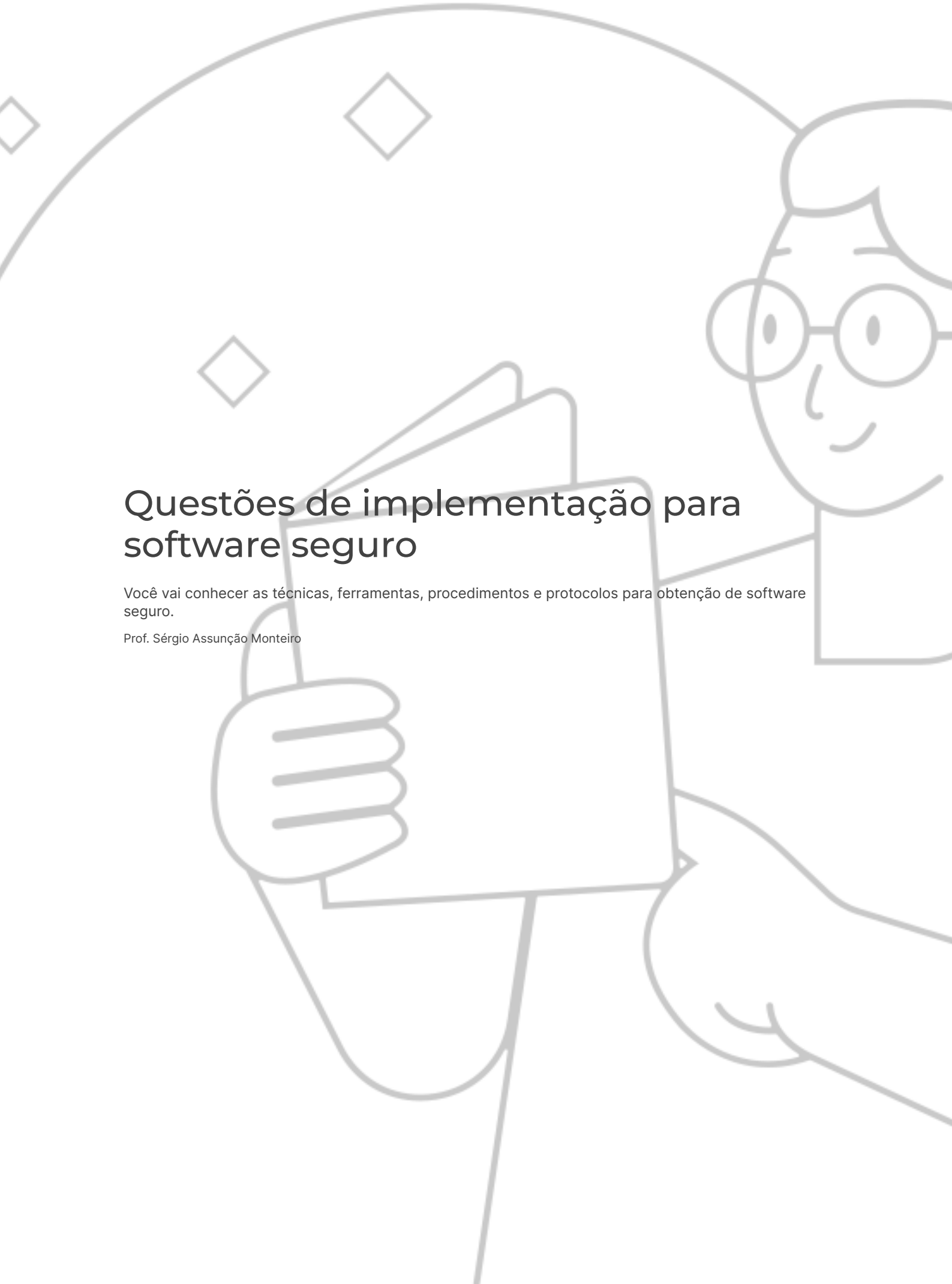


Questões de implementação para software seguro

Você vai conhecer as técnicas, ferramentas, procedimentos e protocolos para obtenção de software seguro.

Prof. Sérgio Assunção Monteiro

Propósito

O conhecimento das diversas técnicas, ferramentas, procedimentos e protocolos de implementação para um software seguro é importante para garantir a segurança dos dados, sendo esta uma das áreas com grande demanda no mercado, em especial, de empresas que usam dados estratégicos para realizar a tomada de decisão.

Objetivos

- Reconhecer a importância da validação de entrada.
- Analisar o processamento de dados com segurança.
- Identificar o processo de chamada de outros programas.
- Analisar o envio da saída.

Introdução

Se você utiliza um sistema bancário para gerenciar suas finanças, certamente não gostaria que outras pessoas tivessem acesso aos seus dados, pois isso poderia gerar grandes transtornos. Temos diversas outras situações nas quais precisamos de garantias de segurança. As aplicações que oferecem essa característica são chamadas de software seguro. No entanto, não é uma tarefa tão simples construir sistemas seguros.

Uma das grandes dificuldades para construir um sistema seguro é ter um conhecimento multidisciplinar. Ou seja, além de dominar uma linguagem de programação em si, é preciso também aplicar algoritmos de criptografia, protocolos de segurança e mecanismos de controle. Temos ainda que acompanhar os avanços na área de segurança e as atualizações feitas pelos fornecedores oficiais, sempre que estes detectam alguma vulnerabilidade nos sistemas ou fazem melhorias, para reduzir os riscos de invasões bem-sucedidas.

Um sistema, basicamente, tem quatro fases de tratamento dos dados: entrada, processamento, chamada a outros sistemas e geração da saída. Para cada uma dessas etapas, precisamos aplicar um tipo de tratamento que, novamente, envolve diversas ações. Além de esse assunto ser extremamente interessante, também vem sendo muito requisitado no mercado e gera alta empregabilidade. Em especial, com as novas tecnologias se popularizando e as técnicas de inteligência artificial se tornando mais presente nas nossas vidas, o conhecimento sobre software seguro é um grande diferencial para o profissional da área.

Fundamentos da validação da entrada

Veja uma breve introdução aos fundamentos da validação da entrada de um software seguro, destacando os seus principais aspectos.

Conteúdo interativo

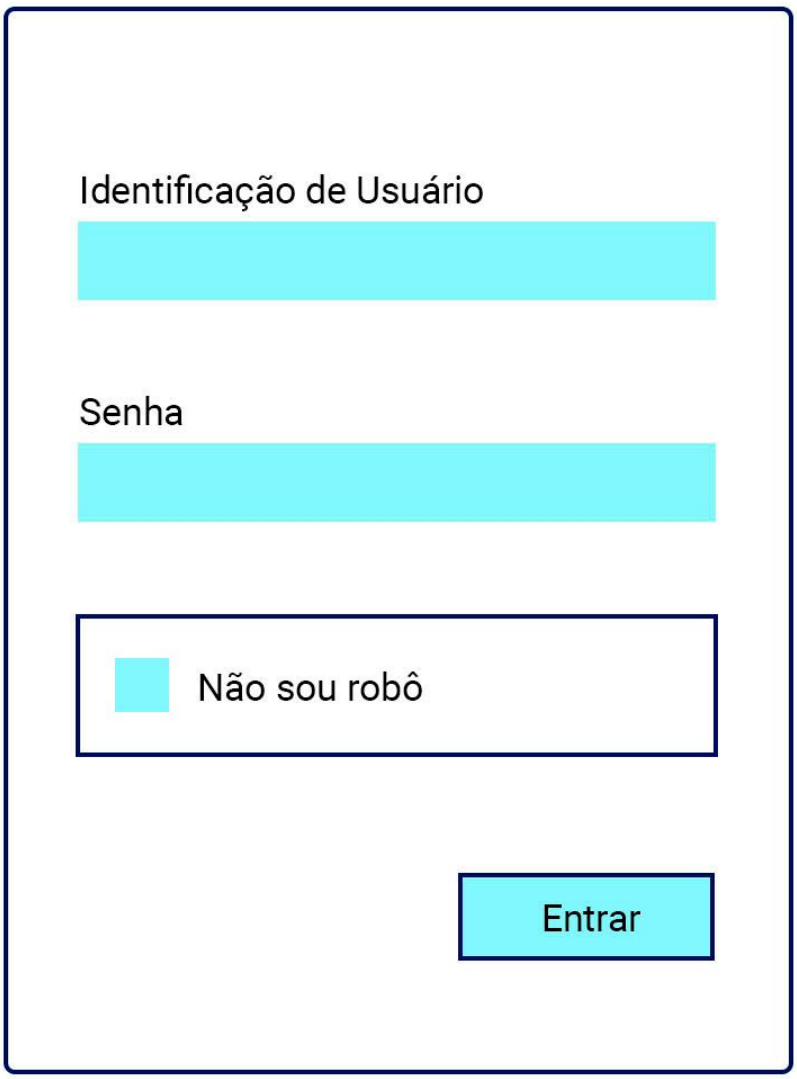
Acesse a versão digital para assistir ao vídeo.

A parte mais básica de qualquer software é o processo de entrada. Nos casos mais simples, precisamos realizar duas ações:

1° Entrar com o login de usuário válido.

2° Digitar a senha.

Depois de validar as credenciais do usuário, muitos sistemas fazem testes para verificar se é uma pessoa e não um robô (programa) que está tentando invadir o sistema. Vejamos um exemplo!



Protótipo do aspecto geral da etapa de entrada no sistema.

Além disso, alguns sistemas exibem diversas imagens para garantir que é um ser humano que está entrando no sistema. E ainda há casos mais elaborados! A validação segura de entrada de software é um aspecto crítico do desenvolvimento de aplicativos de software seguros.

Verificação e validação

Alguns autores distinguem verificação e validação. Observe!

Verificação	Validação
Processo de avaliação de um software para determinar se ele atende aos requisitos especificados e se foi implementado corretamente.	Processo de avaliação de um sistema para garantir que ele atenda às necessidades de seus usuários e às partes envolvidas.

Apesar de haver semelhanças entre os conceitos, ambos se diferenciam de uma forma muito simples: Aquele conceito que responde à pergunta “Fizemos o que foi solicitado pelo cliente do sistema?” é o de verificação. Mas, se a pergunta é “Desenvolvemos o software com a aplicação das melhores práticas?”, significa que estamos preocupados com eficácia e eficiência, ou seja, com o processo de validação.

Vulnerabilidades na entrada de dados

Até agora, falamos apenas da entrada no sistema, ou seja, do processo de autenticação. Depois que entramos, há vários pontos a serem tratados e observados – neste caso, nosso papel é o de desenvolvedor –, de modo a garantir que o usuário utilize o sistema de forma correta.

Várias situações comuns podem ser testadas logo na entrada dos dados, como: Os formatos dos dados estão de acordo com o que o sistema está preparado para aceitar? Os intervalos de dados – tamanho da entrada – estão em conformidade com o que foi especificado?

Existem muitas outras vulnerabilidades associadas a entrada de dados. A mais conhecida delas é a injeção de código malicioso, destacando-se o caso de **injeção de código SQL**.

Nesse caso, o ataque funciona da seguinte maneira: o invasor entra com comandos na entrada, que traz informações sobre como é feito o processo de autenticação. A partir da leitura de um erro gerado pelo programa, o atacante entra com um comando SQL que dá acesso ao sistema e, a partir disso, consegue se passar por um usuário autêntico e, obviamente, não fará nenhuma ação positiva.

A injeção de código SQL é apenas um exemplo entre muitos outros ataques bem documentados, que exploram vulnerabilidades na entrada do sistema. No entanto, existem práticas que podemos implementar para realizar uma validação segura de entrada de software, como:

1

Validar todas as entradas do usuário

Devemos realizar essa validação tanto no cliente quanto no servidor para evitar a entrada de conteúdo ou código mal-intencionado. Exemplos concretos incluem validação de formatos de dados e controle de comprimentos de dados.

2 Usar bibliotecas de validação de entrada

Atualmente, temos diversas bibliotecas para as principais linguagens de programação disponíveis que nos ajudam a fazer esse controle de validação de forma eficiente. Um exemplo concreto é o Pylit para a linguagem Python.

3 Implementar validação de entrada do lado do servidor

Existe um tipo de ataque que ocorre no lado do cliente que tem como alvo o servidor. Um dos mais conhecidos é um ataque de script entre sites, normalmente, referenciado, pela sigla XSS (*cross site scripting*).

4 Criticar todas as entradas do usuário

Antes de entrar no sistema, é necessário termos mecanismos para remover quaisquer caracteres ou conteúdo que possam ser usados para explorar vulnerabilidades de softwares. Um exemplo concreto é a remoção de tags HTML e código JavaScript.

5 Garantir a aplicação de melhores práticas para o código de validação de entrada

Existem excelentes fontes de informação que explicam detalhadamente como muitos ataques funcionam e como combatê-los. Portanto, é responsabilidade do desenvolvedor aplicar essas boas práticas. No caso de corporações, é fundamental investir na capacitação de seus colaboradores. Uma excelente fonte de informação é o site do OWASP, o qual vamos explorar mais adiante.

Portanto, vimos aqui, de forma concreta, que uma das nossas preocupações para construir um software seguro é fazer a validação da entrada. E, para ajudar a alcançar esse objetivo, já existem muitas fontes disponíveis para consultas e bibliotecas. Por isso, é importante investir no planejamento do que vamos fazer antes de sair implementando para, depois, tentar corrigir as vulnerabilidades.

A seguir, vamos estudar as bibliotecas de validação de entrada.

Bibliotecas de validação de entrada

Conheça o uso de bibliotecas para validação de entradas e algumas bibliotecas usadas para facilitar a tarefa de validação de entrada em um software.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

A validação ajuda a evitar vários tipos de vulnerabilidades de segurança. Porém, fazer esse trabalho sem uso de ferramentas específicas dá bastante trabalho e pode demandar muitos recursos financeiros, profissionais especializados e, ainda assim, gerar outras vulnerabilidades. Atualmente, temos a vantagem de contar com muitas bibliotecas (algumas ferramentas usam a expressão “pacotes”) que fornecem a funcionalidade de validação de entrada.

Exemplos de bibliotecas

Vejamos alguns exemplos de ferramentas para validação de entrada:

OWASP ESAPI

A sigla ESAPI se refere ao termo Enterprise Security API (API de Segurança Corporativa). Basicamente, é um conjunto de bibliotecas que, entre outras funcionalidades, implementa recursos de segurança, como validação de entrada.

InputSanitizer

Como o nome sugere, faz a sanitização para validação de entrada. Ela pode ser utilizada em diversas linguagens de programação, sendo mais comum no caso do PHP.

Validator.js

Validação de entrada em aplicativos JavaScript. Ela oferece suporte para regras de validação de e-mail, URL e comprimento.

Apache Commons Validator

Validação de entrada em aplicativos Java. Entre as funções de validação de entrada que ele fornece estão entradas em vários formatos, como strings, datas e números e suporte a regras de validação personalizadas.

Django Forms

Validação de entradas em aplicações para os programadores de Python que trabalham com o framework Django. Entre as validações oferecidas, estão: validar a entrada e manipular os dados do formulário, campos obrigatórios, validação de e-mail e validação de comprimento.
O Django é um framework de código aberto para desenvolvimento web com a linguagem Python, baseado no padrão arquitetural model-template-views.

Laravel Validation

Específica para validar entradas em aplicações web PHP. Ela permite validar a entrada em vários formatos, como strings, números e datas e suporta regras de validação personalizadas.

Portanto, existem diversas bibliotecas de validação de entrada disponíveis para as mais variadas programações. Aqui, apresentamos apenas algumas. O mais importante é que você utilize uma biblioteca para implementar a validação da entrada para reduzir as vulnerabilidades do software que está desenvolvendo e isso certamente vai demandar mais estudo! Porém, tenha certeza de que está investindo o seu tempo para adquirir um conhecimento muito valorizado pelo mercado e capaz de lhe auxiliar em termos de empregabilidade e destaque profissional.

A seguir, vamos falar sobre a implementação da validação de entrada do lado do servidor. O desenvolvimento de um software seguro, realmente, demanda muito estudo, mas estamos aqui para isso, não é mesmo? Então, vamos lá!

Implementação da validação de entrada no servidor

Assista ao vídeo e compreenda conceitos básicos sobre a implementação da validação de entrada do lado do servidor.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

Quando falamos sobre validação de entrada de dados, quase sempre estamos nos referindo à aplicação do lado do cliente. No entanto, também devemos nos preocupar com a validação de entrada do lado do servidor.

Atualmente, é muito comum utilizarmos aplicações que servem apenas para acessar o servidor e visualizar dados. Portanto, é necessário implementar a validação de entrada do lado do servidor, a fim de adicionar mais um elemento na criação de software seguro. Isso nos permite evitar vários tipos de vulnerabilidades, como ataques de injeção e estouros de buffer.

Etapas para validação no lado do servidor

A seguir, apresentamos algumas das etapas que devemos implementar para realizar a validação de entrada segura do lado do servidor:

Identificar as entradas

Desenvolver métodos ou utilizar funções de bibliotecas — o que é a situação ideal - para identificar, explicitamente, quaisquer dados enviados. Esses dados podem vir por meio de formulários, cabeçalhos HTTP, cookies ou qualquer outra forma de entrada.

Definir regras de validação

Definir tipos, comprimentos e formatos de dados esperados.

Validar entrada

Validar as entradas para o servidor para verificar se estão em conformidade com as regras definidas. Caso não estejam, jamais devemos apresentar uma mensagem de erro do sistema para o usuário, mas sim uma mensagem indicando que o formato é inválido, sem fornecer dicas.

Limpar a entrada

Remover qualquer código ou caracteres maliciosos é imprescindível. Em especial, no caso de entradas para consultas ao banco de dados, elas podem conter ataques de injeção de SQL.

Realizar verificações de segurança de vulnerabilidades conhecidas

Conhecer as vulnerabilidades é fundamental para que possamos evitá-las. Existem muitos ataques bem documentados, como *Cross-Site Scripting* (XSS), que permitem a um invasor injetar código malicioso em uma página da web utilizada por outros usuários; e *Cross-Site Request Forgery* (CSRF), que permite a um invasor executar ações não autorizadas em nome de um usuário legítimo.

Manter um log dos erros de validação

Utilizar mecanismos que registrem todas as ações ocorridas no sistema, de modo a investigar, caso seja necessário, os erros de validação que ocorrerem é muito importante para, assim, acionarmos os mecanismos legais para punir os invasores. Apesar de existirem muitas formas de tratar erros de validação de entrada, também há muitos golpes que se renovam.

Manter as regras de validação atualizadas

Revisar os processos e garantir que estejam sendo aplicados é igualmente importante. Nesse sentido, o uso dos logs é fundamental para entendermos o que realmente está acontecendo no sistema e, assim, nos guiar sobre quais são as melhores ações a tomar para melhorarmos a validação da entrada do servidor.

Todas essas etapas nos ajudam a implementar a validação de entrada do lado do servidor e reduzir as vulnerabilidades de segurança do nosso software.

Na sequência, vamos analisar a limpeza dos dados dos usuários, com o objetivo de reduzir vulnerabilidades e aumentar a segurança dos nossos sistemas.

Limpeza de todas as entradas do usuário

Conheça os fundamentos necessários para limpar todas as entradas do usuário.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

Vamos imaginar a seguinte situação: você desenvolveu um pequeno sistema para a área de contabilidade da empresa. O cliente lhe informou que os dados de entrada sempre seriam numéricos e, portanto, você focou na implementação das regras de negócio e sem a suposição (ingênua) de que os usuários do sistema sempre entrariam com dados numéricos.

Infelizmente, uma pessoa entrou com uma cadeia de caracteres – string – e o sistema falhou, apresentando uma mensagem de erro sem nenhum tipo de tratamento. Esta é uma situação muito comum de vulnerabilidade de software.

Etapas da tarefa de limpeza de dados

Esse cenário ocorre com muita frequência e serve como alerta sobre a importância de realizarmos ações de limpeza dos dados de entradas do usuário, para aumentamos a segurança do nosso servidor de software. No entanto, essa tarefa pode implicar a exclusão de dados do usuário.

Por isso, precisamos de um processo bem definido para realizá-la com segurança. A seguir, apresentamos algumas etapas possíveis:

1 Identificar as entradas do usuário

Nesta etapa, precisamos saber onde os dados ficam e como deveriam estar armazenados. Para isso, precisamos mapear as tabelas de banco de dados, os servidores, os arquivos de logs e quaisquer outros meios de armazenamentos de dados.

2 Determinar as políticas de retenção

Aqui, nos referimos a quanto tempo os dados devem ser mantidos, quem deve ter acesso a eles e como devem ser destruídos quando não forem mais necessários.

3 Definir um processo para limpar as entradas do usuário

Esta é uma etapa fundamental, pois precisamos definir um processo que faça verificação da identidade do usuário, validação das permissões do usuário para limpar os dados e um controle para confirmar a intenção do usuário de realmente excluir os dados e, obviamente, registrar essa ação em um arquivo de log.

4 Usar métodos de exclusão segura

Nesta etapa aplicamos métodos que, de fato, apaguem os dados, de modo que não possam ser recuperados por um invasor.

5 Registrar o processo de exclusão

Sempre devemos utilizar uma ferramenta ou uma configuração de sistema que permita fazer o registro do processo de exclusão, ou seja, devemos ser capazes de responder quem excluiu os dados, quando foi realizada a exclusão e qual foi o motivo da exclusão.

6 Notificar o usuário

Por fim, o sistema deve avisar explicitamente ao usuário que seus dados foram excluídos com segurança.

Essas etapas auxiliam na limpeza dos dados de entradas do usuário com segurança e garantem que estejam protegidos segundo as regras corporativas – no caso, de empresas e demais organizações – ou enquanto assim desejarmos – no caso de uso pessoal.

Teste e revisão do código de validação de entrada

Descubra os aspectos a serem considerados para que seja possível testar e revisar o código de validação de entrada.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

Por mais eficiente que seja a técnica que apliquemos no nosso sistema, do lado dos invasores vai existir um grande esforço para violar a segurança. Em especial, no caso de dados valiosos, isso se torna ainda mais crítico. Por isso, precisamos realizar treinamentos e estudar esse assunto continuamente.

Atenção

Uma etapa crítica no desenvolvimento de software seguro é fazer testes e revisões rotineiras do código de validação de entrada.

Etapas dos testes e revisão do código de validação de entrada

Essa é uma forma preventiva de reduzir vulnerabilidades e, caso seja necessário, realizar melhorias no código ou no próprio processo de validação de entrada. A seguir, estão listadas algumas etapas que podemos realizar para testar e revisar o código de validação de entrada em um software seguro:

Verificar a qualidade das regras de validação

É essencial estarmos sempre alinhados com as melhores práticas de segurança. Por isso, precisamos entrar em sites como a OWASP e outros fóruns para acompanhar novas técnicas de validação e discussões sobre o assunto de profissionais que trabalham na área.

Testar entradas inválidas

É necessário criar diversos cenários que forcem entradas inválidas no sistema. Essa abordagem é uma forma muito eficiente de simular ataques, identificar possíveis vulnerabilidades e, posteriormente, corrigi-las.

Testar entradas válidas

É uma boa forma de simularmos o comportamento dos usuários legítimos que farão uso do sistema. Vamos avaliar se o sistema vai se comportar conforme o planejado para entradas válidas de usuário.

Realizar testes de estresse

A ideia é forçar situações extremas, com a finalidade de identificar possíveis vulnerabilidades na validação de entrada.

Usar técnicas de fuzzing

É importante para que seja possível automatizar testes que gerem entradas aleatórias ou, ainda, entradas que possam acionar determinados tipos de vulnerabilidades.

Revisar o código para vulnerabilidades de segurança

É necessário revisar o código de validação de entrada e, se for o caso, realizar melhorias para nos prevenir de vulnerabilidades, visando reduzir as chances de que os invasores sejam bem sucedidos ao aplicar novas técnicas para atacar o sistema.

Usar bibliotecas de validação

É sempre importante utilizar bibliotecas, pacotes e ferramentas que têm como finalidade realizar testes para detectar vulnerabilidades.

Verificando o aprendizado

Questão 1

Um dos primeiros contatos que um usuário tem com um software é com o processo de autenticação. Nas versões mais simples, o usuário precisa fornecer um login e uma senha. Nesse sentido, como esse simples processo pode afetar a segurança de um software?

A

Um usuário pode inserir um código malicioso que, se não for tratado, pode causar danos na segurança de todo o sistema.

B

Um usuário legítimo pode ter sua conta de acesso bloqueada ao errar a senha após algumas tentativas.

C

Um usuário pode esquecer a senha de acesso e, assim, ficar impedido de realizar suas tarefas.

D

O software pode solicitar que o usuário renove a sua senha e, assim, ele pode colocar uma senha fraca.

E

Um usuário legítimo, insatisfeito com o software, pode tentar entrar várias vezes com a senha errada para forçar que o sistema pare de funcionar.

A alternativa A está correta.

Uma das formas que um usuário pode causar problemas na segurança de um software é entrar com um código malicioso. Caso o sistema não tenha um processo de validação de entrada, o usuário pode ter acesso a dados e outras funcionalidades do sistema sem a devida autorização.

Questão 2

Um software corporativo, normalmente, possui diversas funcionalidades. Por isso, é natural utilizar componentes e bibliotecas de terceiros para evitar consumir tempo e recursos de mão de obra com atividades que não são a finalidade do sistema. Nesse sentido, como se prevenir de inserir vulnerabilidades no software a partir de componentes de terceiros?

A

Realizando uma engenharia reversa para analisar o código-fonte dos componentes dos fornecedores em busca de vulnerabilidades.

B

Usando bibliotecas acreditadas pela comunidade de desenvolvimento, baixadas das fontes oficiais e realizando sempre as atualizações sugeridas pelo desenvolvedor.

C

Sempre priorizando a utilização dos componentes com licença-livre.

D

Tentando reproduzir um componente similar ao do fornecedor e mapeando as possíveis vulnerabilidades.

E

Buscando informações em fóruns informais nos quais os usuários expõem suas opiniões com mais liberdade.

A alternativa B está correta.

Sempre vai existir o risco de inserir uma vulnerabilidade no nosso software, quando usamos um componente de outro fornecedor. Assim, é fundamental buscar fornecedores oficiais e que façam atualizações de segurança em ambientes confiáveis, pois eles são especialistas neste tipo de desenvolvimento e não têm nenhum interesse que os componentes deles possam inserir vulnerabilidades nos sistemas dos clientes.

Fundamentos do processamento de dados seguros

Conheça os princípios para realizar o processamento de dados seguros.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

Damos muita importância para os dados, o que está correto, pois eles podem ser muito úteis para agregação de valor. Entretanto, até chegarmos à etapa de melhorar processos ou criar uma compreensão de existirem oportunidades a serem exploradas por meio da análise dos dados, precisamos processá-los.

Atenção

Além das preocupações com a implementação das regras de negócio e outros algoritmos que deem essas respostas, precisamos garantir que o processamento dos dados não afete a segurança do sistema.

A segurança de um sistema também precisa levar em consideração o processamento dos dados de forma segura. Portanto, precisamos implementar processos e controles capazes de prevenir que os dados sejam acessados por pessoas sem o devido acesso. Se a segurança for violada, isso pode levar a roubo, modificação indevida ou divulgação não autorizada. Por isso, precisamos usar recursos lógicos e físicos que limitem o acesso aos dados.

Práticas para processamento seguro dos dados

Listamos, a seguir, algumas práticas que nos ajudam a processar dados com segurança:

Implementar controles de acesso

São mecanismos de controle responsáveis por garantir que apenas pessoas autorizadas tenham acesso aos dados. Exemplos: uso de mecanismos de autenticação que obriguem os usuários a utilizarem senhas fortes, autenticação multifator ou de duas fases, e uso de biometria para acessar o sistema de processamento de dados.

Criptografar dados

Caso uma pessoa sem autorização consiga acessar os dados, os mecanismos de criptografia vão protegê-los, pois não farão sentido para o atacante.
Exemplo: os algoritmos simétricos e assimétricos empregados no processo de criptografia, como o AES (Advanced Encryption Standard) e RSA (Rivest-Shamir-Adleman).

Implementar práticas de codificação segura

Utilizar mecanismos e estruturas de codificação conhecidos, capazes de garantir que o código seja escrito de maneira segura e, assim, o software poder processar os dados com segurança. Exemplos: validação de todas as entradas do usuário, limpeza dos dados, ou seja, procurar por conteúdo malicioso e removê-lo, e não executar entrada de dados sem passar por métodos que os critiquem.

Usar protocolos de comunicação seguros

O exemplo mais clássico de protocolo de comunicação segura é o HTTPS, que protege os dados em trânsito na rede contra interceptação ou modificação, pois garante que os dados sejam criptografados e autenticados durante a transmissão.

Fazer buscas por vulnerabilidade e teste de penetração

Quanto mais complexo for um sistema, maiores são as chances de que ele tenha vulnerabilidades. Por isso, é necessário ter uma rotina para tentar detectá-las e corrigi-las. Uma das formas de fazer isso é por meio do teste de penetração que força situações extremas e, assim, ajuda na identificação de possíveis vulnerabilidades.

Limitar o tempo de processamento

Precisamos de uma métrica capaz de ajudar a estimar o tempo provável para que o programa processe os dados. Se o tempo consumido for muito diferente da média esperada, então, precisamos investigar o que aconteceu, pois é um sinal de alerta, de algo fora do normal.

Controles de acesso

Saiba, assistindo a este vídeo, quais são os fundamentos dos controles de acesso.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

Uma das melhores formas para o processamento dos dados com segurança é garantir que apenas usuários legítimos possam acessá-los. Os controles de acesso são o instrumento responsável por garantir que apenas usuários autorizados possam realizar operações no sistema, tais como acessar os dados ou executar determinadas funções.

Listamos alguns dos principais aspectos sobre os controles de acesso:

Criação de perfis

Os usuários do sistema, por sua vez, serão associados a perfis com papéis de responsabilidade e que terão direitos e deveres. Dessa forma, facilitamos o processo de gerenciamento do controle de acesso.

Princípio do privilégio mínimo

Os perfis de usuário devem ter acesso apenas a recursos e funções estritamente essenciais para que possa realizar suas funções. Caso o usuário precise de mais privilégios, então é necessário rever se ele está no perfil correto, ou, ainda, se devemos ampliar os privilégios do perfil em que está. Por meio do controle por perfis e do princípio do privilégio mínimo, podemos gerenciar melhor os comportamentos dos usuários do sistema e detectar anomalias com mais eficiência.

Separação de funções

Os poderes que um perfil de usuário pode ter tornam-se menores quanto mais valiosos forem os dados que o sistema manipula. Nesse caso, a melhor solução é separar as operações sensíveis entre diferentes perfis de usuários, para evitar que um atacante – simulando um usuário legítimo – tenha controle total sobre uma função crítica. Obviamente, isso aumenta a quantidade de processos para realizar uma tarefa, por isso só deve ser aplicada se, realmente, for justificável.

Controle de acesso baseado em perfil

O monitoramento e aplicação de políticas de controle de acesso baseadas em perfis de usuários do que em indivíduos é muito simples. A consequência direta desse tipo de controle é, no caso de invasão, mensurar os danos que podem ser causados no sistema e, assim, desenvolver contramedidas para prevenir que ocorram, ou, mesmo, para combater invasores.

Autenticação de dois fatores

O controle semelhante aos usados para autenticar a entrada de dados no sistema também pode ser aplicado para verificar a identidade dos usuários que desejam ter acesso aos dados ou processá-los. Por exemplo, podemos usar uma estratégia na qual os usuários forneçam algo que eles conheçam, como uma senha, junto com algo que eles tenham, como um token físico ou um código único enviado para seu dispositivo móvel.

Monitoramento e auditoria

O processo adotado novamente é semelhante ao que fazemos com o controle seguro de dados: geração de arquivos de logs. Por meio desses arquivos, podemos monitorar e, se for o caso, auditar o uso do sistema para garantir que tudo esteja funcionando conforme o esperado. Se houver alguma tentativa de acesso não autorizado, podemos investigar com mais cuidado e tomar medidas legais.

Políticas de senha segura

O acesso por meio de senhas seguras é básico, mas fundamental. Exemplos dessas políticas incluem senhas fortes com combinação de letras minúsculas e maiúsculas e números, comprimento mínimo e máximo, uso de símbolos especiais, período de expiração de senha e impedimento de reutilização de senhas.

Todos esses passos são importantes para o controle de acesso, mas dificultam a vida do usuário legítimo. Portanto, é importante que a implementação dessas técnicas e processos seja justificável e realizada com campanhas de conscientização.

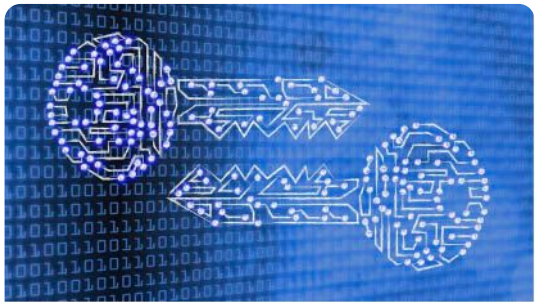
Criptografia de dados

Descubra a importância da criptografia para a implementação de um software seguro.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

Por mais cuidadosos que possamos ser, sempre existe a chance de um atacante encontrar formas de violar a segurança do sistema e ter acesso não autorizado aos dados. Isso pode ser desastroso, pois os dados podem ser modificados e, quando realizarmos o processamento deles, posteriormente, estaremos diante de resultados que não refletem a realidade e podem nos conduzir a tomadas de decisões erradas.



Portanto, precisamos garantir que os dados só tenham valor para os usuários legítimos. Uma das técnicas mais eficientes para fazer isso é a **criptografia**. A criptografia é uma técnica fundamental para proteger a confidencialidade, integridade e autenticidade dos dados.

Ela se baseia em algoritmos de transformação de dados com o uso de chaves. Então, apenas as pessoas com acesso a essas chaves conseguem acesso aos dados originais. No caso de um invasor acessar os dados criptografados ou tentar modificá-los, isso não trará nenhuma vantagem prática para ele.

Saiba mais

A criptografia é um recurso que aumenta bastante a segurança dos dados de um sistema. Atualmente, é muito mais simples utilizá-la na prática, pois muitas ferramentas já a oferecem como recurso para garantir que nossos dados estejam protegidos e, assim, seja possível aumentar a resistência a ataques.

Algoritmos de criptografia

A seguir, apresentamos algumas questões importantes quando utilizar algoritmos de criptografia para aumentar a segurança de um software:

1

Conhecimento sobre os principais algoritmos de criptografia

Devemos buscar o melhor algoritmo de criptografia para nossos dados, pois isso vai protegê-los contra acessos não autorizados. Alguns dos algoritmos de criptografia mais usados atualmente são AES (criptografia simétrica), RSA (criptografia assimétrica) e SHA (função hash).

2

Gerenciamento de chaves

Precisamos de uma política bem definida sobre gerenciamento adequado de chaves que envolve geração, armazenamento e proteção das chaves usadas para criptografar e decriptografar os dados. O algoritmo de criptografia só é seguro se a chave de criptografia for mantida em segurança.

3 Implementação adequada

Precisamos garantir que as chaves sejam geradas de modo aleatório, o processo de criptografia seja executado corretamente e os dados criptografados sejam armazenados em um ambiente seguro, já que, quem vai aplicar a criptografia é um sistema.

4

Integração com outros sistemas

Podemos aumentar a eficácia do processo ao utilizar protocolos de segurança como TLS/SSL (Transport Layer Security — Camada Segura de Transporte/Secure Sockets Layer - Camada de Soquete Seguro), além dos algoritmos de criptografia em si, pois eles garantem que os dados sejam criptografados tanto em trânsito como em repouso.

5

Gerenciamento de certificados

Devemos criar uma política explícita de gerenciamento para os certificados digitais, pois, atualmente, é difícil falar em segurança sem mencioná-los, já que são um recurso lógico usado para autenticar a identidade de usuários, servidores e outros dispositivos em uma rede e, por este motivo, como outros recursos de segurança, eles também precisam de uma política de gerenciamento para garantir que a comunicação seja segura.

6

Atualizações de segurança

Precisamos atualizar os algoritmos de criptografia, da mesma forma que precisamos atualizar outros recursos de segurança. Para isso, devemos visitar regularmente o fornecedor do algoritmo que estamos utilizando no sistema e realizar as atualizações sugeridas pela fonte oficial.

Protocolos de comunicação seguros

Neste vídeo, compreenda os principais conceitos sobre protocolos de comunicação seguros.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

Os protocolos de comunicação segura são especialmente fundamentais em ambientes cliente X servidor, tanto em intranets como na internet. Eles oferecem proteção contra as quebras de confidencialidade, integridade e autenticidade dos dados que transmitimos por uma rede.

Existem algumas questões que precisamos considerar ao utilizar protocolos de comunicação seguros em um software:

Escolha do protocolo

Devemos levar em consideração os requisitos de segurança para o contexto no qual a nossa aplicação vai trabalhar. Os protocolos de comunicação segura mais utilizados são o TLS e o SSL.

Configuração de parâmetros

Temos que configurar diversos parâmetros, como o tamanho da chave de criptografia, considerando que esta é uma das etapas mais complexas no uso de protocolos seguros e uma das principais vulnerabilidades exploradas por invasores.

Gerenciamento de chaves

Precisamos realizar a geração, armazenamento e proteção adequados das chaves usadas para criptografia e decriptografia.

Gerenciamento de certificados digitais

Precisamos de um plano formal para fazer o gerenciamento, monitoramento, atualização e expiração dos certificados digitais para garantir uma comunicação segura, visto que eles são importantes para autenticar a identidade de usuários, servidores e outros dispositivos que estão na rede.

Monitoramento

Precisamos monitorar o protocolo de comunicação seguro em qualquer etapa de um software seguro. Por exemplo, precisamos analisar possíveis violações ou vulnerabilidades de segurança. Fazemos isso por meio do monitoramento de tentativas de acesso não autorizado, padrões de tráfego incomuns ou quaisquer outras anomalias.

Atualizações

Precisamos manter uma rotina de atualização por meio de consultas às fontes oficiais. Normalmente, os fornecedores disponibilizam patches de segurança - programas para corrigir vulnerabilidades. Desse modo, garantimos que estamos usando a versão mais segura do protocolo.

Sem dúvidas, o uso de protocolos de comunicação segura é uma das questões que devemos levar em consideração para realizar o processamento de dados com segurança, pois é uma forma de proteger os dados e, assim, reduzir as chances de que nossa aplicação sofra ataques bem-sucedidos.

Busca por vulnerabilidades e testes

Compreenda, neste vídeo, os principais conceitos relacionados à busca por vulnerabilidades e testes.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

Para realizar um processamento de dados com segurança, sempre precisamos fazer uma busca por vulnerabilidades e realizar testes de segurança. Uma grande vantagem é que já existem passos bem conhecidos para tratar esse assunto.

Listamos os principais:

Varredura de vulnerabilidade

Precisamos usar ferramentas automatizadas para varrer o código do sistema e identificar possíveis vulnerabilidades. Como vimos, a OWASP apresenta diversas vulnerabilidades bem comuns, como injeção de SQL, script entre sites (XSS) e estouros de buffer.

Teste de penetração

Devemos realizar testes manuais para identificar pontos fracos de segurança. Esses testes conseguem detectar vulnerabilidades que dificilmente seriam pegas por ferramentas automatizadas. Fazemos a simulação de ataques para identificar vulnerabilidades e determinar qual o impacto delas para segurança do sistema.

Bibliotecas e componentes de fornecedores confiáveis

Temos, atualmente, acesso a diversas bibliotecas e componentes. Ao utilizarmos esses recursos, porém, podemos introduzir vulnerabilidades no nosso sistema, pois é inviável fazermos testes neles. A forma de reduzir as chances de criarmos um problema é usar fontes confiáveis. No entanto, isso não garante que não existam vulnerabilidades nesses recursos.

Teste de conformidade

Contamos com alguns padrões de testes de conformidade regulatória, como o padrão de segurança de dados do setor de cartões de pagamento (PCI DSS) e a lei de portabilidade e responsabilidade de seguros de saúde (HIPAA). Ambos os padrões são dos Estados Unidos, mas servem como guia para desenvolvermos nossos testes de conformidade de modo semelhante.

Teste contínuo

Devemos lembrar que é importante ter uma rotina de implementação de testes automatizados e testes manuais para identificar possíveis vulnerabilidades, pois, assim, teremos a chance de aplicar correções antes que possam ser exploradas por invasores.

Avaliação de risco

Devemos mapear todas as etapas do processamento dos dados e realizar avaliações de risco. Desse modo, podemos identificar possíveis riscos de segurança e priorizar vulnerabilidades com base no impacto potencial delas.

Documentação

Precisamos ter em mente que, apesar de não ser bem priorizada, essa etapa é fundamental. Sempre que desenvolvermos e realizarmos os testes, devemos documentá-los, assim como seus resultados. Além disso, é essencial mantermos o registro das ações que tomamos para eliminar as vulnerabilidades e corrigir os erros, pois podem ser úteis para outras pessoas envolvidas no projeto.

Todo o processo relacionado ao processamento de dados com segurança exige muito esforço da nossa parte. No entanto, como vimos, existem etapas bem definidas que podemos aplicar para reduzir o potencial de danos e detectar e corrigir vulnerabilidades, antes que elas sejam exploradas por invasores.

Verificando o aprendizado

Questão 1

O processamento dos dados torna o uso de sistemas produtivo. Nesse sentido, selecione a alternativa correta a respeito da melhor forma de evitar que o processamento dos dados gere uma vulnerabilidade de segurança.

A

Colocar avisos sobre as penalidades legais que um usuário pode sofrer ao utilizar o software para fins fraudulentos.

B

Confiar no usuário autenticado, deixando o usuário realizar qualquer transação no sistema, com isso, permitindo que os resultados sejam avaliados posteriormente para tentar detectar falhas ou vulnerabilidades.

C

Utilizar técnicas que acelerem o processamento dos dados e façam testes automatizados nos resultados obtidos.

D

Limitar a quantidade de dados processados para investigar de forma mais eficiente possíveis situações de vulnerabilidades.

E

Utilizar meios para assegurar que apenas usuários com perfis legítimos possam executar apenas determinadas tarefas.

A alternativa E está correta.

A primeira ação que devemos tomar para garantir que um software realize processamentos seguros é validar os dados de entrada. Em seguida, devemos garantir duas condições: I) apenas usuários legítimos tenham acesso ao sistema; e II) que eles estejam associados a perfis que limitem os privilégios de execução, reduzindo as chances de alguém cometer um ato fraudulento.

Questão 2

A política do privilégio mínimo é associada aos perfis dos usuários, e não ao usuário especificamente. Nesse sentido, selecione a alternativa que justifica a eficácia da política de privilégio mínimo para garantir que o software seja seguro.

A

Ao reduzir os privilégios dos usuários, obrigamos que mais pessoas estejam envolvidas na execução de uma tarefa, aumentando, assim, as chances de que ela seja realizada com segurança.

B

O processo de fraude torna-se muito difícil, pois um invasor teria que assumir vários perfis para causar algum dano significativo para a segurança do sistema.

C

Ela se baseia na criptografia dos dados para cada perfil. Assim, é impossível que um invasor sozinho possa prejudicar o processamento dos dados.

D

Ela cria pontos de controle que podem gerar alertas sempre que um dado apresentar alguma inconsistência associado a uma tarefa alocada para determinado perfil.

E

O gerenciamento de perfis é mais eficiente do que o de indivíduos. Além disso, é possível mensurar os danos que um perfil pode causar.

A alternativa E está correta.

A política de privilégio mínimo reduz os direitos dos usuários dentro de um sistema apenas ao essencial para que eles realizem as tarefas para as quais os perfis deles estão mapeados. Portanto, ao identificarmos uma situação insegura, podemos determinar a qual perfil ela está associada. Além disso, ainda somos capazes de medir quais são os impactos que podem ser ocasionados, uma vez que os privilégios estão associados aos perfis.

Fundamentos de Chamadas

Neste vídeo, conheça alguns dos principais fundamentos de chamadas em software seguro.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

Nossa motivação para construir um software é que ele atenda a uma demanda, ou seja, realize ações úteis para seus usuários. Podemos ter softwares utilitários, como os aplicativos de banco, e softwares voltados para o entretenimento, como os jogos eletrônicos. Todos esses sistemas têm em comum uma arquitetura lógica que realiza comandos a partir da chamada de funções específicas.

O uso de funções e medidas preventivas para chamadas seguras

O uso de funções compõe as boas práticas de organizarmos um sistema, pois elas ficam responsáveis por partes específicas do processo lógico. No entanto, realizar essas chamadas de funções ou, ainda, chamar outros programas de dentro do nosso software, pode ser uma vulnerabilidade para a segurança do sistema. Por isso, apesar de essa prática ser muito útil, devemos tomar algumas medidas preventivas para evitar riscos de segurança.

Uma forma bastante comum atualmente de trabalharmos com chamadas de funções é fazer via API (sigla para Interface de programação de aplicativos, do inglês Application Programming Interface). Ela permite que possamos fazer a comunicação entre diferentes componentes de software, de modo que possam interagir entre si e, dessa forma, possibilita integrar nossos sistemas e consumir serviços.

A seguir, apresentamos algumas medidas preventivas que nos ajudam a fazer chamadas seguras:

Validar as entradas do usuário

Devemos ter certeza de que qualquer entrada do usuário passe por um processo de validação antes de ser processada. Uma função deve receber parâmetros, processá-los e retornar um resultado, especialmente se queremos evitar ataques de injeção de código.

Usar APIs do sistema

Podemos nos prevenir de inserir vulnerabilidades no nosso sistema ao utilizarmos APIs e bibliotecas do sistema, pois, teoricamente, elas já foram testadas para evitar problemas de segurança.

Definir permissões

Devemos definir as permissões necessárias que um programa deve ter para chamar uma determinada função e, assim, acessar os recursos do sistema. Isso é essencial. Para isso, é necessário garantir que a função faça uma leitura das credenciais de quem está fazendo a chamada, aumentando a complexidade de uma simples chamada.

Limitar privilégios

Precisamos também limitar os privilégios da função que está sendo chamada, de modo que ela tenha acesso apenas aos recursos necessários para executar a tarefa.

Usar funções com as devidas autorizações

Devemos cadastrar as funções em uma lista, logo após definirmos as permissões e limitar os privilégios de acesso aos recursos do sistema, para controlarmos quem pode realizar chamadas e executar comandos.

Monitorar as chamadas do sistema

Precisamos sempre ter um recurso no sistema que gere arquivos de logs, para podermos monitorar as chamadas. Esse é o procedimento mais clássico. Por mais eficiente que seja nosso processo de segurança preventiva, às vezes, os invasores encontram alguma vulnerabilidade e passam a explorá-la.

Usar canais de comunicação seguros

Precisamos garantir que a comunicação entre as duas máquinas seja criptografada e autenticada. Isso é especialmente importante quando o programa que está sendo chamado estiver sendo executado em uma máquina diferente.

Chamadas de APIs do sistema

Neste vídeo, compreenda o conceito de API e conheça os principais aspectos de chamadas de APIs do sistema em softwares seguros.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

Usamos as APIs para definir um conjunto de regras que servem para realizar a solicitação de dados, fazer o recebimento de dados, executar operações específicas e acessar determinados recursos de um sistema ou serviço de software.

> As APIs podem ser usadas para acessar dados de diversas fontes, como bancos de dados, serviços da web e outros aplicativos de software. Portanto, ao utilizá-las, devemos seguir algumas práticas recomendadas para garantir que nosso software seja o mais seguro possível.

Listamos algumas sugestões de como proceder:

Usar chamadas de API bem documentadas e testadas

Sempre devemos ter uma boa documentação da API e alguma garantia de que ela foi submetida a diversos testes, pois vai nos ajudar a garantir que nosso código permaneça estável, quando a utilizarmos.

Validar a entrada e a saída

Precisamos validar tanto a entrada como a saída das chamadas da API, com os objetivos de evitar receber dados com códigos maliciosos e produzir uma saída maliciosa que comprometa a segurança do sistema.

Usar práticas de codificação seguras

Precisamos considerar que são as mesmas práticas que vimos em outras situações, mas, agora, voltadas para o uso de APIs, como validação de entrada, codificação de saída e tratamento de erros para evitar vulnerabilidades como estouros de buffer e ataques de injeção.

Usar o princípio do privilégio mínimo

Reduzimos as chances de os invasores obterem acesso a recursos confidenciais quando permitirmos que a API tenha acesso apenas aos recursos de que precisa para executar a tarefa específica.

Manter o software atualizado

Devemos sempre utilizar as fontes oficiais para manter as APIs atualizadas, por exemplo, por meio de programas, como patches.

Usar ferramentas de segurança

Devemos nos proteger utilizando ferramentas de segurança como softwares antivírus, firewalls e sistemas de detecção de intrusão, o que vai implicar, na prática, a adição de uma camada de segurança.

Realizar testes de estresse

Precisamos compreender que este importante procedimento gera diversos cenários de situações que forçam a identificação das possíveis vulnerabilidades de segurança.

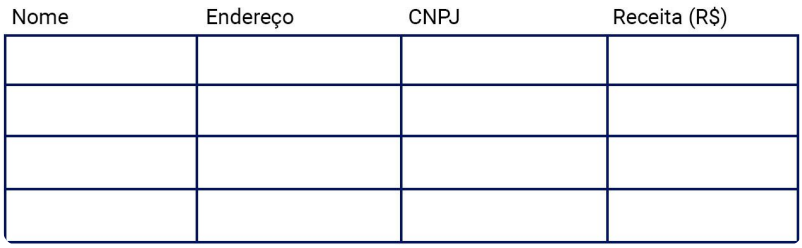
Definição de permissões

Conheça os principais aspectos da definição de permissões no contexto de software seguro.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

Quanto mais módulos um sistema tiver, maior será a complexidade para estabelecer quais permissões devemos conceder aos perfis de usuários. Esse processo, realmente, pode dar muito trabalho. Vamos considerar um módulo CRUD (inserir, visualizar, modificar e excluir dados), conforme vemos na imagem:



Exemplo de um módulo de CRUD.

De imediato, precisamos decidir se todos os perfis terão os mesmos acessos e permissões para realizar visualizações, cadastros, modificações e exclusões dos dados. Obviamente, essa tarefa vai exigir muito da nossa atenção e tem grandes chances de apresentar vulnerabilidades.

Atenção

O processo de definir permissões precisa ser realizado antes que o software seja entregue ao cliente, pois o papel do desenvolvedor é apenas de criar um ambiente no qual um responsável vai associar as permissões aos perfis dos usuários.

Diretrizes para a definição de permissões

Existem algumas diretrizes que devemos seguir para definir permissões. Vamos conhecê-las!

Usar o princípio do privilégio mínimo

Cada permissão deve dar acesso a uma ação específica necessária, exclusivamente, para realizar uma tarefa.

Restringir as permissões de arquivos e pastas

Apenas perfis de usuários autorizados podem ter acesso aos arquivos e pastas. No caso de sistemas operacionais baseados em Unix, podemos usar o comando chmod para definir permissões em arquivos e diretórios.

Usar contas de usuário separadas

Essa parte pode ser incômoda para o usuário, pois a ideia é que ele utilize uma conta separada para usar do servidor da Web e outra conta para acessar o banco de dados. Uma alternativa é usar um processo de autenticação baseado em certificado digital que garanta a autenticidade do usuário.

Usar senhas fortes

Essa recomendação é clássica. Por outro lado, precisamos ter cuidado para não dificultar excessivamente a vida do usuário, pois, caso contrário, ele pode criar situações que gerem vulnerabilidades, por exemplo, anotar a senha e colocar sob o teclado.

Aplicar criptografia

Para garantir a confidencialidade dos dados, precisamos usar criptografia. Isso vale tanto para os dados armazenados, quanto para os que estão em trânsito pela rede.

Verificar as permissões de forma regular

É natural que haja rotatividade dentro de uma empresa. Às vezes, há mudanças de cargos ou profissionais dispensados. Portanto, sempre precisamos manter uma rotina de verificação capaz de nos ajudar a identificar possíveis problemas de segurança.

Realizar testes de estresse

Devemos implementar testes automatizados que verifiquem se as definições de permissões estão, realmente, funcionando. O objetivo é criar situações estressantes para o sistema e estudar como ele se comporta, antes de enviá-lo para produção.

Tipos de processamento e vulnerabilidades

Descubra quais são os principais tipos de processamento de dados e as vulnerabilidades associadas a eles.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

Vimos diversos controles para restringir os privilégios dos perfis dos usuários e de outros aplicativos, visando reduzir a possibilidade de um invasor assumir a identidade de um usuário ou aplicativo legítimo e causar danos durante o processamento de dados.

Contudo, há mais algumas questões interessantes e pouco triviais para as quais precisamos ficar atentos e que são um diferencial em termos de conhecimento: os tipos de processamento e suas respectivas vulnerabilidades.

A seguir, apresentamos os principais tipos de processamento de dados e analisamos as respectivas vulnerabilidades de cada um deles:

Processamento em lote

Envolve o processamento de uma grande quantidade de dados de uma só vez. As vulnerabilidades do processamento em lote incluem a possibilidade de perda ou corrupção de dados durante a transferência, armazenamento ou processamento. Além disso, o processamento em lote pode ser vulnerável a violações de segurança se os dados não forem devidamente criptografados ou protegidos durante o armazenamento ou transmissão.

Processamento em tempo real

O processamento de dados é realizado à medida que são gerados. Há diversas situações em que é muito útil, como controle de condições ambientais e monitoramento de segurança, por exemplo. Uma das vulnerabilidades desse tipo de processamento é em relação a problemas de desempenho e perda de dados. Isso é especialmente crítico quando lidamos com aplicações de Big Data.

Processamento de fluxo

É semelhante ao processamento em tempo real. A principal diferença é ele ser projetado para trabalhar com fluxos de dados contínuos, como transmissões de imagens e áudio ao vivo e dados de sensores. Assim como ocorre no processamento em tempo real, também é vulnerável a problemas de desempenho e isso, certamente, tem impacto na qualidade do dado processado.

Processamento em nuvem

Tecnologia muito utilizada atualmente por diversos motivos, como custos racionais, flexibilidade e segurança. Basicamente, o processamento de dados ocorre na nuvem, utilizando a arquitetura da internet para realizar processamento em equipamentos de empresas especializadas nesse tipo de serviço. Uma das maiores vulnerabilidades ocorre em relação à dificuldade de configuração, pois, se não for feita cuidadosamente, pode ter diversos tipos de vulnerabilidade, como: violações de segurança, caso os dados não sejam criptografados durante o armazenamento ou transmissão; problemas de desempenho, se a infraestrutura de nuvem não for projetada adequadamente; e questões relacionadas à própria confiança no fornecedor do serviço. Mas, na prática, existem fornecedores muito confiáveis e maduros sobre essa tecnologia, como a Amazon, International Business Machines Corporation (IBM) e Microsoft.

Mitigação das vulnerabilidades do processamento de dados

Existem várias ações que podemos tomar para reduzir as vulnerabilidades associadas aos diferentes tipos de processamento de dados, tais como:

- Usar criptografia;
- Aplicar protocolos de segurança;
- Implementar rotinas de backup de dados;
- Desenvolver e implementar uma política de recuperação de desastres.

Além disso, no caso de aplicações de Big Data, precisamos usar tecnologias e técnicas que nos permitam lidar com grandes volumes de dados e sejam escaláveis para atender às demandas futuras. Porém, a ação mais importante de todas é investir na capacitação dos colaboradores e conscientização dos usuários.

Monitoramento das chamadas

Neste vídeo, conheça os principais fundamentos do monitoramento das chamadas em software seguro.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

Depois de estudarmos os diversos tipos de controles necessários para ter um processamento seguro e como esses processamentos podem ocorrer, precisamos monitorar como as chamadas são feitas.

A seguir, apresentamos algumas ações que podem auxiliar na viabilização desse processo:

1

Identificar chamadas de API críticas

Esse papel está mais associado ao responsável pela arquitetura do software, pois ele pode fazer o mapeamento das chamadas de API e estabelecer níveis de criticidade relacionados à segurança. Alguns exemplos de critérios que podemos usar para estabelecer o nível de criticidade da chamada de uma API é se ela trabalha com dados ou recursos confidenciais.

2

Implementar registro de chamadas

Precisamos desenvolver uma forma para registrar informações sobre a chamada que traga informações como a hora da chamada, qual usuário fez a chamada e quais os parâmetros passados para a chamada.

3

Monitorar logs

Sempre devemos ter uma ferramenta que registre as chamadas em arquivos de logs e um processo que nos ajude a procurar por padrões que indiquem atividades maliciosas. São exemplos desse tipo de chamada chamadas repetidas diversas vezes com os mesmos parâmetros em intervalos muito curtos e chamadas fora dos padrões de uso normais.

4

Definir alertas

Precisamos estabelecer alertas que nos notifiquem sempre que houver uma atividade suspeita. Esse é um dos motivos para mantermos arquivos de logs.

5

Revisar os logs regularmente

O ideal seria detectarmos uma situação de tentativa de ataque e, rapidamente, atuar para evitar que ela fosse bem-sucedida. No entanto, há situações mais complexas que só podem ser detectadas depois de muita análise. Por isso, precisamos ter uma política bem definida de revisão dos arquivos de logs na qual existam procedimentos e responsáveis para realizar esse tipo de tarefa.

6

Usar ferramentas automatizadas

A ideia é acelerar o processo de análise para tentar identificar padrões e detectar atividades suspeitas com mais eficiência do que a análise manual.

7

Realizar testes

Regularmente, precisamos estressar o sistema por meio da simulação de chamadas maliciosas e verificar como ele se comporta. Esse tipo de teste pode nos dar informações úteis para que possamos atuar de forma preventiva.

Verificando o aprendizado

Questão 1

Uma das primeiras boas práticas que aprendemos no desenvolvimento de software é utilizar funções. Selecione a alternativa correta sobre os riscos que uma chamada de função pode ocasionar para construirmos um software seguro.

A

O maior problema que pode ocorrer é que a função seja mal documentada e, assim, gere dúvidas quanto ao seu uso correto.

B

O uso de funções torna o sistema muito lento, e isso pode limitar excessivamente a capacidade de resposta do sistema.

C

Caso a função possua uma vulnerabilidade, todos os processos que fizerem chamada para ela estarão expostos.

D

Caso a função não utilize um processo de criptografia, os dados que ela utiliza podem ficar expostos para invasores em potencial.

E

Uma grande limitação no uso de funções é sua restrição à linguagem de programação em que o sistema foi desenvolvido, não podendo utilizar recursos seguros de outras linguagens de programação ou componentes de terceiros.

A alternativa C está correta.

Se as funções não forem bem testadas, podem conter vulnerabilidades. Os riscos de danos aumentam diretamente na proporção em que essas funções precisam ser chamadas. Esse é um dos motivos pelos quais precisamos investir em muitos testes, boa documentação e monitoramento das chamadas realizadas por nosso sistema.

Questão 2

As APIs nos permitem realizar diversas operações que nos ajudam a potencializar a nossa produtividade, mas também oferecem riscos para segurança do nosso sistema. Selecione a alternativa que apresenta uma forma de mitigarmos os riscos à segurança ao fazer chamadas a APIs.

A

Implementar testes de validação dos dados enviados para uma API e dos resultados que ela gerou.

B

Utilizar apenas APIs desenvolvidas pela própria equipe responsável pelo sistema.

C

Usar APIs apenas de empresas que cobrem pela licença.

D

Reduzir as chamadas para APIs, pois diminuímos as chances de expor o sistema a riscos de segurança.

E

Permitir que a API possa ser chamada por apenas alguns perfis de usuário e realizar o monitoramento dos resultados que ela gera.

A alternativa A está correta.

É cada vez mais comum utilizarmos chamadas a APIs. No entanto, não há como termos garantias de que não possuam vulnerabilidades, especialmente, quando não temos acesso aos pormenores da sua implementação. Por isso, precisamos realizar testes que validem os dados de entrada, bem como os resultados gerados por elas.

Fundamentos da saída

Neste vídeo, veja quais são os fundamentos da saída de um software seguro.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

Todas as etapas são importantes para garantir a segurança. Contudo, a saída merece uma atenção especial, pois pode expor dados confidenciais, favorecer potenciais invasores com informações que eles podem usar para explorar vulnerabilidades em seu software e, ainda, é possível gerar uma saída que seja um problema para outros sistemas, ou prejudicar o suporte à tomada de decisão em vez de ajudar.

Devemos estar atentos a alguns procedimentos sobre o envio da saída para aumentar as chances de que o software seja seguro, tais como:

1

Usar criptografia

A saída deve ser criptografada. Dessa forma, se um invasor conseguir interceptá-la, não vai conseguir compreendê-la.

2

Validar a entrada

Como um invasor pode tentar injetar um código para ser executado por nosso programa, precisamos fazer a validação de todas as entradas usadas para gerar a saída.

3

Limpar a saída

Precisamos garantir que a saída contenha apenas as informações estritamente essenciais para quem vai consumi-la. Portanto, devemos remover qualquer informação confidencial que não seja, de fato, essencial.

4

Usar protocolos de comunicação seguros

Precisamos trabalhar com protocolos de comunicação seguros, como o HTTPS, para fazer a comunicação dos dados. Nesse caso, o envio à saída do nosso sistema, com o objetivo de evitar que invasores em potencial interceptem os dados em trânsito.

5

Limitar a saída para usuários autorizados

Precisamos usar ferramentas e outros recursos que permitam que apenas usuários com a devida autorização consigam ler os dados. Por exemplo, podemos usar uma ferramenta que exija a autenticidade do usuário para exibir os dados da saída com o processo inverso da criptografia.

6 Usar registros

Devemos registrar todas as saídas do sistema e também informações sobre quem as acessou. Obviamente, o interesse é identificar qualquer atividade suspeita.

7

Realizar testes

Devemos sempre realizar testes para qualquer do tratamento de dados, inclusive com a saída. Ao simular situações de ataque com ferramentas de teste automatizadas, temos a possibilidade de identificar quaisquer problemas antes que eles entrem em produção.

Uso da criptografia

Conheça os aspectos básicos do uso da criptografia na saída de software seguro.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

O uso de algoritmos de criptografia é essencial para garantimos que os resultados façam sentido apenas para os usuários autorizados. Algumas das ações para enviar dados seguros são:

1

Usar algoritmos de criptografia fortes

Existem vários algoritmos de criptografia bons, como Advanced Encryption Standard (AES) e Triple Data Encryption Standard (Triple DES).

2

Usar o gerenciamento seguro de chaves

A criptografia só tem utilidade prática se conseguirmos fazer um gerenciamento eficiente das chaves usadas para criptografar e decriptografar os dados. Essas chaves só podem ser acessíveis por usuários autorizados e que possam ter suas ações no sistema rastreáveis.

3

Aplicar protocolos de comunicação seguros

Especialmente, no caso em que os dados da saída serão transmitidos via rede, é fundamental usarmos protocolos de comunicação seguros, como SSL/TLS.

4

Autenticar o remetente e o destinatário

Esse mecanismo de autenticação garante que tanto o remetente quanto o destinatário dos dados estejam autorizados a acessar os dados.

5 Verificar os dados de saída

Antes de criptografar os dados, precisamos garantir que os resultados só contenham informações essenciais para quem vai utilizá-las.

6 Testar a criptografia

Nesse caso, precisamos de ferramentas específicas para automatizar esse processo. O objetivo do teste é garantir que o sistema de criptografia da saída esteja funcionando conforme o esperado.

Protocolos de comunicação seguros

Neste vídeo, compreenda aspectos essenciais sobre os protocolos de comunicação seguros.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

Protocolos de comunicação segura são especialmente essenciais quando fazemos transmissão de dados confidenciais pela internet ou mesmo pela rede interna – raramente sabemos onde está o invasor.

A seguir, listamos alguns dos principais protocolos de comunicação segura:

Transport Layer Security (TLS)/Secure Sockets Layer (SSL)

São usados com frequência para proteger o tráfego de dados da web junto com o protocolo HTTPS. O objetivo básico é proteger os dados transmitidos pela internet por meio do uso de algoritmos de criptografia para garantir que não sejam interceptados ou sofram algum tipo de adulteração.

Secure File Transfer Protocol (SFTP)

É outro protocolo seguro que tem como objetivo proteger dados de arquivos transferidos pela internet. Semelhante aos demais protocolos, ele usa criptografia para proteger os dados em trânsito. Além disso, utiliza recursos como autenticação de chave pública para garantir acesso seguro aos dados.

Internet Protocol Security (IPSec)

É um conjunto de protocolos que serve para proteger comunicações de protocolo de internet (IP) por meio da autenticação e criptografia de cada pacote IP. Especialmente depois da pandemia de covid-19, em que o trabalho virtual se tornou mais comum, esse protocolo ganhou grande popularidade por proteger redes privadas virtuais (VPNs).

Simple Mail Transfer Protocol Secure (SMTPS)

É uma extensão segura do protocolo Simple Mail Transfer Protocol (SMTP) que é usado para enviar e-mail. Ou seja, usa técnicas de criptografia para proteger o e-mail em trânsito e, claro, proteger os dados contra a interceptação e adulteração.

Secure Shell (SSH)

É um protocolo usado para acesso remoto seguro a servidores e outros dispositivos. O SSH faz a autenticação e criptografia seguras para acesso remoto. Portanto, o utilizamos para impedir o acesso não autorizado a dados confidenciais.

Virtual Private Network (VPN)

É uma tecnologia usada para criar uma conexão segura e criptografada entre dois dispositivos – computadores ou servidores – pela internet. Junto com IPSec, o utilizamos para proteger os dados transmitidos entre locais remotos e impedir o acesso não autorizado a dados confidenciais.

Restrição da saída para usuários autorizados

Descubra quais são alguns dos principais fundamentos para restringir a saída para usuários autorizados.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

Há muitos dados circulando pela rede que, se caírem nas mãos erradas, podem gerar enormes prejuízos financeiros e de imagem. Portanto, é fundamental utilizarmos mecanismos que restrinjam a saída apenas a usuários autorizados, para proteger dados confidenciais da saída do nosso sistema seguro.

Apresentamos, a seguir, algumas sugestões que auxiliam no processo de restringir a saída apenas a usuários autorizados:

Implementar controles de acesso

Sempre devemos usar mecanismos de controles de acesso para restringir o acesso a dados confidenciais apenas a usuários autorizados, como o uso de ferramentas que exijam autenticação de usuário e façam o controle de autorização com base no perfil de quem está acessando a saída.

Criptografar dados confidenciais

A criptografia garante que os dados só façam sentido para quem tem acesso à chave de criptografia. Isso, obviamente, é feito por meio de ferramentas que precisam de um gerenciamento explícito. Utilizar criptografia é muito seguro. Mesmo que um invasor tenha acesso aos dados e os modifique, o receptor vai saber que os dados estão corrompidos. Além disso, os dados roubados não farão nenhum sentido para o invasor.

Use protocolos de comunicação seguros

Existem diversos protocolos de comunicação seguros para transmitir dados pela rede - seja internet ou intranet - e garantir a sua integridade.

Limpar saída

As informações geradas por um sistema só devem fazer sentido para usuários com perfis autorizados. Portanto, é nossa obrigação ficar atento ao que pode ser exposto para cada perfil de usuário. Precisamos fazer isso antes de criptografar os dados, pois a construção lógica do que será enviado para saída é de responsabilidade do nosso sistema.

Monitorar a saída

Semelhante ao que fizemos nas demais etapas do ciclo de vida dos dados de um software seguro, também precisamos ter uma rotina de monitoramento bem estabelecida que permita analisarmos o uso da saída do sistema e, se for o caso, detectar qualquer atividade suspeita. Esse processo precisa do apoio de ferramentas específicas e de responsáveis para fazer o acompanhamento de quem está acessando os dados de saída.

Realizar testes de estresse

O objetivo é encontrar vulnerabilidade nos nossos controles de acesso, além de garantir que todo o controle de segurança esteja funcionando conforme o esperado.

Esses controles não têm implementação trivial, mas aumentam nossas chances de construir softwares seguros. É muito mais racional investir em ações preventivas do que atuar em medidas corretivas para tentar reduzir danos causados por um ataque bem-sucedido.

Monitoramento da saída e realização de testes

Neste vídeo, compreenda a importância do monitoramento dos usuários do sistema para complementar a segurança do software. Veja ainda as etapas críticas para garantir a segurança do sistema, incluindo a definição de um plano de teste de segurança, identificação de possíveis ameaças, realização de testes de segurança, geração e monitoramento de logs do sistema.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

Estamos, finalmente, na última etapa para a construção de um software seguro! Nosso foco está no envio seguro da saída. Assim como temos preocupações em receber dados com códigos maliciosos de outros sistemas, também precisamos ter muita atenção para não gerarmos dados que causem danos à segurança. Por isso, a segurança de software deve ser vista como um conjunto de partes que se combinam para reduzir as chances de ataques bem-sucedidos.

Uma forma eficiente de complementar as ações para fazer o envio de uma saída segura é por meio do monitoramento dos usuários que estão usando o sistema. Obviamente, esse tipo de ação, junto com outras que já analisamos, precisa ser justificável. Por exemplo, grandes corporações privadas não têm interesse de que outras empresas conheçam suas estratégias de investimento. Da mesma forma, os setores de inteligência dos governos também não querem que seus adversários – e, às vezes, até aliados – conheçam seus planos militares.

Portanto, a definição de processos de monitoramento da saída e ações para identificar problemas são etapas críticas para garantir a segurança de sistemas. Essa é uma área muito dinâmica, exigindo que nos atualizemos constantemente, principalmente, por causa do avanço das técnicas de inteligência artificial, que podem ser utilizadas tanto para combater invasões, como para tentar realizá-las.

Etapas da monitoração da saída

A seguir, listamos algumas etapas que podemos seguir para monitorar a saída e testar problemas de software seguro:

Estabelecer planos de testes de segurança

Estabelecer um plano de teste de segurança, descrevendo metas, metodologias, ferramentas, frequência de teste e responsáveis.

Identificar possíveis ameaças à segurança

Desenvolver o pensamento de um invasor para podermos criar possíveis cenários de ameaças e identificar vulnerabilidades que nosso sistema pode enfrentar.

Desenvolver e executar testes de segurança

Existem muitos testes que podemos aplicar para verificar, se, de fato, o envio da saída é seguro. Por exemplo, podemos usar ferramentas automatizadas para realizar:

Análise estática: faz a análise do código do sistema.

Análise dinâmica: realiza testes com o sistema em funcionamento.

Teste de penetração: estressa o sistema por meio de busca de vulnerabilidades de segurança bem conhecidas.

Gerar e monitorar os logs do sistema

Só podemos analisar dados se eles existirem. Portanto, é essencial utilizarmos recursos que permitam gerar arquivos de logs do sistema, ao mesmo tempo que precisamos implementar rotinas para detectar atividades suspeitas, como muitas tentativas de login com falha.

Usar técnicas de detecção de anomalias

Atualmente, temos ferramentas que utilizam algoritmos de aprendizado de máquina para detectar padrões incomuns tanto no comportamento do usuário, como nas atividades do sistema.

Realizar revisões de código

Sempre é importante fazer revisões do código para tentar identificar e corrigir vulnerabilidades de segurança.

Manter o software atualizado

Em sistemas complexos, é normal utilizarmos muitos componentes, bibliotecas e outras tecnologias fornecidas por terceiros. Por isso, precisamos mapeá-las e, com regularidade, visitar os sites dos fornecedores oficiais e realizar as atualizações recomendadas.

Verificando o aprendizado

Questão 1

A resposta de um sistema é o resultado de várias etapas, que vão desde a validação da entrada dos dados até o processamento deles. Selecione a alternativa correta sobre uma forma de evitar que um resultado sofra alguma corrupção no processo de envio para os usuários.

A

Avisar os usuários legítimos que os resultados serão enviados em determinados dias e horários.

B

Limitar a quantidade de usuários que podem acessar os resultados.

C

Acrescentar informação inútil aos resultados, de modo que os usuários legítimos possam distinguir o que é ou não válido.

D

Usar dispositivos físicos para enviar os resultados para os usuários legítimos, por meio de serviços de entregas de terceiros.

E

Aplicar técnicas que utilizem protocolos seguros para garantir que os resultados sejam criptografados até chegarem aos respectivos usuários legítimos.

A alternativa E está correta.

A criptografia é a forma mais segura de impedir que um invasor, com acesso aos dados, consiga extrair algum valor deles, ou mesmo ocasionar alguma modificação, pois ela seria facilmente percebida quando os resultados passassem pelo processo de decriptografia. Além disso, precisamos usar os protocolos seguros, pois são responsáveis pelo controle dos dados em trânsito na rede.

Questão 2

De modo geral, associamos o uso da criptografia a uma boa prática de segurança. Selecione, porém, a alternativa a respeito de uma situação em que o uso de criptografia pode ocasionar vulnerabilidades de segurança para um software.

A

Utilizar algoritmos de criptografia simétrica.

B

Utilizar algoritmos de criptografia conhecidos.

C

Fazer um mau gerenciamento das chaves de criptografia.

D

Permitir que apenas alguns usuários possam criptografar os dados.

E

Usar ferramentas de criptografia de terceiros.

A alternativa C está correta.

Caso não haja um gerenciamento adequado das chaves de criptografia, um invasor pode ter acesso a uma chave e obter acesso aos dados processados.

Considerações finais

Estudamos diversas questões relacionadas à construção de um software seguro. Tratamos sobre itens importantes, que vão desde a validação da entrada de dados, como o processamento, chamada de outros programas, até o envio da saída de forma segura. Apesar de serem etapas diferentes, o tratamento geral que precisamos dar a cada uma delas em muitos momentos se assemelham: controle de acesso, monitoramento e aplicação de algoritmos de criptografia.

Na prática, no entanto, cada uma dessas etapas demanda um tratamento especial. Por exemplo, certamente, nossa preocupação da validação de uma entrada é diferente da validação de uma saída, pois, nesse último caso, os dados já passaram por todas as fases anteriores de controle. Portanto, é muito mais difícil detectarmos uma vulnerabilidade na saída do que na entrada de dados. Isso torna o trabalho ainda mais desafiador. Por isso, precisamos estar constantemente nos atualizando e estudando as formas de agregar mais segurança aos nossos softwares!

Explore +

Para complementar as discussões trazidas neste conteúdo, acesse o site oficial da Microsoft e procure por:

Lei de Portabilidade e Responsabilidade do Seguro de Saúde (HIPAA) & Tecnologia de Informações de Saúde para a HiTECH (Economic and Clinical Health Health) Act. Nessa página, você conhecerá melhor esses padrões de segurança e, certamente, vai ampliar a sua visão sobre software seguro.

Usar o PyLint para verificar o código do Python. Lá, você aprende sobre a ferramenta PyLint, que é utilizada na prática para fazer análises estáticas de código no Python que, atualmente, é uma das principais linguagens de programação.

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. Nova York: Pearson Education Inc., 2006.

PRESSMAN, R. S. **Engenharia de Software**. 6. ed. São Paulo: MacGraw-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.