2
Conceitos e Propriedades Desejáveis Para Um Software Seguro

Os objetivos deste capítulo são entender os principais conceitos, termos e definições sobre segurança presentes nesta dissertação, compreender a importância dos sistemas de informação para a garantia da segurança da informação e discutir sobre o que é desejável nos softwares para a garantia da segurança.

2.1 Definindo Segurança

O que é segurança? Dependendo do contexto pode significar coisas diferentes para pessoas diferentes, tornando a definição e o entendimento do conceito um pouco difícil. Na língua portuguesa, segundo o dicionário Aurélio (Ferreira, 1999), segurança tem os seguintes significados:

1. Estado, qualidade ou condição de seguro.
2. Condição daquele ou daquilo em que se pode confiar.
3. Certeza, firmeza, convicção.

Ao consultar também o significado da palavra seguro neste mesmo dicionário, presente no primeiro significado acima, é possível encontrar as seguintes definições:

1. Livre de perigo (ameaças)
2. Livre de risco; protegido, acautelado, garantido.
3. Em que se pode confiar.
4. Certo, indubitável, incontestável.

5. Eficaz, eficiente.

Como podemos observar, na lingua portuguesa dizer que algo é seguro tanto pode significar que está livre de perigos, como também algo eficaz. Diferentes significados para o termo segurança também são encontrados na literatura acadêmica (Avizienis et al., 2004), ITSEC (1991), (NBR ISO/IEC 17799, 2001), causando, muitas vezes, uma grande confusão, principalmente quando ampliamos o escopo do conceito de proteção (Chung et al., 2000).

Um software fidedigno (Staa, 2006) precisa ser confiável (Avizienis et al., 2004) e seguro quanto a eventos não intencionais e intencionais. Entender a motivação do evento que causa a insegurança nos ajuda a compreender que tipo de segurança está sendo abordado.

Eventos não intencionais ocorrem quando o agente da insegurança, ou agente de ameaças, não tem a intenção de provocar o dano, como por exemplo, aqueles causados por acidentes naturais ou humanos.

Eventos intencionais ocorrem quando os agentes de insegurança têm como objetivo executar ataques e causar incidentes que podem causar dano. Estes agentes, geralmente, utilizam vulnerabilidades no sistema para comprometer a segurança como um todo. O estudo da segurança para esta classe de evento é abordado na literatura como *security*, isto é, segurança contra eventos intencionais. Eventos decorrentes de ausência de responsabilidade e compromisso com o desenvolvimento de software com qualidade também podem ser incluídos em *security*.

A garantia de segurança (em inglês: *security*) em relação a eventos intencionais contra o software é o objeto de estudo nesta dissertação.

2.2 Propriedades desejáveis para um software seguro

A garantia de segurança pode ser trabalhada com a inclusão de propriedades no projeto de software que necessita ser seguro. Estas propriedades desejáveis podem ser encontradas no ITSEC (1991), que define segurança (em inglês:

security) como uma composição de atributos de confidencialidade, integridade e disponibilidade aplicáveis ao objeto software em um contexto de sistema de informação. Este é o conceito mais tradicional de segurança da informação. A definição mais detalhada de cada propriedade desejável de segurança é a seguinte (Staa, 2006):

Disponibilidade (em inglês: *availability*) – prontidão para o serviço correto. Preservar a disponibilidade consiste na garantia de que usuários autorizados obtenham acesso à informação e aos ativos correspondentes sempre que necessário. Deve ser evitada a destruição desautorizada da informação ou a indisponibilização dos serviços.

Integridade (em inglês: *integrity*) – ausência de alterações não permitidas (corrupção de elementos). Preservar a integridade consiste na salvaguarda da exatidão e completeza da informação e dos métodos de processamento. Normalmente, o interesse comercial por esta propriedade é maior do que a confidencialidade. Pode ser subdividida em:

> Integridade de dados – a propriedade de um dado não ser alterado, destruído ou perdido de maneira acidental ou não autorizada.
>
> Integridade do software – a qualidade de um software em preservar suas funcionalidades (ausência de alterações não permitidas), não sendo corrompido durante o seu desenvolvimento ou execução (Goertzel et al., 2006).

Confidencialidade em (em inglês: *confidentiality*) – preservar a confidencialidade consiste em garantir que o acesso à informação seja obtido somente por pessoas autorizadas a terem esse acesso. A confidencialidade pode ser considerada uma propriedade desejável para a garantia da privacidade (em inglês: *privacy*), que é a habilidade de proteger dados e código de acesso indevido (Staa, 2006).

A Figura 1 representa o conceito mais tradicional de segurança da informação. A informação precisa estar disponível para quem tem o direito de acessá-la. Quem não tem direito de acessá-la deve ser impedido de utilizar

vulnerabilidades para praticar atos que violem a integridade e a confidencialidade do sistema. Neste casos a informação não pode sair do sistema (quebra de confidencialidade) e nem pode ser modificada ou apagada para quebra de integridade. Setas marcadas com um "X" representam ameaças as propriedades de segurança que devem ser impedidas para que a informação não venha ser violada e adulterada.

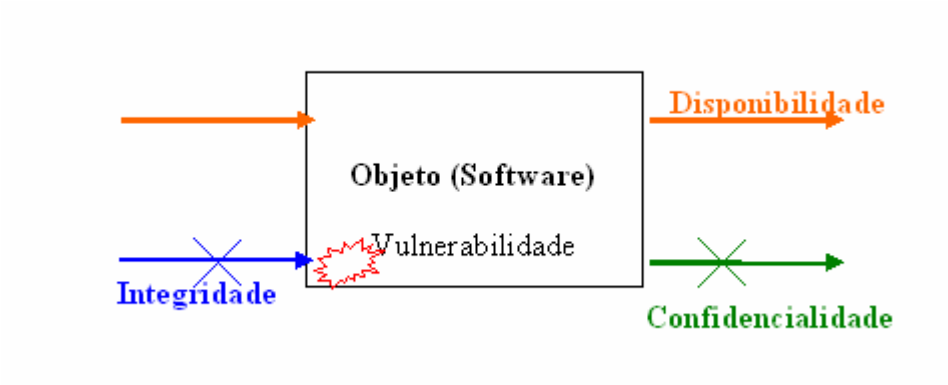

Figura 1: Segurança em seu conceito mais tradicional

Além das propriedades tradicionais, também estão sendo incluidas como propriedades desejáveis para o software seguro a propriedade de não-repudiabilidade (em inglês: *non-repudiation*), que consiste na habilidade em prevenir o software (como um usuário) de refutar ou negar a responsabilidade por ações por ele executadas, e a *accountability* (que em português pode ser traduzida como responsabilidade) determina que todas as ações relevantes para a segurança do software devem ser registradas e rastreadas junto com a atribuição de responsabilidades, isto é, com a possibilidade de se poder responsabilizar alguém por seus atos. O rastreamento deve ocorrer tanto no momento como depois em que as ações ocorrem.

A propriedade não-repudiabilidade e a propriedade responsabilidade são, geralmente, associadas ou com usuários humanos ou com entidades de software que atuam como usuários (*agents proxies, Web services*) (Goertzel et al., 2006).

A segurança de software, então, consiste na preservação da confidencialidade, da integridade, da não-repudiabilidade, da possibilidade de prestar contas e da disponibilidade dos ativos e recursos de informação que o

software cria, armazena, processa ou transmite, incluindo o próprio programa em execução.

2.3 Relacionamento entre Fidedignidade e Segurança

A segurança e a fidedignidade no funcionamento são duas áreas de pesquisa já com cerca de três décadas de existência com muitos pontos em comum e com o mesmo objetivo que é o funcionamento correto de sistemas computacionais (Correia, 2006). Um software fidedigno apresenta garantia da execução de suas funções requeridas, sob todas as circunstâncias possíveis, em um período de tempo satisfatório, sem nunca executar qualquer ação cuja conseqüência leve a resultados indevidos, tais como acidentes sérios, perda de vidas ou propriedades, prejuízos para negócios de empresas ou violação de segurança.

Avizienis (Avizienis et al., 2004) em seu trabalho estabelece definições para caracterizar vários conceitos relacionados à fidedignidade e a segurança de sistemas de computação e comunicação, apresentando um forte relacionamento entre estes dois conceitos a partir dos seus principais atributos.

Figura 2: Relacionamento de Fidedignidade e Segurança (extraída de Avizienis et al., 2004)

Conforme a Figura 2 acima, além dos atributos disponibilidade, integridade e confidencialidade já apresentados nesta dissertação, a fidedignidade de um software é uma composição dos seguintes atributos (Staa, 2006):

- Confiabilidade (em inglês: *reliability*)- continuidade do serviço correto.

- Segurança (em inglês: *safety*) – ausência de conseqüências catastróficas tanto aos usuários quanto ao ambiente, a partir de um evento não intencional.

- Manutenibilidade (em inglês: *maintainability*) – habilidade de ser modificado ou corrigido sem que novos problemas sejam inseridos.

Para melhor entender o relacionamento entre segurança e fidedignidade considere a natureza das ameaças à segurança e, por extensão, a fidedignidade e as conseqüências destas ameaças quando bem sucedidas. Algumas ameaças bem sucedidas contra a fidedignidade do software são resultantes de falhas decorrentes de faltas no software. De acordo com Avizienis et al, faltas de software caem em uma das três categorias a seguir:

1. Faltas no desenvolvimento: um tipo de falta que é introduzida durante o processo de desenvolvimento do software. Quando estas faltas são exercitadas podem ocorrer erros que, uma vez observados, constituem falhas.

2. Faltas físicas: um tipo de falta que é provocada por um defeito, possivelmente transiente, ou anomalia no hardware em que o software executa.

3. Faltas externas: um tipo de falta que é provocada externamente ao software e de sua plataforma de hardware. Faltas externas podem interagir com o software como parte dos dados de entrada de usuários, variáveis passadas pelo ambiente, mensagens recebidas a partir de outro software, etc.

Faltas humanas podem ser intencionais ou não intencionais e podem ou não ser maliciosas. Faltas não maliciosas intencionais muitas vezes são resultantes de uma má avaliação. Por exemplo, uma decisão de um desenvolvedor em investir mais em performance e usabilidade do que em segurança em um projeto que necessita ser seguro. Ferramentas e teorias errôneas também podem levar a perda de segurança.

As faltas existentes no software tornam-se uma ameaça à fidedignidade quando elas são exercitadas. Faltas são sempre latentes, ou seja existem, mas não necessariamente resultam em erros e falhas. Faltas são interessantes por causa de seu potencial em causar problemas quando forem exercitadas. Conseqüentemente, faltas sempre constituem vulnerabilidades que um invasor pode explorar.

A ênfase dos estudos sobre segurança tem sido em problemas de origem maliciosa (ataques, código nocivo), enquanto que o foco dos estudos sobre fidedignidade tem sido mais nos problemas de origem acidental. No entanto, as disciplinas não se excluem, pois a segurança pode tratar problemas de origem acidental e a fidedignidade no funcionamento pode incluir problemas de origem maliciosa (Correia, 2006).

2.4 Terminologias associadas com segurança e seus relacionamentos

Alguns termos empregados relacionados com o tema *security*, representados pela Figura 3, merecem uma breve descrição.

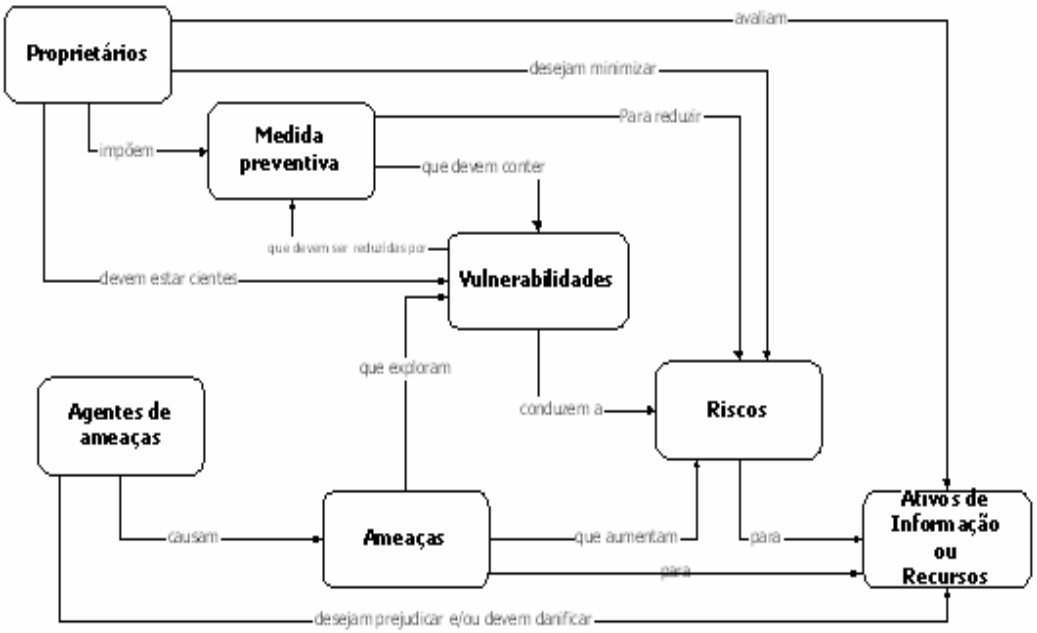

Figura 3: Conceitos relacionados à segurança e seus relacionamentos. (extraída do Common Criteria , 2002)

Proprietários são os indivíduos responsáveis em decidir em nome da organização no que diz respeito ao uso, à identificação, à classificação, e à proteção de um recurso ou ativo específico da informação (Redwine, 2004).

Ativo é uma entidade que possui algum valor para seus proprietários como, por exemplo, uma informação, um serviço ou um bem material. Normalmente, seus proprietários buscam medidas preventivas, que nem sempre são eficientes, para proteger seus ativos e recursos contra ameaças (Redwine, 2004).

No contexto da segurança em software, uma vulnerabilidade é uma falta em um produto de software que juntamente com algum conhecimento sobre o software e seu ambiente operacional, permite que uma invasão ocorra, tornando o software inseguro, mesmo que esteja sendo utilizado adequadamente. Voas (1996) define nível de vulnerabilidade como o grau de acesso não autorizado permitido a um sistema. Por exemplo, o acesso a camadas internas de um sistema operacional precisa ser protegido de usuários sem autorização, porém se tais usuários podem obter este acesso, então o sistema operacional será classificado como tendo um alto nível de vulnerabilidade. As vulnerabilidades podem se originar a partir da tecnologia, pessoas ou processos. Davis (2004) afirma que a maioria das vulnerabilidades de segurança encontradas hoje nos softwares ocorre a partir de faltas que são inseridas de forma não intencional ao longo do processo de desenvolvimento. Políticas e procedimentos organizacionais mal definidos e comunicados são também vulnerabilidades. As vulnerabilidades conduzem a riscos sobre os ativos e recursos de informação (Redwine, 2004).

O termo agente de ameaças é usado para indicar um evento da natureza (meteoro, fogo, alagamento, etc), um grupo de indivíduos ou um indivíduo que possui ou pode possuir capacidade e intenção de criar ameaças (Redwine, 2004).

Ameaças são eventos não esperados (deliberados ou acidentais), externos ao sistema de software, que podem ocorrer, resultando em prejuízos aos ativos e recursos de informação (Redwine, 2004). Normalmente, o software está sujeito a duas categorias gerais de ameaças (Goertzel et al., 2006):

1. Ameaças durante o desenvolvimento (principalmente ameaças

internas): Um desenvolvedor pode sabotar o software em algum ponto em seu ciclo de desenvolvimento, através de exclusões intencionais, inclusões, ou modificações da especificação de requisitos, do modelo de ameaças, dos documentos do projeto, do código fonte, dos casos de testes e evidências de testes, etc. Um processo de desenvolvimento seguro ajuda a reduzir a exposição do software as ameaças internas durante o seu processo de desenvolvimento.

2. Ameaças durante a operação (tanto internas como externas). Qualquer sistema de software que roda sobre uma plataforma conectada a uma rede corre o risco de ter suas vulnerabilidades expostas durante a sua operação. O nível de exposição variará dependendo de se a rede é pública ou privada, conectada a Internet ou não. Quanto maior, mais aberta e descontrolada a rede, mais exposto o software estará a ameaças externas. Mas, mesmo se a rede é privada, pequena, e gerenciada com atenção, esta é uma ameaça a partir de elementos não confiáveis na comunidade de usuários autorizados do software (“pessoal interno mal intencionado”).

Entre os exemplos de ameaças a que um sistema de informação pode estar sujeito temos (Pfleeger , 2003):

1. Falhas do software ou hardware. Perturbações ambientais, incluindo desastres naturais.

2. Erros de operação e administração.

3. Erros de projeto ou de implementação.

4. Ataques hostis (explorando os 2 pontos anteriores relacionados a erros)

Provavelmente jamais será possível a construção de uma lista definitiva de todas as ameaças existentes contra os sistemas de informação, bem como a implementação de proteção em relação a todas elas.

Uma abordagem interessante (figura 4) para entender ameaças e vulnerabilidades está presente no trabalho de Pfleeger (2003) em que a água representa uma ameaça para um compartimento e a rachadura na parede representa uma vulnerabilidade.

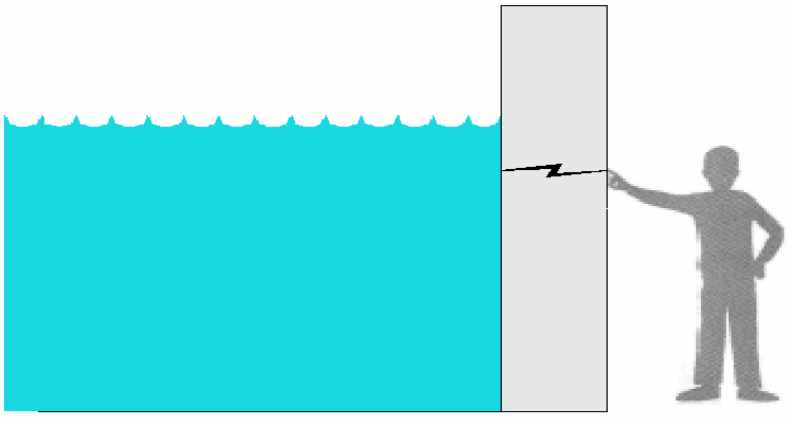

Figura 4: Ameaça e vulnerabilidade (extraída de Pfleeger, 2003)

2.5 Relações entre Segurança da Informação, Segurança de Sistemas de Informação, Segurança de Software

A informação é passiva por natureza (Goertzel et al., 2006). Ela não pode executar ações, mas somente ter ações executadas sobre ela ou a partir dela. Não se torna vulnerável por causa da forma como foi criada. A vulnerabilidade ocorre quase sempre pela forma como é armazenada ou transmitida. Proteger a informação como um ativo quanto à revelação não autorizada, modificação ou destruição é o principal objetivo da segurança da informação (Goertzel et al., 2006).

A segurança da informação estende a necessidade de proteção ao sistema de informação que armazena, transmite, e processa a informação, representada na forma de dado. Cada vez mais presentes e populares em nossa sociedade, sistemas de informação são projetados para atender a uma grande variedade de importantes serviços e aplicações presentes nos seguintes recursos computacionais:

- A Internet (serviços tais como transferência de arquivos, e-mail e *World Wide Web*). Comércio Eletrônico

- Intranets (Uma parte da Internet composta por uma ou várias redes locais)

- *Computação móvel e ubíqua*

A figura 5 a seguir representa um típico sistema de informação com os recursos computacionais descritos acima:

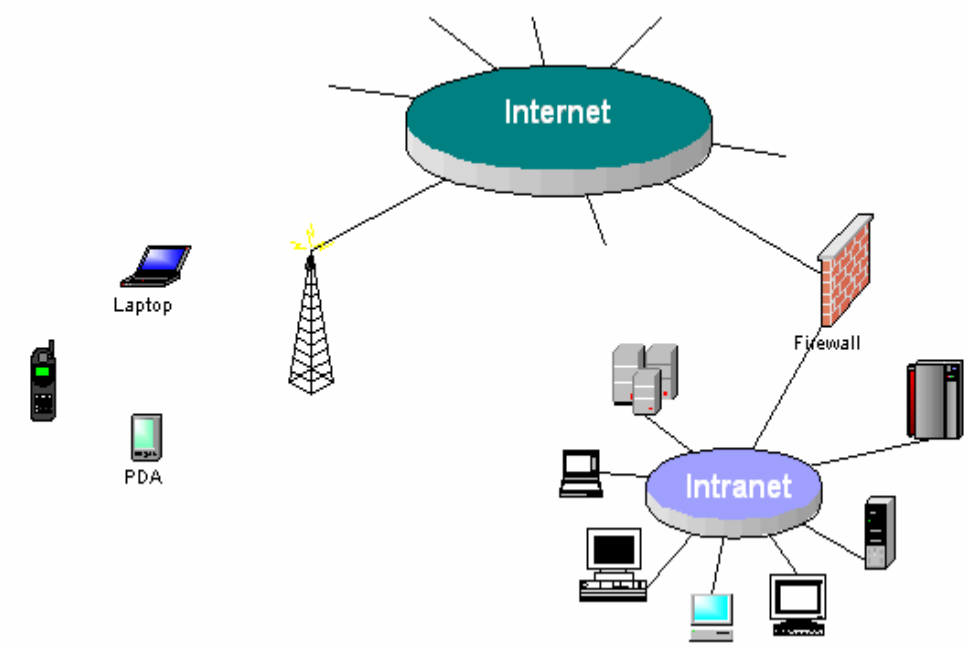

Figura 5: Sistemas de Informação

Laudon (2001) define sistema de informação como um conjunto de componentes inter-relacionados que coleta (ou recupera), processa, armazena e distribui informações destinadas a apoiar a tomada de decisões e o controle de uma organização. Segundo Goertzel et al. (2006), um sistema de informação baseado em computador é constituído por pessoas, softwares, hardwares e recursos (procedimentos e dados), o que torna a segurança em sistemas de informação um assunto amplo que tem que capturar todos os aspectos relevantes do sistema e de seu ambiente (Figura 6).

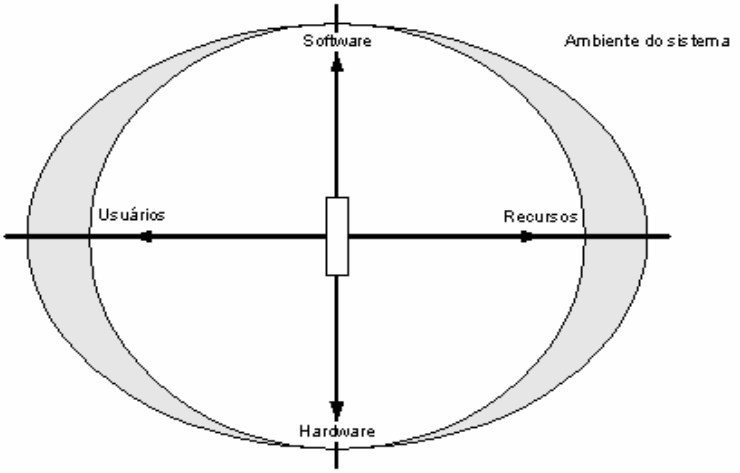

Figura 6: Diferentes dimensões de segurança do sistema (extraída de Goertzel et al., 2006)

Devido às inúmeras possibilidades de interfaces de comunicação com o software, uma pessoa tem a possibilidade de praticar atos inseguros, seja de maneira intencional ou não, ameaçando a segurança do sistema. Implementar segurança em sistemas de informação consiste em evitar ou minimizar desastres e incidentes a partir de eventos inseguros. Isto pode ser tentado através de controles e funções de seguranças necessários, implementados também por software, que busquem garantir que a informação e os serviços por eles processados serão protegidos (consistente com o objetivo do sistema de informação). Sistemas de informação são dependentes da execução de autenticação e autorização de usuários e controle de acesso para gerenciar quem está autorizado a usá-lo enquanto mantém todas as entidades não autorizadas fora.

Alguns fatores contribuem para a exploração de um sistema de informação (Goertzel et al., 2006):

1. O software como o link mais fraco. O número de ameaças objetivando atingir especificamente o software está aumentando e a maioria dos ataques a redes e aos sistemas de informação explora vulnerabilidades de softwares. Segundo relatório do grupo Gartner denominado *"Now Is the Time for Security at the Application Level"* de dezembro de 2005 (Lanowitz, 2005), 75% dos ataques contra a segurança de sistemas de informação ocorre no nível da aplicação.

2. O tamanho, a complexidade do software e as diversas interfaces em um sistema de informação, dificultam a realização e a validação de testes com suficiente abrangência.

3. *Outsourcing* e o uso de softwares não verificados aumentam os riscos de segurança.

4. O reuso de softwares legados interfaceando com outras aplicações em novos ambientes pode introduzir outras conseqüências não analisadas, aumentando o número de vulnerabilidades. Os novos riscos quase sempre não são entendidos.

O software é quase sempre uma parte de um sistema mais complexo e para ser seguro significa que será executado neste contexto sem contribuir para um incidente (acidente ou perda). O software necessita ser seguro por causa daquilo que faz, incluindo como influencia outras entidades. O objetivo da segurança de software é produzir software que não seja vulnerável a modificação não autorizada, tampouco deve possibilitar a um usuário, mal intencionado ou não, ter acesso a dados, podendo difundir, alterar ou mesmo destruí-los. Além disso, o software não deve oferecer oportunidades para que venha a ser adulterado através do uso de operações em princípio autorizadas, não permitindo o *denial of service* durante a sua execução (Goertzel et al., 2006). Usar práticas de desenvolvimento que aumentam a possibilidade dele ser seguro contribuirá para a capacidade do sistema de informação em alcançar seus objetivos de segurança.

Leveson (2002), no contexto de *software safety*, afirma que, ao contrário do que a maioria dos usuários e desenvolvedores imagina, funcionalidades de softwares não necessariamente garantem segurança, mas podem contribuir para alcançá-la. Segundo a autora, sendo o sistema inseguro é impossível que qualquer software construído para este sistema seja seguro. A segurança é uma propriedade que deve ser avaliada a partir do comportamento total de um sistema. Um bom entendimento das propriedades essenciais de sistemas de informação é essencial quando projetamos a segurança de um software. Incidentes podem ser causados não só por um componente isolado deste sistema, mas também devido a uma má interação entre os diversos componentes (humano, digital, eletromagnético, etc)

que compõem o sistema. O tratamento das interações desses componentes é um dos maiores desafios para a avaliação da segurança.

Obviamente, existem sobreposições significantes entre os objetivos principais da segurança da informação, da segurança de sistemas de informação e da segurança de software. Contudo, estas sobreposições não devem ser mal interpretadas quando afirmarmos que os requisitos de segurança de todas as três entidades serão idênticos, ou que seus principais objetivos de segurança podem ser alcançados pelos mesmos meios.

2.6 O que queremos em relação ao desempenho do software quanto à questão da segurança?

Queremos um software fidedigno, com execução previsível e conforme com os seus requisitos de segurança, tanto como um componente individual quanto no nível de segurança do sistema como um todo. Para isto, o software deve ser resistente ou tolerante a ataques.

Para alcançar robustez, habilidade de detectar faltas de modo que as conseqüências possam ser mantidas em um patamar aceitável, os componentes de software e o sistema de informação como um todo devem ser capazes de reconhecer padrões de ataques – nos dados de entrada ou sinais que eles recebem a partir de entidades externas (pessoas ou processos) e resistir à entrada. A resistência à padrões de ataques e faltas externas pode ser alcançada bloqueando dados de entrada que contém os padrões de ataque e encerrando a conexão do software com a entidade externa defeituosa e hostil.

Tolerar falhas resultantes de ataques bem-sucedidos ou de faltas externas intencionais, significa manter a continuidade da operação do software apesar das falhas. Freqüentemente a continuação da operação somente pode ser alcançada por gerenciamento dos processos críticos. Quando o software entra um modo de falha, processos não críticos são terminados de forma ordenada, com uma condizente explicação para o usuário dos motivos do seu encerramento, enquanto os processos críticos para o funcionamento de sistemas são mantidos com um

aceitável nível de funcionamento. Dependendo da capacidade de tratamento de exceção do software, a operação continuada somente é possível até um certo nível de operação degradada, para depois o software terminar completamente a operação (falha severa).

O que o software seguro nunca deve fazer é simplesmente travar (Goertzel et al., 2006). Nenhuma falha deve ser permitida sem que primeiro sejam apagados todos os dados da memória temporária do computador, sinalizando neste momento para todos os usuários a falha. Após estes passos, o software deve então liberar de forma segura todos os recursos, finalizando os processos de software. Isto é o que é entendido por *"fail safe"*, que reduz a possibilidade de um atacante ganhar acesso de leitura a dados sensíveis em memória temporária.

A recuperabilidade do software, habilidade em ser rapidamente reposto em operação fidedigna após a ocorrência de uma falha (ataque bem sucedido), é uma forma de capacidade de sobrevivência que tem provado ser possível de ser atingida somente ao nível do sistema como um todo (Goertzel et al., 2006). Para alcançar o poder de recuperação de ataques, os sistemas de software devem ser capazes de recuperação a partir de qualquer falha resultante de um ataque bem-sucedido ao software (incluindo faltas externas intencionais) e reassumir a sua operação em um nível mínimo aceitável em um período curto, até finalmente conseguir restaurar todo o serviço no nível de performance especificado. A recuperação do sistema de software deve ocorrer tão logo a fonte de ataque tenha sido isolada e bloqueada, e os danos resultantes tenham sido encerrados.

Para conseguir um software com este nível de qualidade, o investimento em segurança, aqui considerando desenvolvimento do software e o estabelecimento do ambiente de operação, não é pequeno. Além das preocupações tradicionais do desenvolvimento de software com qualidade assegurada, as equipes de desenvolvimento e operação dos sistemas se vêem obrigadas a se preocupar com conceitos e aspectos que não estão acostumadas a trabalhar. Para o cliente isto é ruim, pois acaba sendo impactado no prazo e no custo da solução e, muitas vezes, acaba não tendo um software com qualidade assegurada em relação à segurança. Segurança esta que, na maioria dos casos, ele mesmo não entende quanto à

necessidade de ser incorporada à sua solução na maioria dos casos. Além disso, os métodos e práticas atuais de Engenharia de Software têm tido sucesso limitado em gerenciar a complexidade intelectual de projetar e implementar software seguro. Normalmente, no projeto de sistemas de software, conceitos de segurança são endereçados muito tardiamente ao invés de ser uma prática integrada ao projeto como um todo, principalmente nas fases de especificação e arquitetura. Isto significa que os sistemas de software, de qualquer tamanho e complexidade, acabam tendo muitas vulnerabilidades passíveis de serem exploradas.