Você pode usar esta documentação para as seguintes unidades de negócio: mercado libre mercado shops mercado envios

Última atualização em 14/03/2023

Validações e requisitos de segurança

Nesta documentação você encontrará práticas de segurança a serem implementadas que são essenciais para sua integração. Implementá-los permitirá fortalecer o núcleo dos processos de segurança do seu desenvolvimento.

Requisitos específicos

Autenticação

A autenticação é um processo que permite conhecer a identidade digital de um usuário (previamente cadastrado) e consiste em uma série de mecanismos que estabelecem que esse usuário é quem diz ser. Esses mecanismos são geralmente agrupados em 3 grandes categorias:

- Algo que eu sei (senha).
- Algo que eu tenho (telefone celular, token de autenticação, yubikey).
- Algo que eu sou (biometria).

A combinação desses fatores é o que gera uma **autenticação robusta**.

```
# Pseudocódigo petición http

si solicitud http es a un endpoint privado:
      si el usuario está autenticado:
            retorna la información solicitada
      de lo contrario
            retorna al login
si solicitud http es a un endpoint público
      retorna la información solicitada
```

Autorização

A autorização é o mecanismo pelo qual controlamos o acesso a um recurso, geralmente com base em uma identidade de usuário previamente validada. É importante diferenciar esse processo de autenticação: a finalidade da autenticação é determinar a identidade de um usuário, enquanto a finalidade da autorização é determinar se essa identidade possui permissões suficientes para executar a ação que está tentando realizar. Essas validações devem ser implementadas no Backend.

Existem inúmeras metodologias para construir matrizes de autorização. Entre os mais conhecidos estão RBAC, ABAC, ACL, MAC, entre outros.

```
# Pseudocódigo

se solicita visualización de una determinada factura (acción)

si solicitud http es a un endpoint privado:
      si el usuario está autenticado:
            si la factura corresponde al comprador o si la factura
            corresponde al vendedor
                  retorna la factura
            de lo contrario
                  retorna un error general de: no se tiene
                  autorización para acceder a la factura
      de lo contrario
            redirecciona al login
```

Validação de data

A validação de dados é a prática que nos permite validar que todos os valores que não se originam em nosso aplicativo estão bem formados, tanto semanticamente quanto sintaticamente, antes de processá-los, salvá-los ou transmiti-los.

Como regra geral e para segurança da aplicação, a validação de dados deve ser implementada nos serviços de backend, pois são fluxos que não podem ser modificados pelo usuário final, diferentemente das validações de frontend que respondem apenas a uma boa resposta.

É uma prática comum realizar a validação de entrada no front-end (lado do cliente), por exemplo, não permitir que o usuário prossiga no fluxo se não colocar algo na forma de um e-mail em um campo de e-mail. Isso é válido do ponto de vista funcional, mas temos que considerar que, ao nível de segurança, um usuário pode ignorar validações do lado do cliente e, assim, enviar dados maliciosos para aplicativos. Por esse motivo, todas as validações de entrada que realizamos no lado do cliente também devem ser feitas no lado do servidor. .

Validação sintática

Por validação sintática queremos dizer que o valor é formado corretamente em relação ao tipo ou estrutura de dados esperados.

Para tipos de dados primitivos, como **float, double, char, boolean, short** or **long**, uma estratégia recomendada é executar typecasting. A ideia simplificada é a seguinte: HTTP é um protocolo baseado em texto, então todo valor de tipo básico chegando em um request é fundamentalmente uma String (ou array de caracteres) com uma certa codificação.

Na etapa sintática, nosso objetivo é converter a String para os valores essenciais correspondentes que estamos esperando. Por exemplo:

Se esperamos um **user_id**, que sabemos ser um **Int**,podemos realizar a validação de estilo:

```
// Pseudocódigo

si user_id es numérico:
      avanza al siguiente paso
```

Validação semântica

A validação semântica envolve entender se o valor está correto em termos do contexto em que está sendo usado.
No contexto de tipos de dados básicos, podemos pensar em exemplos como os seguintes:

- O valor de uma transação não deve ser negativo e deve ter no máximo dois zeros de arredondamento.
- Um número de pedido não deve ser negativo e deve ser um número inteiro.
- Um sobrenome não deve ter caracteres especiais ou não imprimíveis.
- A data de nascimento de um usuário não deve ser inferior ao ano de 1900.

No contexto de tipos de dados complexos, podemos pensar nos seguintes exemplos:

- Por mais que um documento de escritório esteja bem formado, temos que verificar se ele possui macros presentes com código potencialmente vulnerável ou malicioso.
- Se estivermos recebendo imagens, determine qual é o peso máximo e mínimo que elas devem ter, se devemos excluir a metadata ou se contém apenas informações relacionadas a uma imagem.

Advertências

Allow list vs Block list

Nas metodologias de validação podemos utilizar duas grandes estratégias: **Block list** e **Allow List**.Em termos simples, uma metodologia blocklist consiste em definir e buscar todos os valores inesperados para cada entrada. Por exemplo:

```
var sanitizedName = req.name.findAndDelete('>','<','\'','"',';',',','=');
```

O problema geral dessa estratégia é que é complexo conhecer todos os caracteres perigosos ou inesperados, embora hoje tenhamos toda a lista, as tecnologias e padrões mudam, deixando a blocklist possivelmente desatualizada e vulnerável.

Em vez disso, é recomendável usar uma estratégia de **Allow list** que consiste em definir os caracteres ou formatos de input esperados pelo aplicativo. O racional é o seguinte, é muito mais fácil conhecer e definir quais dados estou esperando do que saber o que não estou esperando.

Seguindo essa estratégia, devemos ser o mais restritivos possível, evitando a todo momento, receber dados de entrada que contenham, por exemplo: "." ou "/".

Caso a lista de caracteres seja muito permissiva porque o aplicativo a exige, será necessário implementar controles adicionais, para evitar comportamentos indesejados.

Por exemplo:

- Se o parâmetro for direcionado de uma consulta para um banco de dados, é necessário parametrizar as consultas.
- Se o parâmetro que esperamos for altamente específico, devemos implementar expressões regulares para validar os dados esperados.
- Se o parâmetro é retornado ao frontend, devem ser implementadas validações adicionais, como: codificação ou sanitização.

Requerimentos gerais

A seguir, uma tabela de resumo é fornecida para uso como uma check-list.

Resumen de los recursos disponibles

Controle	Descrição
Autenticação	Validar que todas as funcionalidades do aplicativo estão autenticadas, que a solicitação foi solicitada por um usuário legítimo.
Autorização	Validar se o usuário autenticado anteriormente tem permissões para acessar ou modificar o recurso específico.
Validação de dados	Certificar que todos os valores ou parâmetros sejam validados semântica e sintaticamente antes de serem usados. Preste atenção aos tipos de dados e use um modelo de allow list (valores permitidos) antes da blocklist (valores não permitidos).
Auditoría e registro de logs	Validar se é possível responder "Quem fez o quê, de onde, quando e com que resultado" com os logs disponíveis em cada funcionalidade do aplicativo. Evite o "logging" de informações do usuário (PII) ou confidenciais.
Proteção contra ataques automatizados	Validar se todas as funcionalidades de modificação de recursos (POST, PUT, DELETE) possuem mecanismos para evitar ataques automatizados, como força bruta ou enumeração de informações.
Segredos	Validar se nenhum segredo está "hardcodeado" no código. Os segredos incluem credenciais de acesso, tokens de autenticação, certificados digitais e qualquer outro tipo de informação confidencial. Todos os segredos devem ser guardados e consumidos por mecanismos adequados ao seu tratamento.
Dependências de software de terceiros	Validar se as dependências externas não possuem vulnerabilidades.
Criptografía Aplicada	Validar se os esquemas de hashing, cifrado, assinatura ou geração de sequência random estão sendo usados corretamente, e adequadas para resolver o caso de uso particular.