

DONT BREAK OUR

RULES

AUTOMATIZING REQUIREMENTS TESTS

DEVSECOPS
VILLAGE



H2
HC
CONFERENCE

Speakers



Allan Kardec

DevSec Lead @ Hakai



Fernando Guisso

AppSec @ Will Bank

Agenda

01 O PROBLEMA

A dor de cabeça diária

02 DESAFIOS

Como fazer o dev trabalhar?

03 O ARSENAL

Preparando o armamento

04 O PLANO

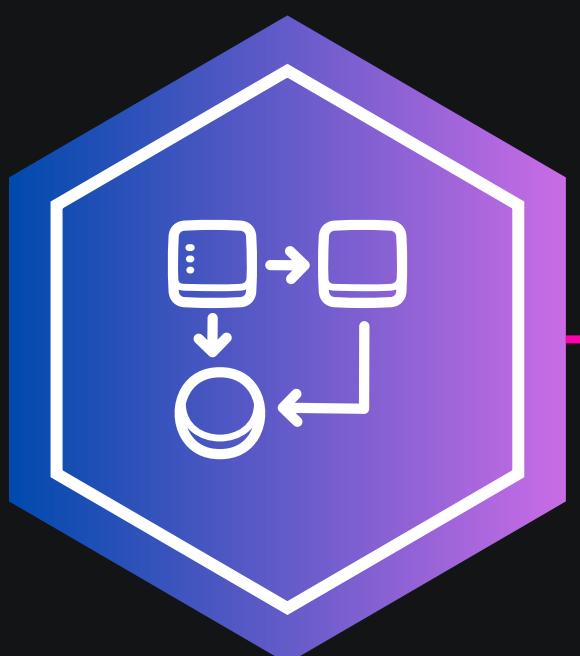
Como vamos fazer?

Show me the code

05 RESULTADOS

protegidos hoje e "sempre"

Etapas

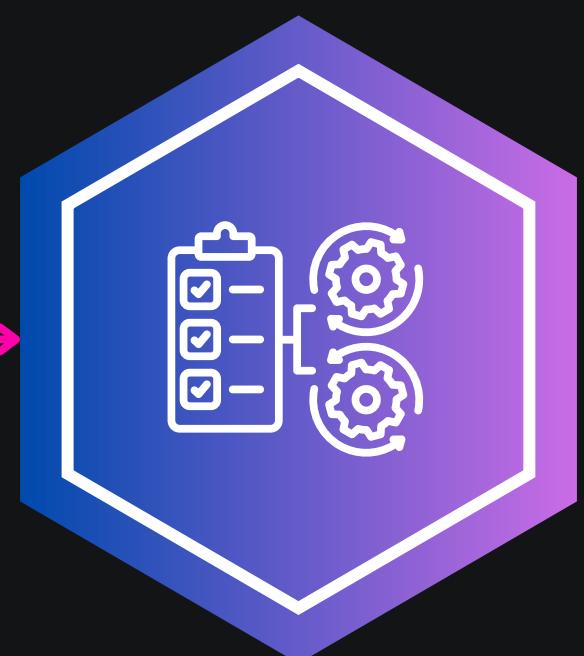


THREAT MODEL

Entender como funciona a aplicação e suas prioridades

REQUISITOS DE SEGURANÇA

Definir majoritariamente quais os principais controles a serem implementados.



TESTES

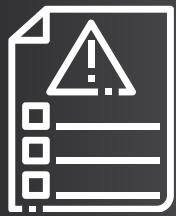
Após OK do time de desenvolvimento, testar se os requisitos foram implementados



Problemas

Incisistência

Analistas diferentes podem interpretar e testar os requisitos de maneiras distintas.



Tempo

Processo repetitivo que demanda muitas horas e consome tempo hábil do time.



Cobertura

Testes manuais tendem a focar em aplicações de maior risco deixando outros vulneráveis



Rastreamento

Documentações inconsistentes dos resultados e correções.



Desafios

COMPLEXIDADE NAS TRADUÇÕES DE REQUISITOS

Converter regras de negócio em testes automatizados

INTEGRAÇÃO DE TECNOLOGIAS

Calibrar as ferramentas para reduzir alarmes falsos

MANUTENÇÃO DOS TESTES

Necessidade de atualização constante conforme a aplicação evolui

COBERTURA DE TESTES

Garantir que todos os cenários críticos sejam verificados

PERFORMANCE

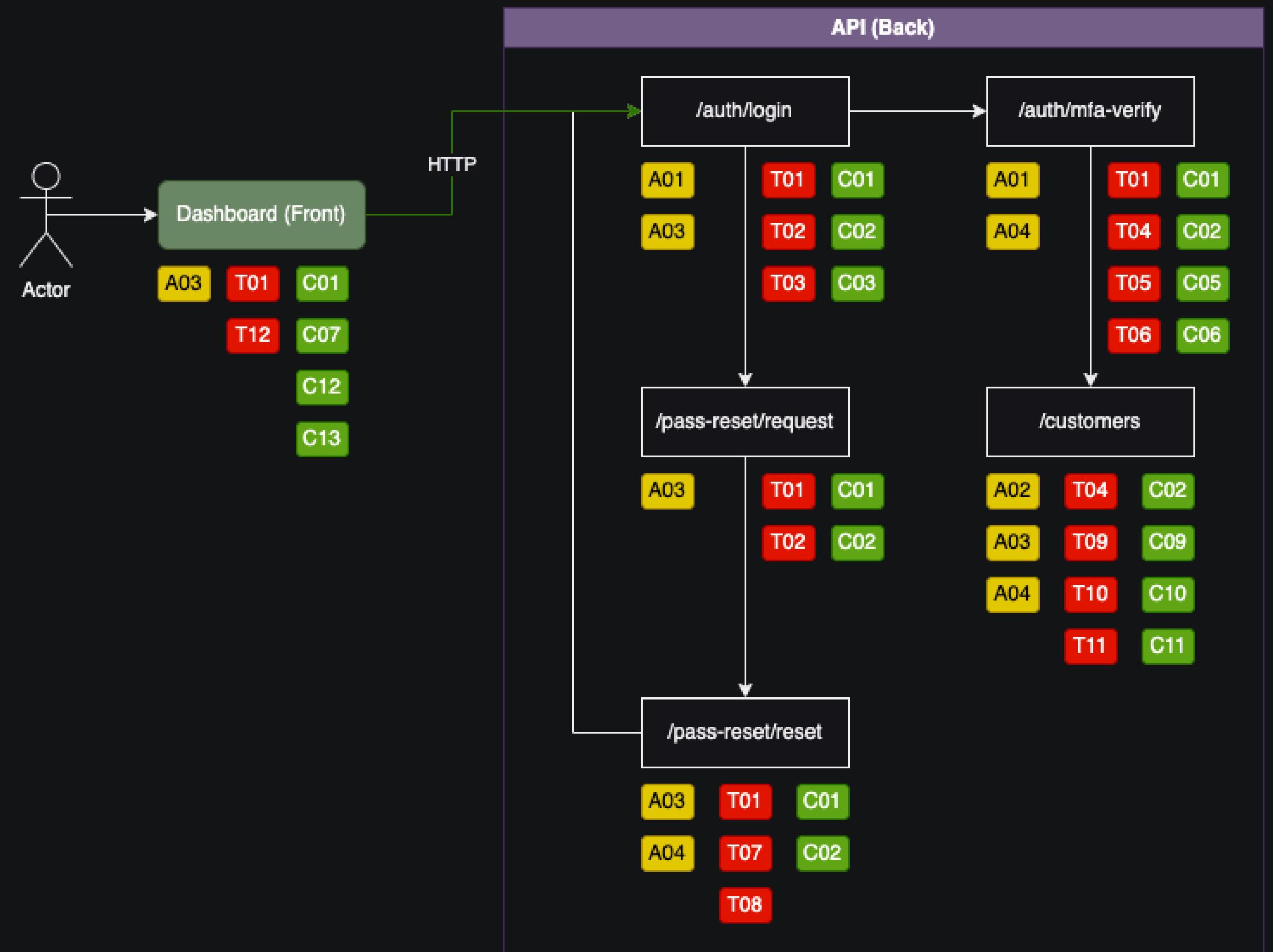
Equilibrar a execução dos testes com o tempo de build

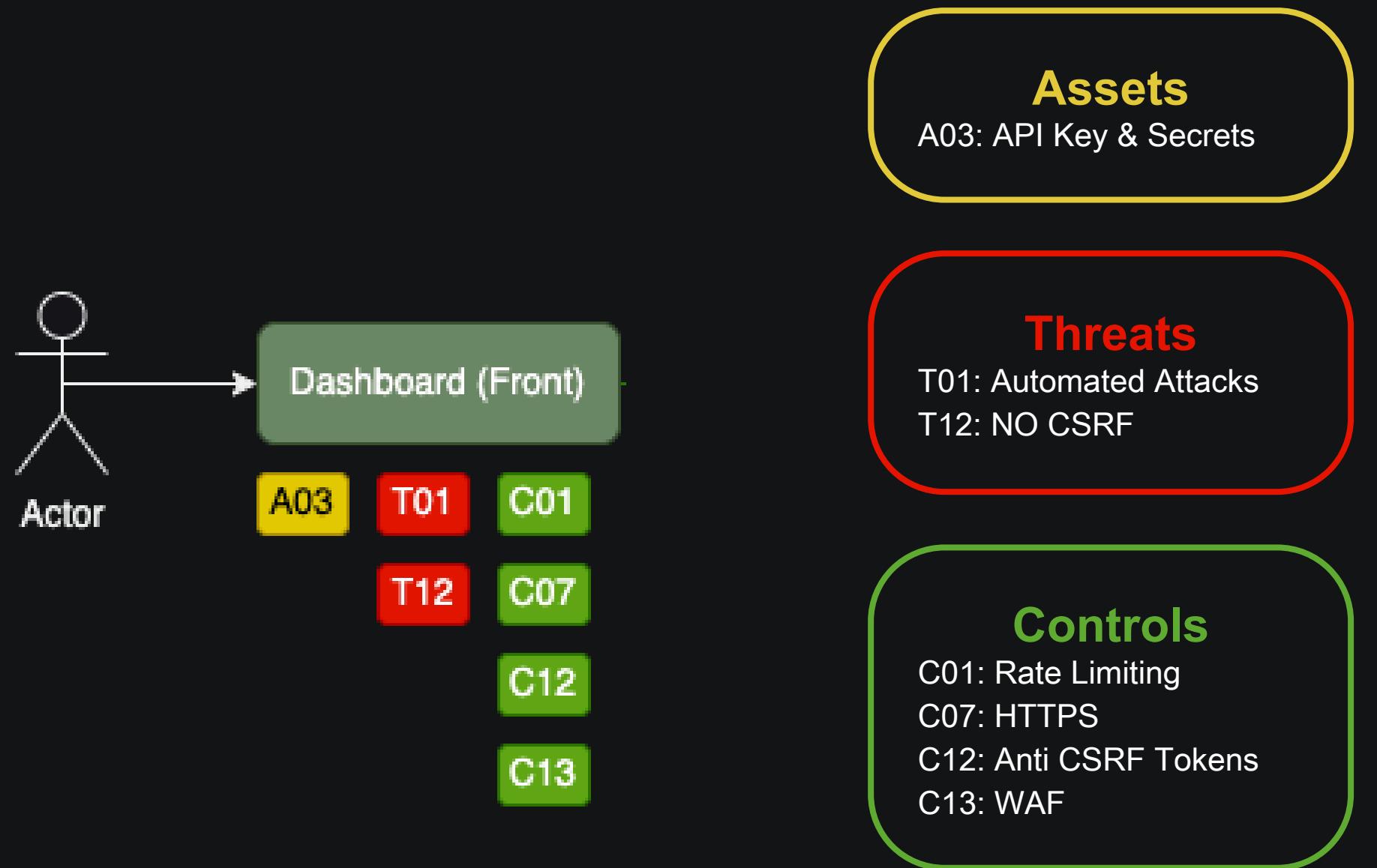
Dashboard App

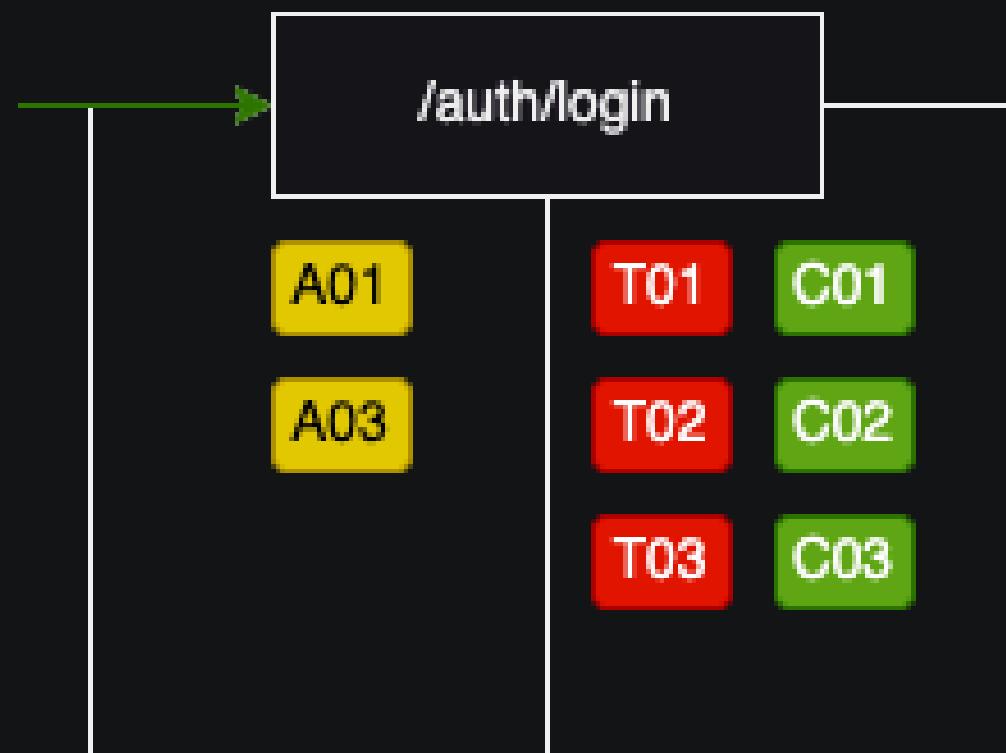
Desenvolvemos uma aplicação de dashboard de um Back Office de uma loja de e-commerce. A ideia é apresentar as informações financeiras sobre os cartões de crédito dos clientes.

Nessa aplicação deixamos de propósito diversas vulnerabilidades no código e a partir dela, vamos fazer nossas prova de conceito.









Assets

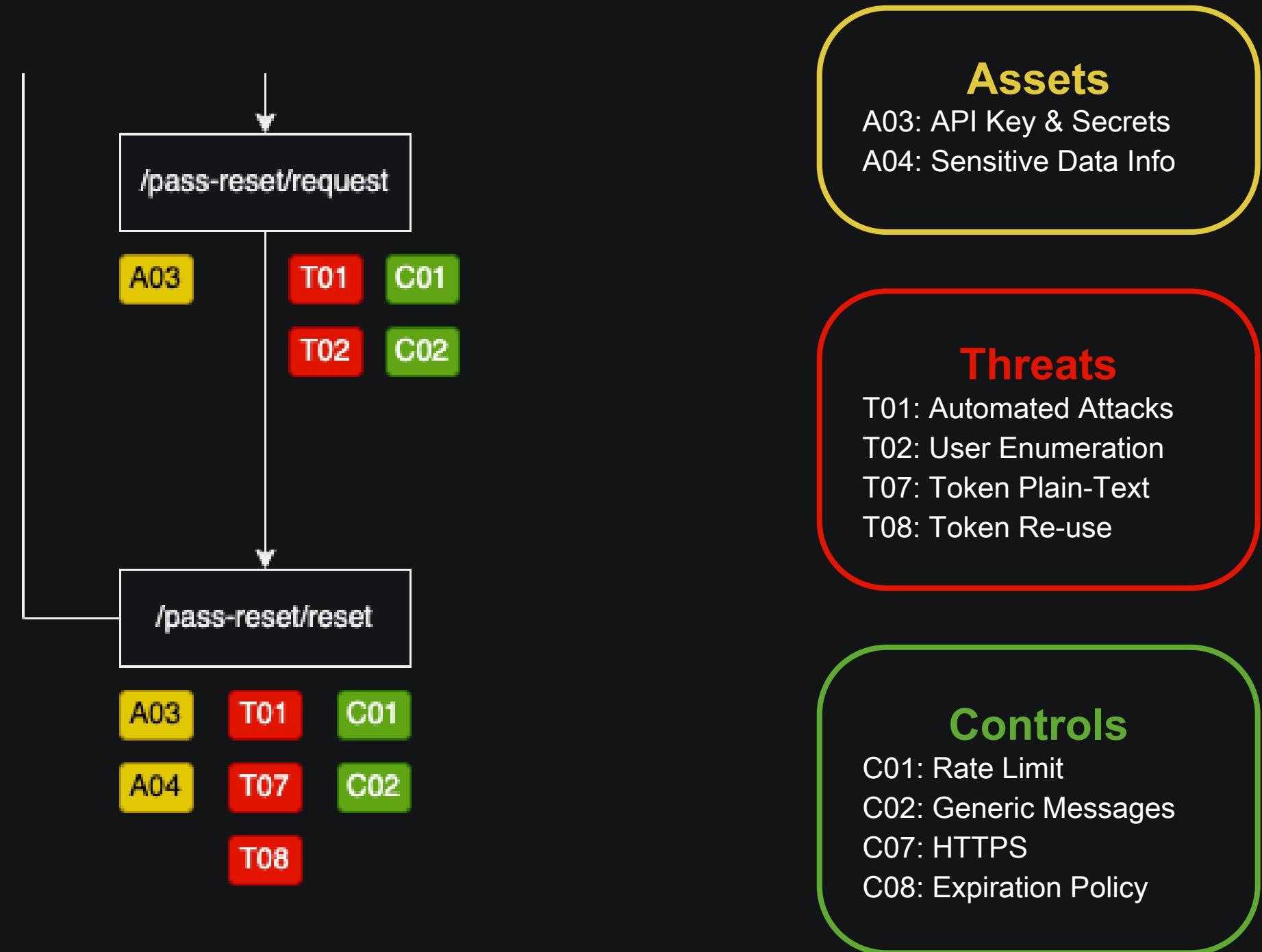
A01: User Credentials
A03: API Key & Secrets

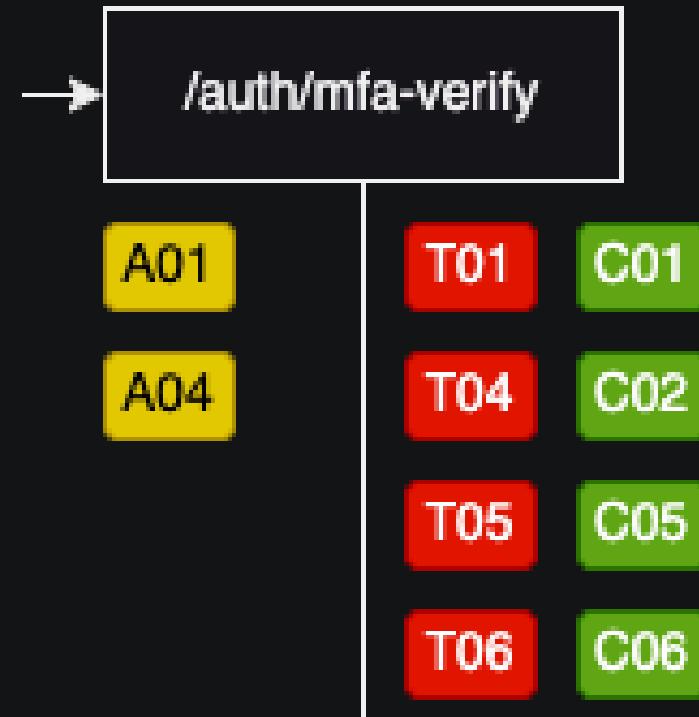
Threats

T01: Automated Attacks
T02: User Enumeration
T12: SQL Injection

Controls

C01: Rate Limiting
C02: Generic Errors
C03: Prepared Statements





Assets

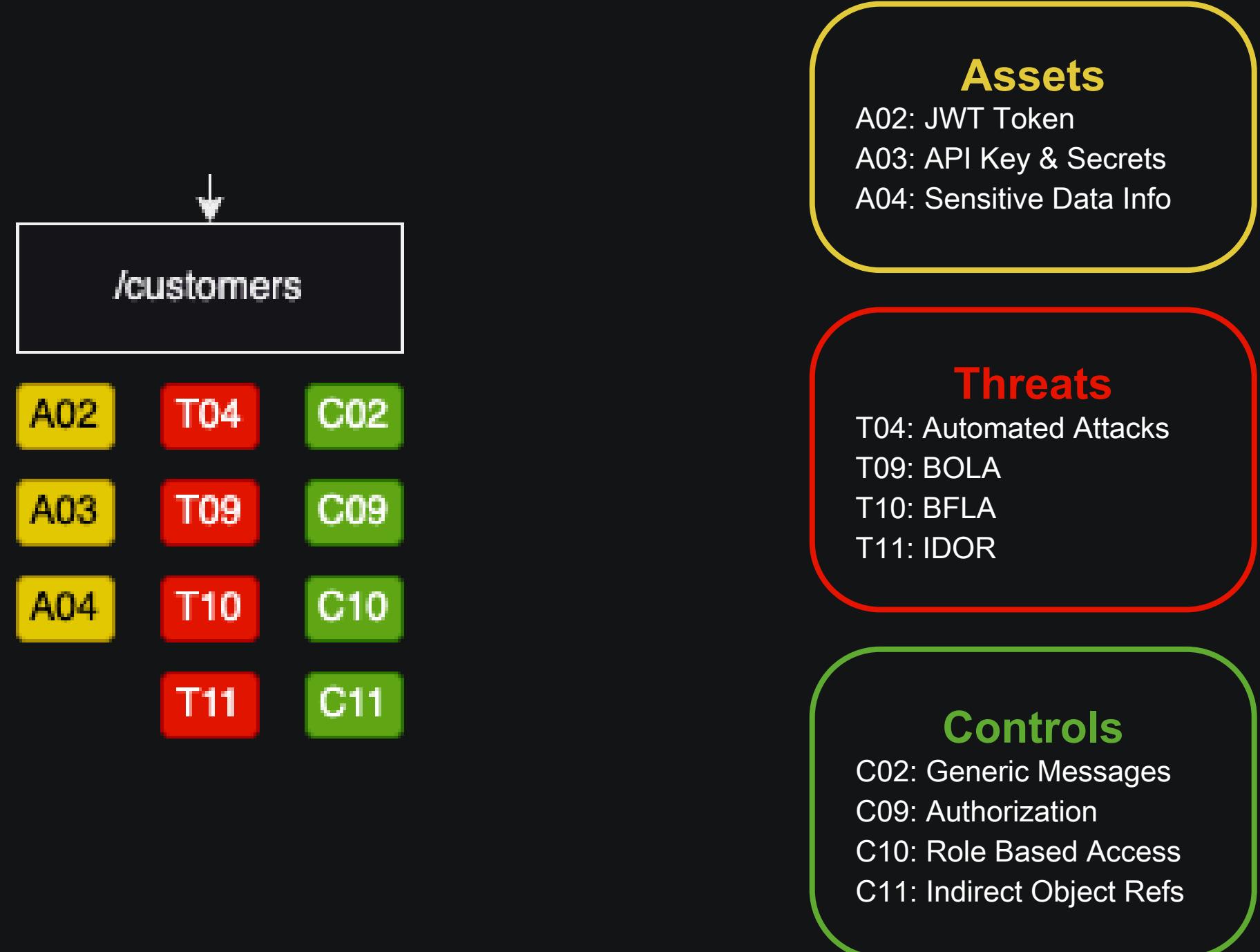
A01: API Key & Secrets
A04: Sensitive Data Info

Threats

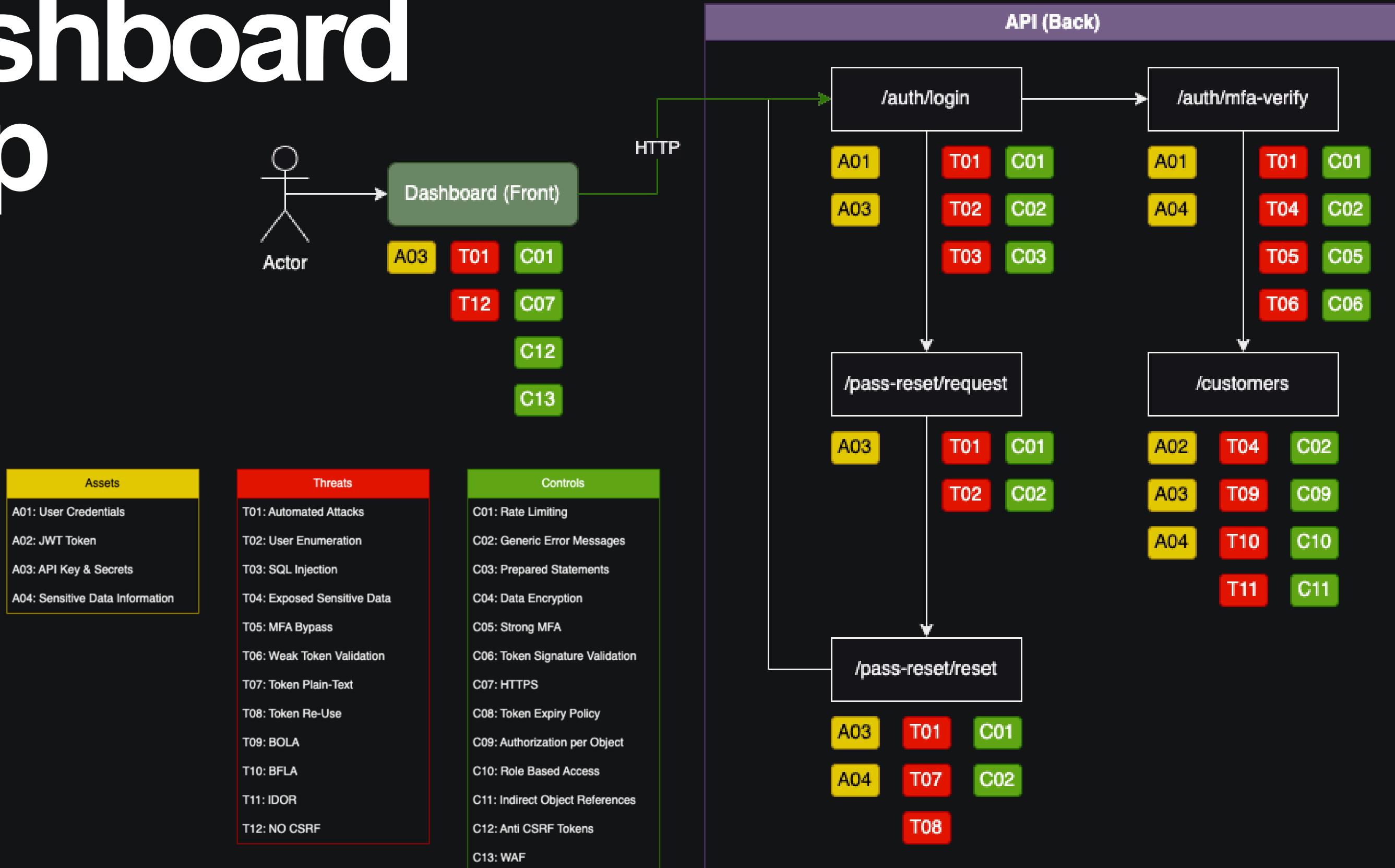
T01: Automated Attacks
T04: Exposed S. Data
T05: MFA Bypass
T06: Weak Token Validat.

Controls

C01: Rate Limit
C02: Generic Messages
C05: Strong MFA
C06: Token S. Validation



Dashboard App



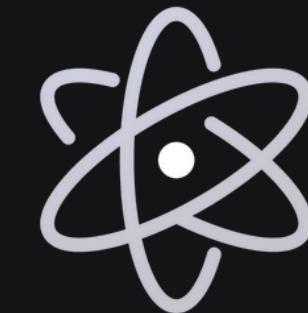
Arsenal

Ferramentas que permitem a criação de templates customizados que possibilitam a automação de testes de requisitos.



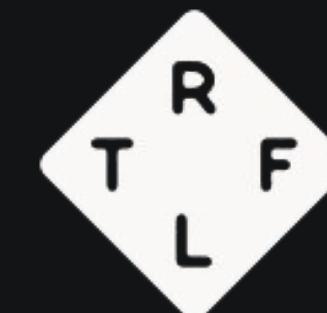
SEMGREP

Semgrep é uma ferramenta de análise estática, rápida e de código aberto, que pesquisa código, encontra bugs e aplica padrões de codificação e proteções seguras.



NUCLEI

Nuclei é um scanner de vulnerabilidades moderno e de alto desempenho que utiliza modelos simples baseados em YAML.



TRUFFLEHOG

TruffleHog é a ferramenta mais poderosa de Descoberta, Classificação, Validação e Análise de segredos.

SEMGREP

- Escaneia o código em busca de vulnerabilidades, padrões de mau uso e problemas de conformidade.
- Suporte a várias linguagens, incluindo JavaScript, Python, Go, Java e mais.
- Permite criar regras personalizadas alinhadas aos requisitos específicos do projeto.
- As regras são escritas de forma simples utilizando YAML, baseadas na estrutura do código.



NUCLEI

- Framework para testes dinâmicos de segurança.
- Usa templates para identificar falhas como IDOR, BOLA, e exposição de informações sensíveis em APIs e endpoints web.
- Assim como o Semgrep, ela tem toda sua engine focada para testes dinâmicos, dando a vantagem para os analistas de segurança se preocuparem apenas com as regras e payloads.



TRUFFLEHOG

- Detecta secrets e credenciais expostas em código, como chaves de API, tokens e senhas.
- Verifica o histórico de commits em repositórios Git para evitar vazamentos.
- Possibilita regras customizadas, como por exemplo, token desenvolvido internamente que foge os padrões de mercado:

company-2025-eyas34das3f3fsdf



Porquê essas Tools?

1

Amplamente utilizadas

- Ferramentas consolidadas no mercado, reconhecidas pela comunidade e com boas práticas já integradas.
- Facilmente integradas a pipelines e fluxos de CI/CD.

2

Personalização

Permitem adição de regras customizadas, adaptando-se a nossos requisitos específicos.

3

Open Source

- Regras podem ser compartilhadas com a comunidade open source:
- Benefício: Crescimento da maturidade coletiva.

4

Privacidade

- Criamos uma comunidade interna na empresa para fomentar colaboração.
- As regras privadas são revisadas e alinhadas aos objetivos de negócio.

5

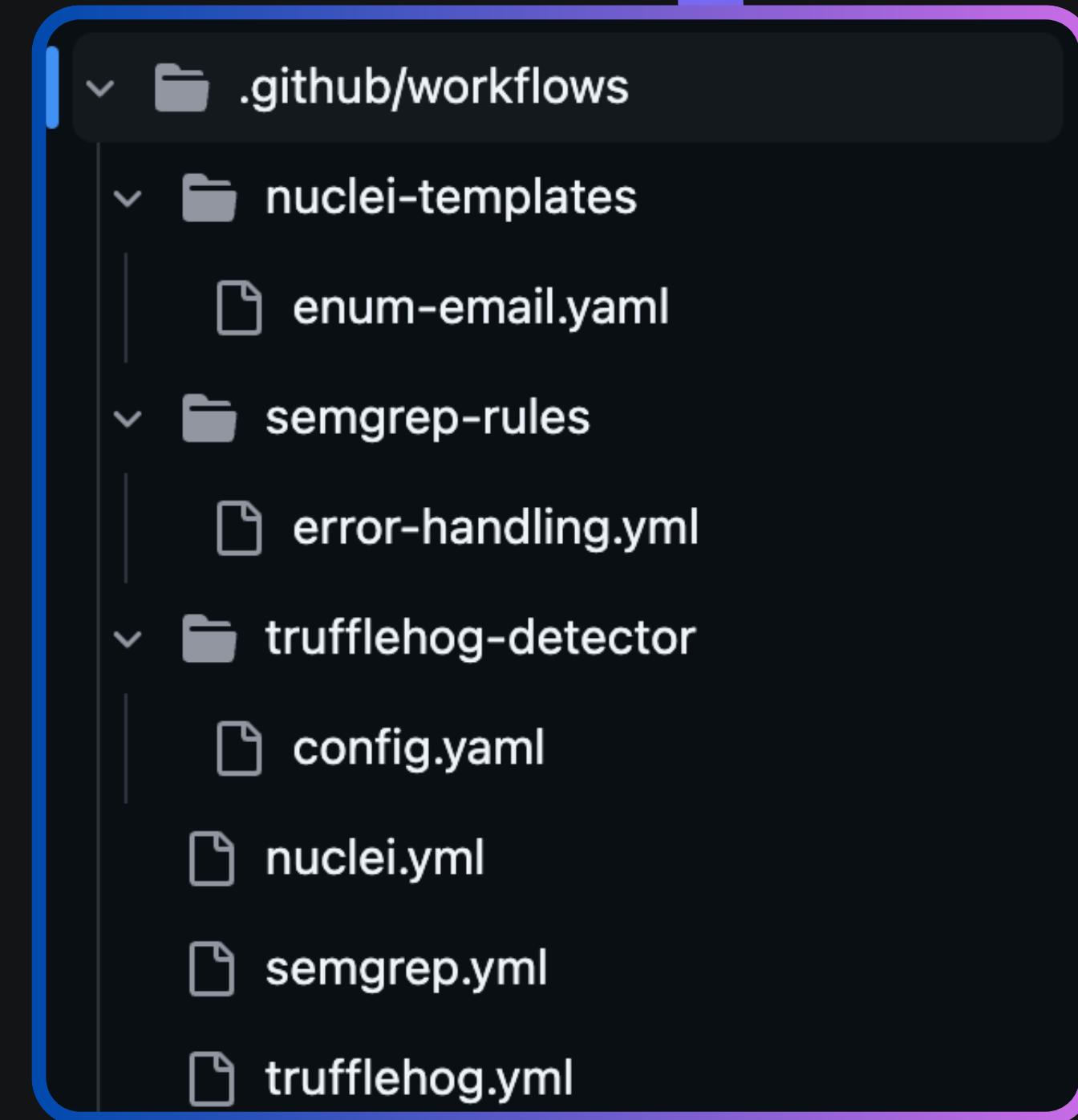
Escalabilidade

Escalamos o uso das ferramentas para várias equipes e projetos, mantendo a consistência nos padrões de segurança.

OPlano

Iremos utilizar o Github Actions como exemplo de CI/CD, porém todas estas ferramentas tem suas possibilidades de implementação em qualquer outra ferramenta moderna de CI/CD como Azure Devops, Gitlab CI, Circle CI e Jenkins.

Vamos utilizar alguns templates chamados de actions que já encapsulam as ferramentas escolhidas, mas a depender do caso, pode ser carregado diretamente os binarios e usados da mesma forma.



Trufflehog



.github/workflows/trufflehog.yml

```
# trufflehog setup
# ...
# Run Trufflehog with custom regex detector
- name: Run Trufflehog
  run: |
    trufflehog filesystem \
    --config config.yaml .github/workflows/trufflehog-detector/config.yaml
```



trufflehog-detector/config.yaml

```
detectors:
- name: DevSecOpsTokenDetector
  keywords:
    - company
  regex:
    devsecopsToken: |
      '\b(company-[0-9]{4}-[a-zA-Z0-9]{15})\b'
  verify:
    - endpoint: http://devsecops.com/token-verify
      unsafe: true
  headers:
    - "Authorization: super secret authorization header"
```

Semgrep



.github/workflows/semgrep.yml

```
name: Semgrep OSS scan

on:
  # Scan changed files in PRs (diff-aware scanning):
  pull_request: {}
  # Scan on-demand through GitHub Actions interface:
  workflow_dispatch: {}
  # Scan mainline branches and report all findings:
  push:
    branches: ["master", "main"]

jobs:
  semgrep:
    name: semgrep-oss/scan
    runs-on: ubuntu-latest
    container:
      image: semgrep/semgrep

    steps:
      - uses: actions/checkout@v4
      - run: semgrep scan --metrics=off --error --config .github/workflows/semgrep-rules
```

Semgrep

```
semgrep-rules/error-handling.yml

rules:
  - id: exposed-secrets-in-error-response
    patterns:
      - pattern: |
          res.status($STATUS).json({
            $KEY: $VALUE,
            ...
            $DETAILS: {
              ...
              $SENSITIVE: {
                ...
                $ENV_VAR: process.env.$VAR_NAME,
                ...
              },
              ...
            }
          })
      - pattern-either:
          - pattern: |
              $ITEM_VAR: process.env.AWS_SECRET_KEY,
          - pattern: |
              $ITEM_VAR: process.env.DB_PASS,

      message: "Sensitive environment variables (such as SECRET or PASS) should not be included
      in error responses."
      languages:
        - javascript
        - typescript
      severity: ERROR
      metadata:
        cwe: "CWE-209"
        owasp: "A5:2021-Security Misconfiguration"
      fix: |
        Ensure sensitive information, such as access keys and secrets, are not exposed in error
        responses. Consider logging these details internally and returning a generic error message to
        the client.
```

Semgrep



semgrep-rules/error-handling.yml

```
- pattern: |
    res.status($STATUS).json({
        $KEY: $VALUE,
        ...,
        $DETAILS: {
            ...,
            $SENSITIVE: {
                ...,
                $ENV_VAR: process.env.$VAR_NAME,
                ...
            },
            ...
        }
    })
- pattern-either:
    - pattern: |
        $ITEM_VAR: process.env.AWS_SECRET_KEY,
    - pattern: |
        $ITEM_VAR: process.env.DB_PASS,
```



middlewares/error-handling.js

```
function errorHandler(err, req, res, next) {
    console.error('Error:', err);
    res.status(500).json({
        error: err.message,
        stack: err.stack,
        details: {
            timestamp: new Date(),
            path: req.path,
            method: req.method,
            environment: process.env.NODE_ENV,
            database: {
                user: process.env.DB_USER,
                pass: process.env.DB_PASS,
                name: process.env.DB_NAME,
                aws_secret_key: process.env.AWS_SECRET_KEY
            }
        }
    });
}

module.exports = errorHandler;
```

Nuclei



.github/workflows/nuclei.yml

```
jobs:
  security-scan:
    runs-on: ubuntu-latest

    services:
      mysql:
        image: mysql:8.0
        volumes:
          - ./init.sql:/docker-entrypoint-initdb.d/init.sql
        env:
          MYSQL_ROOT_PASSWORD: rootpassword
          MYSQL_DATABASE: vulnerable_app
          MYSQL_USER: app_user
          MYSQL_PASSWORD: app_password
        ports:
          - 3306:3306

    steps:
      # Checkout the code
      - name: Checkout Code
        uses: actions/checkout@v2

      - name: Start the API
        run: |
          cd backend
          DB_HOST=mysql DB_USER=app_user DB_PORT=3306 DB_PASS=app_password DB_NAME=vulnerable_app npm
install && npm start &

      - name: Nuclei - Vulnerability Scan
        id: nuclei_scan
        uses: projectdiscovery/nuclei-action@main
        with:
          target: http://127.0.0.1:3000/api/password-reset/request
          templates: .github/workflows/nuclei-templates
```

Nuclei



.github/workflows/nuclei.yml

```
jobs:
  security-scan:
    runs-on: ubuntu-latest

  services:
    mysql:
      image: mysql:8.0
      volumes:
        - ./init.sql:/docker-entrypoint-initdb.d/init.sql
    env:
      MYSQL_ROOT_PASSWORD: rootpassword
      MYSQL_DATABASE: vulnerable_app
      MYSQL_USER: app_user
      MYSQL_PASSWORD: app_password
    ports:
      - 3306:3306
```

Nuclei



.github/workflows/nuclei.yml

```
steps:  
  # Checkout the code  
  - name: Checkout Code  
    uses: actions/checkout@v2  
  
  - name: Start the API  
    run: |  
      cd backend  
      DB_HOST=mysql DB_USER=app_user DB_PORT=3306 DB_PASS=app_password DB_NAME=vulnerable_app npm  
install && npm start &  
  
  - name: Nuclei - Vulnerability Scan  
    id: nuclei_scan  
    uses: projectdiscovery/nuclei-action@main  
    with:  
      target: http://127.0.0.1:3000/api/password-reset/request  
      templates: .github/workflows/nuclei-templates
```

Impactos



- **SEMGREP**

Detectamos uma chave com um padrão de hash único da empresa no histórico do repositório.

- **NUCLEI**

Localizamos um possível vazamento de informações sensíveis em um middleware de erro.

A mensagem do erro poderia ser enviada para o usuário final com dados como tokens e nomes de variáveis internas.

- **TRUFFLEHOG**

Identificamos vulnerabilidade de enumeração de usuários via a rota `/password-reset/request`.

O sistema retornava respostas diferentes dependendo da existência do email no banco, expondo a possibilidade de ataques de força bruta ou automação.

Resultados

Vulnerabilidade	Controle	Ferramenta	Descrição
Vazamento de secrets	ASVS 5.1.4 PCI DSS 3.6	Trufflehog	Detecta chaves e tokens expostos no código ou histórico de commits.
Dados sensíveis em middleware	ASVS 4.10.4 PCI DSS 6.5.7	Semgrep	Identifica mensagens de erro expondo informações sensíveis.
Enumeração de usuários	ASVS 2.3.1 PCI DSS 6.5.10	Nuclei	Detecta mensagens de resposta diferenciadas indicando existência de usuários.

Resultados

Vulnerabilidade	Controle	Ferramenta	Descrição
IDOR	ASVS 5.3.3 PCI DSS 7.1.2	Nuclei	Testa se endpoints permitem acesso a recursos de outros usuários.
BOLA	ASVS 5.4.1 PCI DSS 7.2	Nuclei	Simula modificações em payloads para testar permissões inadequadas.
SQL Injection	ASVS 5.3.2 PCI DSS 6.5.1	Semgrep	Verifica uso de queries concatenadas diretamente com input do usuário.

Resultados

Vulnerabilidade	Controle	Ferramenta	Descrição
No Rate Limit	ASVS 2.5.5 PCI DSS 8.3.1	Nuclei	Detecta tokens em texto plano armazenados ou retornados ao cliente.
Sensitive Info in Logs	ASVS 4.10.5 PCI DSS 10.2	Semgrep	Analisa trechos de código que registram dados confidenciais nos logs.

Obrigado!

Dúvidas ou dívidas?

DEVSECOPS
VILLAGE



H2
HC
CONFERENCE