Análise de desempenho de um método de rastreamento implementado como extensão do SPIFFE/SPIRE

Milton P. Pagliusi[1], **Marco A. Marques**[2], **Charles C. Miers**[1]

[1]Departamento de Ciência da Computação (DCC)
Universidade do Estado de Santa Catarina (UDESC)

[2]Escola Politécnica da Universidade de São Paulo (Poli-USP)
Universidade de São Paulo (USP)

milton.neto@edu.udesc.br, charles.miers@udesc.br, marcomarques@usp.br

Abstract. *The adoption of security measures for virtualized environments is a topic of discussion to ensure secure communication within distributed systems, as these environments have more vulnerability points. This article proposes a performance analysis of an application, as a proof of concept, set in SPIRE, to understand the feasibility of using documents developed as a SPIFFE extension.*

Resumo. *A adesão de segurança para ambientes virtualizados é um tópico de discussão para poder garantir uma comunicação segura dentro de sistemas distribuídos, visto que estes ambientes possuem mais pontos de vulnerabilidade. Este artigo propõe uma análise de desempenho de uma aplicação, como prova de conceito, ambientado em SPIRE, para compreensão da viabilidade do emprego de documentos desenvolvidos como uma extensão do SPIFFE.*

1. Introdução

A discussão sobre o desenvolvimento de métodos para aderir segurança em aplicações baseadas em nuvem envolve os desafios de como garantir a segurança nos serviços e processos inseridos em um ambiente virtualizado [Feldman et al. 2020]. A especificação *Secure Production Identity Framework for Everyone* (SPIFFE) é uma solução de código aberto para projetos e organizações que buscam implementar um sistema de gestão de identidade e controle de acesso com uma integração simples em aplicações nativas em nuvem [SPIFFE 2022]. O desenvolvimento de soluções baseadas em SPIFFE, por parte da comunidade da especificação, tem como objetivo cobrir lacunas para casos de uso não previstos originalmente. O cenário abordado tem foco no desempenho de um *token* de portador estendido a especificação, que visa inserir o conceito de identidade transitiva no fluxo de trabalho, e oferecer rastreabilidade dos serviços inseridos neste fluxo.

O caso de estudo para este trabalho faz parte do projeto intitulado **Gerenciamento seguro de identidades federadas: aprimorando e estendendo a arquitetura SPIFFE (SPIFFE-IdT)**, conveniado com a UDESC, USP, IFC e a *Hewlett Packard Enterprise* (HPE), projeto o qual o autor faz parte como bolsista de iniciação científica. O projeto SPIFFE-IdT tem como objetivo aumentar a expressividade das credenciais, i.e., ampliar o escopo de uso dos documentos de identidade nativos do SPIFFE, nos processos de autorização e autenticação, utilizando um *token* de portador não-nativo. O objetivo

deste trabalho é propor uma análise de desempenho do modelo estendido desenvolvido, utilizando *tokens* aninhados, para poder ter uma compreensão melhor sobre a viabilidade de sua adoção em aplicações distribuídas que lidam com um número expressivo de requisições.

O artigo está organizado como segue. Fundamentação sobre os projetos SPIFFE / *SPIFFE Runtime Environment* (SPIRE) e um cenário estendido da especificação na Seção 2. O cenário estendido, desenvolvido como um *token* de portador e a aplicação, como prova de conceito, na Seção 2.2. A proposta de análise do impacto no desempenho, assim como os critérios definidos para a análise na Seção 3.

2. SPIFFE/SPIRE e Cenário Estendido

A adoção de ambientes em nuvem dentro de projetos de aplicações tem mudado diferentes etapas no desenvolvimento destas, com uma abordagem flexível e dinâmica, a partir da oferta de recursos virtualizados. A tecnologia para disponibilizar este modelo de aplicações é feita a partir da abordagem da arquitetura de microsserviços, em que soluções são separadas em pequenos pedaços de software para garantir sua escalabilidade [Feldman et al. 2020]. Os sistemas de gestão de identidade e controle de acesso, como a especificação SPIFFE, são um dos principais métodos na garantia de segurança dentro dos processos inseridos na abordagem da arquitetura de microsserviços.

2.1. SPIFFE

O SPIFFE é um projeto da *Cloud Native Computing Foundation* (CNCF), uma especificação para a padronização de identidades de microsserviços, e tem como objetivo oferecer um *framework* para gestão de identidades interoperáveis entre ambientes heterogêneos [Feldman et al. 2020]. O conjunto de processos e documentos propostos no SPIFFE garante, de forma automatizada, o manuseio e validação de identidades criptográficas, com o propósito de estabelecer uma comunicação segura entre serviços. Os componentes básicos propostos na especificação SPIFFE são [SPIFFE 2022]: (i) SPIFFE ID: um documento de identidade, uma *string* utilizada como um identificador comum para os microsserviços; (ii) *SPIFFE Verifiable Identity Document* (SVID): um documento verificável de identidade criptográfica, utilizado para provar a identidade de um serviço, inserido em um domínio de confiança, para outros componentes; e (iii) SPIFFE *workload Application Programming Interface* (API): é a API que permite as *workloads* (ou microsserviços) obter seus documentos para suas identidades.

A implementação dos componentes previstos na especificação SPIFFE foi chamada de SPIRE, um projeto da CNCF, de código aberto e pronto para produção [Feldman et al. 2020]. O SPIRE é a ferramenta para organizações que tem interesse em adotar a padronização do SPIFFE e integrar dentro de seus projetos de maneira simplificada os processos necessários para delegação e autenticação [SPIFFE 2022], sendo adotado pela Google em Agosto/2023. A arquitetura do SPIRE é composta de dois componentes principais, o agente e servidor SPIRE [Feldman et al. 2020]: (i) Servidor SPIRE: é o componente que desempenha o papel de gestão e emissão de identidades no contexto de um domínio confiável do SPIFFE. Além de autenticar e delegar documentos a Agentes SPIRE; e (ii) Agente SPIRE: este componente é responsável por oferecer a API das *workloads* com o objetivo de oferecer acesso de documentos de identidades às *workloads*, servindo como um intermediário entre *workloads* e o Servidor SPIRE.

Os componentes que compõe a arquitetura do SPIRE são essenciais para o processo de atestação, i.e., o processo utilizado para provar, com segurança, a identidade de uma *workload* ou agente SPIRE através de atributos e informações disponíveis. A autenticação de componentes, recursos e identidades oferece uma forma segura no estabelecimento de comunicação entre serviços, levando em consideração os diferentes ambientes que uma requisição atinge, mas seu contexto de emprego para soluções seguras e integradas não cobre todos os casos de uso necessários para aplicações modernas. As limitações do emprego do SPIFFE para diferentes soluções que podem ser integradas foram a motivação para um esforço de desenvolvimento, por parte da comunidade, de novas soluções como uma extensão para aprimoramento da especificação como um todo.

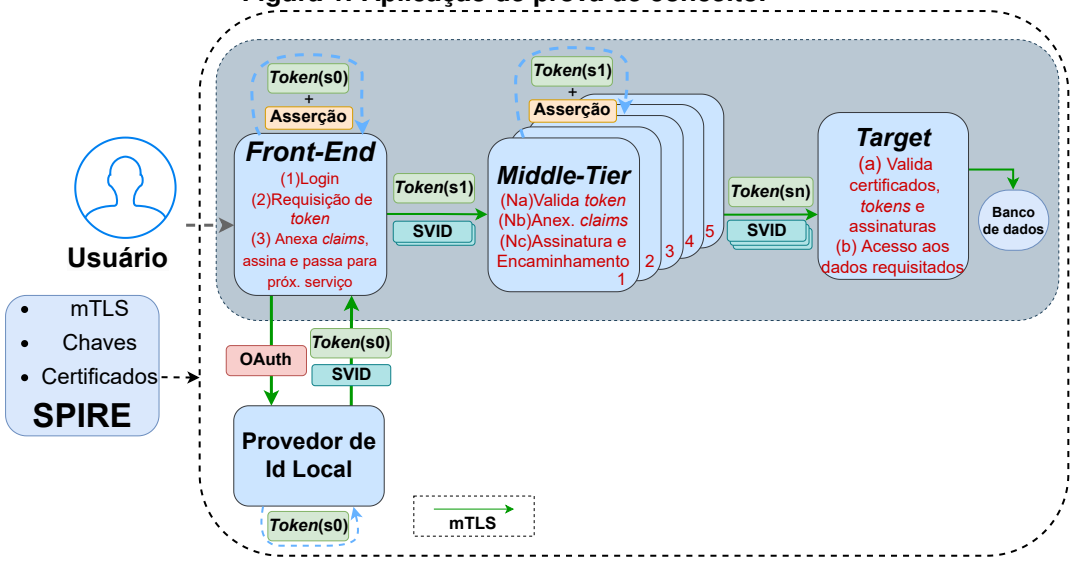
2.2. Cenário Estendido: Identidade Transitiva

A arquitetura proposta no SPIRE é bem definida e sólida, mas diferentes pontos de discussão surgem através das lacunas encontradas na especificação. As lacunas de soluções são alvo de estudo para o desenvolvimento de funcionalidades, documentos e modelos de autenticação estendidos da especificação original. O desenvolvimento do SPIFFE-IdT teve como foco a especificação e o desenvolvimento de uma prova de conceito de um *token* que trouxesse o conceito de identidade transitiva para o SPIRE, utilizando documentos nativos como base, além da emissão, delegação e validação deste *token*. O conceito de identidade transitiva, no contexto empregado pelo projeto, é o de uma identidade que transporta consigo a procedência integral dos nodos que submeteram a mensagem a um processo de autenticação, dado o contexto de uma mensagem inserida em um sistema distribuído. O *token* desenvolvido, adota um formato similar ao formato *JSON Web Token* (JWT), e foi chamado de asserção com assinatura, seu propósito foi servir como um elo entre um usuário-final (solicitante) e a *workload* submetida como responsável desta solicitação, i.e., uma forma de identificar o usuário sem propagar um *token* de acesso (*e.g.*, *token* OAuth) em um ambiente integrado a SPIFFE. O conjunto de asserções é emitido com uma assinatura digital criptográfica (e.g., Elliptic Curve Digital Signature Algorithm (ECDSA)) para garantir a integridade das informações, sendo a emissão feita a partir de uma API não-nativa do SPIRE. Esta API recebe como parâmetros, além da própria requisição de emissão, um *token* OAuth, e utiliza estas informações para emitir a asserção, assinando tal asserção utilizando a chave privada de seu SVID.

O desenvolvimento de um mecanismo de rastreamento fez parte do projeto, através de um modelo de aninhamento de *tokens*, com a capacidade de rastrear o caminho pelo qual o *token* trilhou dentro de saltos entre serviços, i.e., um modelo para *tokens* que tivesse suporte a assinaturas distribuídas e um aninhamento recursivo. O processo de emissão de um *token* ao ser aninhado se faz através de três passos: (i) validação do *token* apresentado (utilizando chaves públicas oriundas do SPIRE); (ii) acrescentar o *token* apresentado como uma *claim* do novo *token*; e (iii) a emissão do novo *token* a partir da assinatura pela *workload* responsável.

O emprego destas novas funcionalidades e documentos foi feito em uma aplicação, como prova de conceito, em que fosse possível integrar num ambiente SPIRE o fluxo de delegação e autenticação de documentos não-nativos, assim como também a operação da API não-nativa para emissão da asserção com assinatura. A prova de conceito (Figura 1 é implementada como uma aplicação web distribuída, simulando um aplicativo bancário, em que um usuário faz requisições (e.g., depósito ou resgatar saldo) via API.

Figura 1. Aplicação de prova de conceito.

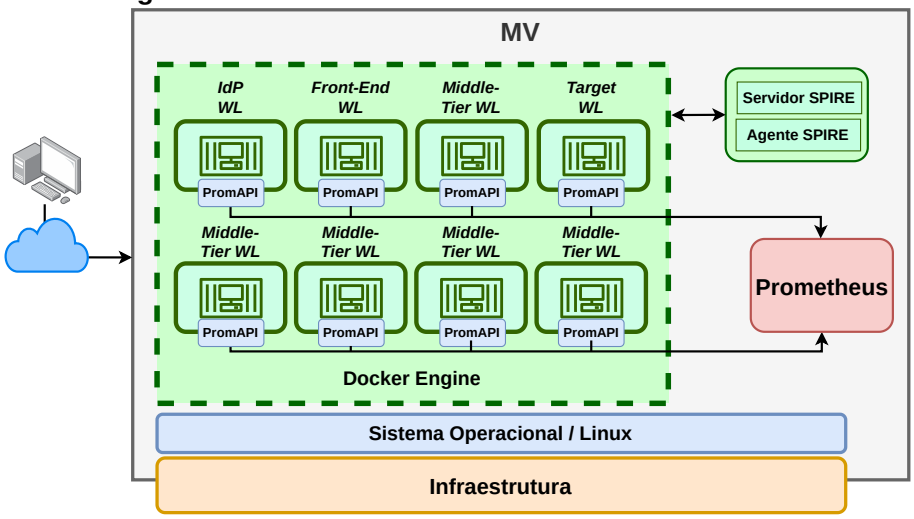


O fluxo de trabalho da aplicação conta com que toda *workload* apresente o *token* aninhado que for emitido por esta, e também apresente todos os certificados (SVIDs) das *workloads* pelo qual o *token* de asserção foi modificado, e utilizam suas chaves privadas para fazer a assinatura digital dos *tokens*, com o propósito de garantir que as asserções sejam verificáveis. A validação de *tokens* no fluxo de trabalho se faz através da asserção com assinatura e o documento de identidade de emissor (SVID), começando pela validação do documento de identidade, a partir da comparação com *claims* inseridas no *token* de asserção, após isso, a chave pública do documento de identidade é recuperada e utilizada para validação da assinatura, e por fim a validação do conjunto de certificados. Qualquer *workload* que recebe requisições em conjunto do *token* aninhado, valida o *token*, adiciona uma nova *claim* de emissor com sua própria chave pública e a *claim* de audiência com a chave pública da *workload* de destino, além de sua própria assinatura. Toda *workload* também encaminha o conjunto de certificados acompanhando o *token*, permitindo a identificação e validação *offline*.

3. Proposta

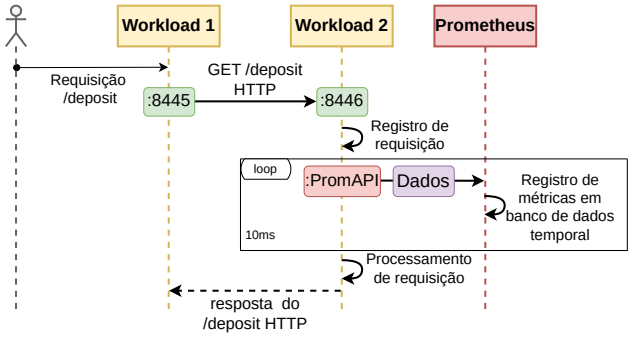
A proposta para a análise de desempenho é fazer uma coleta dos dados relacionados ao custo de recursos computacionais nos processos de emissão, delegação e validação dos *tokens* de asserção com assinatura, com seu emprego dentro de uma aplicação ambientada com a padronização proposta pelo SPIRE. Os dados coletados propostos para a captura são: (i) consumo de Central Process Unit (CPU); (ii) consumo de memória; (iii) *payload* de emissão e validação do *token*; e (iv) tempo de execução dos processos. O conjunto de dados propostos são escolhidos para uma compreensão do impacto que o fluxo de delegação e autenticação estendido tem na aplicação como um todo, e tem como fim servir como uma base referencial do desempenho. A ferramenta escolhida para a coleta destas métricas é a ferramenta de monitoramento chamada Prometheus [Prometheus 2023] (Figura 2), sendo a principal ferramenta para coletar dados de desempenho da aplicação.

Figura 2. *Testbed* utilizando a ferramenta Prometheus.

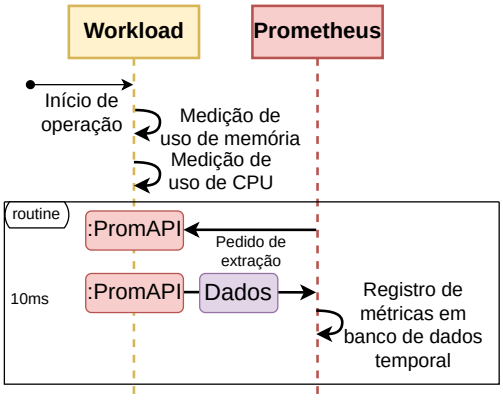


As métricas escolhidas em relação a parâmetros de execução, e.g., consumo de processamento e memória são implantados como uma rotina de medição (Figura 3(b)), em que o uso de CPU é medido como porcentagem de uso de núcleos, e o uso de memória medido em bytes, com a captura feita entre intervalos de dez milissegundos. As métricas escolhidas em relação aos custos das requisições, como o *payload* das asserções, e a latência dos processos de emissão e validação, tem a sua medição implementada diretamente no multiplexador de rotas das APIs (Figura 3(a)), em que os dados de cada requisição é coletado, como número total, tamanho em bytes, tamanho da resposta, tempo de duração da requisição.

Figura 3. Diagramas de captura utilizando Prometheus.



(a) Diagrama de sequência de captura de requisições.



(b) Diagrama de sequência de captura de dados de execução.

Para o plano de testes, os seguintes cenários foram especificados:

1. Cenário simplificado: a prova de conceito em operação, medindo o custo da sua comunicação e recursos computacionais como um todo.
2. Cenário referencial: a prova de conceito em operação, medindo o custo da comunicação, com foco no custo dos recursos envolvendo os cinco saltos intermediários (componentes *Middle-Tier*).

3. Cenário de produção: neste cenário de teste focado em escalabilidade, a aplicação de prova de conceito será implantada em grande escala para avaliar seu desempenho sob condições de estresse.

Os cenários descritos são executados para permitir a comparação entre o desempenho da aplicação, no emprego do *token* para permitir o rastreamento dos fluxos de trabalho em diferentes saltos. O experimento tem a proposta de medir e trazer informações sobre o comportamento da aplicação como prova de conceito e a utilização dos *tokens* não-nativos. O objetivo é tornar um ponto de análise, o comportamento da prova de conceito em um contexto escalável, para ter uma compreensão a fundo do custo da operação do método de segurança, levado ao estresse. A adoção de mecanismos similares, i.e., métodos seguros que utilizam assinaturas digitais criptográficas, ou a implementação de um *bearer token*, é feita com o propósito de aderir segurança e confiabilidade em ambientes em nuvem, porém estes mecanismos não podem comprometer o desempenho das aplicações ao serem implantados em sistemas distribuídos.

4. Considerações

A inspiração para o modelo desenvolvido de asserções com assinatura foi a abordagem proposta por *tokens* de baixo custo computacional, como o *biscuit*, que conta com um *payload* pequeno e um rápido processo de validação [Couprie 2021] A implementação do *token* de asserções acompanhadas por assinaturas digitais por parte das *workloads*, num contexto de validação, oferece segurança contra a quebra da integridade do conteúdo dos *tokens*, também oferece um caso estendido não previsto pelo SPIRE, o recurso de um componente poder fazer declarações (como as *claims*) arbitrariamente. As discussões para trabalhos futuros envolvem a adoção de um algoritmo assinatura digital de criptografia pós-quântica, conhecido como *Crystals-Dilithium* [Ducas et al. 2018], e atualmente, o desenvolvimento um novo esquema de *tokens* leves, similares ao SVID, com o intuito de servir como um *token* de portador, apelidados de *lightweight SVID*.

Agradecimentos: Os autores agradecem o apoio do LARC/USP, LabP2D/UDESC, HPE, FDTE e FAPESC.

Referências

Couprie, G. (2021). Biscuit, the foundation for your authorization systems. https://www.clever-cloud.com/blog/engineering/2021/04/12/introduction-to-biscuit/. Acesso em: 05 Jun. 2022.

Ducas, L. et al. (2018). Crystals-dilithium: A lattice-based digital signature scheme. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 238–268.

Feldman, D. et al. (2020). Solving the bottom turtle — a spiffe way to establish trust in your infrastructure via universal identity.

Prometheus (2023). Prometheus. https://prometheus.io/. Acesso em: 08 Jun. 2023.

SPIFFE (2022). Spiffe. https://spiffe.io. Acesso em: 10 Jul. 2023.