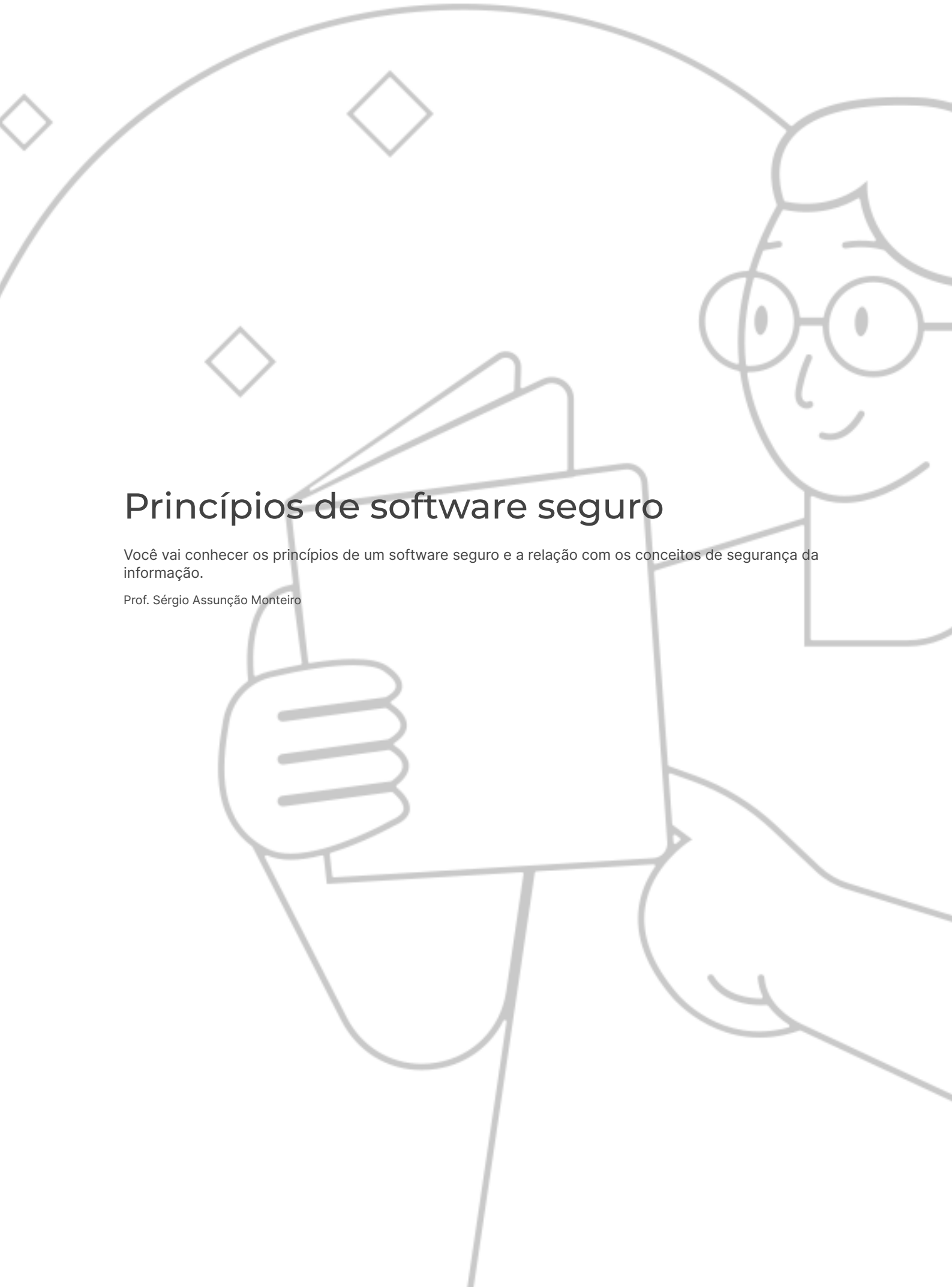


Princípios de software seguro

Você vai conhecer os princípios de um software seguro e a relação com os conceitos de segurança da informação.

Prof. Sérgio Assunção Monteiro

Propósito

Com a popularidade de áreas como a ciência de dados, aprendizado de máquina, Big Data e inteligência de negócios, houve um aumento significativo de garantir a segurança dos dados, afinal de contas, eles são a matéria-prima para essas áreas. Nesse sentido, é importante conhecer os vários aspectos que devem estar presentes em um software seguro, pois é um assunto extremamente atual e muito valorizado no mercado de trabalho.

Objetivos

- Reconhecer a relação entre os princípios de segurança e o software seguro.
- Identificar os princípios de design seguro de software.
- Reconhecer os elementos da cadeia de suprimentos.

Introdução

Cada vez mais, geramos e consumimos dados. O simples ato de fazer uma compra, podendo ser on-line ou em uma loja, já produz uma grande quantidade de dados que podem revelar informações sobre nossas preferências e que podem ser explorados por profissionais de vendas para incentivar o aumento do consumo personalizado. Em outras situações, os dados podem trazer informações importantes sobre a situação de empresas que podem beneficiar algumas pessoas que tenham acesso a eles e prejudicar muitas outras.

Então, não há dúvidas de que os dados são valiosos. E o que fazemos com coisas valiosas? Protegemos para que sejam utilizadas apenas quando for necessário e para alcançarmos objetivos específicos. Nesse sentido, a segurança da informação tem um papel crucial. Mas aí vem outra questão: quem é que, de fato, torna operacional o gerenciamento da segurança dos dados? A resposta é: os diversos aplicativos de softwares.

Então, fica bastante claro que precisamos garantir que um software tenha elementos de segurança que possam ser verificados, para que possa ser chamado de seguro. É exatamente esse assunto que vamos abordar neste estudo. Aqui, teremos a oportunidade de conhecer os diversos aspectos que relacionam segurança ao desenvolvimento, operação e monitoramento de software. Sem dúvidas, é uma das áreas mais “quentes” do momento e que exige muito estudo da nossa parte, mas que é muito valorizada no mercado.

Aspectos essenciais entre princípios de segurança e software seguro

Olá! Neste vídeo, veremos como a implementação desses princípios pode ajudar a prevenir vulnerabilidades e garantir a proteção dos dados e sistemas contra ameaças cibernéticas.

> Conteúdo interativo
>
> Acesse a versão digital para assistir ao vídeo.

Facilmente, reconhecemos a noção de uma situação segura ou não. Por exemplo, você deixaria um objeto pessoal de valor de forma que qualquer um pudesse ter fácil acesso? Obviamente, não. Então, por que você deixaria os dados do sistema que você opera para qualquer um ter acesso?

Provavelmente, você tem cuidado com os dados, mas, certamente, deve conhecer muitas pessoas que ainda não têm cuidados básicos. É aqui que entramos com os aspectos essenciais de segurança e como eles se relacionam com um software seguro.

Software seguro.

Basicamente, os princípios de segurança são aspectos fundamentais bem conhecidos (trataremos um pouco mais adiante) que são usados para projetar e implementar sistemas seguros. Portanto, um software seguro é um elemento lógico – normalmente, chamado de sistema – que foi projetado e desenvolvido para garantir proteção contra ameaças e ataques.

Nesse sentido, precisamos entender a relação entre princípios de segurança e software seguro como a relação entre um projeto de arquitetura de um prédio e sua materialização que, no nosso caso, é o desenvolvimento de software seguro.

Os princípios de segurança são a base estrutural para identificar possíveis ameaças e vulnerabilidades e projetar recursos para reduzir a exposição aos riscos. De forma resumida, o software seguro deve ser construído sobre os princípios de segurança.

Os princípios de segurança fundamentais para o desenvolvimento de software seguro são:

Confidencialidade

Significa que apenas pessoas autorizadas podem ter acesso aos dados.

Integridade

Garante que os dados não sejam modificados ou adulterados de maneira não autorizada.

Disponibilidade

Garante que os sistemas e dados estejam acessíveis a usuários autorizados sempre que precisam utilizá-los.

Autenticação

Garante a identificação dos usuários de um sistema antes de conceder acesso a dados.

Autorização

Garante que os usuários tenham as permissões necessárias para acessar dados ou sistemas específicos.

Ou seja, a palavra mais repetida nos princípios de segurança foi “garantia”. E a ideia é exatamente esta: oferecer garantias sobre identificação, acesso e manipulação dos dados. Neste tópico, vimos a relação entre os princípios de segurança e um software seguro. Nos próximos, vamos explorar cada um desses elementos com o objetivo de criar softwares mais seguros e, portanto, menos vulneráveis a ataques.

O essencial da confidencialidade

Olá! Neste vídeo, falaremos sobre a importância da confidencialidade na proteção de dados sensíveis, os diferentes tipos de informações que devem ser mantidos em sigilo e as técnicas utilizadas para contribuir no desenvolvimento de um software seguro.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

Se a pessoa não tem autorização para acessar ou manipular um dado ou sistema, então deve existir um recurso que dê essa garantia. O nome desse recurso é **confidencialidade**. Ela é um dos elementos básicos de um software seguro, pois auxilia na proteção de dados importantes. Muitas vezes, os dados são classificados de duas formas:

Sensíveis

Podem incluir informações pessoais como nomes, endereços, números de identidade ou CPF, números de cartão de crédito e registros médicos.

Confidenciais

Referem-se a qualquer dado que deva ter acesso restrito. Normalmente, são associados a segredos comerciais, informações financeiras ou projetos de empresas e governos.

No contexto de software seguro, conseguimos garantir a confidencialidade de medidas de segurança, tais como:

1
Criptografia
Utiliza algoritmos específicos para transformar os dados em um formato ilegível que só pode ser decifrado usando uma chave de criptografia.

2
Controle de acesso
Faz a limitação do acesso aos dados utilizando o princípio do menor privilégio, ou seja, o comportamento padrão é que os perfis dos usuários tenham acesso mínimo aos dados.

3
Mascaramento de dados
São técnicas que trocam dados importantes por falsos.

Se um software não consegue garantir a confidencialidade, ele está colocando todos os dados do sistema em risco. Isso é uma seríssima vulnerabilidade e pode trazer terríveis consequências, como roubo de identidade, violação de dados, perdas financeiras e usar uma identidade legítima para cometer crimes.

A garantia da integridade

Olá! Neste vídeo, abordaremos a definição de integridade de dados, a importância de mantê-los íntegros, os principais riscos e ameaças à integridade e as técnicas e medidas que podem ser adotadas para o desenvolvimento de um software seguro.

Conteúdo interativo
Acesse a versão digital para assistir ao vídeo.

Quando você escuta a palavra íntegro, lembra de quê? Provavelmente, de algo que não foi violado, ou de uma pessoa que não é corrupta. A mesma coisa se aplica para a integridade do software seguro. Ela é traduzida, na prática, como um conjunto de recursos capazes de evitar que os dados sofram alterações ou modificações sem a devida autorização. Em outras palavras, a integridade vai proteger os dados de qualquer modificação indevida.

Conceito de ataque a um software.

Essas modificações, quando ocorrem, são feitas através de programas mal-intencionados ou de ataques direcionados. Infelizmente, há situações em que o responsável pelo software facilita a vida dos atacantes, pois não utilizam nenhum recurso que limite o acesso aos dados, nem mesmo tem uma política de backup. Mas aí vem a pergunta: como garantir a integridade dos dados?

Existem algumas ações que podemos fazer para garantir a integridade do software. Alguns exemplos concretos são:

1

Assinaturas digitais

É uma técnica que valida a autenticidade e a integridade de um documento, mensagem ou software. Uma boa comparação é com uma assinatura manuscrita com reconhecimento em cartório, só que muito mais segura.

2

Assinatura de código

É um método que utiliza uma assinatura digital em um programa, arquivo, atualização de software ou executável com o objetivo de garantir sua autenticidade e integridade durante a instalação e execução.

3

Controle de versão

Fazem o controle das alterações de código do software no local onde eles ficam hospedados. Um exemplo de um repositório muito conhecido com essa finalidade é o GitHub. Esses repositórios possuem recursos para que os desenvolvedores identifiquem quaisquer modificações ou alterações.

4

Práticas de codificação segura

Existem várias questões que devem ser tratadas em um software. Por exemplo: validação de entrada; tratamento de explícito de exceções; verificação de limites de memória para evitar ataques de estouro de buffer; controle contra ataques de injeção de SQL e controles de injeção de código de um modo geral.

Tanto as assinaturas digitais como a assinatura de código nos ajudam a garantir que o software seja íntegro. No caso das assinaturas digitais, elas utilizam mecanismos de criptografia para verificar a identidade do desenvolvedor de software. Já a assinatura de código usa os certificados digitais para vincular uma assinatura digital a um código específico. Interessante, não é?

Atualmente, está cada vez mais comum utilizarmos esses recursos. Isso nos ajuda a ganhar tempo, economizar recursos financeiros e, principalmente, aumentar a integridade dos dados e do software. No nosso próximo tópico, vamos abordar o aspecto de disponibilidade.

A importância da disponibilidade

Olá! Neste vídeo, veremos como a disponibilidade é um dos pilares fundamentais da segurança da informação, garantindo que os sistemas e dados estejam sempre acessíveis aos usuários autorizados. Abordaremos as

principais ameaças à disponibilidade e as técnicas e medidas que podem ser adotadas para contribuir para o desenvolvimento de um software seguro.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

Você já ouviu falar na expressão 24 por 7? Ela significa trabalhar 24 horas por dia durante os sete dias da semana. Mas, isso realmente funciona na prática? Bem, todo software precisa ter um plano de manutenção planejada, caso contrário, vai estar exposto a uma manutenção corretiva que é uma situação muito mais complicada. Por isso mesmo, precisamos estabelecer diretrizes que reduzam as chances de que situações desastrosas ocorram, como, por exemplo, as regras para acessar dados.

As regras para acessar os dados de um software seguro precisam ser bem estabelecidas. Vamos ver algumas dessas regras.

1

Quais os dados que o software dá acesso.

2

Quais são os perfis que podem acessar os dados.

3

Quais são as ações associadas a cada perfil.

4

Quais são os horários que os dados podem ser acessados.

5

Quais são os horários que os dados podem ser modificados.

6

Qual a política de backup para garantir que – na eventualidade de um desastre – os dados estejam disponíveis novamente.

É um fato muito importante ter consciência de que a indisponibilidade de um software pode causar sérios danos financeiros, de reputação, queda de produtividade e, em alguns casos, a exposição do risco da vida das pessoas.

Exemplo

Imagine os problemas causados quando os sistemas de controle de trânsito falham. Agora – no caso de uma situação mais extrema ainda –, quais são as consequências quando um sistema que leva oxigênio para mineradores de regiões subterrâneas falha? Ou seja, são situações que exigem cuidados extremos.

Agora, vamos pensar um pouco sobre as causas que podem levar um sistema à indisponibilidade.
Entre elas estão:

- Falhas de hardware;
- Problemas de conectividade de rede;
- Bugs de software;
- Ataques de negação de serviço (DoS).

Em especial, o ataque DoS é bastante sofisticado e pode ser usado em dois níveis. No primeiro nível, os invasores sobrecarregam o sistema com um excesso de solicitações, tornando-o inacessível para usuários legítimos. Enquanto os técnicos ficam focados tentando corrigir esse problema, os invasores utilizam outros tipos de ataques para roubar ou adulterar dados. Então, nesse momento, já deve estar bem claro para nós a importância de garantir a disponibilidade de um software. Mas, o que podemos fazer para garantir essa disponibilidade?

Para garantir a disponibilidade do software, as pessoas e organizações de um modo geral – sejam públicas ou privadas – devem implementar medidas, tais como:

1

Redundância de dados.

2

Versões mínimas de um software para garantir o funcionamento das atividades mais importantes.

3

Planos de recuperação de desastres.

4

Balanceamento de carga para evitar pontos únicos de falha.

5

Realizar testes regulares.

6

Fazer o monitoramento da operação do sistema por meio da análise dos dados de log.

7

Ter um plano de manutenção de sistemas bem definido.

8

Realizar manutenções de software com regularidade.

Portanto, existem caminhos que podemos tomar para garantir a disponibilidade de um software. A questão é realizarmos um equilíbrio entre benefícios e custos. Se a disponibilidade do software é fundamental, então, devemos investir nos métodos que deem essa garantia.

A seguir, vamos estudar mais dois aspectos importantes para ter um software seguro: autenticação e autorização.

A importância da autenticação e autorização

Olá! Neste vídeo, abordaremos a diferença entre autenticação e autorização, a importância de cada uma dessas etapas no controle de acesso aos sistemas e dados, e as técnicas e medidas que podem ser adotadas para contribuir na construção de um software seguro.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

Os conceitos de autenticação e autorização são diferentes, mas estão fortemente relacionados. A autenticação é usada para determinar a identidade do usuário. Ou seja, ela trata do processo de verificação de um usuário ou sistema que tenta acessar um recurso ou um conjunto de dados. Existem diversas técnicas para isso. Algumas delas são:

- Uma combinação de nome de usuário (login) e senha.
- Autenticação biométrica que pode usar a digital, por exemplo.
- Métodos de autenticação multifator: essas são mais elaboradas. Por exemplo, o usuário entra com login e senha para acessar o e-mail, mas precisa passar por uma etapa de validação de um número de celular pré-cadastrado para verificar se, de fato, a identidade é dele mesmo.

Portanto, o objetivo da autenticação é bem simples: garantir que apenas usuários legítimos possam acessar o sistema e seus recursos.

No caso da autorização, o objetivo é determinar quais são os recursos ou funções que um usuário legítimo pode acessar dentro de um sistema.

> Um software seguro utiliza a autorização associada ao perfil do cliente ao invés do indivíduo em si, pois é mais fácil gerenciar direitos e impor limites.

Essas duas características – **autenticação** e **autorização** – são dois elementos que devem compor um sistema de software seguro, pois restringem o acesso de quem pode utilizar o sistema e trabalhar com os dados, além de restringirem o que o usuário pode realizar dentro do sistema.

Dessa forma, vimos que os princípios de segurança são fundamentais para garantir que um software seja seguro. No próximo módulo, nós vamos estudar sobre os princípios de design seguro, ou seja, dos aspectos que devem fazer parte da arquitetura do software. Antes de avançar, faça os exercícios para fixar o conhecimento e, se for necessário, retorne para o texto para entender melhor algum conceito.

Verificando o aprendizado

Questão 1

A noção básica de segurança é um aspecto natural. Em especial, na era em que vivemos, com os grandes avanços da tecnologia e dos algoritmos, os dados, de um modo geral, ganharam grande importância. Nesse sentido, selecione a opção correta a respeito da relação entre os princípios de segurança e software seguro.

A

A segurança é um princípio inerente aos dados e, portanto, é natural que qualquer software implemente técnicas específicas de proteção contra ataques.

B

A proteção dos dados é responsabilidade exclusiva dos usuários de softwares seguros.

C

Só é possível garantir que um software é seguro quando conseguimos constatar a implementação de, pelo menos, um princípio de segurança.

D

Devido à dinâmica que ocorre no mundo digital, é impossível constatar se um software é ou não seguro.

E

Um software seguro restringe o acesso e manipulação aos dados apenas para quem, de fato, deve ter esses direitos.

A alternativa E está correta.

Os princípios de segurança tratam sobre as diversas técnicas e atitudes que devemos usar para garantir a proteção das pessoas e de bens materiais. Então, quando nos referimos a dados, estamos tratando de tecnologia digital que, por sua vez, faz uso de programas que devem implementar os princípios de segurança para protegê-los.

Questão 2

Uma das consequências da popularização da área de ciência de dados foi a conscientização de que os princípios de segurança da informação devem fazer parte dos softwares. Um desses princípios é o da confidencialidade. Nesse sentido, selecione a opção correta a respeito do princípio da confidencialidade.

A

A única forma efetiva de impedir o acesso indevido aos dados é por meio da conscientização de todos os usuários de um sistema.

B

É fundamental tratar todos os dados com o mesmo nível de importância para reduzir riscos.

C

Uma das técnicas usadas para garantir a confidencialidade é a criptografia.

D

Qualquer dado deve ser tratado como confidencial.

E

A forma mais simples de garantir a confidencialidade dos dados é por meio do controle de acesso.

A alternativa C está correta.

O princípio da confidencialidade consiste em se utilizar dos meios necessários para que apenas pessoas autorizadas tenham acesso aos dados. Portanto, um software seguro deve utilizar técnicas que envolvam configurações e algoritmos para impedir o acesso indevido. Entre essas técnicas, estão os algoritmos de criptografia.

Elementos fundamentais do design de um software seguro

Olá! Neste vídeo, veremos os princípios e boas práticas do design seguro, os riscos e ameaças mais comuns durante o processo de design, e as técnicas e ferramentas que podem ser utilizadas para garantir a segurança desde a fase de concepção do software.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

Os usuários de aplicativos utilizam programas prontos para atender às suas necessidades. No entanto, para chegar até esses usuários, os programadores precisam garantir que as funcionalidades estejam funcionando corretamente e que o sistema não tenha vulnerabilidades. Essa segunda parte, em especial, é muito mais desafiante do que parece!

Software vulnerável.

Uma vulnerabilidade nem sempre se trata de um erro de software. Pode ser, inclusive, que ela nunca se manifeste. No entanto, é um risco enorme deixá-la dentro de um sistema, seja de forma proposital – que é uma situação muito estranha! – ou por falta de planejamento de testes. É aí que entram os elementos fundamentais do design de um software seguro.

Quando falamos sobre design de software seguro, nos referimos ao conjunto de diretrizes que os desenvolvedores usam para projetar e desenvolver um sistema para reduzir as vulnerabilidades. Os principais princípios de design de software seguro são:

Privilégio mínimo

Significa que os usuários ou, ainda, os perfis dos usuários devam ter apenas os privilégios essenciais para realizar suas tarefas.

Defesa em profundidade

Utiliza várias camadas de segurança para proteger contra diferentes tipos de ataques. Essas camadas incluem gerenciamento de acesso, acesso condicional, autenticação por multifator, gerenciamento dos privilégios pela identidade e proteção de identidade.

Padrões à prova de falhas

Utiliza configurações padrões seguras que não dependem do usuário.

Separação de funções

É essencial fazer a segmentação de um sistema em camadas. Sendo que cada camada tem um conjunto de funções. Um dos modelos mais conhecidos de arquitetura de software é o MVC. Mais à frente, vamos abordá-lo com mais profundidade.

Security by Design

É a aplicação de boas práticas de segurança em todas as fases do ciclo de vida do desenvolvimento de software.

Economia de mecanismo

Simplificar a configuração da arquitetura do software para reduzir o potencial de erros e vulnerabilidades difíceis de serem detectadas.

Design aberto

Significa que a arquitetura do sistema possa ser revisada por outras pessoas do projeto. É uma péssima prática deixar a responsabilidade do projeto da arquitetura de um sistema para apenas um indivíduo.

Tratamento de erros

É responsabilidade dos desenvolvedores criarem mecanismos para detectar exceções e tratá-las explicitamente. Os erros “raízes” – erros da execução do programa – nunca devem ser expostos para o usuário final, pois podem ser usados para explorar vulnerabilidades.

Os princípios de design de software seguro são excelentes direcionadores, para que os desenvolvedores possam criar softwares mais confiáveis e menos vulneráveis a ataques. Nos próximos tópicos, vamos explorar com mais detalhes esses princípios.

Fundamentos do privilégio mínimo de um software seguro

Olá! Neste vídeo, abordaremos o conceito de privilégio mínimo, a sua importância na proteção dos sistemas e dados contra ameaças cibernéticas, e as técnicas e medidas que podem ser adotadas para implementar essa prática no desenvolvimento de software seguro.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

Como podemos desenvolver um software capaz de reduzir os danos que um invasor pode causar ao se passar por um usuário legítimo? Bem, uma das formas de conseguir esse resultado é conceder privilégios mínimos para cada perfil.

O princípio do privilégio mínimo nos orienta a fornecer apenas o nível essencial para que o usuário realize o trabalho dele. O objetivo aqui é reduzir o risco de danos que um invasor – ser humano ou sistema – possa causar. Mas há outro benefício também no caso de o sistema ser comprometido: ser capaz de medir os danos.

> O ideal é sempre agir com proatividade, mas nem sempre isso é possível. Às vezes, só conseguimos atuar de forma reativa, mas, até mesmo para isso, é importante ter um caminho a seguir. Como sabemos de antemão quais são os privilégios de um determinado perfil, podemos criar mecanismos que bloqueiem a execução das atividades.

O privilégio mínimo é importante para segmentar as atividades e, assim, impor restrições. Um exemplo disso ocorre com um processo de servidor da web que pode apenas fazer leitura de arquivos, ou seja, não tem acesso de gravação. Na prática, isso minimiza o risco de modificações não autorizadas. Mas, como podemos implementar o privilégio mínimo em um sistema?

Para implementarmos o privilégio mínimo em um sistema, precisamos ter um mapa com todas as funções e responsabilidades associadas a cada perfil de usuário. Ou seja, precisamos saber:

- Quais são os processos envolvidos?
- Quais os recursos e funções utilizados nesses processos?

A ideia é atender à necessidade do usuário por meio de um perfil e, a partir disso, traçar os possíveis caminhos que ele pode seguir. Assim, podemos fazer um monitoramento da operação do sistema e observar se há comportamentos inadequados ou privilégios que não condizem com um determinado perfil.

Resumindo

O princípio do privilégio mínimo é mais um aspecto de design de software seguro, pois nos auxilia a reduzir riscos de danos e, no caso de uma invasão, nos orienta a como agir para interromper o ataque. No próximo tópico, vamos conhecer um pouco mais sobre dois princípios de design de software seguro: defesa em profundidade e padrões à prova de falhas.

Defesa em profundidade e padrões à prova de falhas

Olá! Neste vídeo, abordaremos os princípios e benefícios da defesa em profundidade, bem como os padrões à prova de falhas, que ajudam a evitar vulnerabilidades e falhas de segurança em sistemas e softwares.

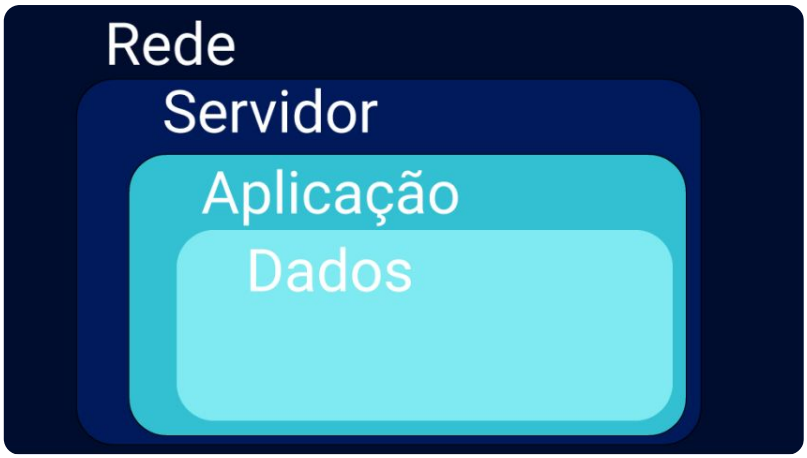
Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

Esses dois princípios de projeto aumentam a segurança de um software. A ideia básica é trabalharmos com camadas semelhantes à criação de trincheiras para dificultar o avanço de um invasor. A seguir, apresentamos os dois princípios com mais detalhes.

Defesa em profundidade

É uma estratégia de segurança que implementa várias camadas de controles de segurança e também contramedidas para reduzir danos. Essas camadas podem incluir uma visão mais externa do sistema – a infraestrutura de rede – como também o servidor, a aplicação em si e os dados que são acessados pelo sistema. Abaixo, apresentamos uma figura que nos dá uma ideia da defesa em profundidade.



Defesa em profundidade.

Os padrões à prova de falhas

Tratam das diversas formas para prevenir ou, pelo menos, reduzir o impacto de possíveis falhas ou erros. Entre essas técnicas, estão:

- Mecanismos para detecção de erros.
- Mecanismos para correção de erros.
- Sistemas de backup e recuperação de dados.
- Redundância de sistemas.
- Técnicas de tolerância a falhas.

Juntos, os dois princípios dificultam a vida dos possíveis invasores. Por outro lado, precisamos ter cuidado para não inserir vulnerabilidades no sistema, pois, agora, ele está ficando mais complexo.

Separação de funções e Security by Design

Olá! Neste vídeo, abordaremos os conceitos de separação de funções e Security by Design, bem como os benefícios e desafios de implementar essas práticas em sistemas e softwares.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

Novamente, voltamos para a segmentação de funções. Agora, no entanto, utilizamos os princípios de separação de funções e segurança por design. Vamos ver cada um deles com mais detalhes.

Separação de funções

Trata da divisão das principais tarefas e responsabilidades entre diferentes indivíduos ou equipes para reduzir as chances de que um invasor possa usar a identidade de um usuário legítimo e ter controle ou acesso a informações confidenciais. Portanto, a ideia é diminuir o poder dos perfis do usuário dentro do sistema.

Security by Design

Significa aplicar as boas práticas de segurança no sistema ao longo de todo o ciclo de desenvolvimento.

É importante notarmos que esses princípios não fazem parte do processo “natural” de desenvolvimento. Os desenvolvedores precisam ser orientados a trabalhar dessa forma desde o começo do projeto e ser conscientizados sobre a importância desses aspectos de segurança. Por isso mesmo, é importante investir tanto na formação técnica como na explicação para os gestores sobre as implicações de negócio associados a um sistema seguro.

Economia de mecanismo, design aberto e defesa contra erros

Olá! Neste vídeo, abordaremos os conceitos de mecanismos de segurança, design seguro e defesa contra erros, bem como os benefícios e desafios de implementar essas práticas em sistemas e softwares.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

Não há dúvidas de que a segurança é um aspecto essencial para um software. O problema disso é que o efeito colateral de implementarmos esses princípios no software é aumentarmos as possibilidades de que existam vulnerabilidades. Parece contraditório, não é mesmo? Mas basta pensarmos que estamos acrescentando elementos que terão de se comunicar por meio de protocolos e que ninguém terá conhecimento total do sistema. Por isso, precisamos fazer um bom planejamento antes de iniciar uma implementação.

Agora, vamos analisar três princípios que auxiliam a manter o software mais seguro, ao mesmo tempo que tentam reduzir as vulnerabilidades devido à complexidade:

1

Economia de mecanismo

São práticas de implementação que desencorajam ataques ou acessos não autorizados. Por exemplo, quando fazemos a implementação de controles de acesso, mecanismos de auditoria e monitoramento e criptografia, estamos aumentando as dificuldades de um invasor.

2

O design aberto

O objetivo aqui é oferecer transparência do código-fonte e, claro, da documentação para outros desenvolvedores. Em especial, para os que têm perfil de desenvolvimento de segurança com o objetivo de revisar e identificar possíveis vulnerabilidades no sistema.

3

A defesa contra erros

Trata-se de mapear o funcionamento do sistema e identificar possíveis falhas. Depois disso, precisamos tratar essas exceções explicitamente e jamais exibir uma mensagem de erro do sistema para o usuário final. Também faz parte desse princípio a utilização de sistemas de backup e recuperação, redundância e princípios de projeto tolerantes a falhas.

Todos os princípios que vimos auxiliam a desenvolver sistemas mais seguros, pois fazem parte da arquitetura do software, ou seja, estão ligados ao projeto e, por isso mesmo, precisam ser bem documentados e testados, pois serão a base para a implementação de outros programas.

No nosso próximo módulo, vamos estudar sobre a cadeia de suprimentos de um software. Ou seja, vamos estudar sobre o impacto dos componentes de terceiros para a implementação de um software seguro.

Verificando o aprendizado

Questão 1

É fato que existem diversos fatores que influenciam para obtermos um software seguro. Um desses fatores, talvez o mais fundamental de todos, está relacionado às decisões do projeto do software, ou ainda, do design do software. Nesse sentido, selecione a opção correta a respeito dos elementos de projeto de um software seguro.

A

Devem se limitar à aplicação de testes em todas as fases de desenvolvimento.

B

Estão focados na implementação de recursos que permitam monitorar o uso do software e como e por quem os dados estão sendo acessados e manipulados.

C

São compostos exclusivamente por decisões técnicas que visam reduzir a exposição a riscos.

D

Fazem a concentração de toda a responsabilidade da segurança dos dados para uma equipe focada exclusivamente na configuração dos mecanismos de controle e acesso aos dados.

E

É composto por um conjunto de práticas que implementam os princípios da segurança, como também permitem monitorar o processo de desenvolvimento e operação do software.

A alternativa E está correta.

Um projeto para a construção de um software seguro envolve decisões sobre restrições de acesso, medidas protetivas e de contra-ataques, utilização de boas práticas, redução de complexidade do projeto, além de outras medidas que se somam para mitigar riscos.

Questão 2

Uma das formas de mitigar riscos em um software seguro é por meio da implementação da política de privilégio mínimo. Apesar de a ideia ser simples, ela é extremamente efetiva na prática. Nesse sentido, selecione a opção correta a respeito da importância da política de privilégio mínimo para construção de um software seguro.

A

Aumenta a complexidade do projeto do sistema e, assim, dificulta a possibilidade de ataques.

B

Utiliza a aplicação de penalizações para os usuários com a redução de direitos dentro de um sistema.

C

Simplifica o gerenciamento das ações que um usuário pode realizar dentro do sistema.

D

Trabalha com a aplicação de algoritmos de criptografia para reduzir a velocidade com a qual um invasor pode acessar os dados.

E

É focado exclusivamente nos algoritmos que garantem a legitimidade de um usuário do sistema.

A alternativa C está correta.

A efetividade da política de privilégio mínimo está exatamente na simplicidade que ela utiliza para atribuir direitos aos perfis de usuários. Dessa forma, é muito mais eficiente monitorar o modo como os usuários estão operando um sistema. Além disso, no caso de um invasor conseguir se passar por um usuário legítimo, também é mais fácil mensurar danos e criar estratégias de contra-ataque.

Elementos da cadeia de suprimentos de um software seguro

Olá! Neste vídeo, veremos os principais desafios e riscos associados à cadeia de suprimentos de software, as melhores práticas para gerenciamento de fornecedores e a importância da verificação de segurança em todo o processo.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

A expressão “cadeia de suprimentos” é bastante utilizada em problemas de logística. Normalmente, ela está associada aos diversos processos que vão desde a extração da matéria-prima até a entrega de um produto para o consumidor final. Mas, o que isso tem a ver com software seguro?

Bem, no contexto de software seguro, a ideia é parecida. Precisamos lembrar que um software é composto por bibliotecas e componentes, além de usar ferramentas de desenvolvimento que são fornecidas por terceiros, ou seja, é uma cadeia de suprimentos tecnológica. Cada elemento dessa cadeia pode influenciar em relação a alguma vulnerabilidade de um sistema. É por isso que precisamos ter políticas de segurança que nos auxiliem a fazer avaliação desses itens que, também, incluem componentes de hardware.

Existem vários casos conhecidos de componentes com seríssimas vulnerabilidades. Alguns exemplos são:

1

Heartbleed

Trata-se de uma vulnerabilidade na biblioteca de criptografia OpenSSL. Ela facilitava a vida dos invasores de modo que pudessem ter acesso a dados confidenciais da memória de um servidor da web.

2

Spectre e Meltdown

São vulnerabilidades que tiveram grande impacto na maioria dos processadores modernos. Elas, simplesmente, possibilitavam que um invasor acessasse dados confidenciais da memória de um sistema.

3

Violação de dados da Equifax

Essa vulnerabilidade, em 2017, causou um prejuízo de, aproximadamente, um bilhão e meio de dólares para a empresa Equifax. A empresa sofreu uma violação de dados que possibilitou que os invasores fizessem a exposição de informações pessoais de milhões de clientes. E qual foi a causa desse problema? Devido a uma vulnerabilidade na estrutura do aplicativo web Apache Struts.

4Spyware Pegasus

É um malware desenvolvido pela empresa israelense NSO Group. Ele age de várias formas, mas, essencialmente, o objetivo é explorar vulnerabilidades em vários aplicativos de software para obter acesso ao dispositivo de um alvo e roubar dados confidenciais.

Esses são apenas alguns exemplos das muitas vulnerabilidades de softwares, mas, infelizmente, é claro que existem muitos outros componentes e recursos de terceiros que podem tornar um software inseguro.

Então, o que devo fazer para proteger um sistema de vulnerabilidades, de um modo geral?

Investir em treinamento e estudo direcionado.

Um exemplo concreto de treinamento e estudo direcionado é visitar os seguintes portais:

OWASP Top Ten

Contém as dez principais vulnerabilidades de aplicações web. Além disso, ele apresenta exemplo de como essas vulnerabilidades se manifestam na prática e o que devemos fazer para combatê-las.

National Vulnerability Database

Apresenta uma base muito extensa de vulnerabilidades de software conhecidas e o como nos proteger em relação a elas.

É importante ressaltar que, no mundo corporativo, esse tipo de estudo deve fazer parte de uma política de segurança formal. Nessa política, deve haver a descrição de procedimentos de estudos, monitoramento e testes que nos auxiliem a tratar com quaisquer riscos a que um sistema esteja exposto.

Vulnerabilidades dos componentes de um software

Olá! Neste vídeo, vamos discutir os aspectos influentes na vulnerabilidade de um software, que são fatores que aumentam a probabilidade de um software ser vulnerável a ataques cibernéticos. Abordaremos aspectos como a qualidade do código-fonte, a complexidade do software, a presença de bugs e vulnerabilidades conhecidas, entre outros fatores.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

Podemos encontrar diversos tipos de vulnerabilidades nos componentes de um software. Como muitas dessas vulnerabilidades são bem conhecidas e documentadas, se elas não forem tratadas rapidamente, podem ser exploradas por atacantes para roubar dados e violar a segurança dos sistemas. A seguir, apresentamos algumas das vulnerabilidades mais comuns.

1 Componentes desatualizados

É bastante comum que os fabricantes descubram vulnerabilidades nos componentes deles e disponibilizem correções. No entanto, é obrigação do usuário fazer a atualização. Aqui, cabe, ainda, uma ressalva: sempre faça atualizações apenas de fontes oficiais, pois um atacante pode tentar enganar um usuário com uma versão atualizada falsa.

2

Configuração inadequada

Sistemas mal configurados ou usados com configuração padrão podem facilitar ataques de modo que um invasor obtenha acesso não autorizado.

3

Injeção de código

Trata-se de um ataque no qual o invasor insere um trecho de código malicioso para ser executado por um programa legítimo. A vulnerabilidade mais conhecida desse tipo é a injeção de SQL que pode permitir que um ataque acesse informações ou assuma o controle do sistema.

4

Falhas de autenticação

Se a autenticação de um componente aceitar senhas fracas ou criar alternativas de autenticação, novamente essa vulnerabilidade pode ser explorada para permitir que um atacante tenha acesso não autorizado.

5

Quebra de autorização

Se o componente de software não validar as permissões de acesso de um usuário, isso pode ser explorado para permitir que um atacante cause danos no sistema ou nos dados.

6

Fragilidade de criptografia

Ocorre quando um componente não faz a criptografia dos dados. Nesse caso, se um atacante tiver acesso a eles, poderá manipulá-los com facilidade.

7

Não tratar a entrada de dados

Se os componentes não criticarem os dados de entrada, então o sistema estará exposto a diversos tipos de ataques, como a injeção de código SQL e de outros tipos de dados que podem gerar interrupções do funcionamento do sistema.

As maneiras que temos para reduzir as ocorrências dessas vulnerabilidades são:

- Implementar práticas de segurança durante todo o processo de desenvolvimento do software.

- Realizar testes de segurança com regularidade.
- Fazer atualizações de fontes seguras.
- Realizar configurações seguras.

Além disso, manter-se atualizado sobre notícias relacionadas à segurança nos fóruns na internet.

Avaliação e gerenciamento dos riscos

Olá! Neste vídeo, abordaremos os conceitos de avaliação e gerenciamento de riscos, bem como as etapas envolvidas nesses processos, incluindo a identificação, análise, avaliação e tratamento dos riscos.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

Como vimos, as vulnerabilidades podem ser a porta de entrada para ataques muito sérios. Por isso mesmo, precisamos ter um bom conhecimento sobre elas para identificá-las e, sempre que possível, eliminá-las. Mas, nem sempre temos a possibilidade de agir da forma ideal. Então, é fundamental que tenhamos um plano que nos permita fazer a avaliação e o gerenciamento de riscos relacionados aos programas que utilizamos.

Para conseguirmos uma identificação e redução de riscos efetivos, precisamos estabelecer processos que devem estar devidamente contextualizados. Por exemplo:

Software desenvolvido pela própria empresa	Software vem de outro fornecedor
Então é necessário aplicar as práticas seguras que envolvem, inclusive, um plano de testes.	Então precisamos conhecer quais são os planos de manutenção, correção e atualização.

Nos dois casos, precisamos de um processo formal que envolve treinamento, muito estudo e realização de simulações e testes de estresse. Ou seja, tudo isso tem o objetivo de identificar riscos potenciais, avaliar a probabilidade da ocorrência de um problema e quais são os impactos deles.

Em linhas gerais, podemos formalizar uma política de avaliação e gerenciamento de riscos de software por meio das seguintes etapas:

1

Identificação de riscos

Devemos listar explicitamente quais são os riscos potenciais que podem afetar o software. Dependendo do contexto, podemos tratar de riscos associados ao desenvolvimento, implantação e operação de software.

2

Análise de risco

Consiste em medirmos as probabilidades reais de um risco se tornar um problema e quais os impactos que esse problema pode trazer para nós, seja no contexto pessoal ou corporativo.

3 Priorização de riscos

Na sequência, precisamos estabelecer uma prioridade em relação aos riscos, apesar de que o ideal é eliminarmos a ocorrência por meio da aplicação de técnicas seguras, sempre que for possível.

4

Mitigação de riscos

Aqui é que entra a implementação de medidas para reduzir ou eliminar os riscos identificados. Entre as ações que podemos tomar, estão:

* A implementação de controles de acesso.
* Formalizar contramedidas para evitar que o risco ocorra ou minimizar seu impacto.

5

Monitoramento e revisão de riscos

Agora, precisamos fazer uma análise periódica do sistema em operação, para estudar o comportamento dele e, quando for necessário, atuarmos para combater ataques.

Em resumo, é fundamental termos processos formais da avaliação e do gerenciamento de riscos de software. Além disso, é muito importante atribuir níveis de responsabilidades para os indivíduos, pois, assim, não haverá dúvidas sobre quem deve tomar qual ação sempre que houver a necessidade de atuar para garantir a confiabilidade do software.

Práticas de codificação segura

Olá! Neste vídeo, veremos algumas das melhores práticas de codificação segura, como a utilização de validação de entrada, tratamento de exceções e gerenciamento de memória, entre outros aspectos importantes.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

Estamos preocupados em entender as melhores práticas que devemos implementar ou verificar para garantir que um software seja seguro. Para garantir a segurança, precisamos reduzir a ocorrência de vulnerabilidades e ter planos formais de como agir, caso ocorra algum problema. Aqui, podemos ver algumas das principais práticas de codificação segura:

Validação de entrada

Como já vimos, os invasores podem tentar explorar as entradas de um sistema. É por isso que precisamos realizar a validação da entrada antes do sistema executar os procedimentos dele.

Tratamento de erros

Muitas linguagens de programação oferecem recursos explícitos para o tratamento de exceções. O objetivo é garantir que o sistema permaneça seguro e estável, caso ocorra um erro. Além disso, ao tratarmos os erros de forma explícita, evitamos de fornecer informações sobre o software que possam ser usadas para explorar alguma vulnerabilidade.

Autenticação segura

Trata-se da utilização de técnicas que garantam políticas de senha fortes, autenticação multifator e armazenamento seguro de credenciais de autenticação.

Controle de acesso

São mecanismos associados a configurações de privilégios relacionados a perfis de usuário.

Comunicação segura

Aqui, usamos algoritmos de criptografia para garantir a segurança de qualquer comunicação entre o cliente.

Armazenamento seguro

Aqui, também precisamos criptografar os dados para impedir o acesso não autorizado, no entanto, o contexto é servidor de banco de dados.

Uso de funções especializadas em codificação segura

Qualquer funcionalidade de um sistema deve ter como base um componente confiável, ou seja, precisamos ter acesso a uma documentação de boa qualidade e ter um local oficial para realizarmos atualizações e informações.

Atualizações regulares

Como já vimos, essa prática envolve atualizar o software regularmente apenas com fontes oficiais para evitar vulnerabilidades e corrigir erros no sistema.

Em resumo, ao seguir essas práticas de codificação segura, estamos reduzindo as probabilidades de exposição ao risco.

Testes de vulnerabilidades do software

Olá! Neste vídeo, vamos explorar um exemplo de testes de vulnerabilidades do software, que são atividades realizadas para identificar possíveis falhas e vulnerabilidades em um sistema de software. Veremos alguns dos métodos e técnicas mais comuns para realizar testes de vulnerabilidades, bem como as ferramentas utilizadas nesse processo.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

O teste de vulnerabilidade de software é um processo de identificação de pontos fracos ou vulnerabilidades em software que podem ser explorados por invasores. Existem vários tipos de teste de vulnerabilidade que podem ser usados para identificar vulnerabilidades no software. Aqui estão alguns dos principais:

Teste estático de segurança de software

É um tipo teste que analisa o código-fonte – por isso é estático – para identificar possíveis vulnerabilidades de segurança. Basicamente, ele verifica o código em busca de padrões conhecidos de vulnerabilidades que podem ser explorados por invasores.

Teste dinâmico de segurança de software

É um tipo de teste que utiliza a execução do software para analisar o comportamento dele e tentar identificar possíveis vulnerabilidades. Aqui, precisamos enviar uma série de requisições ao software e analisar as respostas. Dessa forma, podemos analisar questões como tempo de resposta médio para a execução de uma tarefa.

Teste interativo de segurança de software

É um tipo de teste que combina os testes estáticos e dinâmicos para tentar identificar possíveis vulnerabilidades.

Teste de penetração

Nesse tipo de teste, fazemos a simulação de um ataque ao software para identificar possíveis vulnerabilidades. Como já estudamos, existem muitas vulnerabilidades conhecidas e são elas que esse tipo de teste tenta explorar para analisar o quão seguro é o software.

Teste Fuzzing

Ele gera dados de entrada aleatórios em grandes quantidades e os envia para o software para identificar possíveis vulnerabilidades, tais como estouros de buffer e ataques de injeção.

Por meio desses testes, podemos minimizar os riscos de ficarmos expostos a vulnerabilidades. Mas, obviamente, é necessário fazermos atualizações constantes, pois os invasores são criativos para explorar vulnerabilidades. Tão importante quanto todas as práticas, testes e técnicas que estudamos é a conscientização de gestores, usuários e desenvolvedores sobre a importância de desenvolver e usar um software seguro.

Verificando o aprendizado

Questão 1

Atualmente, é cada vez mais comum utilizarmos componentes de terceiros para construir nossos sistemas. Existem vários motivos que justificam esse comportamento, no entanto também há riscos envolvidos nele.

Nesse sentido, selecione a opção correta sobre os riscos de utilizar componentes de terceiros para construirmos um software seguro.

A

Apesar de aumentar os custos com o desenvolvimento, a única forma de garantir a segurança de um sistema é construir todos os componentes necessários.

B

A única forma de garantir que um componente de software é seguro é adquiri-lo de uma empresa conhecida no mercado.

C

A melhor forma de garantir a segurança de um componente de software é utilizar os que possuem código aberto e são livres de licenças.

D

É muito difícil encontrar informações sobre vulnerabilidades de software, então só devemos confiar nas que são dadas pelos fornecedores dos componentes.

E

É fundamental trabalhar com fornecedores confiáveis e que ofereçam manutenções e atualizações periódicas.

A alternativa E está correta.

Atualmente, é quase impossível construir um software sem utilizar componentes de outros fornecedores. Isso ocorre porque precisamos focar nossa energia produtiva no que faz parte da estratégia do negócio. Por isso mesmo, é fundamental que façamos uso de componentes apenas de fornecedores com boa reputação no mercado e que ofereçam planos de manutenção periódica e atualizações em locais oficiais para reduzir a exposição a riscos.

Questão 2

Garantir que um software seja, de fato, seguro é cada vez mais desafiante. Isso ocorre porque as aplicações estão ficando mais complexas para atender às demandas da sociedade em diversas áreas. Na prática, existem muitas vulnerabilidades conhecidas que podem estar em um sistema e, assim, comprometer a segurança que ele oferece. Nesse sentido, selecione a opção correta a respeito das vulnerabilidades dos componentes de um software.

A

É fundamental utilizar as configurações padrões iniciais de um componente para evitar vulnerabilidades.

B

Diminuir a complexidade de um sistema é a única forma que temos para reduzir a exposição a riscos.

C

Podem estar associadas a decisões de projeto, como a configurações inadequadas.

D

É importante evitar muitos testes durante o desenvolvimento e investir em um único teste completo, quando o sistema já estiver completo.

E

Um componente de software seguro só deve aceitar a entrada de dados de usuários legítimos para eliminar o risco de vulnerabilidades.

A alternativa C está correta.

Infelizmente, as vulnerabilidades de um sistema podem ter diversas origens, como em más decisões de projeto, uso inadequado do sistema e o não tratamento de ataques conhecidos. Por isso mesmo, é fundamental investir em treinamentos na área de segurança com foco na implementação de práticas seguras e na formalização de processos voltados para aplicação de testes, monitoramento de operação e em ações de prevenção e contra-ataque.

Considerações finais

Estudamos sobres os princípios de software seguro. Aprendemos que esses princípios devem estar presentes desde a concepção do projeto do software até o monitoramento da operação dele.

Vimos ainda que existem fontes oficiais que disponibilizam conteúdo sobre vulnerabilidades bem conhecidas que nos ajudam a aprofundar o nosso conhecimento técnico para identificá-las e utilizar recursos técnicos que vão do uso de ferramentas específicas à definição de processos para eliminá-las, quando possível, ou, pelo menos, reduzir os danos que elas podem causar.

Sabemos bem que a ciência e a tecnologia têm avançado a uma velocidade gigantesca. Hoje, temos acesso a diversas ferramentas e pacotes que podem ser usados para produzir coisas excelentes, como também podem ser utilizados para cometer crimes, pois, afinal de contas, são apenas ferramentas e técnicas. No centro de tudo isso sempre estará o ser humano. Por isso mesmo, é muito importante formar profissionais com forte base de conhecimento, estimular a proatividade e conscientizar sobre aspectos éticos. Dessa forma, além de alcançarmos nossos objetivos profissionais, também contribuímos para o bem-estar da sociedade e reduzimos os danos que podem ser causados por pessoas mal-intencionadas.

Explore +

Acesse o site oficial da Microsoft e procure por **Visão geral do desenvolvimento e das operações de segurança**. Lá, você aprenderá como uma das principais empresas de software do planeta trabalha para desenvolver softwares seguros.

Acesse o site oficial do OWASP. Essa é uma leitura obrigatória para um profissional que trabalha ou deseja trabalhar com desenvolvimento de software seguro. Procure pelo termo **OWASP Top Ten**.

Acesse o site oficial do Banco de Dados Nacional de Vulnerabilidades do governo dos Estados Unidos – em inglês: *National Vulnerability Database*. Essa é uma leitura muito útil para um profissional que trabalha ou deseja trabalhar com desenvolvimento de software seguro. Procure pelo termo **Vulnerabilities**.

Referências

CHESS, B.; WEST, J. **Secure Programming with Static Analysis**. Boston: Addison-Wesley, 2007.

HOWARD, M.; LEBLANC, D. **Writing Secure Code**. 2. ed. Microsoft Press, 2002.

MCGRAW, G. **Software Security**: Building Security In. Irving, TX: Pearson Education Inc., 2006.

PRESSMAN, R. S. **Engenharia de Software**. 6. ed. São Paulo: McGraw Hill, 2006.

SOMMERVILLE, I. **Engenharia de Software**. 6. ed. São Paulo: Pearson Addison Wesley, 2005.

VIEGA, J.; MCGRAW, G. **Building Secure Software**: How to Avoid Security Problems the Right Way. Boston: Addison-Wesley, 2002.