Uma proposta para a implantação de um ambiente de desenvolvimento de software com segurança

ANDRÉ LUIZ TEIXEIRA DOS SANTOS [15], MILTON FAGUNDES VALPASSOS[16], MARÇAL DE LIMA HOKAMA[17]

Resumo. Este artigo tem como finalidade orientar na implantação de um ambiente de desenvolvimento seguro, utilizando técnicas que visam minimizar falhas que possam servir como "brechas", e facilitar os acessos indevidos. Tais acessos podem causar alguns transtornos, tais como: obtenção de informações sigilosas, alteração de registros confidenciais e até mesmo a destruição de uma base de dados. São abordadas as causas e conseqüências, assim como possíveis soluções, tendo como base a ISO 15.408, que é originária do *Common Criteria for Information Tecnology Security Evaluation.* Com isso, tem-se como resultado, quando utilizadas tais técnicas, uma aplicação muito mais confiável e segura.

Palavras chaves: Segurança, Aplicação, Desenvolvimento, Invasão e Software

Summary. This article aims to direct in the implantation of a secure software by using techniques that serve the purpose of minimizing faukts that can be used as openings, whose inapropriete accesses can be facilitated. Such accesses may cause prublems, such as secret: secret information access, confidential recording alteration and even the destruction of a database. It deals with some causes and consequeces as well as some possible solutions based upon the ISO 15.408, whose origin comes from then Common Criteria for Information Tecnology Security Evaluation.Therefore, whenever such techiniques are used, they result in a much more reliable and secure softwere.

Key-Words: Security , Aplication, Development, Invasion, Software

1. Introdução

Muito se fala na segurança de sistemas, mas geralmente o enfoque é dado na segurança da rede, nos possíveis ataques de *hackers*, ou seja, uma visão totalmente externa, esquecendo que muitas vezes oferecemos as vulnerabilidades que tanto desejam os possíveis invasores. Estas ocorrem geralmente durante o desenvolvimento de *software*.

As principais falhas tem como origem na não observância das vulnerabilidades existentes no ambiente de desenvolvimento.

Este artigo tem como finalidade auxiliar na elaboração de uma política de desenvolvimento segura, baseado na norma ISO / IEC 15.408 (*Common Criteria for*

[15] Tenente-Aluno do Curso de Formação de Oficiais do Quadro Complementar de 2004.Graduado em Informática. Delcts@bol.com.br.
[16] Tenente-Aluno do Curso de Formação de Oficiais do Quadro Complementar de 2004. Graduado em Informática. miltonval@ig.com.br.
[17] Capitão do Quadro Complementar de Oficiais. Bacharel em Ciências da Computação. caplima@esaex.mil.br.

Information Technology Security Evalution). Serão abordadas também algumas falhas na programação, mostrando suas conseqüências e possíveis soluções.

A norma ISO / IEC 15.408 (*Common Criteria for Information Technology Security Evalution*), na maioria das vezes chamado apenas de *Cammom Criteria* (em português - Critério comum para avaliação de segurança de tecnologia da informação), tem como objetivo fornecer um conjunto de critérios fixos que permitem especificar a segurança de uma aplicação de forma não ambígua a partir de características do ambiente da aplicação, e definir formas de garantir a segurança da aplicação para o cliente final (ALBUQUERQUE, RIBEIRO – 2002*)*.

O presente texto visa expor algumas técnicas relevantes no desenvolvimento de uma aplicação, tais como: o levantamento do ambiente de desenvolvimento, a proteção dos dados dos usuários e das funções de segurança, e a garantia da segurança da aplicação.

2. Política para desenvolvimento de software

Em desenvolvimento de software, normalmente a segurança só é considerada nas fases finais dos projetos, quando é considerada, resultando assim em uma série de adaptações nos sistemas. Incluindo o desenvolvimento de software na política de segurança, reduz-se consideravelmente o retrabalho.

As normas pertinentes ao desenvolvimento de sistemas contemplam as seguintes funções:

- ✓ Processo de desenvolvimento
- ✓ Testes
- ✓ Documentação
- ✓ Controle de revisão e configuração
- ✓ Uso de terceiros
- ✓ Propriedade intelectual

No processo de desenvolvimento, as normas devem garantir que a segurança será considerada durante todo o projeto: especificação, programação e homologação. As políticas devem identificar de quem são as responsabilidades por promover o desenvolvimento seguro e por colocar o sistema em produção. Os desenvolvedores devem conhecer todas as políticas de segurança, padrões, procedimentos e outras convenções de desenvolvimento, tais como: padrão de criação de nomes para os objetos de um sistema (banco de dados, arquivos, variáveis, etc.); senhas não podem ser transmitidas pela rede sem que sejam criptografadas e nem armazenadas em arquivos textos e em dispositivos com acesso livre; e etc.

As seguintes regras devem ser observadas: obrigar que os sistemas sejam especificados, e esta especificação contemple os requisitos pertinentes à segurança e privacidade dos dados; prever a validação da entrada de dados nas interfaces das aplicações; prever a checagem dos dados nos processos de transmissão; e o sistema não poderá ter "portas dos fundos" e nenhum outro mecanismo de interação que permita a entrada ou saída não segura de dados.

Quanto aos testes, deve ser criada uma política que garanta a segurança de todos os dados utilizados, uma vez que estes dados devem ser extraídos do ambiente de produção, para que os testes possam ser realistas. As rotinas de teste devem abordar o aspecto de segurança, de modo a avaliar eventuais vulnerabilidades nos sistemas e devem ser feitas exaustivamente. Para facilitar as rotinas de teste, sempre que possível, deve haver reutilização de código entre projetos, pois estes códigos reutilizados, por terem feito parte de um outro projeto, já sofreram uma boa etapa de testes.

A documentação não é de cunho obrigatório para a segurança. Mas permitirá, no futuro, que outros programadores possam entender com mais facilidade como os

mecanismos de segurança foram implantados, facilitando assim a continuidade e melhoria da política de segurança.

Com o controle de configuração os administradores podem saber se a segurança foi violada através da instalação de algum programa não autorizado. Já que este controle permite saber o que deveria estar instalado nos equipamentos e na rede.

O controle de revisão e configuração permitirá rastrear as mudanças nos sistemas. A política de segurança deve obrigar que todas as solicitações de modificações nos sistemas sejam formalizadas, respeitem uma alça de aprovação e não violem as regras de segurança. (MARTINS, 2003)

É comum que os sistemas tenham *bugs*, fazendo necessária a instalação de *patches* que corrijam este problema. Contudo, os *patches* podem criar vulnerabilidades nos sistemas. Por isto, é de suma importância que a política de segurança exija que os *patches* sejam avaliados, antes de entrarem em produção, num ambiente de homologação. Por vezes se faz necessária a desinstalação de um software ou *patch*, devido a um bug ou incompatibilidade com outro sistema. As políticas que tratam do controle de configuração devem requerer que um sistema ou *patch* seja colocado em produção somente se houver uma rotina para desinstalação.

O uso de terceiros no desenvolvimento de sistemas é uma fonte potencial de problemas com a segurança. Para efeito de minimizar ao máximo tais problemas, as regras e os procedimentos, que dizem respeito à segurança, devem ser incluídos no contrato com o terceiro, sendo este obrigado a respeitar todas as políticas de segurança associadas ao desenvolvimento de software.

No caso de sistemas adquiridos de terceiros, devem ser incluídas nos contratos duas cláusulas de total importância:

- ✓ 1ª - O terceiro não pode vender ou redistribuir os programas nem a documentação desenvolvidos para a empresa. Para haver exceções a esta regra, deve existir uma aprovação formal da diretoria.
- ✓ 2ª O terceiro deve manter os programas fontes e a documentação em custódia e a empresa poderá ter acesso aos mesmos, caso o terceiro venha a encerrar suas operações.

Independentemente de quem tenha feito o desenvolvimento, o resultado final é propriedade da empresa. O sistema contém os processos de negócio e outras informações sobre como a organização opera. Estes programas devem ser considerados como um bem da empresa. Deve haver uma política de propriedade intelectual, compatível com a lei, que garanta a propriedade dos sistemas desenvolvidos.

3. Avaliação do ambiente de desenvolvimento e estratégias de segurança

Não é possível gerar uma aplicação segura em um ambiente não seguro.

O primeiro passo para o desenvolvimento de uma aplicação segura, será o levantamento e avaliação do ambiente no qual esta aplicação será implantada. Verifica-se as ameaças, os pontos críticos, os ativos valiosos, legislações e salvaguardas já existentes no ambiente (ALBUQUERQUE, RIBEIRO – 2002).

Devemos implantar as defesas necessárias e algumas requeridas pela legislação ou pela política de segurança da empresa.

Não podemos esquecer dos sistemas de apoio (sistemas operacionais, banco de dados) que possuem várias características de segurança implementadas. Estes sistemas podem se transformar em armadilhas, se usadas pura e simplesmente, é claro que devem ser usadas, já que não devemos reinventar o inventado, mas é preciso saber quando e o que usar. Usá-las somente porque estas existem, sem haver uma necessidade detectada, pode trazer alguns prejuízos para o desempenho e principalmente brechas de segurança na

aplicação. O correto é levantar o que realmente precisamos, para então decidirmos como atender as necessidades, seja através dos sistemas de apoio ou implementando um novo mecanismo de segurança.

Quatro aspectos devem ser considerados durante o levantamento:

- ✓ Política de segurança (diretrizes, normas, legislações);
- ✓ Ameaças (ativos, mecanismos de ataque e agentes) ;
- ✓ Objetivos de segurança (necessidades do usuário formalizada); e
- ✓ Premissas (considerações sobre o uso do sistema e de seu ambiente)

Devemos primeiro levantar a política de segurança e as ameaças, para então fornecermos subsídio à definição dos objetivos de segurança. Assim, após a definição destes, levanta-se o que já é fornecido pelo ambiente, ou seja, os sistemas de apoio.

3.1. Levantamento da política de segurança

A maioria das empresas possui normas relativas a segurança, privacidade, confidencialidade e diversos outros aspectos de segurança. Antes de tudo, estas são as necessidades que precisam ser levantadas, uma vez que são requisitos do sistema que não podemos alterar (ALBUQUERQUE, RIBEIRO – 2002).

O trabalho pode ser facilitado se a empresa cliente já tiver uma política de segurança definida (ALBUQUERQUE, RIBEIRO – 2002).

O retorno deste levantamento poderá ser uma lista vazia, quando não existir uma política de segurança definida ou uma legislação aplicável, caso contrário retornará uma lista com identificadores que referenciam os aspectos de segurança, os itens que indicam os requisitos de segurança e a descrição desses itens. Esta lista servirá de base para a definição dos objetivos de segurança.

3.2. Levantamento das ameaças

Em virtude da evolução da tecnologia, torna-se impossível levantarmos todas as possibilidades de ameaças. Por isso não devemos querer que uma aplicação tenha defesa para todos os tipos de ameaças. Isso poderá torná-lo falho ou pesado demais. Uma aplicação deverá ter defesas para um grande número de ameaças, pois esta estará com certeza protegida contra muitas outras que não tenham sido levantadas, pois a maioria das ameaças tendem a utilizar os mesmos princípios para realizarem seus ataques.

As ameaças quase sempre possuem as mesmas características: um ativo com valor (tabelas com números de cartões de créditos), um mecanismo de ataque (apoderar-se da conta de um administrador), e um agente (um hacker). Então **AMEAÇA=AGENTE X MECANISMO X ATIVOS.** Faltando um desses itens não haverá ameaças (ALBUQUERQUE, RIBEIRO – 2002).

3.2. 1 Os agentes

De quem se defender? Esta é a pergunta que constantemente fazemos, e a resposta geralmente é: dos "***hackers***". Não é apenas deles que devemos nos proteger, pois as pessoas que estão envolvidas com o sistema, também podem ser grandes ameaças.

Os agentes são classificados de acordo com algumas categorias:

- ✓ Acesso ao sistema

  Quanto maior o acesso, mais fácil será de realizar o ataque. Veja algumas categorias de ataque. Tabela 1 (vide anexo)
- ✓ Conhecimento do sistema

  Quanto maior o conhecimento que se tiver sobre o sistema, mais fácil será de realizar o ataque. Tabela 2 (vide anexo).

- ✓ Capacidade do agente

  É claro que um *hacker* é um agente extremamente perigoso. Porém um usuário comum também pode ser muito perigoso se tiver um amplo acesso e conhecimento do sistema. Tabela 3 (vide anexo).

- ✓ Motivação do agente

  Esta característica não deve ser considerada diretamente em sua especificação de segurança, pois podem ser inúmeras. Porém poderão tornar-se uma informação importante futuramente. Tabela 4 (vide anexo).

- ✓ Classificação dos agentes

  Na especificação de segurança do sistema, utilizamos essa informação para traçar as estratégias a cada ameaça. Tabela 5 (vide anexo).

3.2. 2 Os Mecanismos

Existem diversos mecanismos conhecidos para explorar a vulnerabilidade de um sistema, e muitos irão surgir (ALBUQUERQUE, RIBEIRO – 2002).

Os mecanismos podem ser de alto nível (apoderar de conta de administrador) ou de nível operacional (usar uma aplicação "X"). Devemos nos ater as de alto nível, pois o estudo dos mecanismos de nível operacional, se tornará pouco produtivo, em função da dinâmica do surgimento de novas aplicações.

Destacamos alguns mecanismos na Tabela 6 (vide Anexo).

3.2. 3 Os Ativos

São todas as informações com alguma importância em seu sistema, que se tornem interesse dos agentes.

Veja alguns ativos e sua relevância aos agentes na Tabela 7 (vide anexo).

Existem vários aspectos em um ativo, de interesse de um atacante:

- ✓ Confidencialidade

  Se um agente consegue ler uma informação já obteve ganho, pois só o conhecimento do conteúdo desta informação representa uma perda para o sistema.

- ✓ Integridade

  Para sucesso de um agente, ele precisa alterar ou remover um dado do sistema.

- ✓ Disponibilidade

  Do que adianta todos os recursos possíveis, se estes não tiverem disponíveis quando necessário.

- ✓ Autenticidade

  É a garantia de que o usuário é realmente quem diz ser.

- ✓ Privacidade

  Quando alguém consegue monitorar as ações de um usuário, que deveriam ser privadas.

3.2. 4 Tabela de ameaças

É constituída a partir da ligação entre os ativos e os agentes, constantes na tabela de ativos, definidos quais mecanismos os agentes deveriam utilizar para atingir determinado ativo. Devemos levar em conta a capacidade do agente, seu conhecimento do sistema, o acesso que ele possui, além de sua motivação, para constatar se este realmente tem como utilizar o mecanismo pressuposto (Tabela 8 - Anexo).

3.3 Objetivos de segurança

Levantar as ameaças e necessidades legais, mas estas não oferecem todos os objetivos de segurança, sendo alguns de exclusividade do cliente.

As listas de objetivos de segurança previamente acertadas com o cliente, são a base para toda a segurança a ser implantada no sistema.

3.4 Premissas de segurança

São itens a serem considerados na segurança externa do sistema, utilizados para atender diretamente algum objetivo de segurança ou

alterar sua necessidade (ALBUQUERQUE, RIBEIRO – 2002).

A ISO 15.408, estabelece que as premissas apenas devem ser usadas para indicar as condições prévias do ambiente do sistema. Sendo posteriormente definido os atributos extra–sistema , para indicar aspectos de segurança já atendido pelo ambiente (ALBUQUERQUE, RIBEIRO – 2002).

Basicamente as premissas são de dois tipos: as de uso do sistema e as do ambiente. Como exemplo das premissas de sistema, podemos definir que o sistema somente será usado por um administrador treinado, assim eliminando os objetivos de segurança que tinha como ameaça o desconhecimento do sistema.

As premissas do ambiente dizem respeito ao ambiente esperado pelo sistema. Por exemplo: "os computadores nos quais irão rodar o sistema, não terão um *drive* de disquete". Assim, eliminamos a possibilidade de se usar discos de inicialização, pois este disco poderia burlar as funções de segurança do sistema operacional (ALBUQUERQUE, RIBEIRO – 2002).

Essas premissas devem ser aprovadas pelo cliente, pois afetarão diretamente a operação do sistema (ALBUQUERQUE, RIBEIRO – 2002).

As premissas irão dar origem a duas tabelas distintas, onde a primeira listará todas as premissas com um identificador, a segunda confronta estas com os objetivos de segurança.

Estas listas devem definir claramente as premissas que foram adotadas e como afetam os objetivos de segurança Tabelas 9 e 10 (vide anexo).

4. Proteção dos dados

A função básica da proteção de dados e do controle de acesso é de garantir a confidencialidade e disponibilidade das informações armazenadas (ALBUQUERQUE, RIBEIRO – 2002).

Além do controle de acessos, outros controles são usados para garantir a proteção dos dados, tais como: controle de fluxo de informação, canais de comunicação, informação residual.

A função de controle de acesso, define o que determinado usuário pode acessar ou alterar.

Devemos atentar para alguns problemas práticos no controle de acesso. Como por exemplo garantir que nenhum dos inúmeros meios de acesso a informação, viole o controle de acesso estabelecido.

Esta questão pode facilmente ser resolvida, aproximando a proteção e a informação. Ou seja o controle de acesso a essas informações fica a cargo do banco de dados.

4.1 Política de controle de acesso

É a primeira linha de defesa da aplicação. Sua função primordial é ditada pela necessidade do usuário. Desta forma a fase de especificação do sistema é o melhor momento para definição da política de acesso, sendo realizada em conjunto com o usuário (ALBUQUERQUE, RIBEIRO – 2002).

Quase todos os sistemas necessitam de algum mecanismo de controle de acesso.

De acordo com a ISO 15.408, um atributo de segurança só deve ser usado com o objetivo de atender a determinado objetivo de segurança *(*ALBUQUERQUE, RIBEIRO – 2002).

Geralmente o controle de acesso se liga aos objetivos de segurança nas seguintes formas:

- ✓ Garantindo que os usuários não vejam ou alterem determinadas informações;
- ✓ Facilitar o uso pelo usuário final;
- ✓ Minimizar a necessidade de manutenção do sistema devido erro do usuário final.

Na maioria das vezes não é necessário associar um objetivo de segurança a alguma ameaça, bastando considerar simplesmente uma premissa do sistema, como por exemplo: "realizar o controle de acesso da informação, conforme a política de segurança da empresa".

4.2 Política de controle de fluxo de informação

Existem situações onde uma aplicação trata uma determinada informação e a repassa a outro sistema ou meio de armazenamento. Existindo a necessidade da definição do controle de fluxo informacional.

Devemos também nos preocupar com os fluxos ilícitos de informação, mesmo havendo o controle de acesso e do fluxo de informação, um usuário pode abrir uma conexão direta ao banco de dados através de um acesso oculto, contornando o controle de acesso e fluxo de informação, ou até mesmo roubar o HD.

Na tentativa de eliminarmos estes fluxos ilícitos, nos deparamos com dois problemas:

- ✓ Identificar todos os canais para acesso a informação;
- ✓ Identificar as formas de impedir, controlar e monitorar tais acessos.

O primeiro pode ser resolvido através do levantamento de todos os possíveis canais de acesso a uma informação.

O segundo problema, podemos limitá-los através da autenticação de dados, conseguindo detectar alterações nestes dados. Outra alternativa é diminuir o ciclo de vida da informação, com isso limitando os fluxos ilícitos. Limitar os canais de entrada do sistema, removendo placa de rede e dispositivos de disco flexível, mas sem dúvida alguma a melhor alternativa para evitar acessos ilícitos é através do uso da criptografia.

Geralmente o controle de fluxos ilícitos não são preocupação da maioria dos sistema, porém em arquiteturas em camadas, seu uso pode ser de crucial importância.

Este controle deverá ser usado aos seguintes objetivos de segurança.

- ✓ O acesso do Banco de dados, por qualquer caminho diferente do servidor da aplicação.
- ✓ Os clientes deverão apenas aceitar as mensagens vindas do servidores internos.

É bastante difícil controlar fluxos de informação de um sistema. Por isso é interessante o uso do controle de fluxos por subconjunto. Não há a necessidade de controlar tudo, pois alguns fluxos não são tão importantes assim, sendo mais prudente controlar por completo os fluxos importantes do que vários fluxos de forma incompleta.

4.3 Fluxo de Dados externo

Existem informações que precisam transitar ou serem armazenadas fora do escopo da aplicação.

Na importação dos dados externos é necessário definir os atributos de segurança, seja em sua importação ou criação no sistema, pois geralmente estas informações não possuem atributos de segurança.

As informações quando definidas com controles por níveis, deverão possuir o nível igual ou menor de confidencialidade do nível do usuário que gerou ou importou.

Quando o controle de acesso for discriminado, a informação herdará os direitos de acesso de sua classe.

Todas as entradas devem obedecer a política de segurança, pois de nada vale se a entrada de dados principal seguir a risca esta política de segurança, se existir outras que não siga esta política.

Na exportação de dados, apenas o usuário que ver o dado pode exportá-lo, se este sair sem atributos de segurança, fica o usuário responsável pela segurança do dado exportado, uma vez que o sistema não mais tem controle sobre ele.

Podemos tanto exportar como importar um dado com seus atributos de segurança, um exemplo disso é o *e-mail* .

Quando consideramos uma transação de dados mantendo seus atributos de segurança, precisamos proteger a confidencialidade, integridade e autenticidade da informação.

A confidencialidade pode ser garantida por criptografia, a integridade por um *hash* da informação e sua autenticidade, através

de assinaturas eletrônicas (ALBUQUERQUE, RIBEIRO – 2002).

Sempre que houver uma política de controle de acesso e de controle de fluxo, os atributos de importação e exportação deverão estar sem atributos de segurança, uma vez que estes precisam nascer com o dado, por isso devem serem removidos no momento de sua exportação (ALBUQUERQUE, RIBEIRO – 2002).

Ao exportar um dado para um armazenamento ou linha de comunicação fora de nosso controle, e importar de um outro ponto, é preciso usar a autenticação dos dados, proteção de confidencialidade e integridade, importação e exportação com atributos de segurança. Estas medidas visam primeiramente garantir ao sistema, a capacidade de gerar cópias de segurança e restaurá-la, mantendo sua confidencialidade e integridade (ALBUQUERQUE, RIBEIRO – 2002). Permite também ao sistema ser capaz de se comunicar com qualquer outro sistema com segurança de suas informações.

Segundo a ISO 15.408, a importação, exportação, autenticidade e proteção de confidencialidade e integridade, são atributos separados (ALBUQUERQUE, RIBEIRO – 2002*)*, uma vez que podemos utilizar um ou outro atributo sem a necessidade de uso em conjunto com os outros atributos.

É importante lembrar que usar apenas criptografia, não garante a confidencialidade da informação, pois mesmo que esta tenha sido implementada com algoritmo forte, sua chave pode ser quebrada se esta estiver guardada em local de fácil acesso. Essa confidencialidade é conseguida com a inclusão da autenticação através de assinaturas eletrônicas.

4.4 Informação residual

Um arquivo apagado pode em algumas situações ser recuperado. O mesmo acontecendo com blocos de memórias liberados.

Uma informação depois de descartada, esta continua lá, seja em memória, disco, cache de rede. Se caracterizando em informação residual, o problema está na possibilidade destas informações serem recuperadas por invasores.

Podemos resolver estes problemas escrevendo zeros sobre o bloco ou arquivo descartado, ou ainda zerar a cache depois de eliminar a informação.

É problema identificar onde está ocorrendo informação residual, a melhor alternativa é tratar o fluxo de informações confidenciais no sistema. O simples tráfego da informação até a máquina do usuário pode gerar problemas de informação residual e vazamento por intercepção na rede. Para evitar isso procura-se trazer do banco de dados para a estação local somente as informações solicitada e que tenha acesso. Outra situação é o caso de um sistema de transação eletrônica que deixa informações temporárias em arquivos, tais como número de cartão de crédito.

4.5 Manutenção de integridade de dados internos

A integridade dos dados é um dos aspectos principais da segurança de um sistema (ALBUQUERQUE, RIBEIRO – 2002).

A perda de integridade pode ocorrer por duas causas:

- ✓ Problema de hardware ou influência magnética;
- ✓ Perda de atomicidade de transações.

No primeiro caso pouco pode ser feito no desenvolvimento de sistema, limitando-se como medida a detecção e aviso do problema.

Quanto a perda de atomicidade de transações, devemos implementar na aplicação rotina capaz de desfazer o que foi realizado para uma determinada transação no momento que uma operação gerar um erro. Com isso o sistema voltará a condição anterior ao início da transação, mantendo a integridade das informações.

Lembramos que hoje, nos sistemas mais modernos, estas ações ficam por conta do

próprio SGBD (Sistema Gerenciador de Banco de Dados).

5. Auditoria

E um conjunto de funções que gravam e mantém uma trilha de ações realizadas no sistema, permitindo a análise e visualização destas ações, assim a possibilidade de identificar o que ou quem causou algum problema de segurança.

Simples de ser implementada, porém de difícil de projetada em um sistema, em função da diversidade de parâmetros que devem ser avaliados. Por exemplo quais ações devem ser registradas? Terei que registrar tudo? E se registrar, terei problema de espaço? E muito mais.

Devemos relevar o motivo pelo qual queremos auditoria, desconsiderando objetivos externos à segurança do sistema. Os objetivos da auditoria podem ser:

- ✓ Segunda linha de proteção

  Existe um responsável para um eventual problema, mesmo que o sistema já tenha um evento para evitá-lo.
- ✓ Melhoria do sistema

  Visa medir a funcionalidade do sistema para eventuais melhorias.
- ✓ Aumento do escopo

  Visa identificar ações, mesmo que válidas, que possam causar prejuízos ou exponha ativos de forma desnecessárias.
- ✓ Prevenção

  Ter conhecimento de tentativas de invasão ou ameaças que tentem fraudar os mecanismos de proteção do sistema.
- ✓ Política

  Atendimento a política de segurança.

5.1 Geração dos dados de auditoria

Registrar um grande número de eventos, torna a auditoria completa, porém torna o sistema mais lento e com necessidade de maior armazenamento, além de tornar a revisão das trilhas impossível.

Devemos sempre levar em conta o objetivo de nosso mecanismo de auditoria. A ISO 15.408 sugere vários eventos de auditoria de cada mecanismo de proteção ou atributos de segurança implementado no sistema (ALBUQUERQUE, RIBEIRO – 2002). Devemos, ainda atentarmos apenas para aqueles que dizem respeito aos pontos mais críticos do sistema.

É importante que façamos ligação com as ameaças, para atender aos demais objetivos de auditoria. Para atingirmos os objetivos de aumento de escopo de proteção e prevenção de ataques, é necessário considerarmos os seguintes critérios:

- ✓ Principais mecanismos utilizados pelas ameaças ao sistema;
- ✓ Ativos mais valiosos,
- ✓ Agentes mais capacitados;
- ✓ Itens definidos na política de segurança

Todo sistema que exige alto nível de segurança, principalmente no controle de acesso, precisa de auditoria. O desempenho do sistema precisa ser acompanhado, para que falhas sejam levantadas assim como identificar os usuários maliciosos.

Por fim, é importante salientar que, um sistema de auditoria é caro em sua implantação e diminui o desempenho do sistema (ALBUQUERQUE, RIBEIRO – 2002).

5.2 Análise automática da trilha

Como já vimos, o objetivo da auditoria é detectar as invasões no sistema. Por isso é imprescindível que a trilha de auditoria seja revisada periodicamente.

A revisão manual da trilha de auditoria é uma tarefa tediosa e bastante vulnerável a erros, uma vez que a maioria dos eventos são irrelevantes.

A automação da auditoria é altamente desejável, porém em função de sua alta complexidade de implementação, somente

deverá ocorrer se for realmente necessária. Estes mecanismos podem ser acionados quando:

- ✓ Um evento de auditoria com a finalidade de proteção de escopo, for executado;
- ✓ Um conjunto de eventos, mesmo que inofensivos, indicar tentativa de violação do sistema.
- ✓ Evento de monitoração de quebra do controle de acesso ocorrer.

Após a identificação de qualquer ocorrência, o sistema se manifesta de diversas formas, como por exemplo:

- ✓ Cria um registro em tabela de situações suspeitas;
- ✓ Mensagens ao administrador;
- ✓ *Shutdown* no sistema.

5.3 Armazenamento da trilha de auditoria

O armazenamento de trilhas, tendem a causar alguns problemas, como:

- ✓ A trilha não pode ser alterada por usuário comum(senão este poderia apagar todos os registro que o incriminasse);
- ✓ As trilhas precisam estar integras, mesmo em caso de ataque ou queda do sistema
- ✓ Falta de espaço para armazenamento das trilha.

Fica claro que deve ser preocupação constante do administrador de banco de dados, quanto a exaustão das trilhas.

Existem algumas alternativas como por exemplo, enviar automaticamente as trilhas mais antigas para mídia de armazenamento maior e de menor custo, porém deve existir a preocupação com a disponibilidade destas informações.

Deve-se também definir por quanto tempo estas informações devem estar disponíveis.

Não podemos esquecer que alguns sistemas operacionais, já possuem mecanismos de armazenamento de trilhas, portando devemos evitar a criação de mecanismos já existentes, e que realizam a sua função com eficiência.

6. Autoproteção

Geralmente os sistemas de segurança falham na segurança de suas funções de segurança. Desta forma se estas funções estiverem fora de ação os dados dos usuários estarão desprotegidos.

As funções de segurança do sistema podem ser atacadas em três pontos:

- ✓ Dados e atributos de segurança;
- ✓ Implementação das funções de segurança;
- ✓ Camada subjacente.

6.1 Dados e atributos de segurança

No momento que se perde o controle sobre os dados e atributos de segurança, as funções de segurança perdem sua utilidade.

Os mecanismos para proteção de dados de segurança, são complementares ao dos dados do usuário, ou seja, os mecanismos de segurança de dados dos usuários (controle de acesso, controle do fluxo de informação, importação e exportação de dados, e outros já vistos neste artigo), também se aplicam aos de segurança, devendo ser complementados pelos seguintes mecanismos:

- ✓ Exportação de dados de segurança

  Devemos prover mecanismos mais rigorosos para garantia da integridade, confidencialidade e disponibilidade das informações de segurança exportada (ALBUQUERQUE, RIBEIRO – 2002).
- ✓ Transferência interna de dados

  Deve-se criar critérios rígidos que regem a transferência interna de dados.
- ✓ Capacidade de recuperação

  O sistema deve ser capaz de retornar a um estado seguro mesmo após uma falha ou ações externas
- ✓ Sincronismo de estado da aplicação

Em aplicações distribuídas, é importante gerar mecanismos de garantia de recebimento de informação, mantendo todo o sistema consistente (ALBUQUERQUE, RIBEIRO – 2002).

- ✓ Consistência de dados replicados dentro da aplicação

  Ocorre quando alguns atributos de segurança são mantidos replicados nos clientes por motivo de desempenho ou para garantir a segurança quando o sistema estiver off-line (ALBUQUERQUE, RIBEIRO – 2002).

- ✓ Consistência de dados com funções externas de segurança

  Na maioria das vezes, os sistemas de segurança são compostos por códigos do sistema e do sistema operacional, sendo comum a troca de informação entre o sistema desenvolvido e o sistema operacional. Cabe lembrar que atributos de segurança fora do padrão, também pode se configurar como uma tentativa de invasão pelo sistema operacional. Por isso é conveniente que o sistema deva tratar como erro, todo atributo externo que ele não consiga interpretar (ALBUQUERQUE, RIBEIRO – 2002).

6.2 Proteção das funções de segurança da aplicação

É de vital importância a proteção da aplicação, uma vez que dependendo do sistema operacional usado e a forma de instalação, pode haver alterações indevidas no código executável dos programas

Os sistemas seguros devem possuir um mecanismo de garantia de integridade do sistema, onde este faça verificações, seja na inicialização ou em intervalos periódicos, para garantir que nenhuma alteração tenha ocorrido. Isso pode ser feito através de uma verificação de CRC, ou utilizando redundância de códigos (ALBUQUERQUE, RIBEIRO – 2002).

Deve existir também a preocupação com a parte física, já que o agente pode abrir o gabinete e trocar o HD, e reiniciar o sistema, burlando todo mecanismo de segurança.

Muitas vulnerabilidades surgem por estouro de área de armazenamento, para isso temos como solução a separação de domínios. Existem algumas arquiteturas que permitem, a separação de áreas de dados das áreas de código, assim resguardando a segunda.

A separação de domínio pode ser feita também na forma de armazenamento externo, com controles adicionais.

Outro aspecto de grande relevância, é a verificação de integridade dos canais de acesso ao sistema.

6.3 Proteção contra falhas da camada subjacente

É muito fácil fraudar um sistema, se a base a qual ele roda está comprometida.

Devemos minimizar ao máximo a possibilidade de se contornar o processo normal de inicialização, ou seja impedir o boot pelo drive A: ou substituição do HD, execução por terminais remoto entre outros.

7. Privacidade

A privacidade passou a ser um requisito muito importante após o surgimento da Internet.

Ela pode existir de diversas formas, conforme a necessidade da política de segurança ou da privacidade do sistema (ALBUQUERQUE, RIBEIRO – 2002):

- ✓ Invisibilidade

  O sistema garante a um usuário que os demais tomem conhecimento de seu acesso.

- ✓ Não-rastreado

  O usuário não é identificado no sistema, nem seus passos monitorados, porém pode ser responsabilizado por seus atos, quando usado em conjunto com o pseudônimo.

- ✓ Pseudônimo

Muito utilizado quando há a necessidade de responsabilização do usuário e a privacidade ao mesmo tempo (ALBUQUERQUE, RIBEIRO – 2002). O usuário é identificado e autenticado pelo sistema, mas sendo referenciado por um pseudônimo para cada ação realizada.

A ligação do usuário com seus pseudônimos, somente ocorrerá em situações especiais.

- ✓ Anonimato

  Não existe identificação nenhuma do usuário.

Por fim a privacidade surge por influência da política preestabelecida, de uma solicitação do usuário ou até da legislação, além de ameaças que exigem certo grau de privacidade (ALBUQUERQUE, RIBEIRO – 2002*)*, pois uma vez um usuário tenha seus passos rastreados por outros, fica fácil obter informações sobre ele.

8. Gerenciamento da segurança

Na implementação de uma aplicação segura, torna necessária a existência de funções e interfaces que controlem seus atributos e dados (ALBUQUERQUE, RIBEIRO – 2002).

O gerenciamento de segurança, visa definir os usuários, dividindo em grupos ou papéis, cada um com seus direitos definidos.

O sistema deve ser capaz de identificar cada usuário em cada situação, relacionando-o a seu grupo, e definindo se este terá ou não os direitos reservados a seu grupo.

O gerenciamento ainda deve ser capaz de revogar e expirar atributos de segurança, sendo uma questão de suma importância, como exemplo se o sistema encontra-se sob ameaça através do uso indevido de uma conta, esta deve ser imediatamente revogada, ou seja ter seus direitos totalmente bloqueados. A revogação de direitos, pode ocorrer de várias formas: Na próxima vez que o usuário logar ou através de uma função de verificação, que teste os direitos de tempos e tempos, assim permite a revogação durante este intervalo.

A expiração de direitos ocorre num momento anteriormente estabelecido, temos como exemplo os certificados digitais, que são concedidos, já com um prazo de validade estabelecido. Sendo muito utilizado também em contas de funcionários contratados para serviços temporários, as quais são criadas com data de validade definida.

Enfim, é muito importante o controle de vigência dos direito de usuários sobre o sistema.

9. Testando a segurança

É a etapa final do desenvolvimento de um sistema. Em função de sua importância este teste muitas das vezes se confunde com controle de qualidade.

O principal objetivo dos testes é garantir que a aplicação seja fiel a todos os seus requisitos de segurança.

A ISO 15.408, conta com quatro famílias de garantia de segurança:

- ✓ Cobertura de testes

  Composta por uma família de componentes, que levanta se o testes realizados são suficientes para garantir a funcionalidade prevista do sistema.

  - ✓ Profundidade dos testes

    Esta família de componentes atua no detalhamento da funcionalidade de segurança, baseado no aumento da profundidade das informações utilizadas na análise. Tendo como objetivo impedir algum erro e detectar códigos maliciosos inseridos durante o desenvolvimento.

  - ✓ Testes funcionais

    São executadas geralmente pelos desenvolvedores, buscando garantir que as funções de segurança retornem propriedades necessárias que satisfaçam aos requisitos funcionais (ALBUQUERQUE, RIBEIRO – 2002*)*.

- ✓ Testes independentes

  É a terceirização do teste, ou seja pessoas que não participaram do desenvolvimento, que buscam falhas cometidas pelos desenvolvedores, pois estes tendem a cometê-los, por serem conhecedores da estrutura de funcionamento da aplicação.

10. Avaliação de vulnerabilidades

Tem como finalidade garantir a resistência do sistema contra ameaças do ambiente

A avaliação toma como base a lista de ameaças.

Em sua fase de teste, esta avaliação pode ser feita pela equipe de desenvolvimento, porém o resultado terá maior garantia se esta for realizada por uma equipe independente. Seja quem for, é imprescindível que a equipe tenha um conhecimento compatível com os padrões dos ataques testados (ALBUQUERQUE, RIBEIRO – 2002).

A avaliação deve ser realizada após a aplicação estar pronta para entrega.

Para sucesso da avaliação, deve-se levar em conta o levantamento de alguns elementos:

- ✓ Ambiente que a aplicação será implantada;
- ✓ O nível de conhecimento dos principais agentes de ataque;
- ✓ Toda documentação disponível, inclusive as premissas do ambiente levantado.

Nesta fase, a análise deve ser extremamente detalhista, pois qualquer resultado que contraste, mesmo que um pouco do esperado, pode ser sinal de uma vulnerabilidade.

É importante lembrar que devemos ter cuidado com o fato desta análise ser uma das últimas tarefas a ser realizada antes da entrega, em função disto, ela pode receber pressões de prazo de entrega, assim torna interessante que esta análise seja realizada por terceiros, uma vez que estes serão menos influenciado pelo fator tempo.

Os testes podem ser realizados utilizando de várias técnicas, como por exemplo: fornecendo um número maior de caracteres esperado pelo sistema, assim causando um estouro de campo de entrada. Podemos ainda usar *plics*(‘) ou outros caracteres estranhos, que podem fazer com que o sistema retornem mensagens de erro, onde estas mostram ao atacante uma brecha a ser explorada. Muitas outras podem ser usadas, pois existem inúmeras a disposição e que com certeza surgirão.

O importante e ressaltar que devemos verificar o maior número de possibilidades de ataque, mesmo aquelas menos prováveis, pois assim estaremos dando a garantia de segurança pretendida pela aplicação desenvolvida.

11. Segurança na programação Web

Enquanto as atenções estão voltadas para os problemas de segurança de redes, outro problema de grande gravidade é deixado de lado: falhas de programação capazes de expor um site na *web*.

Estas falhas possibilitam que qualquer amador consiga entrar em sites, para isto basta usar um browser e o *notepad*, não precisando ser um grande *hacker* com muitos conhecimentos e técnicas.

Tudo se inicia na caixa de *login* e senha. É necessário que a aplicação pegue o *login* e senha digitados e pesquise-os no banco de dados. Existe uma regra básica para evitar falhas de segurança: “jamais faça uma pesquisa no banco de dados com algo que o usuário informou sem trocar primeiramente os caracteres reservados”. O não seguimento desta regra leva à algoritmos falhos, do tipo: “vai-se ao banco de dados pesquisando pelo nome e senha, se houver registro de retorno o nome e senha é válido e o usuário está logado”.

Digitando, na caixa de *login*, um apóstrofo e clicando no botão para seguir, podemos ter várias

reações:

Uma mensagem de erro gerada pelo próprio site.

Essa é a melhor reação possível, quando bem utilizada e quando não fica inútil. O fato da própria aplicação no site gerar a mensagem de erro significa que, se é que houve um erro ocorreu um tratamento e o pretenso invasor ficará sem saber o que ocorreu em conseqüência da entrada dele na caixa de *login*.

Alguns sites têm o costume de dar mensagens diferentes para cada tipo de erro, tal como: "usuário inválido", "senha inválida" e "erro desconhecido". Neste caso o procedimento de tratar o erro é inútil, pois a mensagem de erro personalizada está informando ao invasor que a tentativa dele gerou algum resultado.

O ideal é dar uma mensagem padrão, por exemplo "*login* inválido" , para qualquer erro que possa ocorrer durante o processo de *login*. Desta forma o invasor, a princípio, não saberá se as tentativas dele geraram algum resultado ou não, dificultando a ação do pretenso invasor. (TORRES, 2003)

Mensagem do servidor: Caracteres inválidos no login.

Apesar de funcionar e parecer a solução do problema, está não é a melhor forma de se tratar os apóstrofos. Se a mensagem for disparada a partir do servidor significa que o algoritmo de login testou o login e identificou que haviam caracteres inválidos, não indo ao banco de dados. Isto fará com que o invasor de imediato desista do seu site, mas saiba que seu site é mal programado.

Mensagem do Cliente: Caracteres inválidos no *login*.

Em menos de cinco minutos o invasor edita o código fonte, elimina a validação e invade o site. Manter validação apenas no cliente é o mesmo que nada. Se a validação estiver tanto no cliente como no servidor o invasor vai perder tempo a toa. Mas para sorte deste, são poucos os desenvolvedores que tem um verdadeiro conhecimento da arquitetura *web* para desenvolver desta forma.

11.1 Solucionando o problema

Quando lidamos com caracteres reservados de uma linguagem devemos pesquisar de que forma eles poderiam ser representados. O SQL não é uma exceção.

O apóstrofo, caracter reservado no SQL, para ser representado precisa ser dobrado. Assim quando o banco de dados encontra dois apóstrofos dentro de uma *string*, entende que o que se pretende é inserir ou consultar apenas um e tudo funciona corretamente. Em ASP podemos utilizar a instrução *replace* para dobrar os apóstrofos digitados pelo usuário.

Desta forma não fará diferença para o código o que for digitado, o código fará a busca do que o invasor digitar como se fosse um nome de usuário e o resultado será "usuário inválido". Por isso que quando o invasor recebe esta mensagem ou continua recebendo uma mensagem de erro padrão, ele desiste do site, pois percebe que o site fez o tratamento dos apóstrofos corretamente.

12. Conclusão

Este artigo procurou alertar da importância da segurança no desenvolvimento de um sistema. Salientando técnicas de eficiência comprovada internacionalmente, que garantem a obtenção dos resultados pretendidos no que diz respeito a segurança.

Esperamos que seu conteúdo seja de grande valia aos profissionais da área de desenvolvimento e aos interessados pelo assunto, que em conjunto com outras obras sobre o assunto, possam solidificar a prática destas técnicas, que sem dúvida maximiza a segurança de uma aplicação.

Referências

ALBUQUERQUE, Ricardo; RIBEIRO, Bruno. Segurança no Desenvolvimento de

Software. **1. Ed. Editora Campus, Rio de Janeiro, RJ, 2002**

MARTINS, José Carlos Cordeiro. **Gestão de Projetos de segurança da Informação.** 1. Ed. Editora BRASPORT Livros e Multimídia Ltda, Rio de Janeiro, RJ, 2003

TORRES, Dennes. **Segurança Máxima de Software: Como desenvolver Soluções Seguras.** 1. Ed. Editora BRASPORT Livros e Multimídia Ltda, Rio de Janeiro, RJ, 2003

Anexos

Tabela 1 - Acesso ao sistema

Categoria	Descrição
Programador	Acesso eventualmente total ao código do sistema
Administrador	Total acesso à operação e administração do sistema
Usuário do sistema	Acesso a algumas funções do sistema
Acesso não-autorizado	Pessoas, em princípio, sem acesso ao sistema

(ALBUQUERQUE, RIBEIRO – 2002)

Tabela 2 – Conhecimento do sistema

Categoria	Descrição
Grande conhecimento	Usuário poderosos, administradores
Algum conhecimento	Usuários ou ex-usuários do sistema
Acesso a manuais	Embora sem conhecimento do sistema, tem acesso a manuais

(ALBUQUERQUE, RIBEIRO – 2002)

Tabela 3 – Capacidade do agente

Categoria	Descrição
Especialista	Grande experiência em ataques. Usa ferramentas sofisticadas
Conhecedor	Alguma experiência em ataques. Ferramentas mais simples
Curioso	Ferramentas básicas de ataques
leigo	Nenhum conhecimento

(ALBUQUERQUE, RIBEIRO – 2002)

Tabela 4 – Motivação do agente

Categoria	Descrição
Financeira	Interesse em retorno financeiro
Imagem	Interesse em destacar suas habilidades
dano	Vingança ou prejuízo a empresa que tenha trabalhado ou concorrente
Aprendizado	Estudo de ferramenta de ataque

(ALBUQUERQUE, RIBEIRO – 2002)

Tabela 5 – Classificação dos agentes

Agente	Acesso	Conhecimento	Capacidade	Motivação
Usuário do sistema	3	2	3	1,2,3 ou 4
Administrador do Sistema	2	1	2	1,2,3 ou 4
Programadores	1	1	2	1,2,3 ou 4
Funcionário	3	2	2	1 ou 3
Ex-funcionário	4	2	2	1 ou 3
hacker	4	3	1	1 ou 2

(ALBUQUERQUE, RIBEIRO – 2002)

Tabela 6 – Mecanismos de ataque

Mecanismo	Descrição
Abuso do poder	Acesso legal sobre o sistema para realizar um ataque
By-pass de controle de acesso	Utilização de funções externas visando burlar o controle de acesso.
Força bruta	Quebra de senhas e criptografia
Estouro de buffer	Destrutivo, porém de fácil correção pela equipe de desenvolvimento
Caracteres inesperados	Uso de caracteres com a finalidade de confundir o sistema

(ALBUQUERQUE, RIBEIRO – 2002)

Tabela 7 – Ativos com valor

Ativo	Agente	Valor
Confidencialidade da tabela de clientes	Ex-funcionários ou concorrente	Dano à empresa
Integridade da tabela de contas a pagar	Hacker, funcionários, usuários, administrador ou programador	Valor financeiro

(ALBUQUERQUE, RIBEIRO – 2002)

Tabela 8 – Lista de ameaças consideradas

Ameaça	Agente	mecanismo	Ativo
Obtenção de lista de clientes	Hacker, Ex-funcionário ou concorrente	Exploração de vulnerabilidade co-nhecida, caracteres inesperados	Confidencialidade da lista de clientes
Falha no sistema impede seu funcionamento	programador	Abuso do poder	Integridade e disponibilidade da base de dados,

(ALBUQUERQUE, RIBEIRO – 2002)

Tabela 9 - Premissas

Identificador	Premissa
PA.AutentSO	O Sistema Operacional deverá autenticar o usuário antes da execução da aplicação.
PA.AdminSO	Somente o Administrador do SO, poderá alterar senhas e criar novos usuários
PU.TreinAdmin	O Administrador deve consultar o RH antes de cadastrar novos usuários

(ALBUQUERQUE, RIBEIRO – 2002*)*

Tabela 10 – Objetivos de segurança x premissas

Objetivo de segurança	Premissa	Explicação
O.Confidencialidade	PU.CadastroUsuario; PAAutentSO;PU.TreinAdmin	A aplicação utilizará autenticação do SO para garantir a autenticação do usuário na aplicação.
O.ControleAdmin	PU.AdminAuditoria	Existe a figura do administrador de auditoria

(ALBUQUERQUE, RIBEIRO – 2002)