**Relatório de análise de segurança**

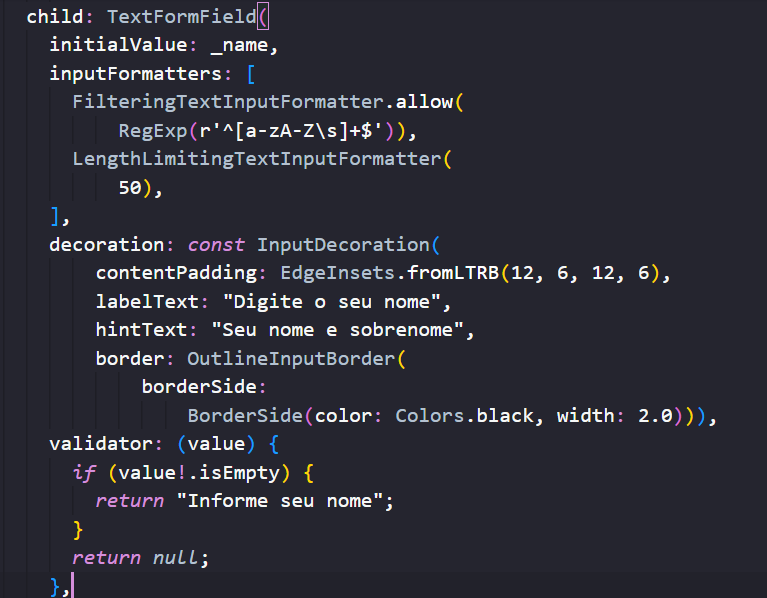
Ariel Andrielli, André Felipe, Jamile de Sousa, João Pedro, Vitor Henrique

Este relatório tem como objetivo apresentar e analisar a capacidade de desenvolvimento de um sistema que adota práticas de segurança. Este documento irá abordar as perspectivas de front-end, back-end e deployment em nuvem, com o intuito de garantir a proteção e integridade dos dados e funcionalidades do sistema.

1. **Perspectiva Front-End**

O projeto foi desenvolvido em Flutter/Dart, tecnologias que podem oferecer uma segurança efetiva por meio de algumas práticas implementadas no front-end. Dentre essas práticas, é possível citar:

* **Proteção contra XSS** – o XSS (Cross-Site Scripting) é uma vulnerabilidade encontrada em aplicações, quando há injeção de códigos maliciosos com objetivo de obter informações sensíveis. Neste sentido, é possível implementar funções como sanitização de entradas de dados, utilização de bibliotecas adequadas, ajuste de escapamento de caracteres, atualizações regulares do produto desenvolvido e armazenamento de informações com criptografia.
* **Autenticação** – a configuração de métodos de autenticação é essencial para garantir a segurança de dados sensíveis dos usuários. Neste sentido, é possível adotar o uso de ferramentas como login por redes sociais ou autenticação por biometria.
* **Atualização constante do aplicativo** – manter os componentes atualizados são essenciais para garantir um ambiente seguro. Se encaixam como práticas de atualização a verificação de compatibilidade de dependências, realização de testes de regressão, práticas de controle de versão (ex. Git) e aplicações de correções de segurança.
* **Validações rigorosas de dados** – certificar-se que apenas informações e formatos válidos estão sendo transitados na aplicação, permitindo somente caracteres e formatos válidos e rejeitando quaisquer dados que potencialmente maliciosos é essencial para manter a aplicação segura. Neste sentido, é possível aplicar, como exemplo, regras de Regex para validação dos formulários presentes no aplicativo.





1. **Perspectiva Back-End**

Com o objetivo de assegurar a proteção dos serviços que constituem a infraestrutura do projeto, foram adotadas medidas com o intuito de fortalecer a segurança contra possíveis ataques virtuais. As medidas de segurança adotas foram subdivididas em segurança:

- da aplicação;

- da infraestrutura.

* **Aplicação:**

As funcionalidades presentes na camada de aplicação do projeto (backend) foram implementadas com técnicas que asseguram a proteção dos dados e credenciais, utilizando configurações armazenadas em variáveis de ambiente restritas ao ambiente de execução. Isso impede o acesso externo a essas configurações, garantindo que não sejam expostas.

No backend do projeto omitimos dados confidenciais do usuário com o uso do método de criptografia ‘Bycript’ para mascarar as informações.

* **Infraestrutura:**

Como o projeto se encontra hospedado em um serviço cloud serverless, asseguramos, assim, a segurança para contra ameaças externas, uma vez que, para que um ataque fosse realizado, o invasor deveria penetrar todas as camadas de segurança da Amazon AWS.

Além do mais, o serviço de hospedagem da Amazon possibilita também a hospedagem do banco de dados e do backend da aplicação, permitindo assim a integridade de toda a estrutura do projeto.

Inacessibilidade das portas:

Os contêineres que alojam os microsserviços da aplicação possuem todas as suas portas TCP/IP não essenciais bloqueadas, garantindo que endereços que possam resultar em vulnerabilidades de segurança não sejam expostos.

* **Inacessibilidade do banco de dados**:

O banco de dados da aplicação possui acesso restrito a qualquer tipo de interação externa, sendo aberto somente para os serviços pertinentes à instância da Amazon EC2.

1. **Perspectiva Cloud**

Considerando que o projeto é composto por um aplicativo em Flutter + Dart com back-end em Spring, é possível realizar o deploy utilizando alguns serviços em nuvem, como exemplo Amazon S3, EC2, Beanstalk e Cognito.

Para todos os serviços, é considerado o modelo de responsabilidade compartilhada, que descreve as responsabilidades entre o cliente e a AWS no contexto nuvem, garantindo a integridade dos serviços, componentes físicos, dados etc.

* S3 – o S3 proporciona funcionalidades como propriedade de objetos com criptografia, gerenciamento de acesso aos dados pelo IAM (serviço de identidade de usuários) e também fornece a possibilidade de integrar a outros serviços, como exemplo o CloudTrail ou GuardDuty para detectar atividades fora do comum em um bucket.
* EC2 – quando falamos de segurança, o EC2 tem duas frentes: a responsabilidade da AWS e a do cliente, garantindo o modelo de responsabilidade compartilhada. Neste sentido, a AWS é responsável pela proteção da infraestrutura dos produtos executados e fornece as ferramentas adequadas para auditoria para o EC2. Já o cliente fica responsável (e deve aplicar) pelo gerenciamento de rede das instâncias EC2 (grupos de segurança ou controle de tráfego), gerenciamento do sistema operacional utilizado, credenciais usadas para se conectar ao EC2 e, por fim, utilizar de forma adequada o IAM para manusear as permissões de cada usuário que terá acesso à instância.
* Beanstalk – o Beanstalk oferece um console intuitivo, onde existem opções para controlar e configurar o acesso ao ambiente aplicando políticas de acesso, bem como podendo implementar firewalls, patches de segurança e configurações de criptografia de dados em trânsito.
* Cognito – o Cognito é uma ferramenta que providencia autenticação e autorização à partir do gerenciamento de usuários conforme a necessidade das aplicações desenvolvidas. Com este serviço, os usuários podem fazer login com usuário + senha ou por formas terceirizadas como login do Google ou Facebook. O Cognito também permite a proteção de dados fornecendo meios de autenticação multifator, uso de criptografia e integração com serviços de auditoria e segurança avançada como o Amazon Macie.