Relatório de Segurança

Este documento relata o processo de um teste de segurança do tipo ***Web Application Pentest - White Box*** na plataforma interna **Unimpact** como requisitado pela UNICAP e pelo Professor Rodrigo Monteiro.

Autor: Vinicius Lôbo - Analista de Segurança
Supervisor: Rodrigo Monteiro - Professor
Cliente: UNICAP - Unimpact Team

Índice

1. Metodologia

1.1 Fase de preparação

Durante a fase de preparação, o analista observou passivamente a aplicação, entendendo melhor suas funcionalidades, identificando suas tecnologias usadas e fazendo uma revisão de arquitetura e código fornecidos.

1.1.1 Escopo

Abaixo segue o escopo definido pelo cliente:

Escopo
Unimpact (localhost)

1.1.1 Credenciais

Foram fornecidas credenciais de usuários com e sem privilégios elevados para os testes.

1.2 Classificação das Vulnerabilidades

A classificação das vulnerabilidade segue o padrão da **MITRE**, constando seu ID de **CVE** e **CVSS v3.1** *Score*:

NÍVEL	CVSS v3.1 SCORE	DESCRIÇÃO
CRÍTICA	9.0 - 10.0	A aplicação ou seus usuários sofrem danos críticos (afetando até mesmo a infraestrutura interna da aplicação e seus servidores) com mitigações complexas, que afetam todos os três pilares da segurança da informação de forma elevada.
ALTA	7.0 - 8.9	A aplicação ou seus usuários sofrem danos altos com mitigações semi-complexas, ferindo os pilares da segurança da informação de forma considerável.
MÉDIA	4.0 - 6.9	A aplicação ou seus usuários podem sofrer danos à sua confiabilidade ou privacidade, com mitigações consideravelmente simples. Pode afetar alguns dos pilares da segurança da informação de forma moderada.
BAIXA	1.0 - 3.9	A aplicação ou seus usuários podem vir a sofrer baixos problemas com uso da aplicação ou seus dados, mas sem nenhuma expectativa de problemas a longo prazo, com mitigação simples. Pode vir a afetar alguns dos pilares da segurança da informação de forma reduzida.
INFORMACIONAL	0.0	Más práticas durante o desenvolvimento da aplicação que podem vir a afetar o uso da mesma ou vazar alguma informação interna.

2. Vulnerabilidades

2.1

Controle impróprio de modificação de atributo de objeto (*Prototype Pollution*)	
CWE-1321	CRÍTICA
Severidade	**ALTO** - Foi possível alterar o *prototype* de um objeto da aplicação por um atributo malicioso, resultando numa leitura arbitrária de comandos na aplicação ou causando negações de serviço.
Viabilidade	**MÉDIA** - É preciso entender o fluxo de autenticação da aplicação e os objetos utilizados por ela para realizar a manipulação de seus atributos.

Ponto afetado
{aplicação}

Impacto
Com uma alteração de um atributo de um objeto, um atacante consegue manipular as funcionalidades daquele objeto no funcionamento em tempo real da aplicação, podendo utilizar de *protypes* vulneráveis para gerar vulnerabilidades que ali não existiam.

Descrição

Ao observar o código, foi possível identificar a utilização de um objeto inseguro durante o fluxo de uso da aplicação, o qual permitiu ao analista alterar seus atributos "poluindo" seu *prototype* e resultando na possibilidade de ataques de negação de serviço e leitura arbitrária de comandos.

Recomendações

Recomenda-se que sejam sanitizadas todas as entradas de usuário que possam ter atribuições de objetos e a paralisação dos seus *prototypes* (um exemplo seria o uso da função: **Object.freeze(Object.prototype)**), impedindo assim uma modificação de seus atributos.

2.2

Mecanismo antiautomação inexistente	
CWE-799	MÉDIA
Severidade	**MÉDIO** - Por não existir a presença de um mecanismo antiautomação na aplicação, um atacante pode realizar ataques automatizados em diversos pontos.
Viabilidade	**SIMPLES** - Nota-se a falta do mecanismo ao realizar qualquer requisição em pontos de autenticação e formulários.

Ponto afetado
{aplicação}

Impacto
Um atacante pode realizar ataques automatizados contra formulários da aplicação, podendo assim se autenticar de forma simples com uso de ataques como ***Brute Force*** ou ***Password Spray***. A vulnerabilidade **2.x(Semântica de senhas inadequada)** em conjunto com este problema pode facilitar os ataques descritos.

Descrição
Ao observar os formulários da aplicação, é possível notar que não há presença de mecanismos antiautomação.

Recomendações
Recomenda-se que a aplicação utilize mecanismos antiautomação, como: CAPTCHA em seus painéis de autenticação e formulários de envio.

2.3

Validação cadastral inadequada	
CWE-284	MÉDIA
Severidade	**MÉDIA** - É possível cadastrar CPFs que não lhe pertencem, além de criar contas com dados gerados proceduralmente.
Viabilidade	**SIMPLES** - É possível constatar a vulnerabilidade durante o processo de cadastro da aplicação e na checagem dos dados pelo código.

Ponto afetado
{aplicação}

Impacto
Um atacante pode cadastrar-se para compra na aplicação com CPFs e números de telefone gerados proceduralmente (com ferramentas como geradores de CPF[1] e de telefone[13]) ou com CPFs que não lhe pertencem.

Descrição
Durante o fluxo de cadastro na aplicação, foi possível observar como a mesma não validava se os dados cadastrais inseridos eram de fato daquele usuário que os inseria. Também foi possível constatar essa vulnerabilidade ao observar como eram feitas as checagens dos dados no código da aplicação

Recomendações

Recomenda-se que a aplicação faça uma validação do tipo ***Know Your Customer*** **(KYC)** para verificar se aquele CPF inserido na conta pertence ao usuário que o está criando e evitar uso de CPFs gerados em sites, e enviar mensagens de confirmação para o número de telefone cadastrado para evitar o mesmo problema.

2.4

Semântica de senhas inadequada	
CWE-521	BAIXA
Severidade	**BAIXA** - Embora ataques de dicionário ainda sejam possíveis, têm chances menores de acerto comparados com ataques como força bruta.
Viabilidade	**SIMPLES** - Pode observar-se no código o uso de uma expressão regular para implementar a política de senhas.

Ponto afetado
{aplicação}

Impacto
Atacantes podem realizar ataques de dicionário contra usuários da aplicação, se baseando em aspectos comuns em criações de senha para criar wordlists personalizadas, como: nome da empresa, datas de nascimento, sequências numéricas, ano de atuação, etc.

Descrição
Ao analisar o código da aplicação, foi possível observar na checagem de senhas de cadastro o uso somente de uma expressão regular para aplicar a política de senhas, implicando que senhas com semântica fraca (como **Unimpact@2023** ou **Senha!1234**) sejam utilizadas.

Recomendações
Recomenda-se que a aplicação aceite somente senhas com entropia superior à 50 bits e faça verificações semânticas para que usuários não cadastrem senhas possivelmente fracas e sejam vítimas de ataques de dicionário.

2.5

Documentação da API exposta	
CWE-200	BAIXA
Severidade	**BAIXA** - A maioria das requisições e alterações que podem ser feitas na API precisam de tokens e chaves com permissões elevadas, a qual usuários comuns não têm acesso.
Viabilidade	**SIMPLES** - Ao observar a presença da API, um leve conhecimento de *endpoints* comumente utilizado por aplicações para chegar ao ponto afetado.

Ponto afetado
{aplicação}

Impacto
Com a documentação da API, é possível um atacante conseguir mais contexto do funcionamento da mesma e realizar requisições com os dados necessários para *endpoints* de criação de contas ou alteração de senhas, por exemplo.

Descrição
Ao realizar as requisições feita pela aplicação, foi possível identificar requisições para a API no ponto afetado. Com uma simples observação do funcionamento da API, foi possível identificar o endpoint da sua documentação.

Recomendações
Recomenda-se que a documentação da API seja acessada somente de forma autenticada, por usuários permitidos e com devido privilégio.

3. Referências e ferramentas

[1] https://www.4devs.com.br/gerador_de_cpf
[2] https://geradornv.com.br/gerador-telefone/