



FGA 0238 - Testes de Software – Turma: 02

Semestre: 2023.2

Equipe: Grupo 10 - Assertivos

Nomes: Letícia Resende Da Silva

Matrículas: 211031118

Artur Jackson Leal Fontinele

211030943

Mateus Vinícius Ferreira Franco

200024868

Luana Souza Silva Torres

190033011

Lucas Rodrigues Monteiro

180125974

Raquel Temóteo Eucaria Pereira da Costa

202045268

Ricardo Augusto Valle Maciel

180077899

Atividade 5 – Teste de Segurança

1 Aplicação Analisada

1.1. Identificação da Aplicação:

- [MEC Energia API](#)
- [MEC Energia Web](#)

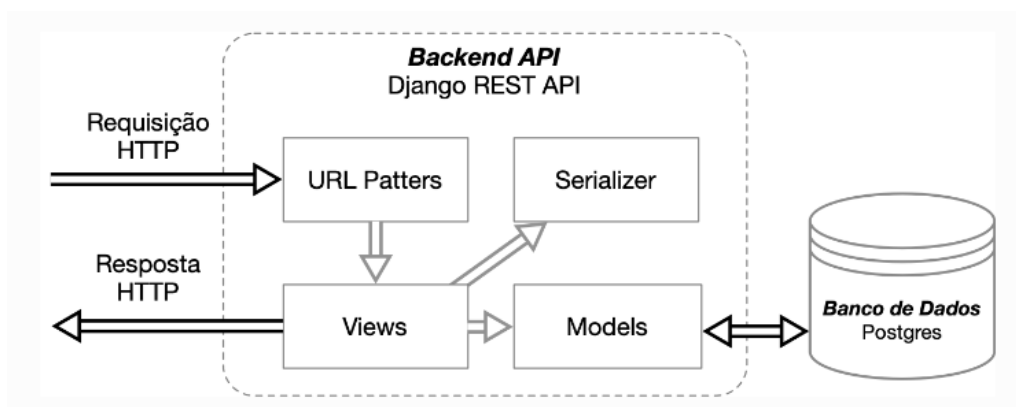
1.2. Descrição

O Sistema MEC-Energia foi desenvolvido com o propósito de oferecer suporte às Instituições de Ensino Superior (IES) no eficiente gerenciamento e na avaliação da adequação de contratos relacionados à conta de energia elétrica. Através do registro detalhado das faturas mensais de energia, o sistema proporciona a geração de relatórios especializados contendo recomendações precisas de ajustes nos contratos vigentes. O objetivo central dessas recomendações é otimizar a utilização de recursos, promovendo uma gestão mais econômica e sustentável da energia elétrica, alinhada às necessidades específicas e à realidade operacional das IES.

Este trabalho tem como objetivo realizar uma análise estática de código para identificar vulnerabilidades de segurança, utilizando o Teste de Segurança Estático (SAST). A ferramenta escolhida para essa análise é o SonarCloud, que examina o código fonte em busca de fragilidades, contribuindo para a identificação precoce de falhas e aprimoramento das práticas de programação. Essa integração reforça o compromisso do Sistema MEC-Energia com a segurança, garantindo eficiência na gestão energética e proteção dos dados, em conformidade com os mais altos padrões de segurança cibernética.

1.3. Linguagens

Python utilizando o Framework Django Rest

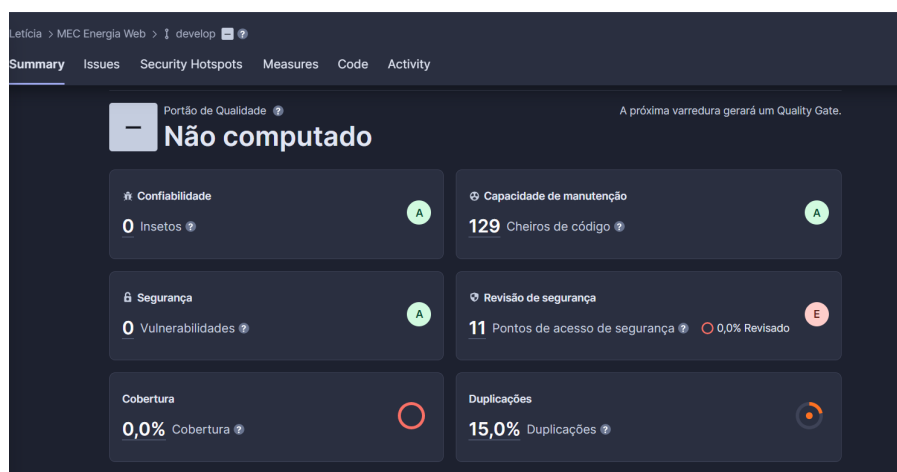


2 Visão Geral do Resultado

MEC Energia API - Não há vulnerabilidades registradas, porém foram verificados 18 hotspots e 13 bugs.



MEC Energia Web - Não há vulnerabilidades registradas, porém foram verificados 11 hotspots.



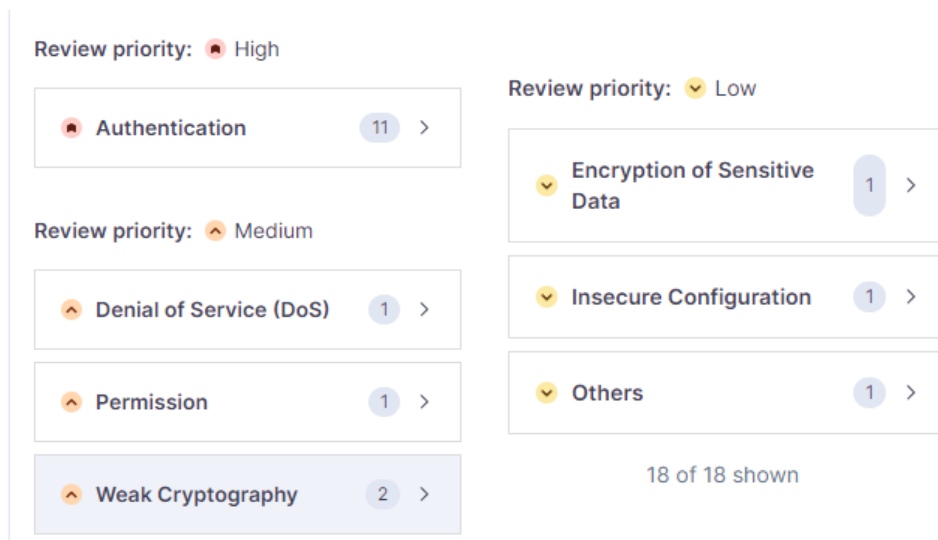
3 Vulnerabilidades

Não foram identificadas vulnerabilidades tanto no Back-End (MEC Energia API) quanto no Front-End (MEC Energia Web).

4 Hot Spots

MEC Energia API - Ao todo, foram verificados 18 Hot Spots, sendo:

- 11 de nível alto
- 4 de nível médio
- 3 de nível baixo



MEC Energia Web:

Ao todo, foram verificados 11 Hot Spots, sendo:

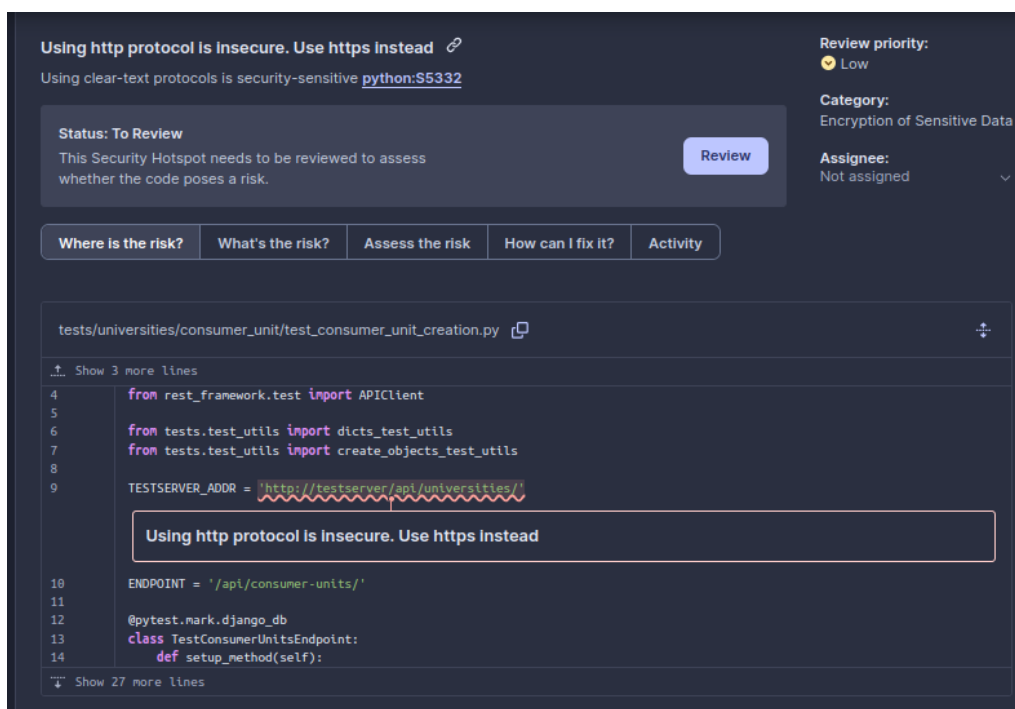
- 7 de nível médio
- 4 de nível baixo



5 Análise das Vulnerabilidades ou Hot Spots

MEC Energia API

5.1. Using http protocol is insecure (Raquel Costa)



Using http protocol is insecure. Use https instead

Using clear-text protocols is security-sensitive [python:S5332](#)

Status: To Review

This Security Hotspot needs to be reviewed to assess whether the code poses a risk.

Review

Where is the risk? **What's the risk?** **Assess the risk** **How can I fix it?** **Activity**

tests/universities/consumer_unit/test_consumer_unit_creation.py

```
4 from rest_framework.test import APIClient
5
6 from tests.test_utils import dicts_test_utils
7 from tests.test_utils import create_objects_test_utils
8
9 TESTSERVER_ADDR = 'http://testserver/api/universities/'
10
11 ENDPOINT = '/api/consumer-units/'
12
13 @pytest.mark.django_db
14 class TestConsumerUnitsEndpoint:
15     def setup_method(self):
```

Using http protocol is insecure. Use https instead

Review priority: Low

Category: Encryption of Sensitive Data

Assignee: Not assigned

5.1.1. Descrição

O uso do HTTP é um ponto crítico em segurança devido à falta de criptografia, expondo informações a interceptação por terceiros. Dados sensíveis, como senhas, ficam vulneráveis a capturas e exploração por atacantes. A ausência de garantias na integridade dos dados possibilita modificações não autorizadas durante a transmissão, facilitando ataques do tipo Man-in-the-Middle. Em um cenário de crescente preocupação com segurança cibernética, a adoção exclusiva do HTTP representa um risco significativo para a privacidade e a segurança online. O SonarCloud identificou corretamente essa vulnerabilidade, reconhecendo a necessidade de abordar o problema iminente de dados suscetíveis a alterações não autorizadas.

5.1.2. Solução

Considerando a segurança na comunicação web, a transição do protocolo HTTP para HTTPS é crucial. A implementação do protocolo HTTPS no servidor web pode ser feita através do Apache ou do Django, para garantir a criptografia adequada dos dados transmitidos.

No contexto do Apache, a configuração do servidor para suportar HTTPS pode ser realizada por meio da aquisição de um certificado SSL/TLS e a integração no arquivo de configuração próprio ("000-default.conf"). Com o Django, é possível habilitar o suporte HTTPS com diversas diretamente no arquivo "settings.py", incluindo as configurações adequadas para redirecionamento seguro, além de utilizar certificados SSL/TLS para garantir uma comunicação segura entre o servidor e o cliente.

MEC Energia Web

5.2. The bode image runs with roto as the default user(Letícia Resende)

5.3.

A imagem do nó é executada com root como usuário padrão.
Certifique-se de que é seguro aqui. [🔗](#)

Executar contêineres como um usuário privilegiado é uma questão de segurança [docker:S6471](#)

Status: Em revisão

Este ponto de acesso de segurança precisa ser revisado para avaliar se o código representa um risco.

Análise

Prioridade da revisão: 🔴 Médio

Categoria: Permissão

Cessionário: Não atribuído ▼

Onde está o risco? Qual é o risco? Avaliar o risco Como posso consertar isso? Atividade

/Dockerfile.dev [📄](#)

```
1  DEEnó:18-alpine
2
3  DIRTRABALHO/aplicativo
```

A imagem do nó é executada com root como usuário padrão.
Certifique-se de que é seguro aqui.

```
# Instala dependências com base no gerenciador de pacotes preferido
CÓPIA DEpacote.json fio.lock* pacote-lock.json* pnpm-lock.yaml* ./
CORRER\
  se [-f fio.lock]; então fio --frozen-lockfile; \
  elif [-f pacote-lock.json]; então npm ci; \
  elif [-f pnpm-lock.yaml]; então o fio global adiciona pnpm && pnpm eu; \
  # Permitir instalação sem lockfile, então o exemplo funciona mesmo sem o
Node.js instalado localmente
  else echo "Aviso: Lockfile não encontrado. É recomendado submeter lockfiles
para controle de versão." && instalação de fios; \
  fi

CÓPIA DESrc./src
CÓPIA DEpúblico ./público
CÓPIA DEnext.config.js .
CÓPIA DETsconfig.json.

# Next.js coleta dados de telemetria completamente anônimos sobre o uso geral.
Saiba mais aqui: https://nextjs.org/telemetry
# Remova o comentário da linha a seguir para desativar a telemetria em tempo de
execução
# ENV NEXT_TELEMETRY_DISABLED 1

# Observação: não exponha as portas aqui, o Compose cuidará disso para nós
```

5.3.1. Descrição

Acima o ConaCloud, identifica um problema relacionado à segurança ao executar um container Docker. A mensagem indica que a imagem do nó está sendo executada com o usuário root como padrão, o que pode representar um risco de segurança, o aviso específico menciona a regra de segurança docker: S6471. Logo, o trecho de código fornecido revela a construção de uma imagem Docker para uma aplicação Next.js, a execução de contêineres como um usuário

privilegiado (root) é considerada uma prática insegura, pois pode aumentar o risco de exploração por parte de ataques maliciosos.

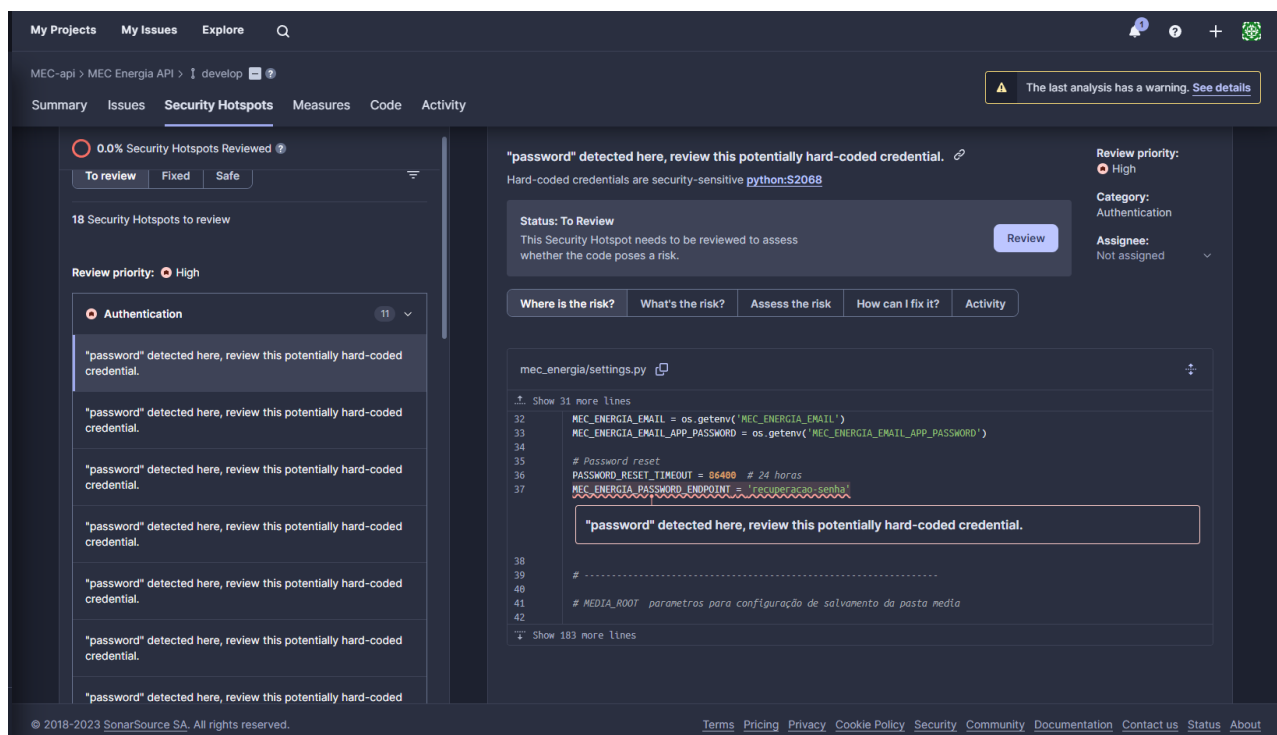
5.3.2. Solução

Uma solução seria garantir que a imagem seja executada de maneira segura, possivelmente alterando o usuário padrão para um não privilegiado e isso pode ser feito na instrução USER no Dockerfile, especificando um usuário que tenha permissões mínimas necessárias para a execução da aplicação.

Dessa forma, o recomendável seria revisar e ajustar as permissões e configurações de segurança em todo o Dockerfile para garantir boas práticas de segurança ao construir e executar containers Docker.

MEC Energia API

5.4. Authentication "password" detected, potentially hard-coded credential(Artur Jackson)



The screenshot displays the SonarSource IDE interface for the 'MEC Energia API' project. The 'Security Hotspots' tab is active, showing a list of 18 hotspots to review. The first hotspot is titled '"password" detected here, review this potentially hard-coded credential.' and is categorized as 'Authentication' with a 'High' review priority. The details panel on the right provides more context, stating that hard-coded credentials are security-sensitive and need to be reviewed. The code snippet shown in the details panel is from 'mec_energia/settings.py' and includes the following lines:

```

32 MEC_ENERGIA_EMAIL = os.getenv('MEC_ENERGIA_EMAIL')
33 MEC_ENERGIA_EMAIL_APP_PASSWORD = os.getenv('MEC_ENERGIA_EMAIL_APP_PASSWORD')
34
35 # Password reset
36 PASSWORD_RESET_TIMEOUT = 86400 # 24 horas
37 MEC_ENERGIA_PASSWORD_ENDPOINT = 'password-reset'

```

The hotspot is located on line 37, where the value 'password-reset' is assigned to the 'MEC_ENERGIA_PASSWORD_ENDPOINT' variable. The interface also shows a 'Review' button and a 'Review priority' dropdown set to 'High'. The bottom of the screen displays the SonarSource logo and copyright information: '© 2018-2023 SonarSource SA. All rights reserved.'

Imagem 1 - password identificada



The screenshot shows the SonarSource web interface. The top navigation bar includes 'My Projects', 'My Issues', and 'Explore'. The breadcrumb trail indicates the project is 'MEC-api > MEC Energia API' and the branch is 'develop'. The 'Security Hotspots' tab is active, showing a list of 11 hotspots under the 'Authentication' category. The selected hotspot is titled '"password" detected here, review this potentially hard-coded credential.' and is categorized as 'Authentication' with a 'High' review priority. The status is 'To Review'. The code snippet shows a Python file 'scripts/seed.py' with a hard-coded password 'user' assigned to 'password' in a 'UniversityUser.objects.create()' call. The interface includes a 'Review' button and tabs for 'Where is the risk?', 'What's the risk?', 'Assess the risk', 'How can I fix it?', and 'Activity'.

Imagem 2 - password identificada

The screenshot shows the SonarSource web interface for a different file, 'scripts/seed_UFMG.py'. The layout is identical to the previous image, with the 'Security Hotspots' tab active. The selected hotspot is also titled '"password" detected here, review this potentially hard-coded credential.' and is categorized as 'Authentication' with a 'High' review priority. The status is 'To Review'. The code snippet shows a Python file 'scripts/seed_UFMG.py' with a hard-coded password 'ufmg' assigned to 'password' in an 'admin_university_user = UniversityUser.objects.create()' call. The interface includes a 'Review' button and tabs for 'Where is the risk?', 'What's the risk?', 'Assess the risk', 'How can I fix it?', and 'Activity'.

Imagem 3 - password identificada

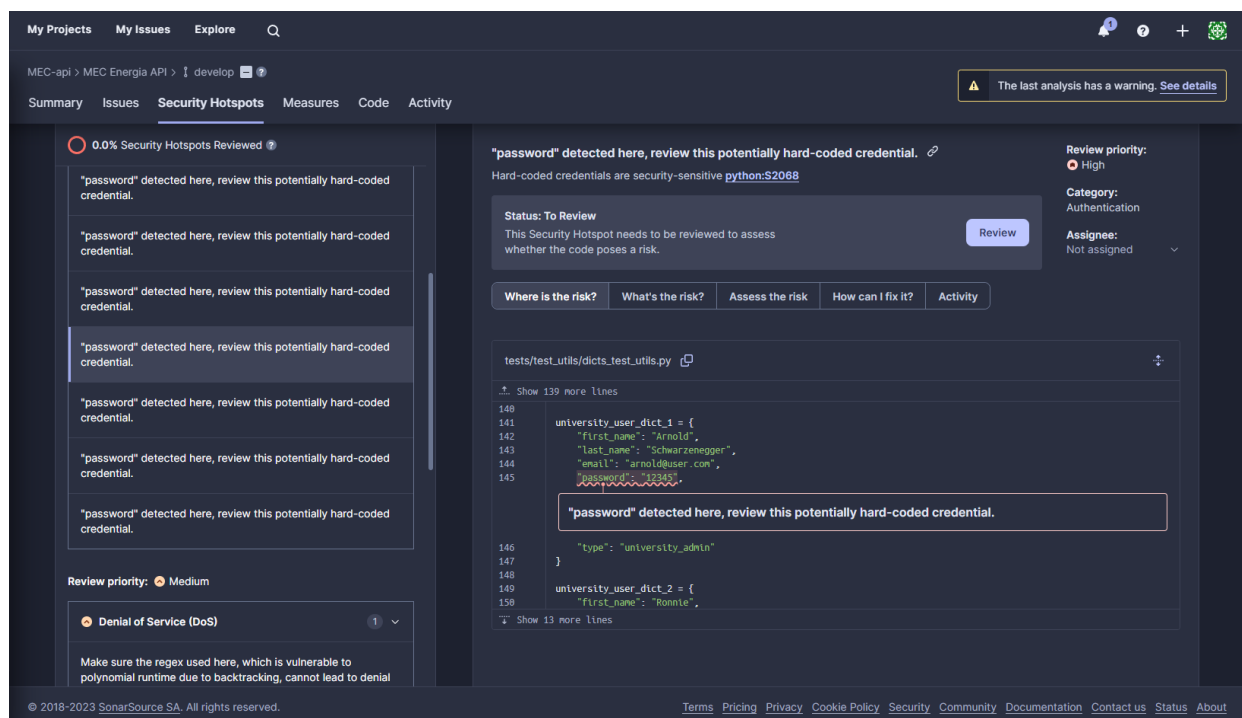


Imagem 4 - password identificada

5.4.1 Descrição

Este é um hotspot de alta prioridade, onde o SonarCloud detectou a palavra “password” em alguns lugares do código do sistema, especificamente foi detectada em 11 lugares. Preferi colocar somente 4 imagens que mostram alguns lugares onde são encontradas essas palavras “password” para que o documento não fique muito grande.

A detecção destas palavras nos indica que há uma credencial, uma provável senha, em "hard-coded", ou seja, estão armazenadas no código fonte no sistema. "Hard coded" é um termo usado na programação para descrever a prática de incluir valores diretamente no código-fonte de um programa. Então nos é alertado como um hotspot de alta prioridade por se tratar de uma senha, que é considerada um conteúdo sensível, estando vulnerável para ser acessada por todos que têm acesso ao código fonte. Portanto, trata-se de um problema real por se tratar de um software open-source, onde uma pessoa má intencionada pode obter acesso a essas credenciais expostas, deixando o sistema vulnerável.

5.4.2 Solução

Uma forma de solucionar o problema seria armazenar essas credenciais em um banco de dados protegido, utilizando uma função de conexão para acessar e recuperar as credenciais quando necessário, garantindo que apenas usuários autorizados tenham acesso às informações. Outras opções seriam armazenar as credenciais em um arquivo de configuração que não é enviado para o repositório, utilizando a função `getenv()`, para se obter as credenciais

do sistema operacional ou utilizar também de algum serviço de gerenciamento de segredos, para armazenar essas credenciais de forma segura, utilizando funções de gerenciamento para acessar e armazenar as credenciais.

MEC Energia API

5.5 Denial of Service (Ricardo Augusto Valle Maciel)

5.5.1 Descrição

- **Expressões regulares**

Expressões regulares (podendo ser conhecidas também por *regex* ou *regexp*) são notações flexíveis utilizadas para descrever padrões em escritas. São extremamente úteis no processamento de textos e realiza 3 principais operações:

1. Validação de dados: Verificar se uma determinada cadeia de caracteres atende a um determinado formato ou padrão;
2. Busca de padrões: Localizar ocorrências específicas de texto em uma string;
3. Substituição de texto: Substituir partes de uma string que correspondem a um padrão por outro texto seguindo outro padrão diferente.

Por exemplo, para buscar datas no padrão dd/mm/aaaa em um texto, pode-se utilizar a expressão regular `^[0-9]{2}/[0-9]{2}/[0-9]{4}$`. Nesse caso, `^` e `$` indicam o início e o fim da string, respectivamente; `[0-9]{2}/` indica que irá ser feita uma busca em dois dígitos de 0 a 9 um ao lado do outro, seguido de uma barra /; e `[0-9]{4}` indica que será feita uma busca em 4 dígitos de 0 a 9 um ao lado do outro, no final formato uma string no formato `dd/mm/aaaa`.

- **Denial of Service (DoS)**

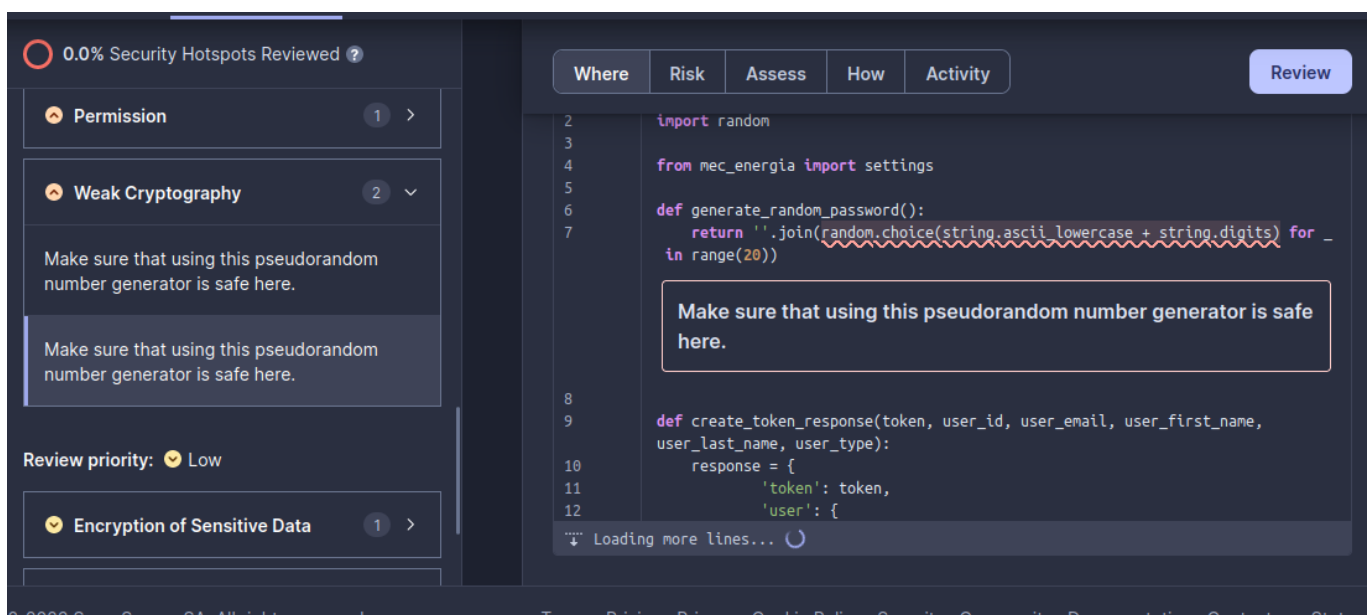
O ataque do tipo DoS (Denial Of Service, em inglês), também conhecido como ataque de negação de serviço, é uma tentativa de fazer com que aconteça uma sobrecarga em um servidor ou computador comum para que recursos do sistema fiquem indisponíveis para seus utilizadores. A mensagem exibida no Sonar Cloud (“Make sure the regex used here, which is vulnerable to polynomial runtime due to backtracking, cannot lead to denial of service.”) explora o fato que a maioria das implementações de expressões regulares podem atingir situações extremas que fazem o sistema funcionar lentamente, para avisar ao usuário que esse erro de backtracking polinomial pode levar a um aumento exponencial no tempo de execução da aplicação, podendo causar uma indisponibilidade de recursos do sistema para seus utilizadores.

5.5.2 Solução

Uma abordagem para resolver o problema é simplificar a expressão regular para torná-la mais eficiente e menos suscetível a um backtracking polinomial. Pode ser feita uma análise dos emails que costumam ser cadastrados no sistema, a fim de verificar se são feitas capturas desnecessárias de subgrupos dentro das expressões e, com isso, melhorar a implementação da expressão.

MEC Energia API

5.6 Weak Cryptography (Luana Souza Silva Torres)



The screenshot displays the SonarCloud interface for a security hotspot. On the left sidebar, the 'Weak Cryptography' issue is highlighted with a count of 2. The main panel shows the code snippet where the issue was found. The code is in Python and defines a function to generate a random password. A red squiggly line underlines the `random.choice(string.ascii_lowercase + string.digits)` expression, indicating the source of the weak cryptography. A tooltip message states: "Make sure that using this pseudorandom number generator is safe here." The interface also shows a 'Review' button and a 'Loading more lines...' indicator at the bottom of the code view.

```
2 import random
3
4 from mec_energia import settings
5
6 def generate_random_password():
7     return ''.join(random.choice(string.ascii_lowercase + string.digits) for _
8                     in range(20))
9
10 def create_token_response(token, user_id, user_email, user_first_name,
11                          user_last_name, user_type):
12     response = {
13         'token': token,
14         'user': {
```

5.6.1 Descrição

Para a realização da criptografia, o sistema utiliza a função 'Random' do Python. Neste cenário, o SonarCloud identifica como um problema de criptografia fraca pois a função 'random' gera números pseudoaleatórios, não sendo considerada uma função segura para este propósito devido a previsibilidade da função.

5.6.2 Solução

Considerando que a aleatoriedade é importante para este contexto, especialmente para a geração de uma senha aleatória, pode-se utilizar a biblioteca 'secrets' do Python. Essa biblioteca fornece funções para gerar caracteres aleatórios de maneira mais segura. Utilizar a função `secrets.choice` é uma boa alternativa para solucionar este problema.

MEC Energia API

5.7 Insecure Configuration (Mateus Vinícius Ferreira Franco)

The screenshot displays the SonarCloud interface for the project 'mateusfrancovinicius > MEC Energia API > develop'. The 'Security Hotspots' tab is active, showing a list of hotspots on the left and a detailed view of a specific hotspot on the right.

Left Panel (Security Hotspots List):

- 0.0% Security Hotspots Reviewed
- Authentication (11)
- Denial of Service (DoS) (1)
- Permission (1)** (highlighted with a red box)
- Weak Cryptography (2)
- Encryption of Sensitive Data (1)
- Insecure Configuration (1)

Right Panel (Detailed View of 'Permission' Hotspot):

- Title:** Copying recursively might inadvertently add sensitive data to the container. Make sure it is safe here.
- Review priority:** Medium
- Category:** Permission
- Assignee:** Not assigned
- Status:** To Review
- Description:** Recursively copying context directories is security-sensitive `docker:S6470`. This Security Hotspot needs to be reviewed to assess whether the code poses a risk.
- Code Snippet:** A Dockerfile snippet is shown, highlighting the `COPY` command: `COPY . .`. The warning message is repeated below the snippet.

5.7.1 Descrição

Esse "hotspot" refere-se a uma preocupação de segurança ao construir imagens Docker a partir de um Dockerfile, especialmente quando o `COPY` ou `ADD` são usados para copiar diretórios inteiros ou múltiplos itens cujos nomes são determinados durante o tempo de compilação.

Quando você constrói uma imagem Docker, você fornece um diretório de contexto para o daemon do Docker. Esse contexto inclui o Dockerfile e todos os arquivos necessários para a compilação bem-sucedida da imagem. Isso pode incluir código-fonte de aplicativos, arquivos de configuração e outros pacotes ou componentes necessários.

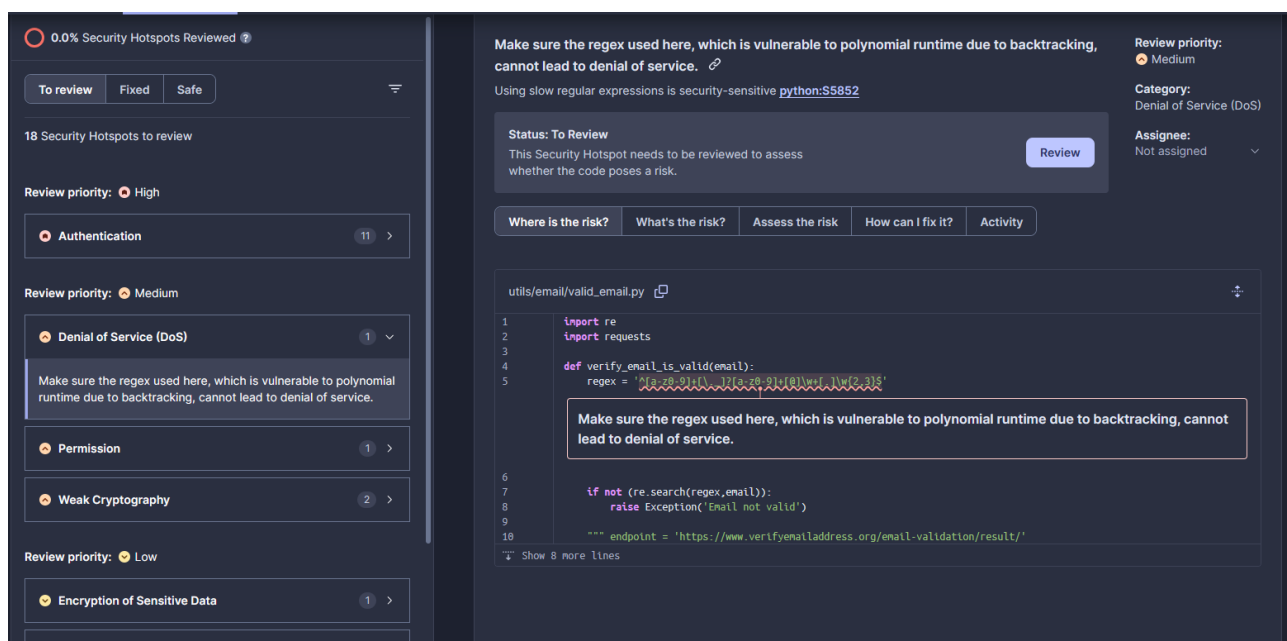
A preocupação específica aqui é que, ao usar COPY ou ADD de forma recursiva para copiar diretórios inteiros, você pode acabar copiando arquivos inesperados para o sistema de arquivos da imagem. Isso pode resultar na inclusão acidental de dados confidenciais ou indesejados na imagem, o que pode representar um risco de segurança.

5.7.2 Solução

Para mitigar esse risco, é importante ser seletivo ao usar COPY ou ADD e garantir que você está copiando apenas os arquivos necessários para a imagem, evitando a inclusão de dados confidenciais ou desnecessários. Isso pode ser feito especificando arquivos específicos em vez de copiar diretórios inteiros, quando apropriado, ou usando .dockerignore para excluir arquivos e diretórios indesejados do contexto de compilação.

MEC Energia API

5.8 Encryption of Sensitive Data (Lucas Rodrigues)



The screenshot displays a security tool interface. On the left, a sidebar shows a list of security hotspots categorized by review priority: High (Authentication), Medium (Denial of Service (DoS), Permission, Weak Cryptography), and Low (Encryption of Sensitive Data). The main panel on the right shows the details of a selected hotspot. It includes a warning message: "Make sure the regex used here, which is vulnerable to polynomial runtime due to backtracking, cannot lead to denial of service." Below this, there is a code snippet for a Python function named `verify_email_is_valid`. The code uses a regular expression to validate email addresses. A warning box highlights the regex pattern, stating: "Make sure the regex used here, which is vulnerable to polynomial runtime due to backtracking, cannot lead to denial of service." The code snippet is as follows:

```
utils/email/valid_email.py
1 import re
2 import requests
3
4 def verify_email_is_valid(email):
5     regex = '[a-z0-9+!_]{1}[a-z0-9+!_]{0,1}[a-z0-9+!_]{1}[a-z0-9+!_]{0,1}'
6
7     if not (re.search(regex, email)):
8         raise Exception('Email not valid')
9
10 """ endpoint = 'https://www.verifyemailaddress.org/email-validation/result/'
11
12 Show 8 more lines
```

5.8.1 Descrição

A funcionalidade de criptografia de dados sensíveis é uma medida de segurança implementada em um aplicativo para proteger a confidencialidade das informações dos usuários.

Ao utilizar essa funcionalidade, o aplicativo criptografa os dados sensíveis antes de serem armazenados ou transmitidos, tornando-os ilegíveis e inacessíveis para pessoas não autorizadas. Isso reduz significativamente o risco de exposição de informações pessoais, como senhas, números de cartão de crédito e outras informações confidenciais.

Ao receber dados sensíveis do usuário, o aplicativo utiliza algoritmos de criptografia avançados para transformar essas informações em um formato criptografado. Esses algoritmos são projetados para garantir que somente o destinatário autorizado possa descriptografar e acessar os dados originais.

O Sonar Cloud identificou uma falha na gestão das chaves, se as chaves não forem adequadamente gerenciadas, pode haver riscos de perda ou comprometimento. Por exemplo, se as chaves são armazenadas em locais não seguros ou compartilhadas indevidamente, pode haver a possibilidade de acesso não autorizado ou perda acidental.

5.8.2 Solução

É importante implementar práticas adequadas de gestão das chaves, como armazená-las em um ambiente seguro e protegido contra acessos não autorizados. Isso pode incluir o uso de sistemas de gestão de chaves seguros e a restrição de acesso somente a usuários autorizados.

6 Conclusão

A avaliação do SonarCloud sobre o projeto da MEC Energia revelou uma quantidade significativa de falsos positivos relacionados aos testes unitários desenvolvidos. Entretanto, destacam-se questões críticas de segurança, como a persistência do uso do protocolo HTTP. Essas situações demandam atenção imediata, sugerindo fortemente a necessidade de reconfiguração e adaptação para padrões mais seguros no ambiente web. O foco na resolução desses problemas de segurança é crucial para garantir a integridade e confiabilidade do sistema, mitigando potenciais vulnerabilidades e fortalecendo a postura de segurança da aplicação. Também foi identificada uma vulnerabilidade crítica na execução do container Docker para a aplicação Next.js. A revisão e o ajuste abrangente das permissões e configurações de segurança são fortemente recomendados para garantir boas práticas ao construir e executar containers Docker. O sistema também possui credenciais/senhas hardcoded, além de utilizar funções que geram números pseudoaleatórios para senhas, o que pode ser um problema real que coloca a segurança do sistema em risco. Por fim, o teste evidencia a necessidade crítica de aprimorar a



configuração de segurança da imagem para proteger contra possíveis explorações maliciosas ao implementar a aplicação Next.js em um ambiente Docker.