UNIVERSIDADE FEDERAL DE CAMPINA GRANDE
CENTRO DE ENGENHARIA ELÉTRICA E INFORMÁTICA
CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

BRENDA LOUISY MORAIS ALVES

CONTROLE DE ACESSO EM API DO PROJETO *SMARTCAMPUS:* UMA ABORDAGEM USANDO SPIRE EM CONJUNTO COM SERVIÇOS DE PROXY

CAMPINA GRANDE - PB

2024

BRENDA LOUISY MORAIS ALVES

CONTROLE DE ACESSO EM API DO PROJETO *SMARTCAMPUS:* UMA ABORDAGEM USANDO SPIRE EM CONJUNTO COM SERVIÇOS DE PROXY

Trabalho de Conclusão Curso apresentado ao Curso Bacharelado em Ciência da Computação do Centro de Engenharia Elétrica e Informática da Universidade Federal de Campina Grande, como requisito parcial para obtenção do título de Bacharel em Ciência da Computação.

Orientador : Andrey Elísio Monteiro Brito

CAMPINA GRANDE - PB

2024

BRENDA LOUISY MORAIS ALVES

CONTROLE DE ACESSO EM API DO PROJETO *SMARTCAMPUS:* UMA ABORDAGEM USANDO SPIRE EM CONJUNTO COM SERVIÇOS DE PROXY

Trabalho de Conclusão Curso apresentado ao Curso Bacharelado em Ciência da Computação do Centro de Engenharia Elétrica e Informática da Universidade Federal de Campina Grande, como requisito parcial para obtenção do título de Bacharel em Ciência da Computação.

BANCA EXAMINADORA:

Andrey Elísio Monteiro Brito
Orientador – UASC/CEEI/UFCG

Reinaldo Cézar de Morais Gomes
Examinador – UASC/CEEI/UFCG

Francisco Vilar Brasileiro
Professor da Disciplina TCC – UASC/CEEI/UFCG

Trabalho aprovado em: 17 de Maio de 2024.

CAMPINA GRANDE - PB

RESUMO

No contexto do projeto do *SmartCampus*, da Universidade Federal de Campina Grande (UFCG), a implementação de APIs que disponibilizam acesso a informações de eficiência energética apresenta desafios relacionados à segurança e ao controle de acesso. As partes interessadas, incluindo desenvolvedores, usuários finais e operadores de produção, necessitam interagir com essas APIs em ambientes variados, criando a necessidade de diferenciar e controlar o acesso de maneira eficaz. Este trabalho explora estratégias fundamentadas no modelo *Zero Trust*, que preconiza a autenticação contínua e autorização granular, garantindo que cada acesso seja verificado e autenticado. Adotando o SPIRE (SPIFFE Runtime Environment) em conjunto com serviços de *proxy*, busca-se assegurar a autenticação segura por meio de identidades criptográficas fornecidas pelo SPIRE, bem como implementar um serviço de autorização personalizado conforme o perfil do usuário. O objetivo é prevenir acesso inadequado, vazamento de dados e manipulações indevidas, garantindo um ambiente seguro e confiável para a implantação do projeto.

ACCESS CONTROL IN THE SMARTCAMPUS PROJECT API: AN APPROACH USING SPIRE IN CONJUNCTION WITH PROXY SERVICES

ABSTRACT

In the context of the SmartCampus project at the Federal University of Campina Grande (UFCG), the implementation of APIs providing access to energy efficiency information presents challenges related to security and access control. Stakeholders, including developers, end users, and production operators, need to interact with these APIs in various environments, creating the need to effectively differentiate and control access. This work explores strategies based on the Zero Trust model, which advocates continuous authentication and granular authorization, ensuring that each access is verified and authenticated. By adopting SPIRE (SPIFFE Runtime Environment) in conjunction with proxy services, the aim is to ensure secure authentication through cryptographic identities provided by SPIRE, as well as to implement a customized authorization service according to user profiles. The goal is to prevent unauthorized access, data leakage, and improper manipulations, ensuring a safe and reliable environment for the project deployment.

Controle de acesso em API do projeto *SmartCampus*: uma abordagem usando SPIRE em conjunto com serviços de proxy

Brenda Louisy Morais Alves
Universidade Federal de Campina Grande - UFCG
Campina Grande, Brasil
brenda.alves@ccc.ufcg.edu.br

Andrey Brito
Universidade Federal de Campina Grande - UFCG
Campina Grande, Brasil
andrey@computacao.ufcg.edu.br

RESUMO

No contexto do projeto do *SmartCampus*, da Universidade Federal de Campina Grande (UFCG), a implementação de APIs que disponibilizam acesso a informações de eficiência energética apresenta desafios relacionados à segurança e ao controle de acesso. As partes interessadas, incluindo desenvolvedores, usuários finais e operadores de produção, necessitam interagir com essas APIs em ambientes variados, criando a necessidade de diferenciar e controlar o acesso de maneira eficaz. Este trabalho explora estratégias fundamentadas no modelo *Zero Trust*, que preconiza a autenticação contínua e autorização granular, garantindo que cada acesso seja verificado e autenticado. Adotando o SPIRE (SPIFFE Runtime Environment) em conjunto com serviços de proxy, busca-se assegurar a autenticação segura por meio de identidades criptográficas fornecidas pelo SPIRE, bem como implementar um serviço de autorização personalizado conforme o perfil do usuário. O objetivo é prevenir acesso inadequado, vazamento de dados e manipulações indevidas, garantindo um ambiente seguro e confiável para a implantação do projeto.

KEYWORDS

zero trust, proxy, controle de acesso, SPIRE

1 INTRODUÇÃO

A Universidade Federal de Campina Grande (UFCG)[1] lançou uma chamada de Pesquisa, Desenvolvimento e Inovação em Inteligência de Negócios, cujo objetivo é promover estratégias aplicadas à gestão universitária. Inicialmente serão priorizadas duas áreas: gestão acadêmica e gestão do consumo de energia do campus UFCG. Para isso, em sua primeira fase, o projeto prevê a implantação de serviços básicos para disponibilizar acessos aos dados do sistema de controle acadêmico e também dados do projeto *SmartCampus*[2], que abrange o consumo de energia de algumas edificações representativas dos campi da UFCG, a exemplo de salas de aula e laboratórios, e também o serviço de atuação em sistemas de IoT (Internet of Things) presentes no campus.

Esses dados serão disponibilizados em APIs para uma variedade de usuários, incluindo desenvolvedores e operadores de sistemas em produção, em diversos cenários. Com isso, surgem diversos desafios relacionados à segurança dos dados e dos sistemas envolvidos. Esses desafios incluem a importância de diferentes níveis de controle de acesso para diferentes usuários, a prevenção de vazamento de dados, proteção contra manipulações indevidas em serviços de IoT, limitações de taxa de acesso para mitigar ataques de sobrecarga, além da garantia de segurança em ambientes diversos.

As soluções convencionais de controle de acesso enfrentam desafios significativos quando se trata de manter as políticas de segurança. Práticas como o compartilhamento de credenciais e a criação manual de identidades para cada usuário são práticas desatualizadas e inadequadas. Essas abordagens enfrentam dificuldades de escalabilidade, tornando-se cada vez mais ineficazes à medida que o número de usuários cresce, resultando em um aumento exponencial das demandas de gerenciamento. A operação dessas soluções também se revela complexa, requerendo uma sobrecarga significativa de recursos e esforços administrativos.

Além disso, a granularidade do controle de acesso oferecido por tais soluções convencionais coloca em risco a segurança, a integridade dos dados e a conformidade com regulamentações de segurança de dados, visto que a falta de acurácia na definição de permissões e privilégios pode resultar em brechas significativas, permitindo acesso não autorizado a informações sensíveis. Portanto, é evidente que as abordagens tradicionais não são mais suficientes para atender às demandas de segurança, eficiência e conformidade em ambientes tecnológicos contemporâneos.

O modelo *Zero Trust* [9], por sua vez, oferece uma abordagem robusta ao controle de acesso, superando as limitações das soluções convencionais citadas anteriormente. Baseado na premissa de "nunca confiar, sempre verificar", exige múltiplas autenticações para estabelecer comunicações confiáveis entre sistemas dentro ou fora da rede da organização.

Neste trabalho, nós iremos focar no controle de acesso aos dados de consumo de energia disponibilizados pelo projeto *SmartCampus*, através da API do LiteMe[3], utilizando um *proxy* que atua como intermediário entre esse serviço e o acesso por parte dos projetos selecionados. Para estabelecer uma comunicação segura entre *proxy* e usuário, será utilizado o SPIRE [1] como provedor de certificados X.509 que serão utilizados tanto na comunicação, quanto para a configuração do controle de acesso de acordo com as identidades apresentadas de cada usuário. Além disso, será implementada uma limitação na taxa de acesso, do inglês (*rate limiting*), para o acesso não autenticado à API, mitigando os riscos de ataque por parte do acesso anônimo. Adicionalmente, ferramentas de monitoramento que permitirão registrar dados de acesso a fim de obter informações importantes para análises de segurança.

[1] Site da UFCG: https://www.ceei.ufcg.edu.br/
[2] SmartCampus UFCG: https://ufcg.liteme.com.br/
[3] Site do LiteMe: https://www.liteme.com.br/

O restante deste trabalho está organizado da seguinte forma. Na Seção 2 apresentamos conceitos e tecnologias utilizadas. O modelo de ameaças assumido é detalhado na Seção 3. Na Seção 4 apresentamos a solução desenvolvida. Na Seção 5 é realizada uma avaliação da solução. Por fim, na Seção 6, sumarizamos as limitações do trabalho e conclusões atingidas.

2 FUNDAMENTAÇÃO TEÓRICA

Nesta seção são introduzidos alguns conceitos e tecnologias importantes para entender o contexto do trabalho apresentado.

2.1 Zero Trust

O modelo *Zero Trust* [9] emerge como uma estratégia essencial na proteção da comunicação entre aplicações nativas em nuvem, enfatizando que a confiança deve ser continuamente verificada, nunca sendo assumida por padrão. Essa arquitetura estende-se desde a identificação e autenticação de usuários e dispositivos, até a gestão de acessos e a segurança da infraestrutura interconectada, promovendo uma abordagem de segurança de dados de forma integrada. Contrapondo-se às práticas tradicionais de defesa de perímetro, nas quais a autenticação interna geralmente resulta em acesso amplo, o *Zero Trust* visa implantar políticas de acesso detalhadas, assegurando controle refinado sobre as permissões de usuários e dispositivos, além de exigir a verificação contínua de identidades em todas as interações, assegurando um ambiente de rede mais seguro.

2.2 Proxies

Nesta subseção serão apresentadas as duas opções de *proxy* consideradas para a solução proposta.

A primeira delas é o Envoy [5] *proxy*, uma ferramenta de código aberto concebida para arquiteturas de microsserviços, oferecendo uma abordagem abrangente à segurança de dados e recursos em ambientes distribuídos. É fundamentado na ideia de que a rede deve ser transparente para as aplicações, para que, caso ocorram erros, seja fácil identificar a origem do problema. Uma característica importante do Envoy é sua capacidade em atuar como *sidecar*, o que permite sua operação ao lado das aplicações e dos clientes, integrando funcionalidades adicionais sem interromper o fluxo existente dos sistemas.

Além de sua flexibilidade e capacidade de interconexão, o Envoy destaca-se por oferecer funcionalidades como observabilidade, monitoramento, balanceamento de carga, limitação de taxa de acesso, além de técnicas de controle de acesso e segurança. Através de sua arquitetura de filtros, permite a inserção de configurações personalizadas para realizar diferentes tarefas de *proxy*, como, por exemplo, o filtro do RBAC [2] (Role-Based Access Control), que tem como objetivo principal a implementação de políticas detalhadas de controle de acesso.

No contexto do Envoy, um *listener* é um componente configurável que define como o *proxy* aceita conexões de entrada. Ele pode ser configurado para ouvir em uma ou mais portas de rede, processando todos os dados que chegam nessas portas conforme as regras definidas nos filtros. Além disso, temos o conceito de *cluster*, que define a configuração dos serviços que o Envoy irá estabelecer conexões.

O Envoy também oferece suporte à segurança do tráfego entre serviços, incorporando o SDS [3] (Secret Discovery Service) à sua configuração, uma funcionalidade projetada para a gestão eficiente de segredos e certificados TLS. O SDS possibilita a recuperação dinâmica desses segredos de um servidor SDS remoto, distribuindo-os automaticamente por todas as instâncias do Envoy envolvidas. Além disso, está encarregado de atualizar esses segredos de maneira dinâmica, mitigando os desafios associados à gestão de certificados expirados e eliminando a necessidade de reiniciar ou reimplantar os serviços para a atualização de segredos, garantindo uma operação mais suave e segura.

Combinando uma arquitetura flexível e interoperável com recursos de controle de acesso e segurança, o Envoy é uma ferramenta importante na construção de infraestruturas de rede seguras e eficientes para aplicações em ambientes de microsserviços.

O Ghostunnel [11] entra como mais uma opção de *proxy* a ser utilizado na solução para o controle de acesso. Este, por sua vez, atua como um *proxy* TLS, no modo cliente e no modo servidor, com suporte para autenticação mútua para garantir a segurança de aplicativos de *backend* que não utilizam TLS. Uma de suas principais funcionalidades é o controle de acesso, que permite a aplicação de políticas de autorização com base em certificados do cliente, garantindo que apenas usuários autenticados e autorizados possam acessar os serviços.

Além disso, assim como o Envoy, o Ghostunnel oferece suporte para a troca dinâmica de certificados em tempo de execução, permitindo que novos certificados sejam carregados sem interromper as conexões existentes. Funcionalidades de monitoramento e métricas também estão integradas ao Ghostunnel, permitindo a coleta de dados sobre o desempenho e a utilização do serviço.

2.3 SPIFFE

O SPIFFE [10] (Secure Production Identity Framework for Everyone) define um conjunto de padrões voltados para a identificação e autenticação de aplicações nativas em nuvem, especialmente em ambientes dinâmicos e heterogêneos. Este *framework* facilita a gestão segura de identidades de *software*, permitindo que sistemas e serviços verifiquem automaticamente as identidades de outros serviços que desejam estabelecer uma comunicação segura. Atuando com base nos princípios do modelo *Zero Trust*, o SPIFFE provê uma camada fundamental para implementar controles rigorosos de segurança. Ao adotar o SPIFFE, as organizações podem automatizar a emissão e a validação de identidades criptografadas e dinâmicas, conhecidas como SVIDs (SPIFFE Verifiable Identity Documents). Esses documentos facilitam a autenticação e a autorização baseada em identidades verificáveis em tempo real, o que é crucial para manter a segurança em sistemas distribuídos.

2.4 SPIRE

O SPIRE é um projeto de código aberto que opera sob os princípios do SPIFFE, fornecendo uma API com ferramentas de autenticação para estabelecer confiança na comunicação entre aplicações em sistemas distribuídos. Juntos, SPIFFE e SPIRE automatizam o processo de identificação de serviços, permitindo autenticação segura e comunicação entre aplicativos de maneira escalável e sem a necessidade de intervenção manual.

No contexto do modelo *Zero Trust*, o documento utilizado pelo SPIRE para verificação de autenticidade é o SVID, que pode ser um certificado X.509 ou um JWT, e possui em sua estrutura uma identidade SPIFFE contendo o domínio de confiança e o nome da carga de trabalho, `spiffeID` (no formato *spiffe://example.org/myagent*), sendo assinadas por uma autoridade certificadora que permite a verificação da autenticidade do documento.

A arquitetura do SPIRE é dividida em dois principais componentes, Servidor e Agente, que atuam em conjunto no processo de autenticação das identidades para ser estabelecida uma comunicação segura, conforme ilustra a figura 1. O Servidor é responsável por gerenciar e emitir todas as identidades em um domínio de confiança, enquanto os Agentes servem às cargas de trabalho. Um Agente recebe um SVID após ser submetido a um processo de atestação de nó, e assim, tem permissão para iniciar o fluxo de atestação das cargas de trabalho às quais ele serve. É através do registro de entrada que o Agente solicita ao Servidor que gere os SVIDs de cada carga de trabalho. Um registro de entrada é um objeto criado e gerenciado pela API do SPIRE, que indica o `parentID` (referência para o `spiffeID` do agente que está solicitando a atestação) e o `spiffeID` da carga de trabalho, além de seletores específicos usados para identificar o ambiente de execução do serviço [6], conforme exemplo abaixo:

```
spire-server entry create \
    -parentID spiffe://example.com/myagent \
    -spiffeID spiffe://example.com/webapp  \
    -selector unix:gid:1000
```

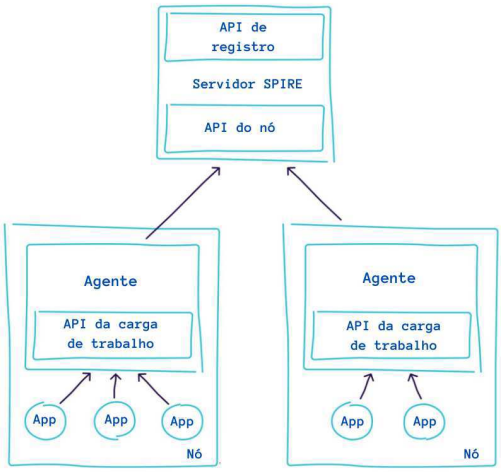

Figura 1: Arquitetura do SPIRE, baseada no livro Solving the Bottom Turtle [6].

Uma vez que a carga de trabalho esteja em posse de sua identidade, é possível utilizar o SPIRE em conjunto com outras ferramentas para implantar políticas de controle de acesso a recursos, por meio de autenticação e autorização. Através do `spiffeID`, o processo de autenticação pode ser validado para que o acesso a um determinado serviço seja permitido. Uma vez concedido o acesso, para implantar as políticas de autorização, é preciso definir quais ações são permitidas a esse `spiffeID`. Através do uso do SPIRE, existem alguns tipos de autorização, como uma simples *allowlist*, que referencia uma lista de identidades que podem ter determinados acessos, assim como formas mais sofisticadas de gerenciar as políticas de acesso, como o RBAC e o *ABAC* (*Attribute-Based Access Control*). Usados em conjunto com *proxies*, esse modelo se torna uma ferramenta bastante segura para implementação do controle de acesso granular [6].

2.5 Graphite

Graphite [8] é uma ferramenta de código aberto que oferece monitoramento sobre o comportamento e desempenho de sistemas. A partir de agentes coletores de informações, as aplicações podem enviar métricas para o serviço do Graphite, a fim de auxiliar na detecção de erros, resolução e melhoria contínua. Para o nosso caso de uso, o coletor usado foi o StatsD, que atua como *proxy* enviando estatísticas de seus servidores para sistemas de monitoramento, como o Graphite.

2.6 Grafana

Grafana [7] é uma ferramenta de código aberto que fornece visualização e análise de métricas por meio de visualizações gráficas. O Graphite, citado anteriormente, atua como fonte de dados de *backend* para o Grafana, que por sua vez oferece um plugin que pode ser usado diretamente para conectar e pegar os dados da instância do Graphite. A partir da coleta de informações, é possível observar dados como métricas, logs, alertas, entre outros dados, agrupando em visualizações capazes de oferecer análises importantes para o processo de monitoramento e observabilidade.

2.7 Caso de uso SmartCampus e LiteMe

Nesta subseção, vamos apresentar o caso de uso em que a solução será aplicada, com foco na gestão dos dados de consumo de energia do campus da UFCG, oferecidos pelo projeto *SmartCampus* através da API do LiteMe.

O LiteMe é uma plataforma que fornece monitoramento através da coleta de informações sobre o consumo elétrico no campus da UFCG. Por meio de medidores que estão conectados a coletores instalados em blocos do campus, os dados são enviados para uma plataforma na nuvem, que os recebe, processa e os apresenta ao consumidor, mostrando onde e quando ele está consumindo mais, e como e quanto pode ser economizado. O sistema envolve medidores que interagem na arquitetura para receber e processar dados elétricos em tempo real.

O sistema possui uma API para ser usada por terceiros. No entanto, apenas algumas informações são públicas e podem ser acessadas por anônimos, sem a exigência de autenticação, por meio de *endpoints* que fornecem dados sobre demanda atual, consumo do dia, consumo mensal e estatísticas de consumo por tempo. A outra parcela de dados só pode ser acessada mediante autenticação, a partir da solicitação de um *token*, sendo necessário ter uma conta cadastrada com e-mail e senha no sistema do LiteMe. Na figura 2 está representado o fluxo de acesso à API.

(1) Sobre o fluxo descrito na cor vermelha:
   - Um usuário anônimo realiza uma requisição para a API do LiteMe, para quaisquer *endpoints* que não exijam autenticação e os dados públicos são retornados na resposta.

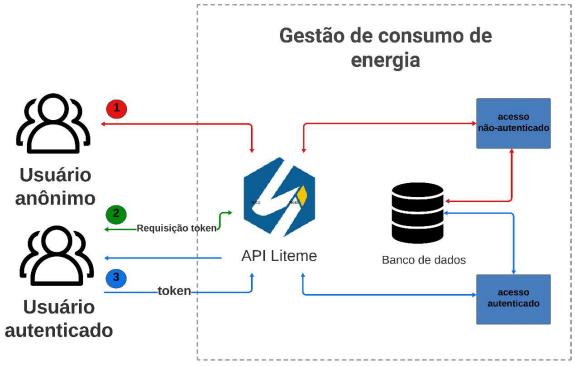

Figura 2: Caso de uso de acesso à API do LiteMe.

(2) Sobre o fluxo descrito na cor verde:
- Um usuário que possui cadastro no sistema do LiteMe, com e-mail e senha, solicita um *token* através de uma requisição.

(3) Sobre o fluxo descrito na cor azul:
- O usuário autenticado, com o *token* em mãos, realiza requisição para quaisquer *endpoints* que exijam autenticação, passando o *token* como parâmetro e os dados privados são retornados na resposta.

3 MODELO DE AMEAÇAS

O modelo de ameaça que será descrito nesta seção considera aspectos do STRIDE[4], *framework* utilizado para facilitar o processo de identificação de ameaças à segurança computacional, buscando compreender o que é esperado do atacante, facilitando, assim, a elaboração de um plano de mitigação de riscos.

Partindo do fluxo em que um atacante pode ser um usuário anônimo malicioso, é possível que ele tente acessar a API não autenticada do LiteMe realizando um ataque de DoS (Denial of Service), caracterizado por um alto volume de requisições, com o intuito de causar sobrecarga e indisponibilidade dos serviços para usuários legítimos.

Considerando o acesso autenticado, o atacante pode se passar por usuários legítimos para acessar a API autenticada, realizando um ataque de falsificação (Spoofing), ao ter acesso às credenciais estáticas de e-mail e senha de um usuário devidamente cadastrado no sistema do LiteMe. Para o nosso caso de uso, ataque de falsificação implica no comprometimento também da confidencialidade dos dados sensíveis (Information Disclosure), considerando que um atacante se passando por um usuário legítimo não possui autorização de acesso a esses dados.

Somado a isso, consideramos que o atacante pode interceptar as requisições realizadas para a API do LiteMe, a fim de comprometer a confidencialidade e a integridade dos dados. Para esse caso, assumimos que a configuração do LiteMe conta com suporte de criptografia TLS, através do uso do HTTPS na transferência de dados sensíveis na comunicação.

[4]https://learn.microsoft.com/pt-br/azure/security/develop/threat-modeling-tool-threats

4 SOLUÇÃO

Nesta seção, é apresentada a solução de controle de acesso proposta usando Envoy como *proxy* e SPIRE como provedor de certificados.

4.1 Descrição geral

A solução propõe a implantação de controle de acesso para a API do LiteMe, combinando o uso do Envoy como *proxy* e o SPIRE como provedor de certificados, que serão providos tanto para o Envoy, como também para os projetos selecionados, que serão usuários autorizados a acessar as APIs. A solução foi implementada sem realizar qualquer alteração na configuração atual do LiteMe.

A escolha do Envoy se apresenta como uma opção mais adequada para a solução no caso de uso do *SmartCampus*. Isso se deve à sua arquitetura altamente flexível, capaz de se adequar a ambientes de microsserviços que demandam escalabilidade. O Envoy se sobressai pela sua capacidade de roteamento e filtragem, suporte avançado a protocolos HTTP, e recursos de observabilidade. Além disso, diferentemente do Ghostunnel, o Envoy possibilita que as políticas de autenticação e autorização sejam feitas a partir de *endpoints* utilizando o filtro do RBAC, em conjunto com a etapa de verificação das identidades fornecidas, tornando as políticas de segurança de autenticação e autorização mais granular e com uma acurácia maior.

O Envoy atua como intermediário entre os usuários e os serviços do LiteMe, gerenciando todas as requisições à API. O tráfego é encaminhado através do Envoy e regras de controle de acesso são aplicadas, incluindo controle por *endpoint*, indicando quais usuários podem acessá-los e também limitações nas taxas de acesso para usuários anônimos.

O SPIRE, por sua vez, atua como provedor de certificados X.509 usados para autenticar a instância do Envoy via SDS, assim como autenticar os projetos selecionados, para que seja possível a validação das identidades no processo de verificação no controle de acesso implementado pelo Envoy.

Na figura 3 é ilustrada a arquitetura da solução, implementando o controle de acesso. Os fluxos serão descritos de forma detalhada nas subseções abaixo.

4.2 Autenticação e Controle de acesso

A implementação de autenticação e controle de acesso com Envoy e SPIRE, no contexto do projeto *SmartCampus*, tem como objetivo central mitigar ataques de falsificação e ataques que comprometam a confidencialidade dos dados sensíveis disponibilizados pela API do LiteMe. Para isso, foi implementada uma infraestrutura utilizando o SPIRE para prover os certificados que atuarão na validação das identidades no processo de autenticação de cada projeto, assim como no processo de enviar os certificados via SDS para a instância do Envoy.

Na figura 3, na VM-Server, temos um servidor SPIRE executando e servindo o domínio *example.org*. Ele é responsável por emitir e gerenciar os SVIDs, identidades no formato de certificado X.509, e entregá-los aos agentes que estão sendo executados nas máquinas dos projetos selecionados e na VM-Envoy. O fluxo 2, referenciado pela cor cinza, indica a entrega dos SVIDs para as instâncias dos agentes. O processo de atestação e configuração é descrito abaixo.

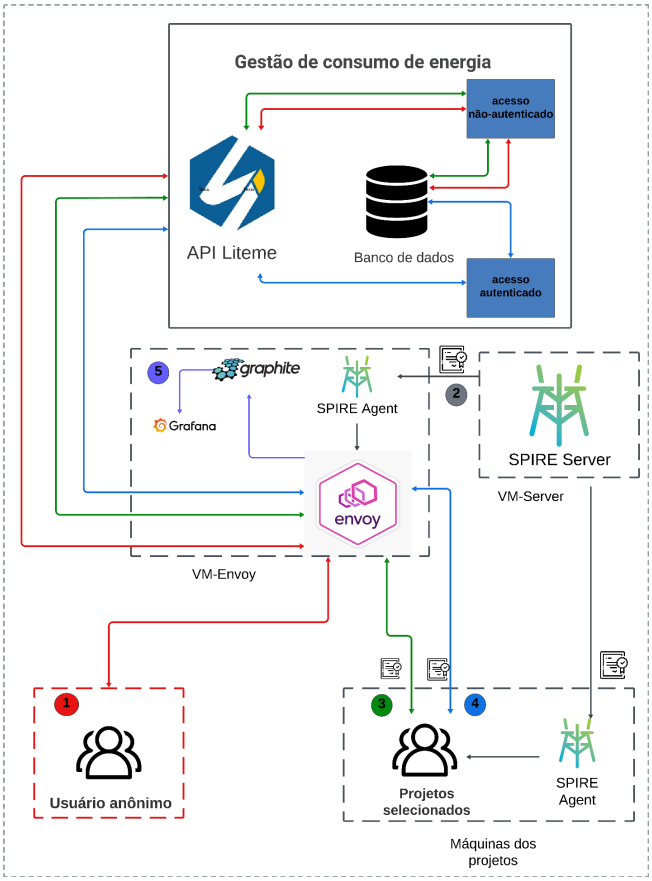

Figura 3: Arquitetura da solução de controle de acesso.

(1) **Autenticação do Envoy:** O processo é iniciado com a atestação do agente que irá servir o Envoy e que está sendo executado na mesma máquina. Isso é feito através do registro de entrada para autenticar o agente. Em seguida, precisamos realizar o registro de entrada para que o agente possa solicitar o SVID do Envoy. Na nossa solução, a instância do Envoy está sendo executada em um container Docker[5]. Então em seu registro de entrada, solicitamos ao servidor SPIRE um SVID para o processo do Envoy, que tem como seletor um identificador da imagem do container que executa o Envoy e `spiffeID`, *spiffe://example.org/liteme-envoy*, além da definição de um DNS (Domain Name System), como *liteme.example.org*, por exemplo. Para que a instância do Envoy possa solicitar ao agente sua identidade, em seu arquivo `YAML`, é configurado um *cluster* que se comunica com o agente, através do seu UDS (Unix Domain Socket), além da configuração do SDS que aponta para o *cluster* configurado. Essa configuração permite que o Envoy tenha acesso aos certificados gerados, de forma remota, garantindo um processo dinâmico na renovação dessas identidades. Abaixo estão descritas as propriedades do SDS configuradas no arquivo `YAML` do Envoy:

[5]Docker: https://www.docker.com/

```
transport_socket:
    name: envoy.transport_sockets.tls
    typed_config:
      "@type": type.googleapis.com/envoy.extensions
                .transport_sockets.tls.v3
                .DownstreamTlsContext
      common_tls_context:
        tls_certificate_sds_secret_configs:
        - name: spiffe://example.org/liteme-envoy
          sds_config:
            resource_api_version: V3
            api_config_source:
              api_type: GRPC
              transport_api_version: V3
              grpc_services:
                envoy_grpc:
                  cluster_name: spire_agent
```

Código 1: Configuração do SDS no Envoy

(2) **Autenticação das máquinas dos projetos:** Essa descrição se aplica aos usuários autenticados no sistema do LiteMe e selecionados. O processo é iniciado da mesma forma, com a atestação do agente rodando na máquina e com a realização do registro de entrada. No registro, é solicitado um `spiffeID` *spiffe://example.org/projeto1*. Para acessar a identidade gerada, o usuário deve usar a API do agente do SPIRE para baixar o SVID no formato X.509, chave privada e certificados da CA (Certificate Authority) entregues pelo servidor SPIRE. Os dois primeiros serão utilizados nas requisições à API, conforme ilustrado na figura 3 nos fluxos 3 e 4, nas cores verde e azul, respectivamente.

Para a configuração do controle de acesso, foi decidido configurar o Envoy para disponibilizar três *listeners*, com base nas APIs não autenticadas e autenticadas, além do perfil do usuário que está acessando:

(1) Listener 1 recebe requisições para a API não autenticada, de usuários anônimos, com restrição de número de acesso e não exige certificados (na figura 3, representado pelo fluxo 1 na cor vermelha)[6];
(2) Listener 2 recebe requisições para a API não autenticada, apenas dos projetos selecionados, exigindo certificados (na figura 3, representado pelo fluxo 3 na cor verde);
(3) Listener 3 recebe requisições para a API autenticada, apenas dos projetos selecionados, exigindo certificados (na figura 3, representado pelo fluxo 4 na cor azul).

No *listener* 3, foram especificadas as regras de controle de acesso usando o filtro RBAC disponibilizado pelo Envoy. Esse filtro permite a definição de uma lista de políticas de acesso, indicando a ação permitida a partir do método HTTP e *endpoints* que podem ser acessados, além da lista de usuários autorizados a terem acesso à política descrita. No Código 2, temos um exemplo de como é realizada a configuração para o *endpoint* `/service/rest/auth`. O filtro está permitindo que sejam realizadas requisições com o método POST pelos usuários autenticados que possuam `spiffeID` *spiffe://example.org/projeto1* e *spiffe://example.org/projeto2*.

```
http_filters:
```

[6]Considerando que está sendo usada uma CA desconhecida pela internet, para fins de testes, o certificado é passado na requisição para que o cliente confie na CA do servidor.

```
- name: envoy.filters.http.rbac
typed_config:
  "@type": type.googleapis.com/envoy.extensions
           .filters.http.rbac.v3.RBAC
  rules:
    action: ALLOW
    policies:
      "authenticate":
        permissions:
          - and_rules:
              rules:
                - header: {name: ":method",
                    string_match: {exact: "POST"}}
                - url_path:
                    path: {prefix: "/service/rest/auth"}
        principals:
          - authenticated:
              principal_name:
                exact: "spiffe://example.org/projeto1"
          - authenticated:
              principal_name:
                exact: "spiffe://example.org/projeto2"
```

Código 2: Configuração do RBAC no Envoy

4.3 Rate Limit

Para o *listener* 1, em que usuários anônimos terão acesso aos dados da API não autenticada, foi aplicado o filtro *Rate Limit* com o intuito de limitar as taxas de acesso, minimizando os riscos de ataques de DoS. Basicamente, a configuração consiste em estabelecer uma quantidade de *tokens* disponíveis para requisições em um determinado tempo definido, válido para o tráfego total recebido pela rede. No exemplo do Código 3, é apresentada parte da configuração da limitação de taxa de acesso, em que para qualquer rota pertencente ao prefixo `/service/rest/withoutauth` é liberada uma quantidade de 10 *tokens*, que permite o mesmo número de requisições em um período de 60 segundos:

```
route_config:
    name: local_route
    virtual_hosts:
    - name: local_service
      domains: ["*"]
      routes:
      - match:
          prefix: "/service/rest/withoutauth"
        route:
          cluster: api_liteme
          host_rewrite_literal: set.lsd.ufcg.edu.br
  http_filters:
  - name: envoy.filters.http.local_ratelimit
    typed_config:
      "@type": type.googleapis.com/envoy.extensions
      .filters.http.local_ratelimit.v3.LocalRateLimit
      stat_prefix: http_local_rate_limiter
      token_bucket:
        max_tokens: 10
        tokens_per_fill: 10
        fill_interval: 60s
```

Código 3: Configuração do Rate Limit no Envoy

Extrapolado o limite de *tokens* para o intervalo de tempo definido, o Envoy bloqueia as requisições recebidas, retornando ao usuário uma mensagem indicando que a quantidade de requisições foi excedida.

4.4 Monitoramento

A configuração do monitoramento é realizada usando a integração entre Graphite e Grafana. No Envoy, é realizada a configuração solicitando ao mesmo que envie uma variedade de métricas ao Graphite, a fim de obter informações em tempo real sobre requisições realizadas, desempenho do serviço e estatísticas de erro. Com as métricas armazenadas no Graphite, o Grafana é utilizado para criar dashboards de visualização que captam os dados em tempo real para fornecer uma análise detalhada das informações colhidas.

A figura 4 ilustra um exemplo de um gráfico de linhas no Grafana que foi configurado para monitorar o Envoy, registrando as requisições realizadas com sucesso em um determinado período. Outras estatísticas podem ser colhidas também, como monitoramento de requisições falhas ao atingir a taxa de limitação de requisições, além da configuração de alertas de segurança em risco de ataques.

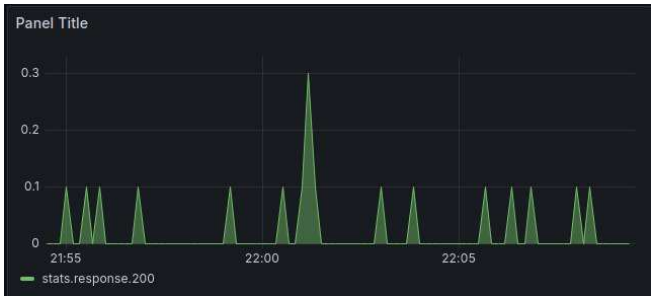

Figura 4: Dashboard do Grafana mostrando métricas do Envoy.

5 AVALIAÇÃO

Nesta seção, apresentaremos alguns casos de testes utilizados para avaliação da solução ao nível de implementação do *proxy* Envoy, seus resultados e discussões.

5.1 Análise de desempenho

Os testes foram planejados para avaliar o desempenho entre o funcionamento atual das requisições feitas para os serviços do LiteMe, que não conta com a atuação de um *proxy*, e a solução implementada, utilizando o *proxy* Envoy como intermediário das requisições. Para avaliação dos aspectos, foi utilizada a ferramenta Apidog [4] que facilita o gerenciamento de APIs. A plataforma fornece suporte para testes automatizados, sendo possível elaborar cenários que avaliam o desempenho da API com base em aspectos diversos. Para avaliação da solução utilizando o Envoy, foi configurado um cenário para realizar um teste de carga, a fim de obter análise dos aspectos de latência, vazão e taxa de erros. Para isso, foi configurado um teste com 10 usuários virtuais, realizando requisições simultaneamente durante um período de 10 minutos.

Para o cenário sem o uso do Envoy, no tempo definido, foram realizadas 5.208 requisições, com média de tempo de resposta de 33ms, e vazão 8,6 requisições por segundo. A taxa de erro obtida foi 0%. Já para o cenário com o uso do Envoy, foram enviadas 5.184 requisições, com média de tempo de resposta de 39ms, e vazão 8,56 requisições por segundo. A taxa de erro foi 0,12% referente ao filtro

de limitação de acesso definido para 900 requisições por minuto, na configuração do Envoy. As figuras 5 e 6 mostram o gráfico gerado pelo Apidog, com a visualização dos valores de requisições por segundo, média de tempo de resposta, erros e quantidade de usuários virtuais em determinado tempo da execução.

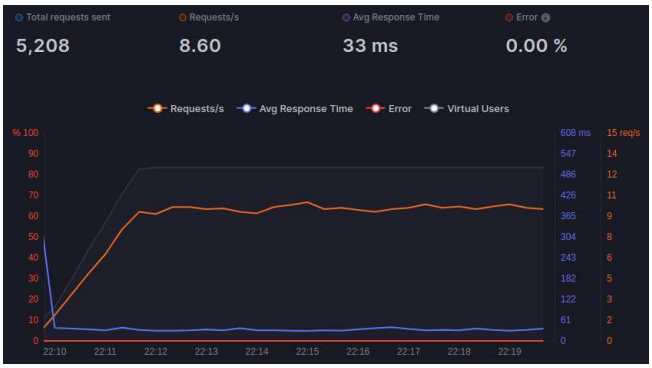

Figura 5: Sem o uso do Envoy.

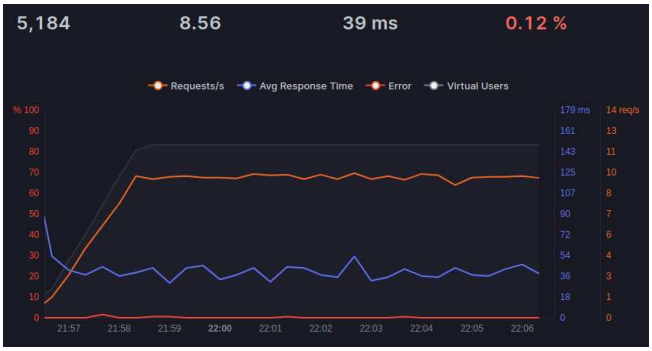

Figura 6: Com o uso do Envoy.

Para o caso de uso do *SmartCampus* com acesso à API do LiteMe, a solução do uso do Envoy como *proxy* para intermediar as requisições não se mostra impeditiva em termos de desempenho. A manutenção dos valores de latência e vazão são indicativos importantes na tomada de decisão sobre a implementação da solução.

5.2 Análise de segurança

Com a implementação da solução, podemos concluir que alguns pontos de riscos de ataques foram mitigados. Em relação ao risco de ataque de DoS, com a implementação do *Rate Limit*, a quantidade de requisições que podem ser feitas em um determinado intervalo de tempo pelos usuários anônimos é limitada, mitigando os riscos de indisponibilidade dos serviços para usuários legítimos. Em relação ao processo de autenticação com o SPIRE, o risco de ataque de falsificação é mitigado pelo processo de verificação e atualização dinâmica de identidades realizado nas requisições e na validação dos usuários autorizados, na configuração do controle de acesso. Consequentemente, ao garantirmos a verificação das identidades autorizadas, mitigamos os riscos de comprometimento da confidencialidade dos dados. Por fim, ao estabelecer uma comunicação mTLS entre Envoy e usuário no processo de requisições, reforçamos a integridade dos dados adicionando uma camada extra de segurança no processo de transferência das informações.

6 CONCLUSÕES

Este trabalho apresentou uma solução de controle de acesso para o caso de uso *SmartCampus* e APIs do LiteMe. Para fornecer segurança no processo de autenticação do proxy e dos projetos selecionados, foi usado o SPIRE como provedor de certificados. Para implementação de um controle de acesso, foi usado o Envoy como *proxy*, intermediando as requisições e garantindo granularidade nas políticas de acesso, verificação de identidades e limitação nas taxas de acesso. A solução foi implementada de forma que não foi necessário alterar a arquitetura do LiteMe e o fluxo de autenticação dos serviços.

Apesar de promissora, a solução possui limitações que devem ser citadas:

(1) É possível que usuários e operadores sintam dificuldades para operar a aplicação e manutenção dos certificados gerados pelo SPIRE, devido à necessidade de adaptar à diversidade de ambiente de execução e também aos diversos perfis de estudantes e profissionais que lidarão com a infraestrutura;
(2) Configurar o *rate limit* de forma que a quantidade de *tokens* seja limitada sobre o tráfego de rede total gera a possibilidade de que as requisições fiquem indisponíveis para usuários legítimos, em caso de ataque de DoS;
(3) Com o aumento do número de projetos e usuários, colocar todas as entradas que representam as identidades se torna pouco eficiente e de difícil manutenção, devido à extensão do arquivo de configuração.

Como sugestão de trabalhos futuros, é importante realizar um aprofundamento nos testes de desempenho, incluindo precisão ao nível de produção e também uso de ferramentas mais robustas de monitoramento. Além disso, a realização de avaliação dos aspectos de segurança do Envoy, de modo a retirar a etapa do uso do *token* no processo de autenticação dos serviços do LiteMe. Implementar formas de otimizar a configuração do arquivo YAML do Envoy, para se adaptar a um cenário maior de usuários de forma eficiente. Por fim, implementar a configuração de rate limit sobre endereço de IP, e não sobre o tráfego total da rede, utilizando serviços externos.

Por fim, como contribuição para o projeto do *SmartCampus*, foi disponibilizada uma documentação para auxiliar usuários[7] e operadores[8] no processo de configuração da infraestrutura necessária para a utilização da solução.

AGRADECIMENTOS

Primeiramente, gostaria de agradecer à minha mãe e às minhas irmãs pelo suporte durante a graduação. A Haniel, pelo suporte e parceria durante todos esses anos. Agradeço também aos amigos feitos ao longo do curso, pela amizade e por agregarem conhecimento.

[7]Link para a documentação do usuário: https://github.com/ufcg-lsd/smartufcg-energia
[8]Link para a documentação dos operadores: https://github.com/ufcg-lsd/smartufcg-energia-ops/

Agradeço a Raisson Souto, pela parceria nas discussões durante esse trabalho, pelo suporte técnico na solução, disponibilizando seu conhecimento em monitoramento e pelas figuras 2 e 3.

Por fim, gostaria de agradecer ao professor Andrey pela orientação neste trabalho, oportunidades dadas e por me guiar no entendimento sobre a profissional que quero ser, e também ao projeto "Ecossistema baseado em IoT para gerência de água e energia"(edital 09/2021, Demanda Universal), uma parceria entre a UFCG e a FAPESQ-PB.

REFERÊNCIAS

[1] [n. d.]. SPIRE Concepts. https://spiffe.io/docs/latest/spire-about/spire-concepts/. Acessado: 12-04-2024.
[2] 2024. RBAC Filter. https://www.envoyproxy.io/docs/envoy/latest/configuration/http/http_filters/rbac_filter. Acessado: 12-04-2024.
[3] 2024. Secret Discovery Service (SDS). https://www.envoyproxy.io/docs/envoy/latest/configuration/security/secret. Acessado: 12-04-2024.
[4] API Dog. n.d.. Testing Process Control. https://apidog.com/help/automated-testing/test-scenarios/testing-process-control Accessed May 3, 2024.
[5] Envoy. 2024. What is Envoy? https://www.envoyproxy.io/docs/envoy/latest/intro/what_is_envoy Acessado: 12-04-2024.
[6] Evan Gilman and Doug Barth. 2021. *Solving the Bottom Turtle: SPIFFE and SPIRE.* https://spiffe.io/pdf/Solving-the-bottom-turtle-SPIFFE-SPIRE-Book.pdf Acessado: 14-04-2024.
[7] Grafana Labs. 2024. Introduction to Grafana. https://grafana.com/docs/grafana/latest/introduction/. Acessado: 02-05-2024.
[8] Graphite. 2024. Overview of Graphite. https://graphite.readthedocs.io/en/latest/overview.html. Acessado: 02-05-2024.
[9] National Institute of Standards and Technology. 2020. *NIST Special Publication 800-207: Zero Trust Architecture.* Special Publication 800-207. National Institute of Standards and Technology, Gaithersburg, MD. 59 pages. https://doi.org/10.6028/NIST.SP.800-207 Available free of charge from https://doi.org/10.6028/NIST.SP.800-207.
[10] SPIFFE. 2024. SPIFFE Overview. https://spiffe.io/docs/latest/spiffe-about/overview/ Acessado: 14-04-2024.
[11] Square. 2024. Ghostunnel. https://github.com/ghostunnel/ghostunnel Acessado: 15-04-2024.