foi criada com o intuito de auxiliar os projetistas e operadores de sistemas a reforçarem a segurança, levando em conta três aspectos:

1. Como o defeito entrou no sistema (gênese);
2. Quando o defeito entrou no sistema (tempo de introdução);
3. Onde o defeito se manifesta (localização).

Esta taxonomia foi o primeiro estudo a abordar o problema das vulnerabilidades de um ponto de vista mais geral, incluindo o ambiente em que o defeito foi inserido. A taxonomia de Landwehr permite a concentração de esforços em áreas e fases de desenvolvimento mais propensas ao aparecimento das falhas.

Uma de suas características mais marcantes, que também é um de seus maiores problemas, é o fato de depender da visão do classificador. Isto torna a identificação ambígua, uma vez que ocorre com base na visão que o taxonomista tem do sistema em avaliação e se ele possui conhecimentos sobre o histórico do defeito.

Diferindo dos estudos RISOS e PA, onde as divisões nas categorias continham sobreposições, foi desenvolvido o modelo de Aslam [10] em 1996. Apresentando um procedimento de decisão que pretendia eliminar a ambigüidade na classificação de falhas (*faults*), Aslam distinguiu tais falhas da seguinte maneira:

1. Falhas de codificação;
    - erros de sincronização;

- erros na validação de condição.

2. Falhas emergentes;
   - erros de configuração;
   - falhas do ambiente.

A taxonomia de Aslam mostrou-se muito útil na organização das vulnerabilidades em um banco de dados. Este tipo de organização será melhor discutida na próxima Seção.

Em 1998, Krsul criou uma taxonomia constituída por uma árvore de categorização, que deveria ser construída *a priori*, contendo propriedades preditivas e descritivas [3]. A taxonomia de Krsul possui quatro classes de alto nível, com múltiplas subdivisões, e dá-se como segue (subníveis omitidos):

1. Projeto;
2. Suposições do ambiente;
3. Defeitos de codificação;
4. Erros de configuração.

Tal taxonomia pode ser aplicada a qualquer sistema onde é possível especificar suposições a respeito do ambiente como limites em atributos de objetos, por exemplo: UNIX, Windows NT, MAC OS ou sistemas operacionais orientados a objetos, distribuídos e *microkernels*. Em linhas gerais, a taxonomia de Krsul é uma extensão da taxonomia de Aslam, com algumas melhorias no processo de classificação das vulnerabilidades.

Recentemente, alguns pesquisadores investiram em definir taxonomias mais especializadas, cobrindo apenas uma determinada área de segurança de software.

Com a intenção de auxiliar desenvolvedores de software a identificar as causas fundamentais de vulnerabilidades em softwares utilizados na Internet, Frank Piessens propôs em 2002 uma taxonomia que pudesse ser utilizada para estes fins [14]. Esta taxonomia é basicamente uma hierarquia de dois níveis que procura identificar em que fase do desenvolvimento de um software uma falha é introduzida, como pode ser visto na Tabela 1.

Em 2003 Paxson et al descreveram uma Taxonomia de *Worms* [9]. Neste trabalho eles procuram entender as classes de *worms*, os atacantes que podem utilizá-los e os seus potenciais conteúdos. A taxonomia é descrita com base nas seguintes categorias:

- Seleção e descobrimento de alvos: baseado em varreduras, lista pré-gerada, lista gerada externamente, lista interna, passivo;
- Propagação e distribuição: auto-propagação, canal secundário, embarcado;
- Ativação: interação humana, atividade gerada por humanos, ativação programada, auto-ativação;
- Conteúdo: inexistente/não funcional, controle remoto via Internet, *relay* de *spam*, *proxies*, negação de serviço, coleta de dados, acesso para venda, danificação de dados, controle remoto no mundo físico, negação de serviço no mundo físico, reconhecimento no mundo físico, destruição no mundo físico, mecanismo de atualização;
- Motivações e atacantes: curiosidade, orgulho e poder, vantagem comercial, extorsão e crime, protesto político, protestos em geral, terrorismo, guerra eletrônica.

Tabela 1. Taxonomia de Piessens

Fase	Descrição
Análise	Sem análise de risco / sem política de segurança
	Análise de risco parcial
	Riscos não antecipados
Projeto	Erros de projeto em protocolo de criptografia
	Confiar em abstrações não seguras
	Balanço entre segurança / conveniência
	Sem mecanismos de geração de eventos
	Projeto não prevê todos os riscos
Implementação	Checagem defensiva de entrada insuficiente
	Checagem e uso não atômicos
	Erros de validação de acesso
	Implementação incorreta de primitivas criptográficas
	Manipulação insegura de condições de exceção
	Erros na lógica de segurança
Operação	Reutilização em ambientes mais hostis
	Configuração complexa ou desnecessária
	Configuração padrão insegura
Manutenção	Interação entre recursos
	Fallback inseguro

Já em 2004 Pothamsetty e Akyol propõem uma taxonomia específica para vulnerabilidades introduzidas no projeto e implementação de protocolos de rede [15]. De modo a ajudar os times de engenharia a não cometerem os mesmos erros novamente eles estudaram, classificaram e documentaram aproximadamente 500 vulnerabilidades relacionadas com protocolos de rede. Esta taxonomia possui três grandes ramificações:

- taxonomia de vulnerabilidades;
- taxonomia de técnicas de teste;
- taxonomia de boas práticas.

Apesar da existência de todas as propostas de taxonomias citadas anteriormente, não se tem notícia de que alguma delas esteja em uso. O que se observa é que na prática a tendência tem sido a adoção de esquemas de classificação de vulnerabilidades, como será abordado na Seção 5.

5. Classificações de Vulnerabilidades

Atualmente, não existe um consenso sobre a forma correta de se classificar uma vulnerabilidade, porém existem diversos grupos de pesquisa que mantêm classificações diferentes. Entre estas classificações podem-se destacar as seguintes: CVE, ICAT, SecurityFocus e OSVDB.

5.1. CVE

O CVE (*Common Vulnerabilities and Exposures*) é uma lista de nomes padronizados para vulnerabilidades e fraquezas de sistemas e softwares, mantida pelo MITRE [16].

O CVE tem por objetivo padronizar o nome de todas as vulnerabilidades e fraquezas publicamente conhecidas. Não se trata de um banco de dados de vulnerabilidades, mas sim de um dicionário, cujo foco principal está em facilitar o compartilhamento de informações.

A lista do CVE é criada em três etapas. A primeira é a etapa de submissão, a única de responsabilidade do MITRE. As outras etapas são as que criam entradas candidatas (CAN) e entradas CVE de fato. As entradas candidatas são criadas ou por mantenedores de softwares, que as utilizam para reportar problemas detectados, ou pelo próprio grupo do CVE, quando um novo problema crítico está sendo amplamente divulgado. Das mais de dez mil entradas, 70% são entradas candidatas e outros 30% são entradas CVE de fato.

5.2. ICAT

O ICAT, da *Computer Security Division* do NIST (*National Institute of Standards and Technology*), é uma base de dados de vulnerabilidades baseada no CVE [17]. Distingue-se do CVE por permitir que os usuários realizem buscas por vulnerabilidades específicas.

As buscas no ICAT podem ser realizadas sobre todas as entradas ou sobre as entradas do último ano, últimos seis ou três meses. Também é possível realizar a pesquisa através de um índice alfabético do nome do software ou do fabricante, além de ser possível procurar por um grau específico de severidade.

Existem ainda alguns filtros gerais que permitem que as vulnerabilidades sejam classificadas. Estes filtros possibilitam, por exemplo, encontrar as vulnerabilidades remotamente exploráveis, que possuam como conseqüência acesso com privilégios de superusuário e que sejam resultantes de um *buffer overflow*. Os possíveis filtros estão ilustrados na Tabela 2.

O ICAT também mantém um documento com estatísticas das vulnerabilidades subdivididas em cada um dos possíveis filtros. Neste documento é possível identificar, por exemplo, que no ano de 2004 das 1443 vulnerabilidades listadas, 1147 são remotamente exploráveis.

5.3. SecurityFocus

A SecurityFocus é uma comunidade de segurança da informação e possui uma coleção de vulnerabilidades com mais de treze mil entradas [18]. Estas podem ser encontradas através do mecanismo global do próprio sítio da SecurityFocus. Porém a informação submetida não é validada ou verificada [19].

A pesquisa no sítio da SecurityFocus pode ser filtrada pelo fornecedor, pelo programa que este fornecedor distribui e pela versão deste programa. Estes filtros são específicos para o banco de vulnerabilidades.

5.4. OSVDB

O OSVDB (*Open Source Vulnerability Data Base*) é uma base de dados independente e de código aberto [20]. Segundo os autores esta base é feita pela comunidade, para comunidade e tem como objetivo primordial prover informações técnicas precisas, detalhadas, atualizadas e imparciais.

Tabela 2. Alguns possíveis filtros do ICAT.

Fontes mais comuns	SecurityFocus
	Microsoft
	CERT
Exploit relacionado	Local
	Remoto
Conseqüência da vulnerabilidade	Disponibilidade
	Confidencialidade
	Integridade
	Proteção de Segurança
Tipo da vulnerabilidade	erro de validação de entrada
	erro de validação de acesso
	erro de condição inesperada
	erro de ambiente
	erro de configuração
	condição de corrida
	erro de projeto
	outros erros
Tipo de componente exposto	sistema operacional
	pilha do protocolo de rede
	aplicação de usuário
	aplicação de servidor
	hardware
	protocolo de comunicação
	módulo de criptografia
Tipo da entrada	Entradas CVE
	Entradas candidatas

Assim como o CVE, todas as vulnerabilidades catalogadas possuem um identificador, chamado no projeto de `OSVDB id`. Estas vulnerabilidades são listadas em categorias, sendo que as mais utilizadas são `NEW` e `STABLE`.

Detalhes das vulnerabilidades marcadas como `NEW` não podem ser vistos até que um membro do OSVDB modifique seu *status* para `STABLE`. Neste caso, a única informação aparente é o título da vulnerabilidade.

O banco de dados pode ser pesquisado por um índice alfabético organizado pelo título da vulnerabilidade que, geralmente, é iniciado pelo nome da aplicação afetada. As vulnerabilidades podem ser classificadas conforme as características listadas na Tabela 3.

Os esquemas de classificação aqui citados estão sendo utilizados amplamente pela comunidade, mas possuem algumas limitações, como a falta de padronização para os termos e informações armazenadas e a dificuldade de troca de informações sobre vulnerabilidades entre grupos de pesquisa e análise de vulnerabilidades. Com o intuito de permitir uma maior padronização, facilidade de busca e informações mais completas sobre as vulnerabilidades, surgiram no último ano novas iniciativas na área, descritas na Seção 6.

Tabela 3. Características da vulnerabilidade.

Localização	Físico
	Local
	Remoto
	Telefonia
	Desconhecida
Tipo de Ataque	Autenticação
	Criptográfico
	Negação de Serviço
	Seqüestro
	Informação
	Infra-estrutura
	Manipulação da Entrada
	Condição de corrida
	Outro
	Desconhecido
Impacto	Perda de Confidencialidade
	Perda de Integridade
	Perda de Disponibilidade
	Desconhecido
	Divulgação
Exploit	Disponível
	Não Disponível
	Rumor
	Desconhecido
	Web Check
OSVDB	Verificado
	Mito/Falso
	Boas Práticas
	Preocupação

6. Propostas Recentes

No último ano diversas iniciativas despontaram na comunidade de segurança com a intenção de melhor descrever e classificar vulnerabilidades e de utilizar estes dados como auxiliares no processo de definição de estratégias de recuperação.

Em 2004 o *European Task Force on Computing Security Incident Response Teams* criou um subgrupo para trabalhar na definição do VEDEF (*Vulnerability & Exploit Definition and Exchange Format*), um padrão para definição de um modelo de dados para troca de informações a respeito de vulnerabilidades entre instituições interessadas em classificá-las e estudá-las [21].

Uma das motivações para criação do VEDEF foi a constatação de que estavam surgindo diversas iniciativas distintas no sentido de definir padrões e que seria necessário unir esforços para que realmente fosse definido um único padrão para troca de informações. As diversas iniciativas são:

- *Common Format for Vulnerability Advisories*, mantido pelo *European Information Security Promotion Programme* (EISPP);

- *Common Announcement Interchange Format* (CAIF), mantido pelo Grupo de Resposta a Incidentes da Universidade de Stuttgart (RUS-CERT);
- *Advisory and Notification Markup Language* (ANML), mantido pelo *The Open Security Project* (OpenSec);
- *Application Vulnerability Description Language* (AVDL), mantido pelo *Organization for the Advancement of Structured Information Standards* (OASIS);
- *VulDEF element of Vendor Status Notes* (JVN), mantido pelo *Japan Computer Emergency Response Team Coordination Center* (JPCERT/CC).

Apesar do VEDEF não propor um esquema de classificação, ele propõem um modelo de dados que define diversas categorias de informações necessárias para descrever satisfatoriamente uma vulnerabilidade. Além disso, o VEDEF pretende envolver as iniciativas citadas acima, de modo a realmente produzir um padrão que possa ser utilizado amplamente pela comunidade.

Outra iniciativa recente é a do *Software Engineering Institute*, da *Carnegie Mellon University*, que publicou um relatório técnico em Janeiro de 2005, onde é descrita uma abordagem estruturada para classificação de vulnerabilidades de segurança [2]. Neste relatório é proposto um esquema de classificação de vulnerabilidades e um modelo de representação para permitir a comparação entre diversas vulnerabilidades.

O esquema de classificação é baseado em pares de valores e atributos, de modo a prover uma visão multidimensional das vulnerabilidades. Os atributos e seus valores são escolhidos de acordo com as características que permitam que as vulnerabilidades sejam exploradas por determinadas técnicas ou que determinem quais contramedidas devem ser adotadas. Os atributos definidos são os seguintes:

- Propriedades da memória: local de memória sobrescrito, tipo de dados modificado;
- Interface de funções: cópia de memória sem checagem, tamanho incorreto, argumento provido pelo usuário;
- Operações com números inteiros: aplicação, erro, conjuntos, sinais, tipo padrão, tipo extendido;
- Formato da cadeia de caracteres: função de entrada/saída, controle de formato;
- Propriedades da vulnerabilidade: impacto, produtos afetados, solução, porção conhecida, requerimentos para exploração;
- Propriedades do *exploit*: código, localização do código, conseqüências.

O modelo de representação para análise comparativa das vulnerabilidades consiste em criar um *bitmap* que represente todos os possíveis pares de atributos e valores. Uma vez tendo uma base de *bitmaps* seria possível, ao se saber de uma nova vulnerabilidade, fazer um XOR das informações conhecidas com todas as informações já contidas na base de *bitmaps*. O resultado deste XOR representa a proximidade entre duas vulnerabilidades.

A mais recente das iniciativas é um esquema de pontuação de vulnerabilidades. Em 23 de fevereiro de 2005 foi tornado público o relatório do *National Infrastructure Advisory Council* (NIAC) entitulado *Common Vulnerability Scoring System* (CVSS), que descreve um sistema de pontuação para vulnerabilidades a ser utilizado para definir a criticidade de uma determinada vulnerabilidade [22].

Este sistema especifica três níveis de métricas: base, temporal e ambiental. A métrica base é definida pelo produtor do software e é obtida através das características da vulnerabilidade que independam do ambiente onde o software esteja sendo utilizado e que não sofram alterações com o passar do tempo. A métrica temporal também é definida pelo produtor de software e se baseia em características que possam mudar com o passar do tempo, como por exemplo o fato de existir ou não um método de mitigação do problema. Já a métrica ambiental é calculada pelo usuário do software e leva em conta aspectos do ambiente onde software está sendo utilizado para definir a criticidade da vulnerabilidade.

Este sistema foi definido por um esforço conjunto de diversos fabricantes de software e já está começando a ser utilizado por alguns deles para definir a criticidade e urgência no tratamento de vulnerabilidades. O trabalho do CVSS será mantido por um grupo especial do *Forum of Incident Response and Security Teams* (FIRST) [23].

7. Conclusões

Após estudar as diversas taxonomias propostas foi possível observar que elas não aderem aos fundamentos do desenvolvimento de taxonomias, não satisfazendo às características preditivas e descritivas desejáveis. Muitas delas fazem mau uso do termo, pois o utilizam para descrever elementos agrupados em categorias que não apresentam as características de uma taxonomia.

Também pôde-se observar que devido à complexidade da área de vulnerabilidade de software, ainda não existe um conhecimento profundo sobre todos as possíveis categorias de vulnerabilidades que possam existir. Isto torna difícil o processo de propor uma taxonomia que seja exaustiva e aceitável.

Em parte por este motivo, nota-se que a tendência atual é de se utilizar simplesmente esquemas de classificação ou de padronização do modelo de dados para definir as características de uma vulnerabilidade. Do ponto de vista prático, os esquemas existentes têm demonstrado que podem ser úteis para reunir informações sobre vulnerabilidades e permitir a troca de informações entre grupos de pesquisa.

A classificação de vulnerabilidades também é importante para permitir que se determine se uma dada vulnerabilidade é nova ou já conhecida, seu método de exploração e como foi introduzida. Estas informações permitem que desenvolvedores possam evitar a inserção dos mesmos tipos de vulnerabilidades em seus softwares.

Mesmo com todas as iniciativas atuais, ainda não existe uma base extensa de informações confiáveis sobre todos tipos de vulnerabilidades existentes. Desta forma, conclui-se que ainda existe a necessidade de um esforço continuado para se obter todas as informações necessárias para chegar à definição de uma taxonomia que permita uma maior padronização, facilidade de busca e informações mais completas sobre as vulnerabilidades.

Referências

[1] CERT Coordination Center, "CERT/CC Statistics 1988-2005: Vulnerabilities reported." `http://www.cert.org/stats/cert_stats.html`.

[2] R. C. Seacord and A. D. Householder, "A structured approach to classifying security vulnerabilities," Tech. Rep. CMU/SEI-2005-TN-003, CMU/SEI, January 2005.

[3] I. V. Krsul, *Software Vulnerability Analysis*. PhD thesis, Purdue University, May 1998.

[4] J. D. Howard and T. A. Longstaff, "A Common Language for Computer Security Incidents," tech. rep., Sandia National Laboratories, October 1998.

[5] G. Hoglund and G. McGraw, *Exploiting Software: How to Break Code*. Addison-Wesley Professional, 1st ed., February 2004. ISBN 0-201-78695-8.

[6] C. Landwehr, A. Bull, J. McDermott, and W. Choi, "A Taxonomy of Computer Program Security Flaws," *ACM Computing Surveys*, vol. 26, no. 3, pp. 211–254, 1994.

[7] ANSI/IEEE, *IEEE Standard Glossary of Software Engineering Terminology*. New York: IEEE, 1983.

[8] R. Shirey, "RFC 2828: Internet Security Glossary." `http://www.ietf.org/rfc/rfc2828.txt`, May 2000.

[9] N. Weaver, V. Paxson, S. Staniford, and R. Cunningham, "A Taxonomy of Computer Worms," in *Proceedings of The First ACM Workshop on Rapid Malcode (WORM)*, October 2003.

[10] T. Aslam, I. Krsul, and E. H. Spafford, "Use of a Taxonomy of Security Faults," in *Proceedings of the 19th National Information Systems Security Conference*, pp. 551–560, October 1996.

[11] R. Abbott, J. Chin, J. Donnelley, W. Konigsford, S. Tokubo, and D. Webb, "Security Analysis and Enhancements of Computer Operating Systems," Tech. Rep. NBSIR 76–1041, National Bureau of Standards, April 1976.

[12] R. B. II and D. Hollingworth, "Protection Analysis: Final Report," Tech. Rep. ISI/SR-78–13, University of Southern California Information Sciences Institute, May 1978.

[13] M. Bishop, *Computer Security: Art and Science*. Addison Wesley Professional, 1st ed., December 2002. ISBN 0-201-44099-7.

[14] F. Piessens, "A taxonomy of causes of software vulnerabilities in internet software," in *Supplementary Proceedings of the 13th International Symposium on Software Reliability Engineering*, pp. 47–52, 2002.

[15] V. Pothamsetty and B. Akyol, "A Vulnerability Taxonomy for Network Protocols: Corresponding Engineering Best Practice Countermeasures," in *Proceeding of Communications, Internet, and Information Technology 2004* (M. H. Hamza, ed.), IASTED, ACTA Press, November 2004. ISBN 0-88986-445-4.

[16] MITRE, "Common Vulnerabilities and Exposures (CVE)." `http://www.cve.mitre.org/`.

[17] National Institute of Standards and Technology (NIST), "ICAT." `http://icat.nist.gov/`.

[18] Symantec, "SecurityFocus." `http://securityfocus.com/`.

[19] R. Gopalakrishna and E. H. Spafford, "A trend analysis of vulnerabilities," tech. rep., CERIAS, Purdue University, 2005. CERIAS TR 2005-05.

[20] "Open Source Vulnerability Data Base (OSVDB)." http://www.osvdb.org/.

[21] European Task Force on Computing Security Incident Response Teams, "Vulnerability & Exploit Definition and Exchange Format (VEDEF)." http://www.vedef.org/.

[22] J. T. Chambers and J. W. Thompson, "Common Vulnerability Scoring System: Final Report and Recommendations by the Council," tech. rep., National Infrastructure Advisory Council, October 2004.

[23] Forum of Incident Response and Security Teams, "FIRST to host CVSS." http://www.first.org/cvss/.

[24] A. Houaiss, *Dicionário Houaiss da Língua Portuguesa*. Editora Objetiva, 1ª ed., 2004. ISBN 8-573-02383-X.