Lucas R. Cupertino Cardoso

Proposta de melhoria da expressividade de credenciais SPIFFE para gerenciamento de identidades em ambientes federados

Brasil

2023, v<1.0>

Lucas R. Cupertino Cardoso

Proposta de melhoria da expressividade de credenciais SPIFFE para gerenciamento de identidades em ambientes federados

Trabalho de conclusão de curso apresentado ao Departamento de Engenharia de Computação e Sistemas Digitais da Escola Politécnica da Universidade de São Paulo para obtenção do Título de Engenheiro.

Universidade de São Paulo – USP

Escola Politécnica

Departamento de Engenharia de Computação e Sistemas Digitais (PCS)

Orientador: Prof. Dr. Marcos Antonio Simplício Jr.

Brasil

2023, v<1.0>

Agradecimentos

Dedico este trabalho à mulher quem me criou e deu forças para seguir meus sonhos desde criança, minha avó, Yvone Cupertino Cardoso, com todo seu otimismo inabalável mesmo nos momentos de maior tormenta. Além disso, a outras três pessoas fundamentais na minha vida: Beatriz Pires de Melo Marques Pazine, minha namorada, agradeço por todo o companheirismo, doçura, gentileza e persistência inigualáveis dia após dia; aos meus amigos Maria Renata Godoy e Gabriel Ramos de Oliveira pela amizade e lealdade mesmo nas horas em que a luz parecia não existir. Vocês três, no espetáculo que é a vida, estiveram sempre ali na primeira fileira me incentivando a seguir em frente.

Além disso, não poderia esquecer de agradecer meus colegas e amigos, Gabriel Kenji Godoy Shimanuki e Otávio Felipe de Freitas, que estiveram comigo durante o caminho até o sonho de concluir a graduação e me tornar engenheiro. Aos meus professores e professoras que contribuíram a minha formação, em especial ao professor Marcos Antonio Simplício Junior pela orientação e explicação das inúmeras dúvidas que eu tive (e, eventualmente, repetem-se).

Finalmente, agradeço aos colegas, não só de projeto, mas do LARC, pelas discussões e reuniões.

A essas e outras pessoas fundamentais na minha vida, agradeço, com todo o coração, pela aventura que esses 26 anos de existência trouxeram.

Resumo

Identidades federadas são de particular importância atualmente dada a crescente disseminação de microsserviços. Elas servem para vincular a identidade eletrônica e os seus respectivos atributos espalhados por diferentes sistemas de gerenciamento de identidades. Nesse sentido, os sistemas gerenciadores de identidade – *Identity Management Systems* – têm ganhado importância cada vez maior a fim de conferir maior segurança e confiança. O SPIFFE – *Secure Production Identity Framework For Everyone* – surge como uma tecnologia *open-source* alternativa com o intuito de gerenciar de maneira segura a produção de identidades em ambientes federados. O presente trabalho tem como objetivo explorar modificações não implementadas no SPIFFE, a fim de garantir maior expressividade de credenciais e otimizar o desempenho do *framework*. Mais especificamente, a modificação estudada está relacionada aos documentos de identidade verificáveis SPIFFE – SVIDs – um documento que permite que uma *workload* prove sua identidade caso haja requisição para tanto.

Palavras-chave: engenharia. computação. segurança. criptografia. identidades digitais. identidades federadas. SPIFFE.

Lista de ilustrações

Figura 1 – Federeação SPIFFE ilustrada conforme (FELDMAN et al., 2023). . . . 30
Figura 2 – SPIFFE *Runtime Environment* 31
Figura 3 – Diagrama de funcionamento do modelo aninhado. 32
Figura 4 – Tecnologias utilizadas para o processo de obtenção de métricas na *PoC* com requisições automáticas e sucessivas. 38
Figura 5 – Operação de *append* entre dois *tokens* LSVID. 40
Figura 6 – Arquitetura do modo de funcionamento ID para a prova de conceito discutida nesse documento 43
Figura 7 – Passo-a-passo de autenticação mútua com mTLS. Imagem retirada do site da Cloudfare. 44
Figura 8 – Exemplo do uso do mTLS dentro da *Proof of Concept* ilustrando-se a mútua autenticação entre *asserting workload* e *subject workload*. 44

Lista de quadros

Lista de tabelas

Tabela 1 – Especificação dos campos de um *token* LSVID. 39

Lista de abreviaturas e siglas

API	*Application Programming Interface*
CPU	*Central Processing Unit*
IdM	*Identity Management*
JWT	*JSON Web Token*
mTLS	*Mutual Transport Layer Security*
NIST	*National Institute of Standards and Technology*
PKI	*Public Key Infrastructure*
SPIFFE	*Secure Production Identity Framework for Everyone*
SPIRE	*SPIFFE Runtime Environment*
SVID	*SPIFFE Verifiable Identity Document*
TLS	*Transport Layer Security*
TTP	*Trusted Third Party*
USP	Universidade de São Paulo
XML	*Extensible Markup Language*
ZKP	*Zero-knowledge proof*

Lista de símbolos

$\sigma_i\,(\mathsf{msg})$	Assinatura digital do ente P_i sobre a mensagem msg
$\mathcal{H}$	Função de *hash* universal
$\mathcal{H}_{\mathsf{SHA-512}}$	Função de *hash* SHA-512
$s_1 \| \| s_2$	Concatenação de duas *strings* s_1 e s_2
pk_i	Chave pública do ente P_i
sk_i	Chave privada do ente P_i
id_i	SPIFFE-ID do ente P_i
p_i	*Payload* do ente P_i
c_i	*Claim* do ente P_i
$\mathcal{L}_i \equiv (\mathsf{p}_i, \mathsf{c}_i, \square, \sigma_i)$	LSVID associado ao ente P_i com $\square$ sendo um parâmetro que pode ou não ser vazio.

Sumário

1 INTRODUÇÃO 19
1.1 Motivação 21
1.2 Objetivo 22
1.3 Organização do Trabalho 23

2 ASPECTOS CONCEITUAIS 25
2.1 SPIFFE 25
2.1.1 SPIFFE Verifiable Identity Document – SVID 25
2.1.1.1 Certificados X509 e JWT dentro do ambiente SPIFFE 26
2.1.1.2 Vantagens do uso do X509 + TLS 26
2.1.1.3 Formação de Tokens 27
2.1.2 OAuth 27
2.1.3 Trust Bundles 28
2.1.4 Ciclo de Vida de Identidades e Certificados 28
2.1.4.1 Emissão de Identidade 28
2.1.4.2 Revogação de Certificados 28
2.1.4.3 Expiração de Certificados 29
2.1.5 Delegação e Validação de Cadeias de Confiança 29
2.1.6 SPIFFE Federation 29
2.1.7 SPIFFE Workload API 30
2.2 SPIRE 30
2.2.1 SPIRE Server 30
2.2.2 SPIRE Agent 31
2.3 Modelo Aninhado – Nested Model 32

3 METODOLOGIA 33
3.1 Revisão Bibliográfica 33

4 ESPECIFICAÇÃO DE REQUISITOS 35

5 DESENVOLVIMENTO DO TRABALHO 37
5.1 Tecnologias Utilizadas 37
5.1.1 Linguagem Go 37
5.1.2 Prometheus 37
5.1.3 Apache Jmeter 37
5.2 Projeto e Implementação 38

5.2.1 Componentes . . . 38
5.2.2 Funcionaliddes . . . 39
5.2.3 Ciclo de Vida de um LSVID . . . 39
5.2.3.1 Criação . . . 39
5.2.3.2 Extensão de Identidade . . . 40
5.2.3.3 Extensão de Token . . . 41
5.2.4 Modos de Funcionamento . . . 41
5.2.4.1 ID . . . 41
5.2.4.2 Outros modos: Anonymous e Double Mode . . . 42
5.3 Arquitetura . . . 42
5.3.1 Gerenciamento de Plugins e SVIDs . . . 42
5.3.2 mTLS - Mutual TLS . . . 42
5.4 Aplicação . . . 44

6 CONSIDERAÇÕES FINAIS . . . 45
6.1 Conclusões . . . 45
6.2 Contribuições . . . 45
6.3 Perspectivas de Continuidade . . . 46

REFERÊNCIAS . . . 47

1 Introdução

Dezenas de milhares de ataques cibernéticos ocorrem anualmente segundo técnicas de *phishing* (COOK, 2023) ou roubo de credenciais de funcionários. Com o advento de aplicações em *cloud* e a disseminação da força de trabalho móvel – *i.e.* grupos de funcionários que não estão fisicamente restritos a uma dada localidade – o acesso a recursos é feito a partir de diferentes redes o que aumenta a suscetibilidade a eventuais ataques. Somado a isso, num cenário de desenvolvimento de arquiteturas cujo objetivo é o máximo desempenho, os modelos de segurança devem evoluir também.

Softwares vêm sendo desenvolvidos de modo a abrigar quantidades cada vez maiores de microsserviços os quais espalham-se através de máquinas virtuais em localidades diferentes executando serviços distintos, *e.g.* ambientes *cloud* ou *data centers* privados. À medida que novas tecnologias *e.g. containers*, microsserviços, computação em *cloud* e *serverless functions*, um fato é claro: existe um número maior de fragmentos de serviços. Isso aumenta tanto o número de potenciais vulnerabilidades que um atacante pode explorar, mas também faz com que o gerenciamento de defesas perimétricas praticamente impossíveis.

Uma identidade digital é uma representação única de uma entidade que seja suficiente para identificar esta entidade em uma transação *online* (SILVA, 2022; GRASSI; GARCIA; FENTON, 2017). Uma identidade digital é sempre única no contexto de um serviço, mas não define univocamente o sujeito em todos os cenários de utilização. Uma entidade, por sua vez, é qualquer coisa existente no mundo real e capaz de possuir múltiplas identidades. A prova de identidade estabelece que uma entidade é quem ela afirma ser em um processo de autenticação digital.

Autenticação digital é o processo de determinação da validade de um ou mais autenticadores usados para reivindicar uma identidade digital. Uma autenticação estabelece que um sistema tentando acessar um serviço tem sob controle as tecnologias usadas para autenticar. O sucesso do processo garante que o sistema acessando o serviço hoje é o mesmo que acessou o serviço anteriormente (GRASSI; GARCIA; FENTON, 2017). O elemento que será provado chama-se *applicant*. Quando esse completa com sucesso a autenticação, ele é referido como *subscriber*. Identidades digitais representam um desafio técnico porque o processo geralmente envolve a validação e autenticação de sistemas através de redes públicas de modo a existir diversas oportunidades para fraudar a identidade de um sistema.

Nesse contexto, o gerenciamento seguro de identidades (*Identity Management* – IdM) refere-se ao conjunto de métodos usados durante o manuseio de identidades digitais de participantes de um sistema, sejam eles indivíduos ou sistemas em si (*e.g. hardware* ou *software*). Não obstante, um sistema IdM engloba políticas e ferramentas cujas facilidades

estão desde a criação e verificação até revogação de identidades, principalmente em ambientes federados (MALER; REED, 2008) (*e.g.* ambientes de computação nativa da nuvem). Mais especificamente, ambientes federados são um padrão arquitetural que permite a interoperabilidade e compartilhamento de informações entre diferentes aplicações, sistemas ou programas [A FEDERATED ARCHITECTURE FOR DATABASE SYSTEMS].

A gestão de identidade e acesso integra, portanto, processos de negócios e tecnologias para permitir a autenticação e autorização de sujeitos antes e durante uma transação *online*. Sendo assim, num cenário de desenvolvimento de sistemas cada vez mais complexos, arquiteturas desse tipo contribuem para a distribuição dinâmica de identidades e, portanto, a portabilidade de identidades digitais.

Autenticação federada ou identidade federada, é um meio de vincular a identidade eletrônica e os seus atributos, que estão espalhados por múltiplos sistemas de identidade. O Single Sign-On (SSO, ou, login único) é um subconjunto da identidade federada, o que damos ênfase neste artigo. O ganho com a autenticação federada é haver uma experiência única para o usuário e um controle centralizado da identidade deles.

1.1 Motivação

Uma solução de IdM federado capaz de permitir a identificação segura de sistemas de *software* em ambientes dinâmicos e heterogêneos é o *Secure Identity Framework for Everyone* – SPIFFE (FELDMAN et al., 2023; AN. . . , 2023). De maneira resumida, o SPIFFE consiste numa ferramenta de código aberto (*open source*) baseada numa implementação de referência também de uso aberto – SPIRE – capaz de emitir credenciais apenas para cargas de trabalho que tenham sido atestadas em infraestruturas modernas e heterogêneas. Essa característica é de grande destaque uma vez que a não emissão prévia de credenciais estáticas de longa duração para *workloads* é sinônimo de menor exposição à captura por atacantes.

SPIFFE e SPIRE visam fortalecer a identificação de componentes de *software* de uma maneira padronizada capaz de ser utilizada através de sistemas distribuidos por qualquer um e em qualquer lugar. Além disso, num cenário de negócios, o SPIFFE e o SPIRE podem reduzir significativamente custos associados ao *overhead* no gerenciamento e emitindo certificados criptográficos (*e.g.* certificados X.509), e acelerar o desenvolvimento e *deployment* por meio da exclusão da necessidade dos desenvolvedores em entender as tecnologias de identidade e autenticação requisitadas para garantia de uma comunicação segura entre serviços.

O SPIFFE traz ainda um entendimento da constituição da identidade de um componente de *software*. Aproveitando-se da federação do SPIFFE, componentes em sistemas e organização diferentes podem estabelecer uma comunicação confiável para, de maneira segura, se comunicar sem lidar com o *overhead* da criação de VPN's, *one-off certificates* ou compartilhamento de credenciais. O SPIRE ainda detém um mecanismo de atestação multi-fator executando em tempo real para determinar, com precisão, a emissão de identidades criptográficas. Isso também emite, distribui e renova automaticamente credenciais com tempo de vida curtos a fim de garantir que a arquitetura de identidade de uma organização reflita, com precisão, o estado das *workloads*.

Sendo assim, no contexto de uso do SPIFFE, o desenvolvimento de um documento verificável mostra-se fundamental ao monitoramento dos processos em curso dentro da organização, bem como favorece a consolidação de um ambiente mais seguro.

1.2 Objetivo

O objetivo desse trabalho é propor um modelo de documento de identidade capaz de otimizar o funcionamento do SPIFFE com a criação de *tokens* mais leves juntamente com o fornecimento de funcionalidades de segurança mais refinadas *e.g.* delegação, atestação e validação de cadeias inteiras de confiança.

As modificações propostas visam atender as demandas da comunidade no sentido de desenvolver soluções para casos de uso não previstos originalmente pelo SPIFFE. Além disso, os objetivos são norteados pelo levantamento bibliográfico discutido no Capítulo 2.

Finalmente, como consequência do cumprimento dos objetivos desse trabalho, tem-se a geração de uma prova de conceito – *Proof of Concept* (PoC) – para simulação do funcionamento das propostas de melhoria do documento de identidade.

1.3 Organização do Trabalho

Esta monografia está organizada da seguinte forma: no Capítulo 2 são apresentados os conceitos fundamentais ao estudo da produção de documentos de identidade dentro do *framework* do SPIFFE-SPIRE. Optou-se por dividir o capítulo em três partes: SPIFFE, SPIRE e Modelos de Identidade. Nas duas primeiras partes são apresentadas as características, aplicações e organização sob o ponto de vista de uma PKI. Discute-se, também, os modelos de certificados usados – X509 e JWT – para criação do documento de identidade contrapondo-se as duas opções. Tópicos como a formação de *tokens* destacando-se a presença de propriedades de segurança, delegação, atenuação e validação de cadeias de confiança. Por fim, na última parte do capítulo é apresentado o *Nested Model* – modelo aninhado.

O terceiro capítulo aborda a especificação de materiais e métodos utilizados no projeto, descrevendo as principais tecnologias utilizadas juntamente com a especificação dos requisitos funcionais e não funcionais do projeto. O capítulo começa com a revisão bibliográfica citando algumas das principais diferenças cruciais ao entendimento do projeto e a seguir são abordados os requisitos propriamente ditos. Ao final comenta-se brevemente as fases anteriores do projeto que contribuíram, em certa medida, ao desenvolvimento da aplicação final com o *lightweight* SVID.

O quarto capítulo dá enfoque ao desenvolvimento do projeto de modo a discutir e descrever as principais melhorias trazidas ao uso do novo documento de identidade do SPIFFE com discussão de casos de uso. O capítulo 5 discute as etapas de desenvolvimento do trabalho. O último capítulo conclui o trabalho apresentando algumas das principais considerações bem como rumos futuros à pesquisa desenvolvida.

2 Aspectos Conceituais

2.1 SPIFFE

O SPIFFE é um framework que visa fortalecer a identificação de componentes de *software* tirando-se vantagem de sistemas distribuídos. Construído com base no SPIRE (Seção 2.2) define padrões para atribuição de identidade a *softwares* visando garantir duas propriedades: interoperabilidade em uma organização e independência da plataforma. Visando-se cumprir esses preceitos, o SPIFFE define as interfaces e documentos necessários à obtenção e validação de identidades criptográficas de maneira integralmente automatizado (FELDMAN et al., 2023). As partes fundamentais do SPIFFE são as seguintes:

- **SPIFFE ID**: representação de uma identidadde (*i.e.* o nome de um serviço de software);
- **SVID**: um documento criptograficamente verificável usado para provar a identidade de um serviço a outro;
- **SPIFFE Trust Bundle**: formato de representação da chave pública em uso para uma dada autoridade emissora SPIFFE, ou seja, representa o conjunto de chaves públicas de um domínio de confiança e;
- **SPIFFE Federation**: mecanismo de compartilhamento dos *trust bundles*;
- **SPIFFE Workload API**: uma API usada pelos serviços para que eles obtenham um ID sem a necessidade de autenticação.

2.1.1 SPIFFE Verifiable Identity Document – SVID

No contexto do SPIFFE, um SVID é um documento de identidade criptograficamente verificado usado para prova de identidade de um serviço a outro (FELDMAN et al., 2023). Um SVID contém um único SPIFFE ID e é assinado por uma autoridade emissora representando o domínio de confiança ao qual o serviço pertence.

O SPIFFE ID, por sua vez, é uma *string* que funciona como o nome único do sistema modelada segundo uma URI composta de diferentes partes. Um exemplo fictício seria dado pela seguinte URI: `spiffe://exemplo.com/poli`. Podemos observar a presença de três partes principais. São elas:

1. Esquema URI `spiffe://`: serve para diferenciar um identificador SPIFFE de um identificador URL ou outro tipo de localizador de rede;

2. *Trust domain* `exemplo.com`: pode ser composto por um ou mais termos representando os domínios de confiança atrelados a uma organização inteira. São usados para gerenciar regiões limítrofes em termos de segurança e administração entre organizações distintas;

3. Serviço `poli`: representa a própria *workload*.

É importante notar alguns pontos: o primeiro deles é que um domínio de confiança é uma parte do SPIFFE ID sobre o qual um conjunto de chaves públicas é considerado oficial. Isso é de fundamental importância uma vez que cada domínio de confiança pode ter diferentes autoridades emissoras e, portanto, o eventual comprometimento de uma dessas autoridades não interfere nas demais uma vez que as chaves públicas não são as mesmas. Isso confere comunicação segura entre partes que não confiam completamente umas nas outras – preceito das arquiteturas de confiança zero [CONFIANÇA ZERO].

Um segundo ponto importante de se notar é que cada organização é livre para escolher a construção do SPIFFE ID uma vez que, como premissa de projeto do *framework*, deve-se haver flexibilidade na representação da identidade de modo a facilitar o entendimento tanto por humanos quanto máquinas.

Essas duas ressalvas contribuem para o melhor entendimento da federação SPIFFE discutida na seção 2.1.6 de maneira mais detalhada. No entanto, é essa organização do SPIFFE que, por exemplo, permite a integração de serviços distintos dentro de uma organização.

2.1.1.1 Certificados X509 e JWT dentro do ambiente SPIFFE

Como premissa de projeto, o SPIFFE visa facilitar o gerenciamento de identidades de modo a conferir propriedade agnóstica do ponto de vista dos *softwares* utilizados em diferentes domínios (*e.g.* organizações ou microsserviços). Nesse sentido, são dois os principais formatos utilizados dentro do contexto do SPIFFE: X509 e JWT.

O primeiro consiste na codificação de um SPIFFE ID segundo o padrão de um certificado X509 [RFC 5280]. O segundo corresponde ao JWT-SVID [RFC7519]. São utilizados como *tokens* ao portador para provar a identidade a pares na camada de aplicação. Esse último fato será importante na seção 5.2 uma vez que a prova de conceito para demonstração do novo modelo de credencial será feito segundo a simulação de uma aplicação bancária.

2.1.1.2 Vantagens do uso do X509 + TLS

Os JWT-SVIDs são suscetíveis a uma classe de ataques chamados ataques de replay [REPLAY]. A fim de mitigar as consequências desse vetor de ataque, três mecanismos são

implementados pelo SPIFFE. São eles:

1. JWT-SVIDs devem ser transmitidos somente através de canais seguros *e.g.* mTLS [mTLS e ZTA] e a implementação de TLS contempla documentos baseados em certificados X.509;

2. A *claim* de *audience* (`aud`) deve ser definida para uma correspondência inequívoca somente ao *token* da entidade ao qual se destina. No caso do JWT-SVID a confiança é indiretamente concedida aos destinatários do documento *e.g.* se *Alice* possui um *token* com *audiences* *Bob* e *Eve*, então, ao enviar o *token* a *Eve*, *Eve* pode se passar por *Alice* ao enviar o mesmo token para *Bob*;

3. JWT-SVIDs devem possuir um tempo limite de uso restringindo a janela de oportunidade de um atacante de comprometer o sistema.

Os X509-SVIDs apresentam melhores propriedades de segurança em relação aos JWT-SVIDs uma vez que podem ser utilizados em conjunto com TLS de modo a inviabilizar ataques de *replay* por um intermediário. [DAR EXEMPLO DE ATAQUE DE REPLAY]

2.1.1.3 Formação de Tokens

Tendo-se em vista o que foi discutido nas seções 2.1.1.1 e 2.1.1.2, dentro do SPIFFE, os documentos de identidade apresentam-se como estruturas assinadas e são comumente chamadas de *tokens*.

O *token* pode conter uma série de asserções sobre o *subject* – ou tópico – como um identificador ($\mathsf{id}_{\mathsf{token}}$) e autorizações de acesso enquanto que a identidade do tópico e o *token* do emissor podem ser verificados usando seus respectivos documentos de identidade. Existe a possibilidade de representação da identidade por meio de certificados X509 ou inserindo-a no próprio *token*.

2.1.2 OAuth

O *OAuth* (*Open Authorization*) é um padrão aberto de delegação de autorização utilizado em contextos de ganho de permissões (*e.g. login* em sites de terceiros) por meio do *login* a uma conta de provedor de identidade (aqui comumente referido como IdP). No contexto do projeto, como será descrito na Seção 5.2, o OAuth é de fundamental importância no momento em que o *front-end* da aplicação (ver Seção **??**) requer o acesso a OKTA [OKTA], faz a requisição do *token*, anexa as informações (*i.e.* as *claims*), assina e transmite aos *middle tiers*. É apenas após receber o OAuth que o IdP emite o primeiro *token* para ser utilizado no início da cadeia de validação.

2.1.3 Trust Bundles

A definição de *trust bundle* é dada no início da Seção 2.1. No entanto, aqui iremos comentar, brevemente, algumas particularidades deste documento. A primeira delas é que cada documento SVID contém um conjunto de particularidades em termos da forma de representação de um *trust bundle*. Cada domínio de confiança SPIFFE possui um *bundle* associado e o conteúdo desse é usado na validação dos SVIDs que dizem pertencer a um dado domínio. O processo é aquele de verificação de um algoritmo de assinatura digital [ECDSA, RSA, SCHNORR].

A presença de apenas chaves públicas em um *trust bundle* permite evitar desperdício de recursos para assegurar confidencialidade, mas apenas integridade. Eles são formatados segundo um documento JWK e compatíveis com tecnologias de autenticação *e.g. OpenID Connect*

2.1.4 Ciclo de Vida de Identidades e Certificados

Dentro do cenário das PKIs existem mecanismos que tornam o gerenciamento e a autenticação de identidades digitais mais fáceis e seguros. Nessa seção serão destacados a emissão de identidades, revogação e expiração de certificados.

2.1.4.1 Emissão de Identidade

À medida que novos *softwares* passam a compor um sistema, novas identidades digitais devem ser emitidas. Mais precisamente, um certificado do tipo X.509 chamado CSR – *Certificate Signing Request* – é criado com uma respectiva chave privada sk. No entanto, após a criação, esse documento carece de algo fundamental em uma PKI: a assinatura da CA. Nesse ponto, o certificado não é válido. É necessário enviar o CSR a uma CA para que haja a validação.

2.1.4.2 Revogação de Certificados

Certificados são válidos até o momento em que expiram a menos que haja revogação em alguns instante de tempo anterior. São diversas as causas: desde a perda ou destruição irreversível da chave privada, comprometimento da sua segurança ou a atualização de um dos identificadores do certificado de modo a torná-lo inválido [BRANDS].

Enquanto a revogação é uma circunstância excepcional, a verificação do *status* de revogação de um certificado que não inspirou não segue o mesmo padrão. A entidade responsável pela verificação pode seguir uma das seguintes estratégias: ter o *status* de validação do certificado *online* ou, periodicamente, fazer o *download* de uma lista atualizada e assinada digitalmente, a CRL – *Certificate Revocation List*.

No caso de certificados X.509v3, o conjunto de todos os certificados emitidos é dividido de modo que cada um tenha a própria CRL, ou seja, cada certificado possui um ponteiro em direção à parte da CRL que indica seu *status* de revogação – *CRL Distribution Points* [PERLMAND & KAUFMAN, PAG 38 DO BRANDS].

2.1.4.3 Expiração de Certificados

Todo certificado tem uma data de expiração. São inúmeras as razões para tanto: do ponto de vista de segurança, por exemplo, a presença de uma data de expiração pode diminuir a janela de oportunidade de um atacante; enquanto que, sob a perspectiva de desempenho, limitar o tamanho CRLs e melhorar gerenciamento da obsolência do componente de *software* proprietário do certificado.

2.1.5 Delegação e Validação de Cadeias de Confiança

É importante destacar a presença do processo de delegação *i.e.* a autorização de Autoridades Certificadoras (do inglês, *Certification Authorities*) – CAs – de baixa ordem por CAs de ordem superior. Esse processo se dá pela assinatura do certificado da CA de baixa ordem pela CA em maior nível na hierarquia. Sabendo-se que cada CA tem sua chave e certificado, este, ao ser assinado, gera uma cadeia de confiança que permite que menos chaves sejam conhecidas.

No entanto, isso também acarreta a uma desvantagem no uso de certificados X509: qualquer CA pode assinar qualquer certificado sem restrições. Nesse cenário, um atacante poderia agir como uma CA intermediária e conseguir aprovação de qualquer CA intermediária presente originalmente na PKI e, finalmente, emitir qualquer identidade desejada. Nesse sentido, é imprescindível que cada CA e suas respectivas filhas sejam completamente confiáveis.

2.1.6 SPIFFE Federation

Uma PKI é definida como o conjunto de funções, políticas, *hardwares*, *softwares* e procedimentos necessários a criação, gerenciamento, distribuição, uso, armazenamento, revogação de certificados digitais e gerenciamento de cifração de chave pública. As PKIs permitem que documentos de identidade digitais sejam validados localmente, *offline* utilizando-se um conjunto pequeno de chaves.

No SPIFFE, partindo-se da premissa de comunicação entre diferentes ambientes e somando-se os preceitos de arquiteturas de confiança zero, é necessário um mecanismo de compartilhamento das chaves públicas de um domínio. Apenas ele pode ser utilizado para verificar e autorizar serviços que, quando em domínios estrangeiros (aqueles diferentes

do qual o serviço pertence) possam validar as identidades do *trust domain* local. A esse mecanismo dá-se o nome de *bundle endpoint*.

De maneira ainda mais simplificada, *bundle endpoints* são serviços HTTP protegidos por TLS – *Transport Layer Security*. Ao processo de vínculo entre dois domínios distintos dá-se o nome de federação (ou *SPIFFE federation*).

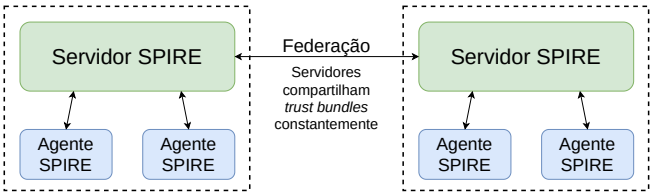

Figura 1 – Federeação SPIFFE ilustrada conforme (FELDMAN et al., 2023).

2.1.7 SPIFFE Workload API

Corresponde a uma API local usada pelas *workloads* para obter seus documentos de identidade (SVIDs), *trust bundles* (conjunto de chaves públicas) e informações relacionadas. É interessante observar que o fato da API ser local permite o aproveitamento de funções relacionadas ao sistema operacional da máquina *host* para, por exemplo, identificar *workloads* sem requerer autenticação direta.

Em termos de arquitetura, a API funciona como um servidor gRPC [EXPLICAR] com *stream* bidirecional permitindo a atualização dos campos das *workloads* à medida que é necessário. Além disso, não são necessárias quaisquer autenticações de segredos conjuntamente às *workloads*.

2.2 SPIRE

O SPIRE – *SPIFFE Runtime Environment* – é a implementação das cinco partes das especificações do SPIFFE. Ele possui dois componentes principais: o *server* e o *agent*. O primeiro é responsável pela autenticação dos agentes e MINTING os SVIDs enquanto o segundo encarrega-se de SERVIR a *Workload API*. Nas subseções 2.2.1 e 2.2.2 são detalhadas as características dos dois componentes principais da arquitetura do SPIRE.

2.2.1 SPIRE Server

Em um determinado domínio de confiança, o SPIRE *server* age como uma CA raiz em uma PKI, ou seja, ela é responsável pelo gerenciamento e emissão de documentos de identidades (os SVIDs). O servidor encapsula informações sobre os agentes e as *workloads*, principalmente. Por meio de entradas de registro – *registration entries* – é que o servidor é

Figura 2 – SPIFFE *Runtime Environment*

avisado de quais *workloads* estão sob sua responsabilidade. Essas entradas nada mais são que regras para atribuir *SPIFFE IDs* aos nós e *workloads*.

É válido notar que não é obrigatória a atuação do servidor SPIRE como autoridade certificadora raiz. Em sistemas de larga escala, por exemplo, podem ser necessário que o *server* seja um nó não raiz. Na prova de conceito apresentada nesse documento, iremos contemplar o cenário descrito no parágrafo anterior. Sendo assim, o *server* gera um certificado autoassinado para então emitir os demais SVIDs. Finalmente, no contexto que o *SPIRE server* é a CA raiz, sua proteção a eventuais ataques é fundamental a fim de garantir a solidez de uma cadeia de confiança em uma PKI.

2.2.2 SPIRE Agent

Citando (FELDMAN et al., 2023): *O SPIRE Agent tem uma única função, obedecer uma outra muito importante – servir a Workload API.* Para tanto, engloba as seguintes funções secundárias: determinação das identidades e chamada das *workloads*; e, de maneira segura, apresentar-se ao *server*. Isso quer dizer que existe uma diferença crucial na atuação de agentes e servidor: o caráter do gerenciamento desempenhado. Ou seja, enquanto o *server*, estando ou não na raiz da PKI, precisa assinar os SVIDs de todo domínio, o agente recebe informações apenas da *workloads* que, em um dado domínio de confiança, podem chamar o SPIRE *server*. A representação das informaçoes apresentadas nas subseções 2.2.1 e 2.2.2 são mostradas na figura seguinte:

2.3 Modelo Aninhado – Nested Model

A fim de contemplar a necessidade pelo aumento da expressividade de documentos de identidade com mecanismos de validação com formatos de assinatura digital, a solução foi a contrução de um modelo de asserções e *tokens* completamente novo no contexto do SPIFFE. Esse modelo – chamado modelo aninhado – nada mais é que um *token* cuja principal característica é, à medida que ocorrem os saltos entre as diferentes *workloads*, agregam-se diferentes assinaturas de modo a, por exemplo, rastrear o caminho do *token* durante toda cadeia de atestação. Esse *nested model* é constituído de uma asserção, um conjunto de *claims* e acompanhado de uma assinatura. Toda vez que novas *claims* são trazidas, agrega-se o *payload* e a *assinatura* ao *token* original.

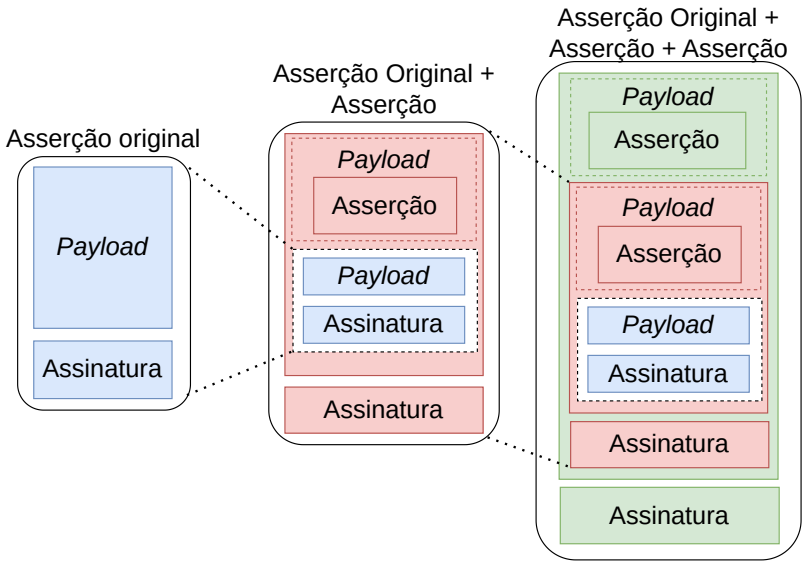

Figura 3 – Diagrama de funcionamento do modelo aninhado.

3 Metodologia

3.1 Revisão Bibliográfica

Muitas tecnologias e padrões usados para permitir autenticação e autorização federada ou centrada no usuário em aplicações *web* (*e.g.* SAML, OpenID Connect), foram projetadas para usufruir de funcionalidades de propósito geral presentes nos padrões *web*, como redirecionamento de URL, parâmetros de URL, *iframe* e *cookies*(SILVA, 2022). Segundo (GRASSI; GARCIA; FENTON, 2017), o processo de autenticação digital busca garantir que um determinado sujeito possui controle sobre um ou mais autenticadores (*e.g.* senha, chave privada etc.) que estejam associados à sua identidade digital. Um dos novos desafios na área de autenticação está no provisionamento de identidades para serviços e componentes de *software*. Assim como os usuários humanos, componentes de *software* precisam acessar sistemas como bancos de dados ou APIs diversas. Assim, já que atualmente é cada vez mais comum que componentes de *software* tenha responsabilidades específicas e mínimas (*e.g.* paradigma de microsserviços), componentes devem ter identidade únicas.

Outra tendência que reforça a necessidade de identidades específicas para componentes de *software* é o modelo de confiança zero – *zero trust* – (ROSE et al., 2020). Este modelo define que todos os recursos de um sistema devem ter identidades e que a segurança da rede não deve ser orientada a perímetros considerados confiáveis, mas sim que cada serviço deve usar autenticação e autorização forte em todas as comunicações. Atualmente, diversas alternativas estão disponíveis para a geração de identidades para componentes de *software e.g. Kubernets*, *Google BeyondProd* e SPIFFE/SPIRE. Estas identidades estão tipicamente embutidas em certificados X.509 ou *tokens* JWT. A primeira categoria é a mais popular devido à vasta gama de aplicações e ao aspecto da transparência para aplicações legadas *e.g.* colocando um componente *proxy* para encapsular conexões entre aplicações egadas ou como terminadores de conexões TLS que usam tais certificados. Os *tokens* JWT, por outro lado, estão mais sujeitos a ataques de reutilização e não são suportados de forma transparente apesar de serem populares quando incorporados já no processo de desenvolvimento da aplicação.

O SPIFFE possui identidades implementadas como URIs – *Uniform Resource Identifiers* – *e.g. spiffe://example.com/database* poderia sere associada a um servidor de banco de dados localizado no domínio administrativo *example.com*. Além do formato da identidade, outros componentes chave do SPIFFE são as definições dos formatos de ientidades verificáveis, os SVIDs – *SPIFFE Verifiable IDs* – e a especificação da API para emissão e resgate de SVIDs – a *Workload API*. Tanto o SPIFFE quanto sua implementação de referência – o SPIFFE – são projetos considerados estáveis para uso em produção. O

SPIRE fornece APIs que permitem o estabelecimento de confiança entre componentes de *software* através da atestação das propriedades destes componentes e a emissão de identidades verificáveis, os SVIDs, para os mesmos. Ele permite permite que clientes e servidores legados possam integrar um ambiente de confiança zero através de *proxies* intermediando todas as conexões(FELDMAN et al., 2023; AN. . . , 2023).

Este projeto propõe-se à lidar com um problema estrutural referente ao modo de construção da *Internet* – a ausência de uma camada de identidade uma vez que não há como saber com quem ou o que a conexão está sendo estabelecida (CAMERON, 2004; SILVA, 2022). O conhecimento exclusivo do endereço IP – o qual pode ser forjado – do dispositivo não informa nada em termos de qual entidade está acessando o serviço.

No contexto do SPIFFE, esse trabalho justifica-se pela busca pela construção de um conjunto de componentes integrável ao *framework* de modo a permitir que credenciais sejam construídas com um nível de expressividade maior que as atuais. As análises e soluções propostas devem ser suportadas pela literatura existente, particularmente em padrões bem estabelecidos e documentações de órgãos internacionais *e.g.National Institute of Standadards and Technology* – NIST.

4 Especificação de Requisitos

Para a implementação de um novo formato de identidade segundo as especificações exigidas pelo SPIFFE, é necessário o conhecimento da tecnologia *protobuf* – *Protocol Buffers*. O *protobuf* é um mecanismo para serializar dados estruturados de maneira semelhante ao formato *json* a menos de algumas diferenças. São elas, por exemplo: o tamanho e a velocidade. A ideia é definir a estrutura do dado e gerar código específico para leitura e escrita de dados a partir de diversas *streams* usando uma variedade de linguagens (PROTOCOL. . . , 2023). No contexto da pesquisa, o uso do *protobuf* vai de encontro ao aumento do nível de compactação do armazenamento de dados, maior rapidez no *parsing* de dados, versatilidade no uso de várias linguagens de programação e otimização da funcionalidade com a autogeração de classes para manipulação dos dados recebidos; além de, como será detalhado a seguir, permitir a integração com a principal linguagem de programação *Go* utilizada no desenvolvimento do projeto.

Desse modo, a criação de uma nova credencial funcionando em conjunto com as *attested claims* – documentos de atestação e comprovação típicos no *framework* do SPIFFE – mostra-se uma ideia sujeita à utilização do *protobuf* de modo a otimizar a forma e velocidade com a qual a verificação e atestação são feitas. Para simular os cenários de funcionamento da nova identidade, será feita a orquestração de vários contêineres segundo a utilização do *Kubernets* (KUBERNETS, 2023). Ele é um produto *open source* utilizado para automatizar a implantação, o dimensionamento e o gerenciamento de aplicativos em contêiner – que é o caso. Existe ainda a possibilidade de simular cenários de teste com o uso do *Docker* – tecnologia para criação e replicação de contêineres simulando todo o *stream flow* de uma identidade através do SPIFFE. A criação desses contêneres, a princípio, será feita segundo a seguinte lógica: nos dois extremos do fluxo de tráfego da identidade haverá uma *subject workload* no início e uma *target workload* no final; entre essas duas, vários *middle tiers* agindo como *obstáculos* para analisar o fluxo da informação.

Uma última especificação está relacionado ao uso de criptografia para tornar o processo de troca de dados mais seguros. Nesse sentido, será feito uso de algoritmos pós-quânticos (BOS et al., 2017; BOTROS; KANNWISCHER; SCHWABE, 2019; DUCAS et al., 2017) de criptografia. Tanto o *Kyber* como o *Dilithium* são algoritmos bem estabelecidos e implementados segundo os padrões do NIST. Além deles, o projeto tem o intuito de usar implementações já consolidadas de alguns dos principais algoritmos para efeito de estudo bem como eventuais otimizações. A linguagem escolhida para escrever as modificações será a *GoLang*(THE. . . , 2023) dado o escopo de utilização da mesma em cenários semelhantes ao do projeto. Nesse cenário, a criação de um novo padrão de identidade usando algoritmos de criptografias modernos permite unir desempenho e

segurança num contexto de gerenciamento de identidades digitais com o SPIFFE de modo a trazer recursos que permitam a diferenciação desse *framework* comparado a seus concorrentes, bem como trazer um diferencial dado o caráter *open source* da tecnologia.

5 Desenvolvimento do Trabalho

Este capítulo destina-se a apresentar as tecnologias utilizadas para o desenvolvimento do projeto, bem como aspectos da implementação que permitem destacar as vantagens de utilizar o LSVID em vez de certificados X.509. Na Seção 5.2 são discutidos aspectos arquiteturais da prova de conceito usada, componentes e funcionalidades. Ao final, são discutidos os testes e métricas de desempenho extraídas.

5.1 Tecnologias Utilizadas

5.1.1 Linguagem Go

Toda a prova de conceito é feita utilizando-se a linguagem de programação *Go*(THE. . . , 2023). As principais vantagens na utilização dessa tecnologia se dão pela possibilidade de ser *multicore* e orientada ao desenvolvimento de aplicações em rede (como é o caso da *proof of concept* em questão). Ela é, principalmente, motivada pela dificuldade de desenvolver aplicações análogas em C++ unida ao desenvolvimento mais produtivo de aplicações concorrentes com *garbage collection*.

5.1.2 Prometheus

Para a captação dos dados da aplicação, utilizou-se o *Prometheus* [SITE PROMETHEUS]. Trata-se de uma tecnologia capaz de coletar e armazenar métricas de desempenhos como séries temporais de dados. Além disso, permite a criação de bancos de dados e coleta com base em modelo executando sobre protocolo HTTP. Seu uso se dá pela criação de *probes* espalhadas pelos principais *endpoints* dos componentes da prova de conceito visando extrair as métricas de interesse. Em termos de arquitetura, considerando-se a prova de conceito desenvolvida, temos o seguinte arranjo:

5.1.3 Apache Jmeter

Corresponde a uma aplicação para teste de desempenho de recursos estáticos e dinâmicos. Pode ser usado para simular um elevado número de requisições em um servidor de modo a, juntamente com o *Prometheus*, automatizar a coleta de dados na prova de conceito [SITE APACHE]. É uma ferramenta que funciona em nível de protocolo permitindo avaliar o uso dos LSVIDs em toda a cadeia de confiança do modo ID (ver Seção 5.2.4.1)

(a) *Prometheus* – ferramenta de *profiling* e obtenção de métricas de desempenho (*e.g.* uso de CPU, tamanho em *bytes* das credenciais).

(b) *Apache Jmeter* – ferramenta para realização de várias requisições intervaladas e teste de desempenho em aplicações *web* como a *PoC*).

Figura 4 – Tecnologias utilizadas para o processo de obtenção de métricas na *PoC* com requisições automáticas e sucessivas.

5.2 Projeto e Implementação

5.2.1 Componentes

Em termos da estrutura, um LSVID possui duas partes principais. São elas:

- **Payload** p: corresponde aquilo que engloba as *claims*. Isso quer dizer que as *payloads* são vinculadas às *claims*. Por outro lado, as *claims* não precisam depender de nenhum outro ente. Conforme visto na Figura 5, o processo de *append* de um LSVID faz com que todas as *claims* pertencentes ao LSVID $\mathcal{L}_i$ agora estejam no $\mathcal{L}_{i+1},\ i \in \mathbb{N}_{\geq 0}$. Representando, por exemplo o processo de *append*, teríamos a seguinte descrição matemática:

$$\mathcal{L}_{i+1,t+\delta t} \leftarrow \mathsf{Append}\left(\mathcal{L}_{i,t},\ \mathcal{L}_{i+1,t}\right) \Rightarrow$$
$$\begin{cases} \mathsf{c}_{i+1,t+\delta t} & \leftarrow \mathsf{c}_{i+1,t} \bigcup\limits_{0 \leq j \leq i} \mathsf{c}_{j,t} \\ \mathsf{p}_{i+1,t+\delta t} & = \bigcup\limits_{0 \leq j \leq i} \mathsf{p}_{j,t} \end{cases} \quad \therefore \mathcal{L}_{i+1,t+\delta t} \equiv \left(\mathsf{p}_{i+1,t+\delta t},\ c_{i+1,t+\delta t},\ \sigma_{i+1}\right)$$

  Onde a notação $\mathcal{L}_{i,t}$ representa a estrutura do LSVID i no instante t. A representação matemática visa contemplar a atenuação durante o processo de aninhamento (2.3). Exceto as *claims* consideradas obrigatórias, todas as demais são opacas relativamente ao *framework* implicando, assim, na necessidade do devido tratamento pela camada de aplicação. O processo de adição de novos *tokens* é feito de maneira segura durante o processo de aninhamento.

- **Assinatura**[1] σ: a assinatura digital de cada *payload* é feita segundo algoritmos de criptografia comumente adotados (DSA, principalmente). O campo da assinatura inclui tanto o algoritmo utilizado quanto o valor da assinatura. De maneira semelhante a [RFC7519] para *tokens* JWT, quaisquer alterações nesse campo acarretam

[1] Comumente referenciada como *sig*, abreviatura de *signature*.

a invalidação do *token*. Na *PoC*, visando facilitar a manipulação, os *tokens* são representados por JWTs segundo a seguinte especificação:

Campo	**Formato**	**Referência [RFC4648, sec 5]**
Payload	`Base64`	$0-9,\ a-z,\ A-Z-\_.$
Assinatura	`Base64`	$0-9,\ a-z,\ A-Z-\_.$

Tabela 1 – Especificação dos campos de um *token* LSVID.

5.2.2 Funcionaliddes

O formato LSIVD – *Lightweight* SVID – baseia-se no modelo de token aninhado de modo a permitir a geração de um documento de identidade capaz de ser extendido de acordo com a necessidade da camada de aplicação. A organização de um LSVID começa com a assinatura por uma TTP de modo a garantir a relação unívoca entre um o dono de uma chave privada e um SPIFFE ID *i.e.* $\mathsf{P}_i \equiv (sk_i, \mathsf{id}_i)$. O LSVID, então, pode ser extendido embutindo-se mais informações de modo a criar uma estrutura aninhada (2.3) por meio da operação de *append* (ou ainda, representando-se por meio do algoritmo $\mathcal{L}_{i+1} \leftarrow \mathsf{Append}\,(\mathcal{L}_i, \mathcal{L}_{i+1})$ conforme a Seção 5.2.1).

O processo de extensão pode ser realizado por uma TTP ou por qualquer *workload*. No entanto, existem diferenças. No primeiro caso, tratando-se de uma TTP, é possível incluir informações específicas relacionadas à identidade do dono do *token e.g.* seletores do processo de atestação; já no segundo caso, uma *workload* pode agregar asserções de autorização fazendo com que o proprietário do *token* LSVID tenha permissão de desempenhar uma dada ação por um determinado período de tempo. A Figura 5 ilustra o processo de *append* entre os LSVIDs $\mathcal{L}_0$ e $\mathcal{L}_1$.

5.2.3 Ciclo de Vida de um LSVID

5.2.3.1 Criação

O processo de criação de LSVID começa com a criação periódica de um par de chaves pelo SPIRE *agent* e as respectivas requisições de assinatura. Ambas são enviadas ao SPIRE *server* – *trusted time party* do sistema – para atestanção do agente, verificação se o campo *audience* é equivalente ao SPIFFE Id do agente, adição de novos campos, assinatura do LSVID e retorno do documento ao requerente – o SPIRE *agent*. Com o LSVID pronto, a *workload* pode então iniciar o ciclo de requisição-resposta pelo documento de identidade verificável do sistema.

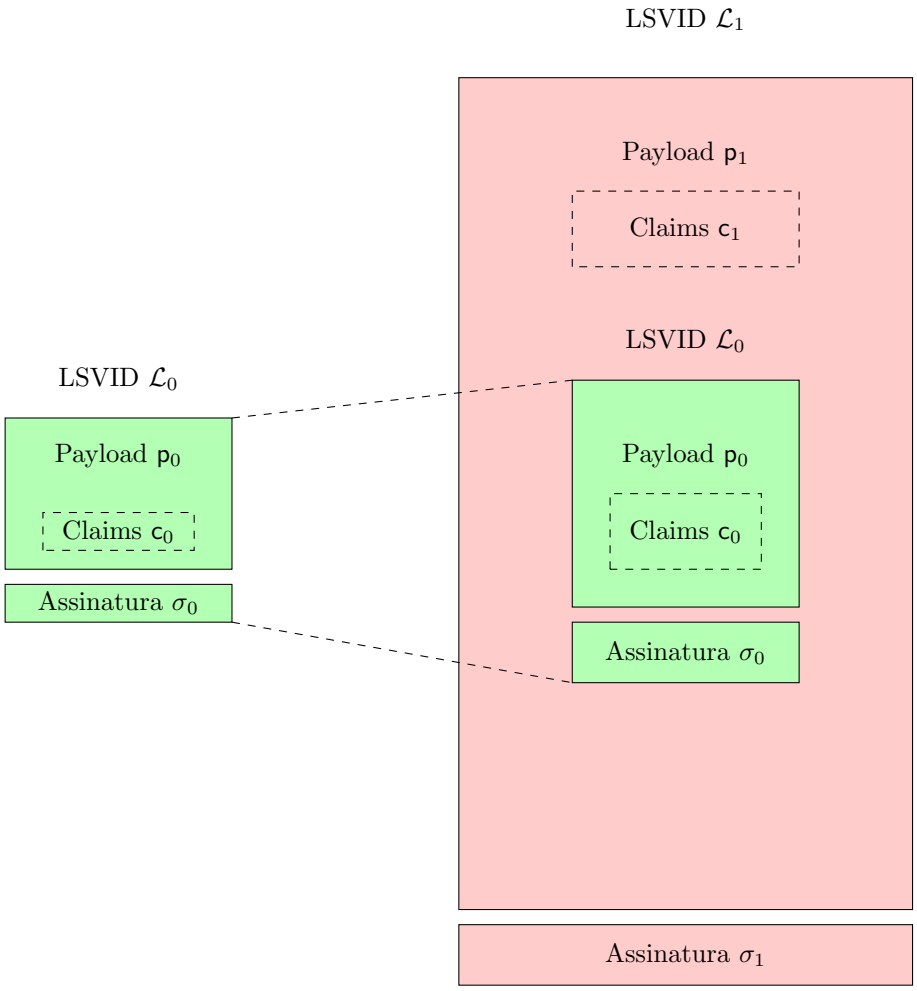

Figura 5 – Operação de *append* entre dois *tokens* LSVID.

5.2.3.2 Extensão de Identidade

Em termos de representação, valendo-se dos escritos da Seção 5.2.1, a estrutura aninhada gera a seguinte representação $\mathcal{R}_n$ de um *token* LSVID $\mathcal{L}_n$ aninhado a outros $n-1$ *tokens* $\mathcal{R}_n = \mathsf{p}_n.\mathsf{p}_{n-1}.\mathsf{p}_{n-2}.\ldots.\mathsf{p}_0.\sigma_0.\ldots.\sigma_{n-2}.\sigma_{n-1}.\sigma_n \equiv \Pi_n.\Sigma_n$. É interessante notar o nível de simetria entre a representação de um único LSVID e a presentação do *token* extendido. A representação matemática dos termos que compõe a *string* que representa o *token* é obtida por meio da aplicação dos algoritmos de concatenação $\mathsf{concat}\,(.)$ e de inversão de uma *string* $\mathsf{reverse}\,(.)$ conforme mostrado abaixo:

$$\begin{cases} \mathsf{concat}\,(s_1,\ s_2,\ldots,\ s_n) \equiv \mathsf{concat}\,(s_i)_{1\leq i\leq n} = s_1||s_2||\ldots||s_n \equiv s_1.s_2.\ldots.s_n \\ \mathsf{reverse}\,(\mathsf{concat}\,(s_1,\ s_2,\ldots,\ s_n)) \equiv s_n||s_{n-1}||\ldots||s_1 \\ \Pi_n \equiv \mathsf{p}_n.\mathsf{p}_{n-1}.\mathsf{p}_{n-2}.\ldots.\mathsf{p}_0 \leftarrow \mathsf{concat}\,(\mathsf{p}_i)_{0\leq i\leq n} \\ \Sigma_n \equiv \sigma_0.\ldots.\sigma_{n-2}.\sigma_{n-1}.\sigma_n \leftarrow \mathsf{reverse}\left(\mathsf{concat}\,(\sigma_i)_{0\leq i\leq n}\right) \end{cases}$$

5.2.3.3 Extensão de Token

Outra funcionalidade permitida é a utilização do *token* LSVID para autenticação e autorização em cenários distribuídos por meio da adição de *claims* não anteriormente previstas. Aqui destaca-se o uso dos *tokens OAuth* discutidos na Seção 2.1.2 como parte crucial no processo de extensão de *token* LSVID. O processo se inicia com o envio do LSVID e *OAuth token* com as devidas permissões; feita a validação, a *Asserting Workload* extrai as *claims* necessárias presentes no *OAuth*, cria uma nova *payload* p', assina o LSVID e p' e extende o *token* segundo o equacionamento matemático da Seção 5.2.3.2 *i.e.*

Eh condiçao necessaria ao funcionamento correto desse processo de vinculaçao de um *token OAuth* e outro LSVID seja feito por uma *workload* confiavel uma vez que o *token* LSVID resultante podera ser utilizado nos recursos associados ao *OAuth* enquanto que o tempo de validade continuara sendo definido pelo LSVID – o que visa garantir maior segurança ao sistema uma vez que a janela de tempo para um atacante agir e reduzida.

5.2.4 Modos de Funcionamento

5.2.4.1 ID

O modo de funcionamento ID precisa de um provedor de identidade seguro e disponível a fim das *workloads* conseguirem obter um documento verificável. Para cada *workload* tem conhecimento da identidade da *workload* seguinte graças ao campo *audience* (ou aud). Desse modo, sendo w_i e w_{i+1} duas *workloads*, temos a seguinte sequência de passos observada no algoritmo abaixo:

No modo de funcionamento ID, os seguintes campos são obrigatórios:

É válido observar que existem pelo menos duas possibilidades de gerenciamento do LSVID:

1. Enviar o LSVID como parte da requisição. Nesse caso, o campo *issuer* deve conter o mesmo id do campo *subject*. A figura a seguir ilustra esse comportamento.

   Vale destacar que o SPIFFE Id é a parte $\mathsf{id.spiffe}$ da representação em formato URI conforme explicado no Capítulo 2.

2. w_n colocar o LSVID como *issuer* no *payload* antes de assinar e enviar para w_{n+1}. Nesse caso, w_{n+1} conseguiria verificar todas as $n-1$ *workloads* anteriores no *token* foram previamente assinadas no caminho:

$$\mathsf{true} \overset{?}{\leftarrow} \mathsf{Verify}\left(\mathcal{L}_n,\ \mathcal{L}_n.pk_n\right) \Rightarrow \mathsf{true} \leftarrow \mathsf{Validate}\left(pk_{n+1}, \mathcal{L}_n\right)$$

O modo ID funciona como uma PKI (*e.g.* necessidade de uma CA *root* aqui representada pelo *identity provider* – ou *SPIRE Server*). Desse modo, toda a cadeia deve

ser validada (Seção 2.1.5). O processo parte da requisição da *workload* pelas chaves públicas do LSVID (*LSVID bundle*) ao agente que, por sua vez, faz a mesma requisição ao servidor. O processo de resposta se dá no sentido contrário, ou seja, até chegar a *workload*.

5.2.4.2 Outros modos: Anonymous e Double Mode

Outro modo de funcionamento possível é o *Anonymous*. Nesse caso, a presença de um IdP não é mais necessária, uma vez que as chaves públicas podem ser usadas como campo *issuer*. A seguir são utilizados métodos de concatenação de assinaturas dos novos LVIDs [GALINDO, BISCUITS] com agregação de chaves. Os campos obrigatórios nesse caso são menores em função de que, no último LSVID, existe a chave de agregação que deve ser extendida.

No *Double Mode* a *audience* não é conhecida a princípio. Esse modo consiste na aplicação conjunta tanto dos modos ID e *Anonymous* numa mesma *payload*. A assinatura σ_{DM} seria dada por

$$\sigma_{DM} = \sigma_{IDM} \cup \sigma_{AM}$$

5.3 Arquitetura

Nesta Seção são discutidos alguns pontos da arquitetura da prova de conceito. São abordados alguns pontos vantajosos que a arquitetura do SPIFFE-SPIRE (*e.g.* gerenciamento de *plugins* e de SVIDs) e como são organizados os componentes do sistema. Esquematicamente, tem-se o arquitetura descrita na figura 6.

5.3.1 Gerenciamento de Plugins e SVIDs

Cada componente do SPIRE faz uso de *plugins* para atestação de nós. O próprio *Node Attestor Plugin* possui dois lados: *server* e *agent*. O agente, por outro lado, deve possuir um *plugin* atestador de *workload* que comunica-se com uma *workload* (que, por sua vez, deve comunicar-se à *workload* API).

Durante o processo de atestação de um nó o agente deve autenticar-se quando em contato com o *SPIRE Server* e obter documentos verificáveis para as *workloads* que são responsabilidade do agente. De acordo com a necessidade das *workloads*, os SVIDs podem ser renovados (lembrando que existe um campo relacionado ao tempo para um SVID expirar).

5.3.2 mTLS - Mutual TLS

Trata-se de um método de autenticação mútua capaz de garantir que ambos os entes de uma comunicação são quem eles afirmam ser por meio da verificação que cada

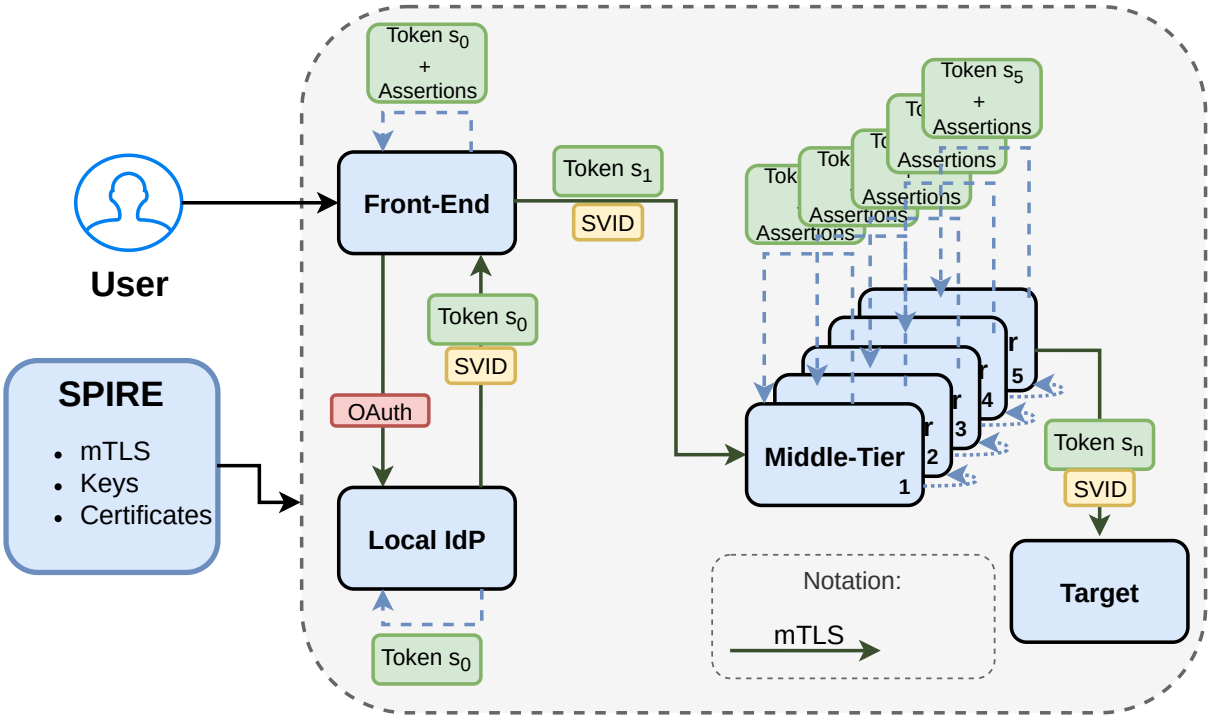

Figura 6 – Arquitetura do modo de funcionamento ID para a prova de conceito discutida nesse documento

um tem a respectiva chave privada *sk*. A informação presente no certificado TLS fornece informações extras para verificação. O mTLS é muito usado em sistemas de confiança zero [ZTA-NIST] e é importante na construção da prova de conceito como forma de fornecer maior segurança ao sistema bem como demonstrar a aplicabilidade prática da solução. Em contraposição, o TLS – *Transport Layer Security* – é um protocolo de cifragem usado para autenticar o servidor em uma conexão cliente-servidor e cifra as comunicações entre cliente e servidor para que atacantes externos não consigam decifrar as mensagens. O arquivo contendo informações relacionadas à verificação do servidor é o certificado TLS usado também no mTLS. No entanto, o cliente também possui um certificado mTLS (daí o nome *mutual*). O processo com mTLS envolve os seguintes passos:

1. Cliente conecta-se ao servidor;
2. Servidor apresente o certificado TLS;
3. Cliente verifica o certificado do servidor e atesta se ele é quem diz ser;
4. Cliente apresenta seu certificado TLS ao servidor;
5. Servidor verifica o certificado do cliente e atesta se ele é quem diz ser;
6. Em caso de sucesso, servidor fornece acesso;
7. Cliente e servidor trocam informações por meio de conexão TLS.

O uso do mTLS é recomendado na prevenção de uma série de ataques (*e.g. credential stuffing*, ataques de força bruta, *phishing* e *spoofing*). Especialmente no caso de credenciais vazadas, num cenário de computação em nuvem com diferentes serviços sendo utilizados, o comprometimento do sistema pode ser catastrófico em caso de sucesso por parte do atacante. No cenário da *PoC*, o uso do mTLS pode ser exemplificado pela figura **??**.

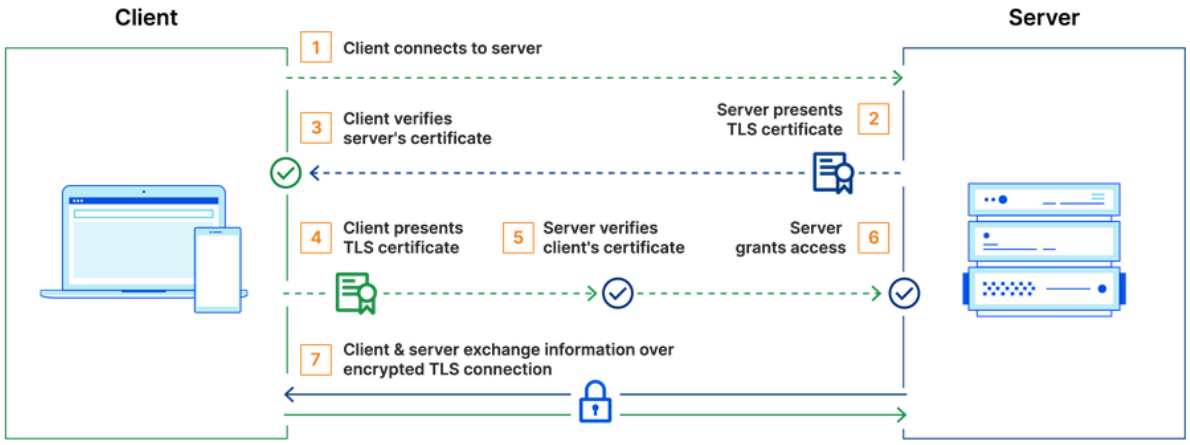

Figura 7 – Passo-a-passo de autenticação mútua com mTLS. Imagem retirada do site da Cloudfare.

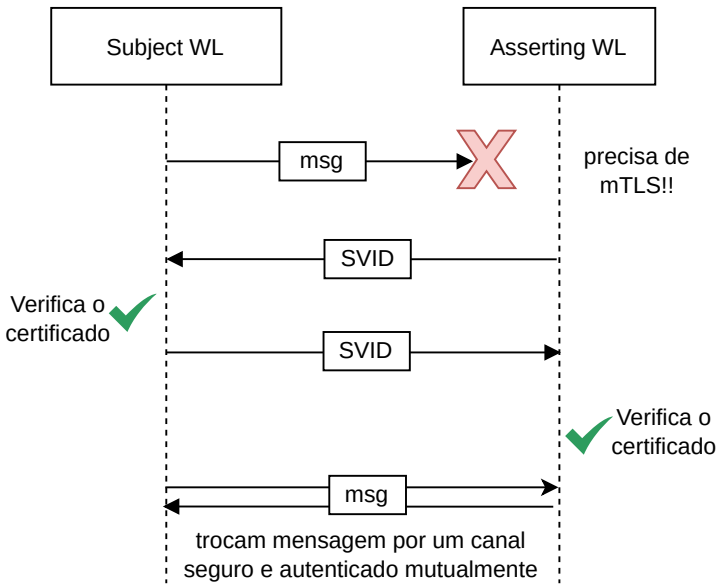

Figura 8 – Exemplo do uso do mTLS dentro da *Proof of Concept* ilustrando-se a mútua autenticação entre *asserting workload* e *subject workload*.

5.4 Aplicação

Ainda que, conforme discutido na Seção 5.2, os certificados X.509 possuam vantagens, a prova de conceito e posterior análise do LSVID é feita baseando-se em JWTs.

6 Considerações Finais

6.1 Conclusões

A proposta de melhoria proposta e descrita no presente documento juntamente com a relação com os conceitos próprios do SPIFFE-SPIRE se mostraram satisfatórias e como uma alternativa possível e viável de ser utilizada no *framework* em questão. Unido a esses fatos o contato com ferramentas de extração de métricas (*e.g.Prometheus* e *Apache Jmeter*) e esquemas de agregação de identidades contribuíram para uma discussão mais aprofundada dos impactos atrelados a um novo documento verificável de identidade.

Além disso, a arquitetura SPIRE se mostrou eficiente durante todo o processo analisado no presente trabalho. Não houveram problemas (a menos da limitação de recursos disponíveis) para a utilização do SPIRE. Toda a estrutura da prova de conceito foi implementada sem qualquer prejuízo e a linguagem *go* se mostrou extremamente eficiente para os objetivos da *PoC*.

A capacidade da aplicação, mesmo como uma *Proof of Concept* (PoC), em processar as requisições, efetuar o aninhamento e validação pode ser considerada satisfatória.

6.2 Contribuições

O estudo mais aprofundado da insfraestrutura de chave pública e de tópicos mais aprofundados de criptografia foram fundamentais ao melhor entendimento do SPIFFE. Além disso, do ponto de vista de tecnologia, o SPIFFE-SPIRE trata-se de um projeto *open-source* e, portanto, orientado às necessidades de uma comunidade. Nesse sentido, o contato com esse *modus operandi* de constante contato com as pessoas que estão envolvidas na adição de *features* foi algo extremamente enriquecedor.

Em termos de monitoramento o estudo de ferramentas focados em soluções baseadas em nuvem, além da análise de formas de implementar uma maneira automatizada de introduzir estresse na aplicação foi parte importante para o desenvolvimento desta análise. Os objetivos propostos para este trabalho, como um todo, foram atingidos, que compõe não somente o comportamento do mecanismo de rastreamento sob o ponto de vista de desempenho e consumo de recursos computacionais, mas também uma caracterização sobre como o esquema proposto traz benefícios ao *framework* e tem implementação viável.

6.3 Perspectivas de Continuidade

Para trabalhos futuros, existem lacunas para o emprego de diferentes va- lidações e assinaturas criptográficas na extensão do *token* aninhado utilizando-se criptografia de curva elíptica segundo o modelo de (GALINDO; GARCIA, 2009). O aumento da escalabilidade é um ponto importante para a evolução da prova de conceito, com o intuito de poder trabalhar com diferentes requisições HTTPS simultâneas trazendo maior fidelidade com uma aplicação real também é outro ponto a ser pensado (*e.g.* uso do padrão grpc). No projeto, atualmente, são objetos de discussão a adoção de um algoritmo assinatura digital de criptografia pós-quântica, conhecido como *Crystals-Dilithium* (BOS et al., 2017). Finalmente, o conjunto das declarações (claims) a serem usadas no *token* em seu processo de emissão é um objeto de estudo para projetos futuros bem como a utilização de orquestradores mais robustos (*e.g.*(KUBERNETS, 2023)) e a mudança para outro serviço no lugar da OKTA.

Referências

AN overview of the SPIFFE specification. 2023. <https://spiffe.io/docs/latest/spiffe-about/overview/>. Citado 2 vezes nas páginas 21 e 34.

BOS, J. W. et al. Crystals - kyber: a cca-secure module-lattice-based kem. *IACR Cryptology ePrint Archive*, v. 2017, p. 634, 2017. Disponível em: <http://dblp.uni-trier.de/db/journals/iacr/iacr2017.html#BosDKLLSSS17>. Citado 2 vezes nas páginas 35 e 46.

BOTROS, L.; KANNWISCHER, M. J.; SCHWABE, P. Memory-efficient high-speed implementation of kyber on cortex-m4. *IACR Cryptology ePrint Archive*, v. 2019, p. 489, 2019. Disponível em: <http://dblp.uni-trier.de/db/journals/iacr/iacr2019.html#BotrosKS19>. Citado na página 35.

CAMERON, K. The laws of identity. 2004. Disponível em: <http://www.identityblog.com/stories/2004/12/09/thelaws.html>. Citado na página 34.

COOK, S. *Phishing statistics and facts for 2019–2023*. 2023. <https://www.comparitech.com/blog/vpn-privacy/phishing-statistics-facts/>. Citado na página 19.

DUCAS, L. et al. Crystals - dilithium: Digital signatures from module lattices. *IACR Cryptology ePrint Archive*, v. 2017, p. 633, 2017. Disponível em: <http://dblp.uni-trier.de/db/journals/iacr/iacr2017.html#DucasLLSSS17>. Citado na página 35.

FELDMAN, D. et al. *Solving the Bottom Turtle-a SPIFFE Way to Establish Trust in Your Infrastructure via Universal Identity*. [s.n.], 2023. ISBN 9780578777375. Disponível em: <www.booksprints.net>. Citado 6 vezes nas páginas 7, 21, 25, 30, 31 e 34.

GALINDO, D.; GARCIA, F. D. A schnorr-like lightweight identity-based signature scheme. In: . [S.l.: s.n.], 2009. v. 5580 LNCS, p. 135–148. ISBN 3642023835. ISSN 03029743. Citado na página 46.

GRASSI, P.; GARCIA, M.; FENTON, J. *Digital Identity Guidelines*. [S.l.]: Special Publication (NIST SP), National Institute of Standards and Technology, Gaithersburg, MD, 2017. Citado 2 vezes nas páginas 19 e 33.

KUBERNETS. 2023. <https://kubernetes.io/pt-br/>. Citado 2 vezes nas páginas 35 e 46.

MALER, E.; REED, D. The venn of identity: Options and issues in federated identity management. *IEEE Security and Privacy*, IEEE Educational Activities Department, USA, v. 6, n. 2, p. 16–23, mar 2008. ISSN 1540-7993. Disponível em: <https://doi.org/10.1109/MSP.2008.50>. Citado na página 20.

PROTOCOL Buffers Documentation. 2023. <https://protobuf.dev/overview/>. Citado na página 35.

ROSE, S. et al. *Zero Trust Architecture*. Special Publication (NIST SP), National Institute of Standards and Technology, Gaithersburg, MD, 2020. Disponível em: <https://tsapps.nist.gov/publication/get_pdf.cfm?pub_id=930420>. Citado na página 33.

SILVA, C. E. da. Autenticação e autorização: antigas demandas, novos desafios e tecnologias emergentes. 2022. Citado 3 vezes nas páginas 19, 33 e 34.

THE Go Programming Language. 2023. <https://go.dev/>. Citado 2 vezes nas páginas 35 e 37.