PUC

ISSN 0103-9741

Monografias em Ciência da Computação
n° 21/11

Relações Entre Riscos, Erros, Vulnerabilidades e Boas Práticas de Software, Uma Análise com Base em Diagramas De Influência: O Caso da SANS/MITRE.

André Luiz de Castro Leal

Departamento de Informática

PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO DE JANEIRO
RUA MARQUÊS DE SÃO VICENTE, 225 - CEP 22453-900
RIO DE JANEIRO - BRASIL

Monografias em Ciência da Computação, No. 21/11 ISSN: 0103-9741
Editor: Prof. Carlos José Pereira de Lucena Dezembro 2011

Relações Entre Riscos, Erros, Vulnerabilidades e Boas Práticas de Software, Uma Análise com Base em Diagramas de Influência: o Caso da SANS/MITRE.

André Luiz de Castro Leal

aleal@inf.puc-rio.br

Abstract The present work makes an analysis of informed errors by SANS/MITRE establishing a relationship among these errors, software vulnerability and best practices through a casual loop diagram.

Keywords: Error, Fault, Vulnerability, Best Practices, Software Engineering, Risk, Casual Loog Diagram.

Resumo: O presente trabalho apresenta uma análise de erros relatados pela SANS/MITRE e estabelece relações desses erros com vulnerabilidades e boas práticas de software por meio de diagramas de influência.

Palavras-chave: Erros, Vulnerabilidades, Boas Práticas, Engenharia de Software, Riscos, Diagramas de Influência.

Responsável por publicações:

Rosane Teles Lins Castilho
Assessoria de Biblioteca, Documentação e Informação
PUC-Rio Departamento de Informática
Rua Marquês de São Vicente, 225 - Gávea
22453-900 Rio de Janeiro RJ Brasil
Tel. +55 21 3114-1516 Fax: +55 21 3114-1530
E-mail: bib-di@inf.puc-rio.br

Sumário

1. INTRODUÇÃO 1

1.1. ORGANIZAÇÃO DO TRABALHO 2

2. CONTEXTUALIZAÇÃO 2

2.1. ASPECTOS METODOLÓGICOS 2
2.2. LISTA DE ERROS MAIS PERIGOSOS 3
2.3. FATORES DE RISCO 5
2.4. VISÃO GERAL DE DIAGRAMAS DE INFLUÊNCIA 7

3. VULNERABILIDADES DOS 25 ERROS SANS/MITRE 9

4. ANÁLISE COM SUPORTE DE DIAGRAMAS DE INFLUÊNCIA 13

4.1. BOAS PRÁTICAS EM SOFTWARE 13
4.2. DIAGRAMAS DE INFLUÊNCIA 14

5. CONSIDERAÇÕES FINAIS 18

REFERÊNCIAS BIBLIOGRÁFICAS 19

Lista de Figuras

Figura 1: Resumo gráfico do método aplicado 3
Figura 2: Princípios de um processo moderno de software 13
Figura 3: Diagrama de influência de elementos para o erro CWE-79 (*Cross-site scripting*) 15
Figura 4: Diagrama de influência de elementos para o erro CWE-89 (*SQL Injection*) 16
Figura 5: Diagrama de influência de elementos para o erros distintos 17
Figura 6: Diagrama de influência de elementos para o erros distintos 18

Lista de Tabelas

Tabela 1: Lista de erros da categoria “Interação Insegura entre Componentes” 4
Tabela 2: Lista de erros da categoria “Risco de Gerenciamento de Recurso”..... 4
Tabela 3: Lista de erros da categoria “Fragilidade na Defesa”..... 5
Tabela 4: Vulnerabilidades - Categoria “Interação Insegura entre Componentes” 10
Tabela 5: Vulnerabilidades - Categoria “Risco de Gerenciamento de Recurso”... 11
Tabela 6: Vulnerabilidades - Categoria “Fragilidade na Defesa” 12
Tabela 7: Discriminação de boas práticas de software 14

1. Introdução

A automação industrial moderna faz com que diversos tipos de software estejam instalados em equipamentos cada vez mais sofisticados utilizados pelo homem no seu cotidiano. Tais equipamentos são utilizados para operações simples como no caso de celulares, mas também equipamentos críticos que são parte, por exemplo, de controles de aeronaves que transportam vidas e equipamentos médicos, que auxiliam em diagnósticos de doenças. O uso acentuado desses equipamentos criam um novo contexto de operações que resulta no aumento da carga cognitiva humana e ambientes de software mais propícios a erros [LIMA e TURNELL, 2006].

Dentro dessa mesma realidade de sofisticação, há uma vertente importante no contexto de software que também passa a ser exposto a maiores probabilidades de erros, os projetistas e desenvolvedores.

Um estudo recente, abril de 2010, da *SysAdmin, Audit, Network, Security Instituite* (SANS) [2010] em parceria com a Mitre Corporation (Mitre) [2010] e publicado na *Commom Weadness Enumaretion – A Community-Developed Dictionary of Software Weakness Types (*CWE) [2010] identifica uma lista dos mais generalizados e críticos erros de programação que podem conduzir a problemas em software [CWE/SANS, 2010]. Os erros são considerados perigosos porque freqüentemente permitem que ataques externos possibilitem que agentes[1] assumam de forma ilegal a execução do software, permitem o roubo de dados ou impedem o funcionamento total do software.

A SANS é uma organização de pesquisa cooperativa e educação estabelecida em 1989, que atua em assuntos ligados à segurança da informação, onde desenvolve, mantém e disponibiliza, sem custo, sua coleção de documentos. Estão entre esses documentos o popular *Internet Storm Center* (sistema de alerta da Internet), um site com notícias semanais sobre um resumo executivo de segurança em computadores (*NewsBites*[2]), e cerca de 1200 trabalhos de pesquisa [SANS, 2010]. Enquanto o Mitre é uma organização sem fins lucrativos estabelecida nos EUA em 1958 e que tem por objetivo trabalhar para o interesse público a partir de recursos nacionais em assuntos ligados a engenharia de sistemas, tecnologia da informação, conceitos operacionais e modernização de empresas. Estão envolvidos no Mitre, além de centros de investigação e desenvolvimento na área de tecnologia, o Departamento de Defesa Americano e o Centro Avançado de Desenvolvimento de Sistemas de Aviação, além de outras importantes organizações do governo americano.

A publicação da lista dos 25 erros mais perigosos foi efetuada no CWE [CWE, 2010] pelo fato de ser é uma iniciativa estratégica de garantia de *software (software assurance SWA*[3]*)* patrocinada pela *National Cyber Security Division*[4] do *U.S. Department of Homeland Security*[5] que objetiva estabelecer um nível de segurança para indicar que o

[1] Agente está sendo considerado nesse trabalho como sendo um usuário, um programa, função ou qualquer ator externo ao software.

[2] http://www.sans.org/newsletters/newsbites.

[3] https://buildsecurityin.us-cert.gov/swa

[4] Estabelecido para apoiar o Poder Executivo Cível Federal Americano em questões de ataques cibernéticos. Além de compartilhar as informações observadas com os governos estadual e locais, como também indústria e parceiros internacionais (http://www.us-cert.gov/).

[5] Departamento interno de segurança americano que atua em missões de segurança da nação contra diversos tipos de ameaças, entre elas as cibernéticas.

software está livre de vulnerabilidades, ou erros inseridos intencionalmente durante em fase de projeto ou acidentalmente inserido durante seu ciclo de vida.

O presente trabalho pretende abordar sobre os erros listados pela SANS/Mitre, [CWE/SANS, 2010] identificar e analisar as principais vulnerabilidades que podem acarretar nos erros. Além disso, o trabalho pretende classificar as vulnerabilidades a partir de critérios de risco e relacioná-las às boas práticas de software através de digramas de influência [SENGE, 1990] com o objetivo propor uma visão sistêmica para dar suporte a decisão das escolhas das práticas.

1.1. Organização do trabalho

O documento está subdividido em seções onde a seção 2 apresenta a proposta metodológica e contextualiza a lista de erros mais importantes, sobre fatores de risco e faz uma apresentação geral sobre diagramas de influência; na seção 3 estão relatados e categorizados alguns dos erros relatados pela SANS/MITRE. A seção 4 esta dividida com uma conceituação de boas práticas de software e apresenta a análise dos erros e vulnerabilidades a partir dos diagramas de influência. A seção 5 encerra o trabalho com as considerações finais.

2. Contextualização

2.1. Aspectos Metodológicos

A realização do trabalho foi composta de atividades que podem ser discriminadas como:

- **Atividade 1 (1A)**: trabalho de revisão bibliográfica e registro sobre erros e fatores humanos no uso de software.

- **Atividade 2 (2A)**: revisão bibliográfica e registro sobre gerenciamento de riscos e os fatores de potencialização de riscos: vulnerabilidade, ameaças e impacto.

- **Atividade 3 (3A)**: estudo e registro sobre a lista dos 25 erros considerados pela SANS/Mitre como mais perigosos em software.

- **Atividade 4 (4A)**: levantamento das principais vulnerabilidades (causas) que levaram à classificação dos 25 erros mais perigosos.

- **Atividade 5 (5A)**: Classificação de vulnerabilidades segundo critérios de risco.

- **Atividade 6 (6A)**: revisão bibliográfica sobre boas práticas em software e diagramas de influência.

- **Atividade 7 (7A)**: estabelecer relações de precedência e potencialização entre as vulnerabilidades, os erros e as boas práticas em software e apresentá-la em diagramas de influência com o objetivo de se permitir uma visão sistêmica os elementos envolvidos.

A **Figura 1** apresenta o diagrama das atividades envolvidas neste trabalho de pesquisa.

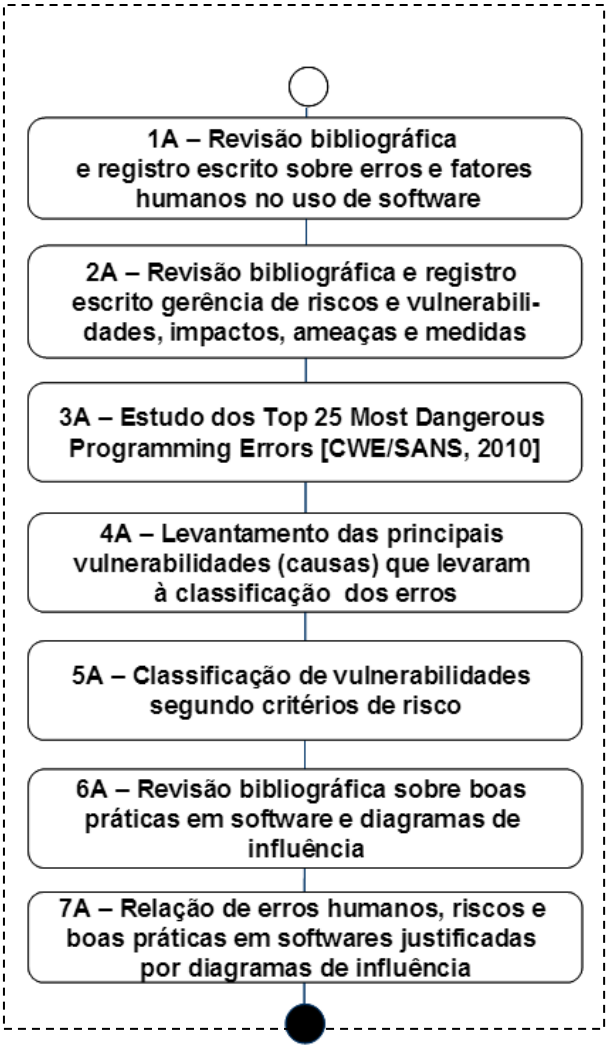

Figura 1: Resumo gráfico do método aplicado.
Fonte: do autor.

2.2. Lista de erros mais perigosos

Os 25 erros mais perigosos [CWE/SANS, 2010] em software estão dispostos em uma lista com o objetivo de apresentar os principais erros de software que permitem ou causam invasões externas, roubo de informações, erro de funcionamento, entre outros. A intenção da criação da lista foi de auxiliar a clientes a entender como softwares podem se tornar mais seguros, pesquisadores podem focar seus esforços em um grupo específico de problemas comprovadamente mais perigosos, e por fim a lista pretende auxiliar gerentes de software como um instrumento de classificação e permitir que verifiquem se estão livres desses principais erros [CWE/SANS, 2010].

Os erros foram categorizados e agrupados em três seções: a) interação insegura entre componentes, b) risco de gerenciamento de recurso e c) fragilidade na defesa[6].

[6] Do original *Porous Defenses*.

Na seção "Interação Insegura entre Componentes", os autores agruparam erros comuns que possam levar softwares a ter deficiências relacionadas as intercomunicações entre componentes, módulos, programas, processos, *threads* ou entre sistemas. Na seção "Risco de Gerenciamento de Recurso", estão relatados os erros que acarretam software a não gerenciar a criação, o uso, transferência ou o término de importantes recursos de sistemas. Por fim, na última seção, a "Fragilidade na Defesa", estão agrupados os erros mais comuns que ocorrem pelo desprezo, ou o abuso, ou falta de planejamento sobre questões de técnicas defensivas [CWE/SANS, 2010].

A **Tabela 1, 2** e **3** apresentam a lista de erros já agrupados de acordo com sua categoria.

Tabela 1: Lista de erros da categoria "Interação Insegura entre Componentes". Fonte: [CWE/SANS, 2010]

Ordem	CWE ID	Descrição
[1]	CWE-79	Falha ao manter estrutura de página web (*Cross-site scripting*).
[2]	CWE-89	Limpeza inadequada de elementos utilizados em comandos SQL[7] (*SQL Injection*).
[4]	CWE-352	Falsa requisição (*Cross-Site RF – CSRF*).
[8]	CWE-434	*Upload* irrestrito de arquivos de tipos perigosos.
[9]	CWE-78	Limpeza inadequada de elementos especiais usados nos comandos do sistema operacional (*OS Command Injection*).
[17]	CWE-209	Exposição de informações impróprias através de mensagens de erro.
[23]	CWE-601	Redirecionamento de URL[8] para locais não confiáveis (*Open Redirect*).
[25]	CWE-362	Falha no uso de recurso compartilhado (*Race condition*).

Tabela 2: Lista de erros da categoria "Risco de Gerenciamento de Recurso". Fonte: [CWE/SANS, 2010]

Ordem	CWE ID	Descrição
[3]	CWE-120	Cópia para *buffer* sem checar o tamanho permitido de entrada (*Classic Buffer Overflow*).
[7]	CWE-22	Limitação inadequada de caminho para restringir diretório (*Path Traversal*).
[12]	CWE-805	Acesso a *buffer* com valor do tamanho incorreto.
[13]	CWE-754	Verificação inadequada de condições sem uso ou excepcionais.
[14]	CWE-98	Controle inadequado de declarações externas de inserção ou requisição a partir de nomes de arquivos (*PHP File Inclusion*).
[15]	CWE-129	Validação inadequada de índices em *array list.*
[16]	CWE-190	Estouro de inteiro (*Integer overflow*)
[18]	CWE-131	Cálculo incorreto de tamanho de *buffer.*
[20]	CWE-494	*Download* de código sem verificação de integridade.
[22]	CWE-770	Alocação de recurso de forma ilimitada.

[7] Structure Query Language
[8] Uniform Resource Locator

Tabela 3: Lista de erros da categoria "Fragilidade na Defesa".
Fonte: [CWE/SANS, 2010]

Ordem	CWE ID	Descrição
[5]	CWE-285	Controle de acesso inadequado (*Authorization*).
[6]	CWE-807	*Imputs* confiáveis ou não nas decisões de segurança.
[10]	CWE-311	Falta de criptografia de dados sensíveis.
[11]	CWE-798	Uso de códigos fixos (*Hard coded.)*
[19]	CWE-306	Falta de autenticação em funções críticas.
[21]	CWE-732	Atribuição incorreta de permissão para rercusos críticos.
[24]	CWE-327	Uso de algoritmo de quebra de criptografias.

Em CWE/SANS [2010] pode ser lida uma discriminação mais detalhada dos erros. Os erros podem ser encontrados a partir do seu ID, como por exemplo o CWE-798 é o identificador do erro de uso de códigos fixos (*Hard coded)*, por isso foram deixados nas tabelas supracitadas os identificadores originais. A discriminação mais detalhada dos erros serão utilizadas nesse trabalho apenas para aqueles erros que forem utilizados nas relações entre vulnerabilidades e boas práticas de software e que não possuam uma descrição suficientemente clara.

2.3. Fatores de Risco

A preocupação com o risco está nos efeitos negativos, falhas tecnológicas, infração legal, comprometimento da imagem, reputação, roubo, fraude, dentre outros fatores que possam prejudicar determinada situação, seja empresarial ou pessoal. Risco pode ser interpretado como sendo uma incerteza, portanto está diretamente relacionado à "exposição à chance de perdas ou danos" [PRITCHARD, 1998]. O risco como incerteza está diretamente relacionado à eficiência operacional e o gerenciamento deste risco consiste em técnicas para diminuir a variação do resultado projetado e o real obtido. Neste caso é difícil avaliar o risco existente, e os gestores normalmente utilizam intervalos de valores para fazer as previsões, como por exemplo: 5 - Crítico, 4 - Acima do Normal, 3 - Normal, 2 - Abaixo do Normal e 1 - Leve [PRITCHARD, 1998].

De acordo com o *Project Management Body of Knowledge* (PMBOK) [2010], risco é um evento ou condição incerta que, se acontecer, tem um efeito positivo ou negativo para a segurança da informação da empresa. Segundo a definição obtida pelo Dicionário Aurélio risco é [FERREIRA, 1999]: *Risco: s. m., perigo, inconveniente, probabilidade de perigo; em risco de: em perigo de*. O PMBOK ainda estabelece a categorização da gerência de risco como uma área para identificar, analisar, planejar e controlar riscos essas atividades podem ser discriminadas como:

- Identificação dos riscos inerentes a uma etapa do desenvolvimento (fase, processo, iteração). Feito através do levantamento das ameaças presentes e do impacto que podem provocar caso se realizem.

- Análise dos riscos identificados, a partir da identificação do nível de exposição do projeto. É realizada uma classificação dos riscos, baseando-se no relacionamento entre a exposição e conseqüência negativa do risco e o benefício da oportunidade, decide-se quais serão eliminados, quais serão mitigados, quais serão aceitáveis e quais serão acompanhados.

- Planejamento determina como e quando os riscos serão abordados ao longo do projeto. São elaborados planos de mitigação, eliminação e acompanhamento de riscos que serão utilizados como base para a gerência de riscos.

- Controle à execução e ao acompanhamento dos planos elaborados para o projeto. Os riscos identificados são analisados constantemente para a identificação do seu estado atual e atualização dos planos elaborados. Novos riscos podem ser identificados também.

O risco pressupõe três componentes: 1) Um evento; 2) A probabilidade de ocorrência do evento; e 3) O impacto decorrente do evento. Ao avaliar esses componentes, os gestores atuam fortemente para prevenir e minimizar o impacto para a organização a partir da sua identificação, análise, planejamento e controle, conforme já citado anteriormente.

Para Gordon e Loeb [2002], risco é a probabilidade de que agentes, que são as ameaças, explorem vulnerabilidades, expondo os ativos a perdas da segurança causando impactos nos negócios. Sêmola [2003] propõe uma equação em que sugere que o risco é diretamente proporcional às ameaças, vulnerabilidades e o impacto ligados ao ativo. Quando esses fatores são reduzidos, o risco será diretamente amenizado.

$$R = \frac{V \times A \times I}{M}$$, fórmula segundo Sêmola [SÊMOLA, 2003].

Onde,

R = Risco
V = Vulnerabilidade
A = Ameaças
I = Impacto
M = Medidas de Segurança (firewall, backup, política de segurança, capacitação técnica, entre outros)

- Vulnerabilidade (V): são as circunstâncias que aumentam a possibilidade de uma ameaça ser concretizada, aumentando sua freqüência e seu impacto. Na análise do risco, vulnerabilidade é a falta de segurança para determinado ativo ou grupo de ativos [SÊMOLA, 2003].

- Ameaça (A): o conceito de ameaça é definido por possíveis danos aos ativos, intencionais ou não. Ameaças são subdivididas em: desastres naturais (enchentes, incêndio, terremoto), humanas (subdividida em intencional, onde ocorre diretamente por hackers e funcionários descontentes e não intencional, onde funcionários com pouco conhecimento sobre a tecnologia aplicada) e ambientais (engloba toda parte tecnológica, software, hardware, falhas de sistema operacional, falha elétrica. Resumidamente, ameaças são "os meios pelos quais a confidencialidade, integridade e disponibilidade da informação podem ser comprometidas." [SÊMOLA, 2003]

- Impacto (I): impacto é o dano causado por uma ameaça à vulnerabilidade, tanto no ambiente tecnológico quanto na imagem da empresa, analisando o

impacto na confiabilidade, disponibilidade e integridade do ativo [SÊMOLA, 2003].

A importância de ser abordada a questão de riscos nesse trabalho é a de evidenciar o papel da vulnerabilidade na fórmula proposta. Tal fator é potencializador de risco e ao ser mitigado, conseqüentemente há uma minimização dos riscos e em se tratando de software, dos erros inerentes.

Com foco em gerência de risco a identificação das vulnerabilidades e sua análise poderão levar a uma classificação, grau e descrição, de intervalos de valores como segue:

- 5: Crítica;
- 4: Acima do Normal;
- 3: Normal;
- 2: Abaixo do Normal;
- 1: Leve.

Esses valores, aplicados à fórmula de Sêmola [2003], poderão junto aos fatores de impacto e ameaças, classificar os erros em intervalos que irão identificar os graus de risco de determinada situação em software. Esses graus podem ser elencados como:

- de 25 a 21: Elevado;
- de 20 a 16: Alto;
- de 15 a 11: Atenção;
- de 10 a 6: Médio;
- de 5 a 1: Seguro.

Entende-se que será válido qualquer esforço para minimizar os fatores que potencializam riscos. Dessa forma, justifica-se a tratativa desse trabalho no que diz respeito a identificação de vulnerabilidades e a apresentação conjunta e relacionada dessas vulnerabilidades com os erros de software. A indicação de boas práticas de software irá preencher uma lacuna que pretende, juntamente com diagramas de influência, apresentar uma visão sistêmica e de relações que poderão servir como apoio a tomada de decisão no que diz respeito a adoção das técnicas.

2.4. Visão Geral de Diagramas de Influência

Antes iniciar as discussões técnicas acerca dos diagramas de influência é importante contextualizar sua origem. Eles tiveram seu fundamento no pensamento sistêmico onde a dinâmica de sistemas teve papel importante. A dinâmica de sistemas foi criada pelo Prof. Jay W. Forrester[9] na década de 50 na MIT *Sloan Management School*[10], que reconheceu a necessidade de uma forma melhor para entender os sistemas sociais. De forma uma interdisciplinar a dinâmica de sistemas estuda o comportamento dinâmico de sistemas complexos e tem origens na Teoria Geral dos Sistemas de Bertalanffy [1977], onde busca o entendimento dos sistemas enquanto um todo, de uma forma interdisciplinar, focando na complexidade e na interdependência, de uma maneira holística.

É importante constar que no mundo dos negócios quando se deixa de ver a organização como a soma de eventos isolados e suas causas para desenvolver a visão

[9] http://mitsloan.mit.edu/faculty/detail.php?in_spseqno=SP000041&co_list=F
[10] http://mitsloan.mit.edu/

de um sistema composto de partes que interagem, desenvolve-se também a percepção de que, a partir dessas interações, surgem padrões de comportamento [SENGE, 1990].

Para que a tomada de decisões possa ser eficaz é importante que existam ferramentas que dêem suporte a uma análise do todo e das relações de suas partes, assim há uma maior chance de se entender melhor os problemas e os padrões de comportamento. Peter Senge [1990] propõe a utilização de uma notação gráfica conhecida como diagrama sistêmico ou de influência *(casual loop diagram)* para representar as estruturas responsáveis por esses padrões de comportamento.

Um diagrama de influência consiste basicamente de uma representação de influências causais entre os elementos de um sistema chamadas de *feedback*, *loop*, de *feedback loop causal* ou ainda ciclo causal. Um *loop* de *feedback* é uma seqüência fechada de causas e efeitos, ou seja, um caminho fechado de ação e informação. Assim, um diagrama de influência representa os *loops* de *feedback* de um sistema [RICHARDSON e PUGH, 1981].

Fazem parte da notação de um diagrama de influência os seguintes componentes [SENGE, 1990]:

a. **Elementos**: simplesmente escritos de uma forma clara e direta, em linguagem natural. Declarado como substantivo. Exemplo: "*Aceite do Usuário*", "*Registro de Ocorrências*", "*Testes e Aceite do Analista*", "*Controle de Medições e Indicadores*", entre outros [LEAL, 2009];

b. **Setas**, une os elementos indicando suas relações causais. Para representar uma defasagem, que significa que existe um determinado tempo entre uma ação e o seu efeito, pode-se cruzar uma seta com duas linhas perpendiculares, desenhar a seta de forma tracejada ou, ainda, utilizar um ícone que represente o tempo (um relógio, por exemplo);

c. **Sinais (-)** e **(+)**, define o tipo da relação causal: - os sinais indicam uma relação no sentido de que uma mudança em um elemento A produz uma mudança em B, na mesma direção ou em proporção direta, portanto são diretamente equivalentes ao sinal representado. Se o sinal for positivo, indica uma relação positiva, de forma que uma alteração em A produz uma alteração em B na mesma proporção positiva, caso o sinal seja negativo há uma relação de negatividade, causando algum tipo de perda no elemento subseqüente;

d. **Sinal do Loop (-)** e **(+)**, Os *loops* positivos são chamados de *feedback*, em que os sinais de *loops* positivos são *feedbacks* de reforço e os negativos são equilíbrio. Opcionalmente ao sinal, pode-se utilizar um ícone "bola-de-neve", para representar o *loop* positivo e um ícone "balança", para representar o *loop* negativo.

Podem-se descrever, a partir das sugestões de Richardson e Pugh [1981] e Kim [1992], as etapas para a construção de um diagrama influência, deixando clara a ordem e as funções de todos os seus componentes:

1. Deve-se pensar nos elementos de um diagrama influência como se fossem variáveis, cujos valores podem crescer ou decrescer.

a. Usam-se substantivos, e não verbos, para descrever os elementos, uma vez que as ações (verbos) são representadas pelas setas no diagrama – "*Aprovação do Usuário*" ao invés de "*Aprovar pelo Usuário*";

b. A definição de uma variável deve deixar clara qual posição significa "mais" para essa variável – "*Aceite do Usuário*" ao invés de "*Testes do Usuário*" [LEAL, 2009];

c. Normalmente, o conceito fica mais claro se o nome do elemento denota um senso positivo – "crescimento", ao invés de "declínio";

d. Os links causais devem implicar uma relação de causalidade e não simplesmente uma seqüência temporal. Isto é: um link positivo do elemento A ao elemento B não significa "primeiro A ocorre e então B ocorre". Particularmente, significa "quando A aumenta, B aumenta";

2. À medida que se colocam os links no diagrama, deve-se avaliar os possíveis efeitos colaterais das inter relações. Uma vez identificados, decide se os links adicionados realmente representam um padrão de comportamento importante;

3. Sempre existe um objetivo para os *feedbacks* de equilíbrio. O diagrama fica mais claro se o objetivo é mostrado juntamente com a "diferença" que dirige o *loop* para o objetivo. Isso pode facilmente ser visto nos diagramas apresentados anteriormente;

4. Diferentes estados do processo podem ser representados para facilitar a análise do estado atual e o percebido de forma a encontrar diferentes padrões de comportamento ao longo da execução;

5. As conseqüências das ações de curto e longo prazos das ações devem ser representadas por loops diferentes;

6. Se um link entre dois elementos necessita de muitas explicações, provavelmente há a necessidade de maior detalhamento, ou seja, a inclusão de novos elementos ou outras relações;

7. Deve-se manter o diagrama o mais simples possível. O objetivo principal do diagrama de influência não é descrever todos e cada um dos detalhes do processo, mas aspectos de relações que possam identificar padrões de comportamento.

3. Vulnerabilidades dos 25 Erros SANS/Mitre

Como já contextualizado, a importância de se entender e mitigar as vulnerabilidades irá significar a redução de riscos, uma vez que essas fazem parte de um grupo de elementos pontecializadores.

Diante disso, é importante identificar quais as vulnerabilidades foram significativas nos 25 erros mais perigosos classificados pela SANS/Mitre [CWE/SANS, 2010]. Essa identificação irá ser fundamental para estabelecer as relações com os erros e conseqüentemente a indicação de boas práticas de software. Dessa forma, o trabalho irá cumprir um dos objetivos que é utilizar técnicas de gerência de riscos, identificação

e análise, para auxiliar na mitigação de vulnerabilidades de erros de forma a apresentar uma proposta de ferramenta para auxilio à análise de padrões de comportamento, que é o caso do diagrama de influência. Além disso, a classificação das vulnerabilidades a partir de graus de risco irão melhorar a percepção da importância de se mitigar tal vulnerabilidade.

Por se tratar de um trabalho de pesquisa de cunho limitado de tempo e espaço, serão eleitos, aleatoriamente, 2 (dois) erros de cada categoria da lista dos 25. Outros trabalhos poderão seguir a mesma sistemática metodológica para complementar o restante da lista.

O **Tabela 4** apresenta as vulnerabilidades dos erros da categoria "Interação Insegura entre Componentes".

Tabela 4: Vulnerabilidades - Categoria "Interação Insegura entre Componentes". Fonte: [CWE/SANS, 2010]

Ordem	**CWE ID**	**Descrição**		
[1]	CWE-79	Falha ao manter estrutura de página web (*Cross-site scripting*).		
Vulnerabilidade		**Origem**	**Apelido para Diagrama de Influência *****	**Grau ****
Scripts javascripts, *tags* como *onload*, *onerror* ou *style* não são neutralizados ou são incorretamente tratados pelo software.		Implementação	Inadequada neutralização *Scripts*	4
A aplicação não filtra scripts externos.		Implementação	Ausência filtro *scripts* externos	5
A aplicação faz controle por *Black-list* para neutralizar possíveis ataque XSS[11], ou a lista pode não contemplar todas as possibilidades.		Arquitetura e projeto, Implementação	Ausência de uso de *Black-list* ou lista desatualizada	4
Ordem	**CWE ID**	**Descrição**		
[2]	CWE-89	Limpeza inadequada de elementos utilizados em comandos SQL (*SQL Injection*).		
Vulnerabilidade		**Origem**	**Apelido para Diagrama de Influência *****	**Grau ****
Execução de declaração SQLs dinâmicas controladas por usuários externos para modificar SQLs internas da aplicação.		Arquitetura e projeto, Implementação	Ausência controle de execução SQLs externas	5
A aplicação utiliza tabelas que não são accessíveis por agente externo, mas permite que esse possa alterar chaves primárias através de execução de SQL. Exemplo, a chave pode ser alterada na declaração SQL para obter acesso a		Arquitetura e projeto, Implementação	Ausência de controle de acesso à chaves de tabelas	5

[11] Através de um XSS (*Cross-site scripting)*, o *hacker* injeta codigos *JavaScript* em um campo texto de uma página já existente e este *JavaScript* é apresentado para outros usuários. Fonte: http://pt.wikipedia.org/wiki/Cross-site_scripting.

informações não autorizadas.			
Limpeza inadequada de regras de permissão poderão permitir que instruções SQLs externas possibilitem acesso inadequado a dados.	Implementação	Limpeza inadequada de regras de permissão	4

** Classificado pelo Autor.
Opções consideradas: - 5: Crítica; 4: Acima do Normal; 3: Normal; 2: Abaixo do Normal; 1: Leve.
*** Nem todos os apelidos estarão representados nos diagramas de influência.

O **Tabela 5** apresenta as vulnerabilidades dos erros da categoria "Risco de Gerenciamento de Recurso".

Tabela 5: Vulnerabilidades - Categoria "Risco de Gerenciamento de Recurso". Fonte: [CWE/SANS, 2010]

Ordem	**CWE ID**	**Descrição**		
[3]	CWE-120	Cópia para *buffer* sem checar o tamanho permitido de entrada (*Classic Buffer Overflow*).		
Vulnerabilidade		**Origem**	**Apelido para Diagrama de Influência *****	**Grau ****
A aplicação não trata a verificação de valores atribuídos aos *arrays* podendo incorrer em um erro de *array índex*.		Implementação	Ausência de controle de *array índex*	5
A aplicação não trata corretamente o cálculo de tamanho ao alocar recursos em *buffer*.		Implementação	Tratamento inadequado a alocações de *Buffer*	5
Ordem	**CWE ID**	**Descrição**		
[16]	CWE-190	Estouro de inteiro (*Integer overflow*)		
Vulnerabilidade		**Origem**	**Apelido para Diagrama de Influência *****	**Grau ****
A aplicação executa algum cálculo incorreto não previsto para o tamanho do inteiro declarado para uma variável, causando alocação incorreta de recursos, atribuições de permissão indevidas, falha de comparação e até mesmo interrupção de funcionamento.		Arquitetura e projeto, Implementação	Tratamento inadequado de atribuição de valores	5

** Classificado pelo Autor.
Opções consideradas: - 5: Crítica; 4: Acima do Normal; 3: Normal; 2: Abaixo do Normal; 1: Leve.
*** Nem todos os apelidos estarão representados nos diagramas de influência.

O **Tabela 6** apresenta as vulnerabilidades dos erros da categoria "Fragilidade na Defesa".

Tabela 6: Vulnerabilidades - Categoria "Fragilidade na Defesa".
Fonte: [CWE/SANS, 2010]

Ordem	CWE ID	Descrição		
[5]	CWE-285	Controle de acesso inadequado (*Authorization*).		
Vulnerabilidade		**Origem**	**Apelido para Diagrama de Influência *****	**Grau ****
Funções ou métodos do aplicativo disponibilizados para agentes externos, mas sem o devido tratamento de restrições, possibilitando assim o uso incorreto.		Arquitetura e projeto, Implementação e Operação	Ausência de restrições de acesso à funções	5
Utilização de passagens secundárias para acessar arquivos ou páginas web que tenham como pré-requisitos outras páginas ou funções que passam parâmetros.		Arquitetura e projeto, Implementação e Operação	Permissão de acesso através de páginas secundárias	4
A aplicação permite ler ou modificar recursos de segurança críticos por meio de ações não intencionais.		Arquitetura e projeto, Implementação, Instalação e Operação	Ausência de controle para ações não intencionais.	5
Ordem	**CWE ID**	**Descrição**		
[10]	CWE-311	Falta de criptografia de dados sensíveis.		
Vulnerabilidade		**Origem**	**Apelido para Diagrama de Influência *****	**Grau ****
A aplicação armazena informações importantes de forma clara e com acesso a recursos externos ao sistema, quando deveriam essas informações estarem criptografadas ou protegidas de alguma forma.		Arquitetura e projeto	Ausência de criptografia no armazenamento	5
A aplicação transmite informações importantes sem criptografia ou proteção por meio de canais que possam ser monitorados por agentes externos.		Arquitetura e projeto, Operação e Configuração do Sistema	Ausência de criptografia na transmissão de dados	5

** Classificado pelo Autor.
Opções consideradas: - 5: Crítica; 4: Acima do Normal; 3: Normal; 2: Abaixo do Normal; 1: Leve.
*** Nem todos os apelidos estarão representados nos diagramas de influência.

4. Análise com Suporte de Diagramas de Influência

A partir da contextualização de riscos, da escolha dos erros a serem analisados e da identificação das vulnerabilidades é possível iniciar os desenhos dos gráficos dos diagramas de influência com o acréscimo das boas práticas de software, que possam atuar na mitigação das vulnerabilidades identificadas.

4.1. Boas Práticas em Software

O processo moderno de desenvolvimento de software proposto por Royce [1998] está dividido em cinco princípios, conforme apresentado na **Figura 2**:

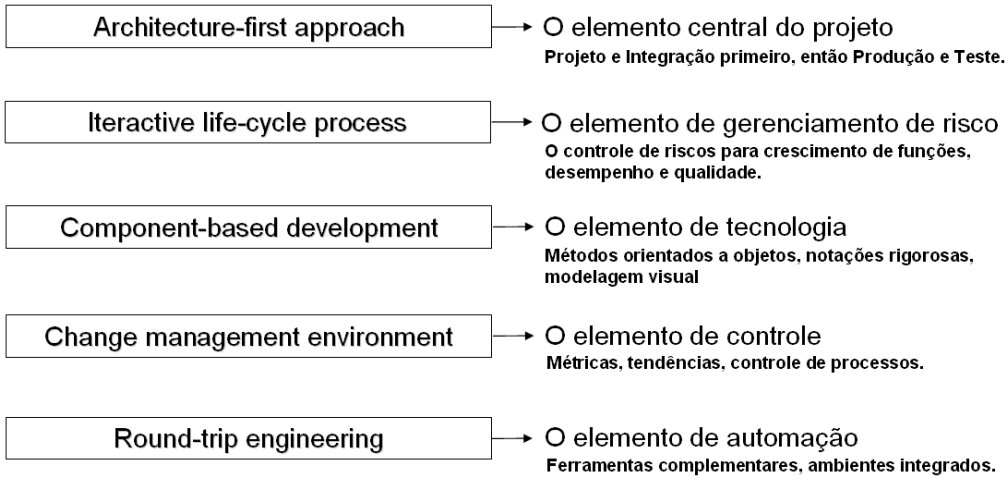

Figura 2: Princípios de um processo moderno de software.
Fonte: [Royce, 1998].

1) *Architecture-first approach* (o elemento central do projeto): nesse princípio é importante o foco dos esforços no projeto de implementação e de integração do software e, só após isso, os esforços devem ser concentrados na produção e nos testes.

2) *Iterative life-cycle process* (o elemento de gerenciamento de risco): a abordagem nesse princípio deve ser concentrada no controle dos riscos a fim de suportar o crescimento das funcionalidades do software, suportar o crescimento e garantir o desempenho e a qualidade.

3) *Component-based development* (o elemento de tecnologia): este princípio sugere a adoção de modelos formais de documentação, de modelagem visual e métodos orientados a objetos para representar os requisitos que estão sendo especificados.

4) *Change management environment* (o elemento de controle): um processo de software moderno deve estar amparado por métricas, por indicadores de tendência que permita a tomada de decisão para a correção de desvios do planejado e de um controle organizado de todas as etapas do processo de desenvolvimento.

5) *Round-trip engineering* (o elemento da automação): no processo moderno, há a necessidade de se amparar o processo com ferramentas complementares de apoio ao desenvolvimento, o controle do projeto e a utilização de ambientes integrados.

Esses princípios estão vinculados a processos de desenvolvimento. Nesse trabalho boas práticas de software está mais aderente ao contexto indicado por Sommerville [2003], que propõe que a Engenharia de Software é uma disciplina que envolve boas práticas de software, ferramentas, métodos e técnicas, cujo foco é o desenvolvimento de sistemas de software de alta qualidade com eficiência relativa ao custo, para solucionar problemas cuja análise demonstrou a necessidade do uso de um sistema de software.

4.2. Diagramas de Influência

Na gerência de risco, a atividade de análise é utilizada com a intenção de minimizar os riscos identificados [PMBOK, 2010]. A análise deve ser feita com o uso de ferramentas ou técnicas que possibilitem ao tomador de decisões uma melhor verificabilidade dos elementos envolvidos. Dessa forma, os diagramas de influência apresentados nesse trabalho tem a intenção de prover uma ferramenta visual para análise sistêmica das correlações entre os erros, suas vulnerabilidades e as boas práticas. As boas práticas utilizadas nos diagramas tem suas descrições conforme apresentadas na **Tabela 7**.

Tabela 7: Discriminação de boas práticas de software.
Fonte: [CWE/SANS, 2010]

Descrição	Apelido no digrama de influência
Checar parâmetros de entrada de acordo uma especificação pré-estabelecida.	Validação de parâmetros
Combinar *Black-lists* com *White-lists*[12].	Combinação entre *Black* e *White lists*
Não exiba a validação da *Black-list* nas caixas de dialogo ou mensagens para o usuário.	Não exiba validações de *Black-lists*
Utilizar mecanismos estruturados que separam automaticamente o código dos dados.	Separação de código fonte e dados
Criar contas (*accounts*) com privilégios limitados para execução de tarefas.	*Accounts* exclusivas para execução de determinadas tarefas
Utilizar assertivas para comparar parâmetros transmitidos com parâmetros especificados.	Validação de parâmetros por assertivas
Usar criptografia para dificultar legitimidade de valores passados como parâmetros ou associe assinatura digital junto ao parâmetro.	Uso de criptografia em parâmetros ou união de assinatura digital
Usar regras de privilégios de acesso.	Uso de regras de privilégios de acesso
Limpar variáveis da seção vinculadas ao cursos aberto com regras de permissão de acesso e fechar o cursor.	Limpeza de variáveis de seção e fechamento de cursores

[12] *White-list*: lista de parâmetros válidos.

As **Figuras 3 e 4** apresentam digramas com as relações entre os elementos da **Tabela 1** descrita na seção *3. Vulnerabilidades dos 25 Erros SANS/Mitre*.

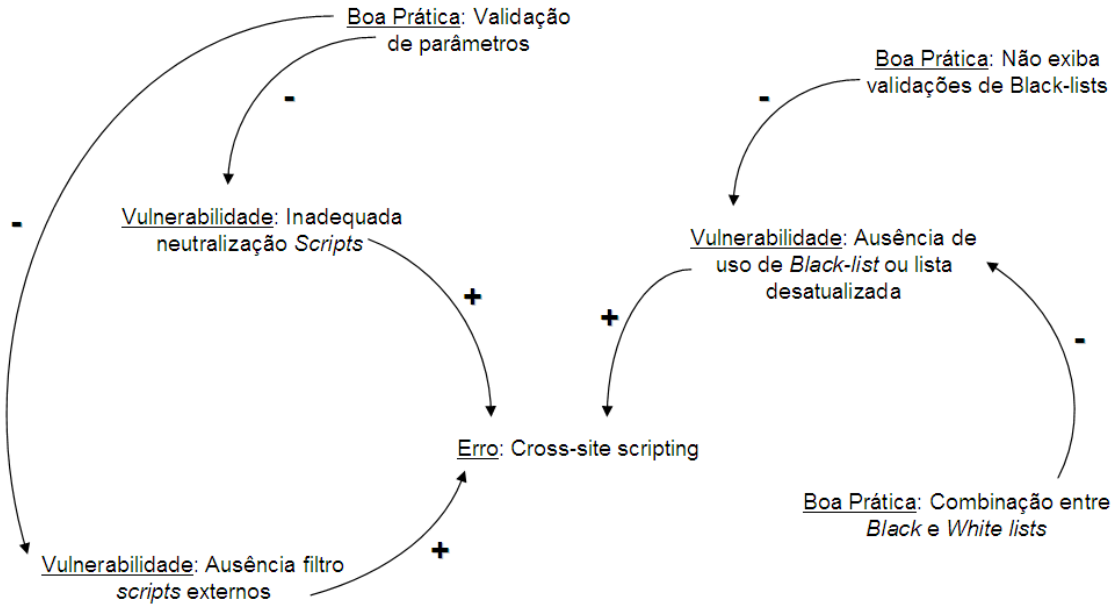

Figura 3: Diagrama de influência de elementos para o erro CWE-79 (*Cross-site scripting*).
Fonte: Elaborado pelo autor. Adaptado de Senge [1990].

No diagrama em questão, pode-se fazer uma análise isolada de pequenos arcos entre os elementos envolvidos. Dessa forma, é possível, de forma obvia, verificar que vulnerabilidades influenciam aumentando o erro e que boas práticas influenciam de forma a minimizar vulnerabilidades. Mas é importante uma análise conjunta dos elementos e suas co-relações.

A análise isolada de elementos não permite uma visão mais ampla da influência entre as partes. Ao serem agrupados em uma única visão é possível perceber que a influência de um elemento pode indicar uma melhora ou piora em mais elementos. Isso ocorre ao contrário também, pode-se verificar que mais de um elemento, possibilitando opção de escolha poderá influenciar um elemento central, seja de forma positiva ou negativa.

No caso do diagrama da **Figura 3**, há um exemplo dessa afirmação. A boa prática *Validação de parâmetros* influencia duas vulnerabilidades, permitindo sua mitigação. As vulnerabilidades *Inadequada neutralização de Scripts* e *Ausência filtro scritps externos* são influenciadas de forma a serem mitigadas com o uso da boa prática em questão. Como as duas vulnerabilidades acarretam a maior incidência de erro de *Cross-site scripting* é fortemente recomendando o uso da boa prática identificada para que se possa influenciar de forma a minimizar o erro, uma vez que ao reduzir a vulnerabilidade irá conseqüentemente atuar sobre o erro de forma inversa ao sinal da relação isolada de vulnerabilidade e erro.

Para a vulnerabilidade *Ausência de uso de Black-list ou lista desatualizada* pode-se verificar que há duas opções de boas práticas. Portanto, a indicação de N*ão exiba*

validações de Black-lists e *Combinação entre Black e White lists* são duas opções para se mitigar a vulnerabilidade e conseqüentemente reduzir o erro em questão.

No diagrama **Figura 4** pode ser visto que o padrão de comportamento estabelecido entre os elementos é mantido, ou seja, as co-relações entre erro, vulnerabilidades e boas práticas se mantém conforme discriminados no digrama anterior. As vulnerabilidades continuam afetando de forma a aumentar a chance do erro e o uso das boas práticas sugeridas iriam mitigar as vulnerabilidades e conseqüentemente reduzir chances do erro.

No diagrama pode-se ainda observar que há mais de uma possibilidade de boa prática para uma única vulnerabilidade, portanto, possibilitando ao tomador de decisões uma variabilidade de escolhas das práticas para mitigar vulnerabilidades. Isso pode ser verificado pelas boas práticas *Uso de criptografia em parâmetros ou união de assinatura digital* e *Uso de regras de privilégios de acesso* que reduzem a vulnerabilidade *Ausência de controle de acesso à chaves de tabelas.*

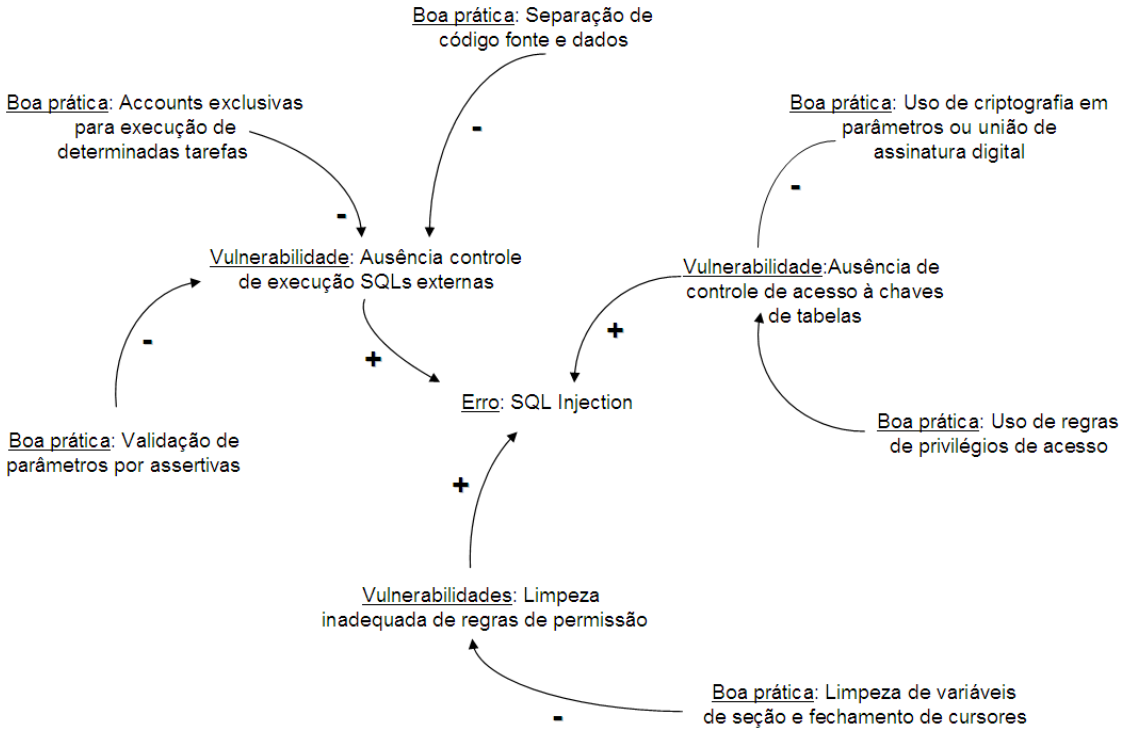

Figura 4: Diagrama de influência de elementos para o erro CWE-89 (*SQL Injection*).
Fonte: Elaborado pelo autor. Adaptado de Senge [1990].

Outras relações importantes e que não estão claras no documento CWE/SANS [2010] são quais boas práticas podem mitigar vulnerabilidades de erros distintos. Portanto, é importante analisar por diagramas de influência esse tipo de relação, pois interessa em muito aos tomadores de decisão de quais ações seriam mais eficientes, ou seja, minimizariam custos de desenvolvimento e aumentariam a eficácia da qualidade do software sob o aspecto de redução de erros.

A análise do diagrama **Figura 5** sugere que a boa prática *Uso de criptografia em parâmetros ou união de assinatura digital* afeta vulnerabilidades de erros distintos. Assim, é importante que sejam feitas ações para que tal boa prática possa ser implementada uma vez que sua aplicação sugerem uma redução significativa em três tipos de vulnerabilidades reduzindo as chances de ocorrerem os erros relativos ao acesso a informações importantes não criptografadas (*Falta de criptografia de dados sensíveis*) e também de *SQL Injection*.

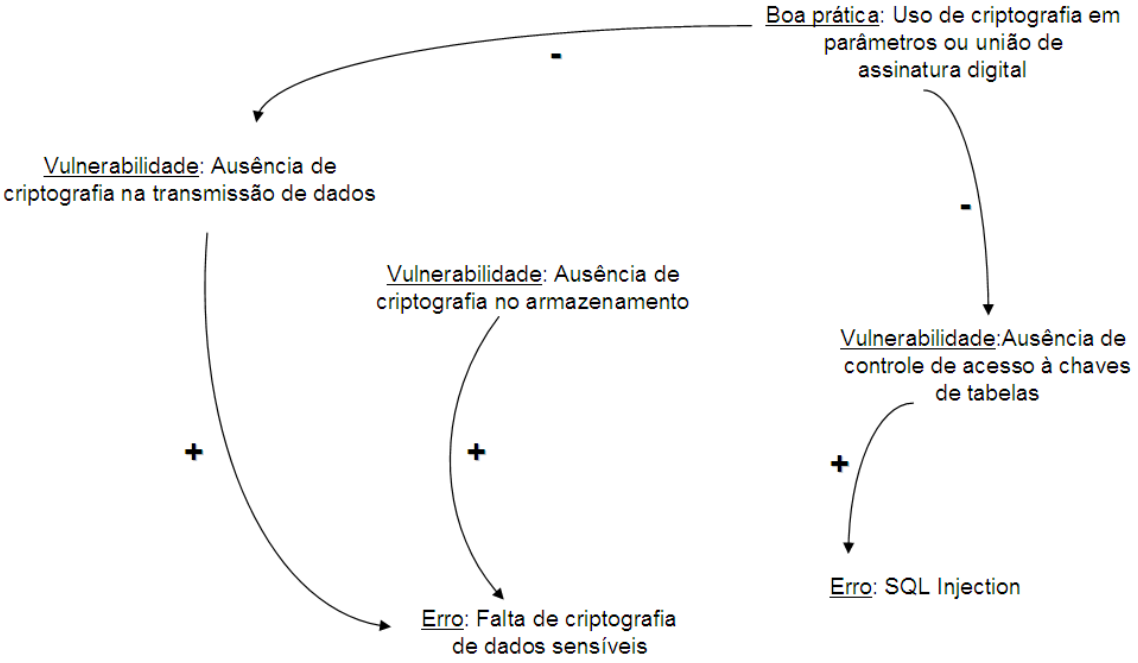

Figura 5: Diagrama de influência de elementos para o erros distintos.
Fonte: Elaborado pelo autor. Adaptado de Senge [1990].

O mesmo acontece com os elementos dispostos no diagrama da **Figura 6** em que as boas práticas sugeridas afetam de forma a mitigar vulnerabilidades e erros de categorias distintas conforme exposto na seção *3. Vulnerabilidades dos 25 Erros SANS/Mitre*. Ou seja, ao agruparmos elementos da categoria "Interação Insegura entre Componentes " e "Fragilidade de Defesa" pode-se observar um diagrama em que a boa prática *Limpeza de variáveis de seção e fechamento de cursores* afeta mitigando duas vulnerabilidades de erros distintos.

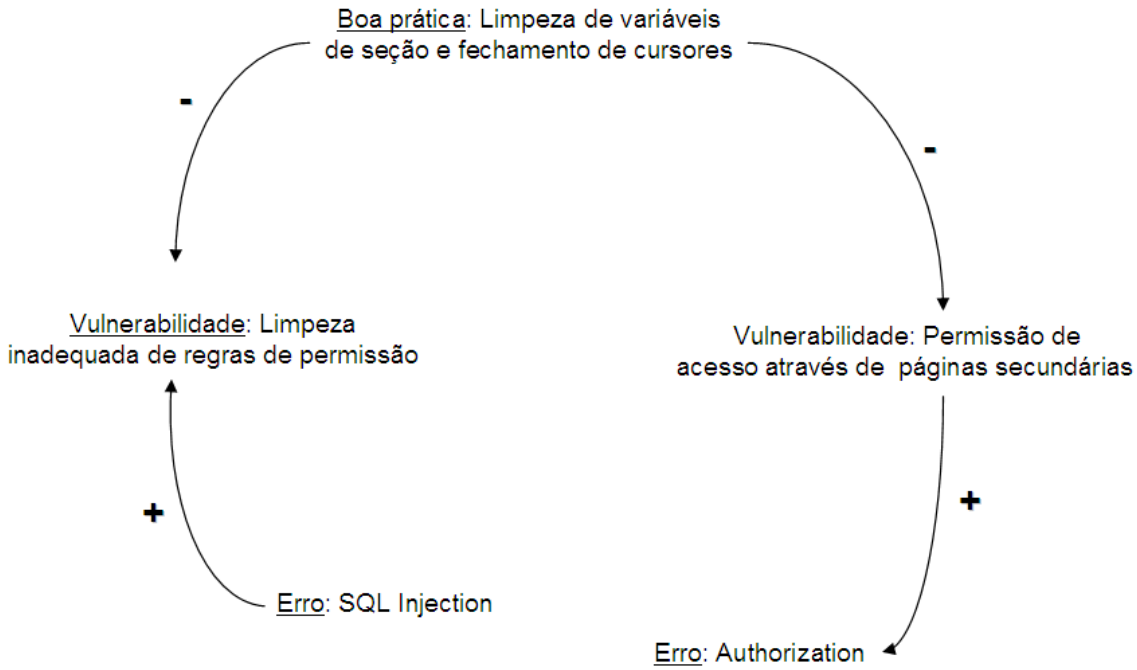

Figura 6: Diagrama de influência de elementos para o erros distintos.
Fonte: Elaborado pelo autor. Adaptado de Senge [1990].

5. Considerações Finais

As relações estabelecidas entre erros, vulnerabilidades e boas práticas de software nem sempre estão claras no cotidiano dos tomadores de decisão, sejam eles desenvolvedores no momento de produzir e testar software, ou mesmo gestores, quando necessitam decidir quais as melhores ações podem ser escolhidas para que a eficiência do processo de software seja sempre maximizada.

Os humanos estão suscetíveis a erros a qualquer momento do ciclo de vida do software, seja ao produzir código ou projetar determinada interface. Dessa forma, é importante que existam ferramentas que dêem suporte a uma análise mais sistemática e global para que analistas de situações possam também incorrer em menos erros. Os digramas de influência, quando bem implementados, podem demonstrar padrões de comportamento que passam desapercebidos quando necessitam ser encontrados em artefatos textuais ou simplesmente em uma análise empírica dos acontecimentos.

O documento produzido pela CWE/SANS [2010] tem bastante informação sobre erros, suas causas, exemplos, códigos anexados, conceituações, entre outros, mas não nos permite uma análise conjunta entre as diversas vulnerabilidades expostas e as ações que possam mitigá-las, quando se faz necessário avaliar os impactos de adoção de uma única boa prática de software e sua influência em demais fatores. Tal análise só é possível quando dispomos de uma ferramenta gráfica como o digrama de influência, onde é possível incluir diversos elementos dos diferentes erros apontados pela SANS/MITRE e avaliar seus impactos em diferentes vulnerabilidades ou erros.

Referências bibliográficas

[BERTALANFFY, 1977] Bertalanffy, L. Von. **Teoria Geral dos Sistemas**. Petrópolis: Vozes, 1977.

[CWE, 2010] **CWE Commom Weadness Enumaretion – A Community-Developed Dictionary of Software Weakness Types**. <Disponível em: http://cwe.mitre.org/> <Consultado em: Julho/2010>.

[CWE/SANS, 2010] **The 2010 CWE/SANS Top 25 Most Dangerous Programming Errors. The MITRE Corporation Copyright © 2010**. Document version: 1.03 Date: April 5, 2010. <Disponível em: http://cwe.mitre.org/top25/> <Consultado em: Julho/2010>.

[FERREIRA, 1999] Ferreira, A. B. de H. **Novo Dicionário Aurélio Século XXI: o dicionário da língua portuguesa**. 3. ed. Rio de Janeiro: Nova Fronteira, 1999.

[GORDON e LOEB, 2002] Gordon, L. A.; Loeb, M. P. **The Economics of Information Security Investment**. ACM Transactions on Information and System Secutity, Vol. 5, Nº 4, November 2002.

[KIM, 1992] Kim, D. H. **Toolbox: Guidelines for Drawing Causal *Loop* Diagrams**. The Systems Thinker, v.3, n.1, p.5-6, 1992.

[LIMA e TURNELL, 2006] Lima, A.T.P. Turnell, M.F.Q.V. **O contexto de trabalho, as IHMs e o erro humano na operação de sistemas elétricos**. In: Simpósio Brasileiro de Sistemas elétricos SBSE, 2006, Campina Grande/PB. Simpósio Brasileiro de Sistemas Elétricos, 2006, Art. no. 168. <Disponível em: http://www.labplan.ufsc.br/congressos/ SBSE/anais/168_sbse2006_final.pdf> <Consultado em: Junho/2010>.

[LEAL, 2009] Leal, A. L. de C.: **Uma proposta de taxonomia de boas práticas em desenvolvimento de software.** Master's thesis, Universidade Federal de Viçosa, CCE/DPI, dissertação de Mestrado. 2009.

[MITRE, 2010] **About Mitre. The Mitre Corporation**. <Disponível em: http://www.mitre.org/about/index.html> <Consultado em: Julho/2010>.

[PMBOK, 2010] **Pmbok Guide 2010**. PMI, 2010.

[PRITCHARD, 1998] Pritchard, E. et al. **Risk management: concepts and guidance**. Pensilvânia: Project Management Institute Press, 1998.

[RICHARDSON e PUGH, 1981] Richardson, G.P. Pugh III, A .L. **Introduction to System Dynamics Modeling with DYNAMO**. Cambridge, Massachusetts: Productivity Press, 1981.

[ROYCE, 1998] ROYCE, W. **Software Project Management - A Unified Framework**. Addison-Wesley, 1998.

[SANS, 2010] **SysAdmin, Audit, Network, Security Instituite - SANS**. Disponível em: <http://www.sans.org/about/sans.php>. <Consultado em: Julho/2010>.

[SÊMOLA, 2003] Sêmola, M. **Gestão da segurança da informação: uma visão executiva**. Rio de Janeiro: Campus, 2003.

[SENGE, 1990] Senge, P. **The Fifth Discipline: The Art and Practice of the Learning Organization**. ISBN 0-385-26095-4, New York, Currency Doubleday, 1990.

[SOMMERVILLE, 2003] SOMMERVILLE, I. **Engenharia de Software** / Ian Sommerville; tradução André Maurício de Andrade Ribeiro; revisão técnica Kechi Hirama. São Paulo: Addison Wesley, 2003.