ESTUDO COMPARATIVO DE TOKENS DE AUTORIZAÇÃO EM APIS REST

André Luiz Lourenço de Andrade[1], Robson Oliveira de Albuquerque[1], Fabio L. L. de Mendonça[1], Daniel Alves da Silva[1], Carlos Eduardo Lacerda Veiga[2], Rafael Timóteo de Sousa Júnior[1]

[1]Pós-graduação Profissional em Engenharia Elétrica – PPEE – Departamento de Engenharia Elétrica, Faculdade de Tecnologia, Universidade de Brasília (UnB), Brasília, Brasil, Zip Code 70910-900

[2]Advocacia-Geral da União - SIG-Quadra 06-Lote 800, Brasília, Brasil, Zip Code 70610-460

RESUMO

Após o processo de autenticação, é necessário que os sistemas atribuam permissões apropriadas às entidades autenticadas que solicitam por respostas. Em um ambiente diversificado com diferentes níveis de permissão, as aplicações HTTP devem determinar o nível de autorização para cada solicitação que é requisitada para si. Este estudo apresenta uma análise comparativa de três tokens de autorização usados em conjunto com APIs REST: JSON Web Tokens (padrão amplamente utilizado na indústria); PASETO (oferece uma versão enxuta e simplificada do JWT); e macaroon (método padrão para delegar autorização em ambientes descentralizados proposto por desenvolvedores da Google). Tokens de autorização são projetados para restringir ações e aumentar a segurança em sistemas distribuídos. No entanto, dependendo da implementação, cifras criptográficas ou complexidades desnecessárias, vulnerabilidades significativas podem ser introduzidas com o seu mal uso. Este artigo fornece uma abordagem para comparar tecnologias de autorização e destaca seus principais benefícios e desvantagens com o objetivo de ajudar administradores de sistemas e desenvolvedores a selecionar o método de autorização mais apropriado para cenários específicos.

KEYWORDS

JSON Web Token, PASETO, Macaroon, tokens, autorização, REST API.

1. INTRODUÇÃO

Embora as medidas de segurança baseadas em perímetro possam fornecer alguma proteção contra ameaças cibernéticas, elas são insuficientes por si só, pois dependem da suposição de que todas as ameaças virão de fora da rede em administração (ROSE, 2020). Para conter as ameaças de forma adequada, as organizações devem se adapar e favorecer abordagens de defesa baseadas em camadas, que inclui uma variedade de medidas para proteger contra ameaças que se originam tanto de dentro quanto de fora da rede.

Com a modernização de aplicativos, dar atenção aos processos de autenticação e autorização tornou-se indispensável no design de serviços seguros que exigem a identificação adequada de seus usuários (Wilson & Hingnikar, 2019). A evolução dessas tecnologias ocorre em conjunto com a robustez dos ataques que visam contornar o processo de identificação e obter privilégios indevidos.

Por este motivo, a obtenção de controle de identidade e nível de permissão de cada requisição que consuma por APIs REST ganhou relevância substancial, tornando-se necessário a transmissão de tokens de autorização em cada requisição HTTP realizada pelos clientes (Wijayarathna, 2019).

Este trabalho além de expor um contexto de implementação de segurança em arquiteturas de aplicações modernas, sobretudo REST APIs, tem como contribuição o estudo de tokens estruturados JWT, PASETO e Macaroon, estabelecendo critérios de comparação que sejam aplicáveis no processo de escolha dos métodos de autorização que decorrem no desenvolvimento de aplicações.

O artigo está dividido em 6 partes que incluem a Introdução, que realiza uma breve contextualização e explicita a contribuição do trabalho, os Conceitos Básicos que explica o conceito de APIs REST e introduz os tokens de autorização estudados no trabalho, realizando uma comparação geral entre eles, a Contextualização do Problema que evidencia a relevância do estudo mais aprofundado de tokens de autorização, a Metodologia de Implementação, que coloca em evidência os passos realizados para a obtenção dos Resultados que, por sua

vez, define os critérios de comparação entre os tokens e evidencia os melhores tokens a serem utilizados de acordo com cada critério. Ao final do artigo encontra-se a Conclusão que realiza um resumo do artigo com as conclusões obtidas dos resultados e explorando possibilidades de extensão de estudos que seguem uma linha similar.

2. CONCEITOS BÁSICOS

2.1 APIs REST

APIs REST é um conjunto de regras e convenções que estabelecem como as comunicações entre sistemas devem ser feitas, utilizando o protocolo HTTP para definir a estrutura de suas requisições e respostas (Li L, 2011). Uma das características mais importantes das APIs REST é que elas são *stateless* (Lee, 2017), isto é, não mantêm nenhum estado da comunicação entre as requisições, fazendo com que cada requisição seja tratada de forma independente e que suas respostas não dependam de nenhuma informação anterior. Essa característica torna as APIs REST mais escaláveis e fáceis de implementar e manter.

Tokens de autorização são utilizados pelas APIs para permitir ou não o acesso a determinados recursos. Eles funcionam como uma forma de autenticação e autorização em que o usuário ou sistema que faz a requisição precisa fornecer um token válido para acessar os recursos desejados. Os tokens de autorização comumente são gerados após um processo de autenticação bem-sucedido e possuem informações sobre o usuário ou sistema que fez a requisição. Quando uma API recebe uma requisição com um token de autorização, ela verifica se o token é válido e se o usuário ou sistema tem permissão para acessar o recurso solicitado. Se o token for legítimo e as permissões válidas, a API retorna a resposta com os dados solicitados.

2.2 JSON Web Token (JWT)

JSON Web Tokens representam um conjunto de declarações em um objeto JSON que é apresentado em uma estrutura JWS (JSON Web Signature) ou JWE (JSON Web Encryption). O JSON é o Conjunto de Declarações do JWT que consiste em pares de nome e valor em que os nomes são *strings* e os valores são valores JSON arbitrários (Jones, 2015). Para a transmissão do JWT, o objeto JSON é codificado em Base64 que contém uma carga útil de informações, como dados de usuário, bem como uma assinatura digital para garantir que as informações não tenham sido alteradas durante a transmissão (Shulin, 2020). O JWT é amplamente utilizado na autenticação e autorização de usuários em aplicativos web e serviços de API, pois é portátil, escalável e, se implementado corretamente, seguro. Além disso, o JWT pode ser facilmente integrado com outras tecnologias de segurança, como HTTPS, OAuth2 e OIDC.

O JWT vem em duas formas: simétrica e assimétrica. No JWT simétrico, o hash da concatenação do cabeçalho e *payload* é assinado usando uma chave secreta, que pode ser gerada usando algoritmos SHA-256, SHA-384 e SHA-512. A mesma chave é usada para assinar e validar o token. Por outro lado, o JWT assimétrico usa um par de chaves pública/privada. A chave privada assina o token, enquanto a chave pública realiza a validação. RSA, ECDSA e RSASSA-PSS podem ser usados para gerar o par de chaves. A chave privada assina digitalmente a mensagem, enquanto a chave pública verifica sua integridade.

2.3 PASETO

O Platform Agnostic Security Token (PASETO) é um padrão de token de autorização que compartilha algumas similaridades com o JWT, mas impõe um conjunto rigoroso de diretrizes na seleção de algoritmos criptográficos disponíveis para os desenvolvedores (Paragon Initiative, 2023). Essa característica é considerada vantajosa porque o JWT fornece uma abundância de opções, algumas das quais são inseguras. O PASETO é composto por três elementos obrigatórios (*version*, *purpose* e *payload*) e um elemento opcional (*footer*). Assim como o JWT, os tokens do PASETO são codificados em base 64 antes de sua transmissão (Terjesen & Haussmann, 2022).

O elemento de *purpose* do PASETO pode ser configurado como local ou público. A opção local é recomendada em cenários em que uma chave secreta compartilhada pode ser armazenada com segurança entre os pontos finais que irão validar o token. O token é criado e assinado com a mesma chave secreta compartilhada entre os sistemas e é sempre transmitido em forma criptografada. Por outro lado, quando o *purpose* do PASETO é público, pares de chaves são gerados com a *API libsodium* e transferidos em texto simples, mas sempre assinados. Neste caso, é recomendado que a aplicação evite enviar informações sensíveis do usuário por meio dos tokens.

2.4 Macaroons

Os macaroons são credenciais de autorização que fornecem suporte flexível para compartilhamento controlado em sistemas descentralizados e distribuídos (Birgisson, 2014). Os Macaroons têm uma característica chave de delegação, que permite que o proprietário de um macaroon gere um sub-macaroon com autoridade reduzida. Um macaroon é composto por quatro componentes: *location, public ID, caveats* e assinatura. O componente de *location* é uma cadeia de caracteres que especifica a origem do macaroon e é normalmente representado por um Localizador de Recursos Uniforme (URL). O *public ID* é usado para identificar o macaroon e está vinculado à sua chave privada. Os *caveats* podem ser vistos como a carga útil do macaroon e declaram um predicado que deve ser verdadeiro para qualquer solicitação que o macaroon derivado autorize no serviço pretendido. Elas são usadas para restringir as capacidades do macaroon, indicando quando, onde, por quem e para que finalidade o serviço alvo deve conceder acesso. O componente final é a assinatura, que é calculada usando o HMAC com a chave privada salva no servidor. Ao contrário do JWT e PASETO, os Macaroons não têm uma codificação definida para a transmissão, mas seu conteúdo é seguro para ser comunicado por meio de cabeçalhos HTTP, cookies ou URLs (Birgisson, 2014).

Também é possível utilizar macarrons com *third-party caveats* que são úteis em situações em que se tem serviços externos que podem realizar verificações adicionais de autorização relevantes para o token de acesso, restringindo ainda mais o token inserindo validações (ou *statements*) adicionais.

2.5 Comparação Geral dos Tokens de Autorização

Para a criação da Tabela 1, foram destacados os recursos de segurança, tipo de implementação, algoritmo de *hash*, padrão de transferência e partes envolvidas. Para o critério de segurança, levou-se em conta se o token de autorização possui falhas em sua composição, sendo possível usá-lo incorretamente ou de forma ineficiente, para o JWT este quesito foi considerado médio por ser possível utilizar em sua implementação algoritmos de criptografia considerados ultrapassados como RSA com PKCS #1v1.5 *padding*, RSA com OAEP *padding*, ECDH com curvas de *Weierstrass* e AES-GCM (Degges, 2019). Para o critério de complexidade de configuração, foram consideradas as opções possíveis de implementação do token. Quanto mais opções, seja de modo ou algoritmos criptográficos possíveis, mais complexa foi considerada a configuração.

Tabela 1 - Comparação Geral dos Tokens de Autorização

	JSON Web Token	**PASETO**	**Macaroon**
Segurança	Médio	Alto	Alto
Tipo de Implementação	simétrico, assimétrico	local, public	regular e third-party caveats
Algoritmo de Hash	Simétrico:HS256/384/512 Asimétrico:RS256/384/512 ES256/384/512; PS256/384/512	local:xChaCha20+Blake2B public:EdDSA Overcurve25519	N/A
Padrão de Transferência	base64	base64	N/A
Partes	Header, Payload, Signature	Version, Purpose, Payload, Footer	Identifier, Caveats, Signature

3. CONTEXTUALIZAÇÃO DO PROBLEMA

Os tokens de autorização desempenham um papel crucial na segurança de aplicações e serviços modernos, permitindo apenas que usuários ou entidades autorizadas acessem recursos protegidos. No entanto, a falta de padronização na implementação desses formatos de token pode tornar desafiador para os desenvolvedores escolherem o tipo de token mais adequado para suas aplicações. Cada token abordado neste trabalho possui características únicas e compensações que os tornam adequados para casos de uso específicos. Levando em consideração a lista da OWASP publicada em 2021 (OWASP, 2021) que coloca como design inseguro e erros de configuração com um dos maiores riscos no desenvolvimento de aplicações, este estudo torna-se relevante tendo em vista que não há uma solução única de implementação de tokens de autorização - apesar de algumas tentativas de unificação de gerenciamento de identidade em microsserviços (Cao, 2018) - inferindo-se que a equipe disposta para o desenvolvimento de APIs REST deva avaliar cuidadosamente seus requisitos e as compensações de cada formato de token antes de escolher o mais apropriado para uso.

A indústria percebeu esse problema e lançou frameworks que simplificam o processo de autorização para aplicações, como (Permit.io, 2022) e (SuperTokens, 2021). No entanto, a dependência de ferramentas de terceiros pode se tornar um fator prejudicial em si. As questões de preços embutidos nos produtos e vulnerabilidades de bibliotecas das quais o desenvolvedor não tem conhecimento podem ser consideradas uma desvantagem no uso dessas soluções.

4. METODOLOGIA DE IMPLEMENTAÇÃO

Para estre trabalho, foi desenvolvido um ambiente que implementa a autorização usando as três abordagens de token em estudo: JWTs, PASETO e Macaroon. As especificações do sistema em que o teste foi realizado se encontram na Tabela 2.

Tabela 2. Tabela de propriedade do sistema de testes

Propriedades do Sistema	
Processador	Intel(R) Core (TM) i7-6700T, 2.80GHz 2.81 GHz
RAM	16GB
Sistema Operacional	Windows 11 Enterprise v21H2
Linguagem	Python 3.10.6
Bibliotecas	hmac v20101005, hashlib v20081119, json v2.0.9, pyseto v1.7.0, pymacaroons v0.13.0, datetime v4.9

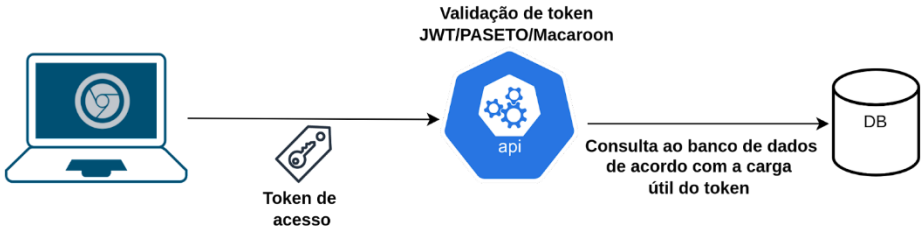

Figura 1 - Fluxo de validação de tokens implementado no experimento

Os métodos de autorização foram simulados por este trabalho implementam um sistema de registro de estudantes universitários que, após a devida autenticação, verifica em cada token estruturado as informações básicas como o nome do estudante, nome da universidade, número de matrícula e data de validade do vínculo do estudante para fornecer conteúdo ao usuário. É importante ressaltar que os atributos escolhidos podem ser configurados para características específicas de novos sistemas e que a estrutura dos códigos implementados pode ser editada para novas implementações.

O experimento realizado consiste na validação de tokens de autorização estruturados que carregam consigo as informações de cadastro de estudantes que foram supracitadas, portanto, torna-se responsabilidade da API realizar as validações de integridade do token e retornar uma resposta válida caso o token seja devidamente validado e identificado conforme apresentado na Figura 1.

4.1 Cenário 1 – Descrição da Implementação de Autorização com JWT

Para o cenário implementando autorização com JSON Web Token com a linguagem *Python*, foram utilizadas as bibliotecas *hmac, hashlib, base64* e *JSON*. Primeiro, definiu-se a *string "this is our secret"* como a chave secreta que será usada para criar e validar o JSON Web Token (simétrico). O cabeçalho foi codificado com seus metadados, indicando que será usado SHA-256 como algoritmo de criptografia para criação da assinatura. O payload foi preenchido com quatro *claims* de universidade, estudante, número de matrícula e expiração do token. A assinatura do token foi criada concatenando o cabeçalho e o *payload* dentro da função HMAC com a chave privada estabelecida. Para realizar de validação do token JWT, é necessário checar a integridade do token através da sua assinatura, assim como as *claims* contidas em seu *payload*. Para isso, a API de validação decodifica a *string* recebida em base64 e realiza o *parsing* das partes a serem validadas contidas no objeto JSON.

4.2 Cenário 2 – Descrição da Implementação de Autorização com PASETO

Para este cenário, foi utilizada a biblioteca *pyseto*, onde foi realizada a implementação do PASETO na sua quarta versão. Foi utilizado o método *new()* para criar a chave secreta representada pela string *"this is our secret"*, e o método *encode()* para codificar o token em base64 respeitando o formato do PASETO. Para a implementação do PASETO, foram utilizadas as *claims* de universidade, estudante, número da conta e expiração do token, representadas em formato JSON como um dos argumentos do *encode()*. Para a validação do token PASETO é necessário decodificar a *string* em base64 utilizando o método da biblioteca *pyseto decode()* que, além de decodificar realiza a validação de integridade do token, tendo em vista que o método recebe como argumentos a chave da criação do token e o próprio token. Após a decodificação a aplicação se torna responsável pela validação das *claims.* Os detalhes da implementação do token PASETO utilizando a biblioteca *pyseto* pode ser verificado em detalhes na documentação da biblioteca (PYSETO, 2023).

4.3 Cenário 3 – Descrição da Implementação de Autorização com Macaroons

Para este cenário foi utilizado a biblioteca *pymacaroons*. Inicialmente, para criar o token macaroon, é necessário configurar a chave privada, representada pela variável *secret*, o recurso em que o macaroon será utilizado, representado pela variável *location*, e o identificador de chave, representado pela variável *identifier*.

Foi utilizado o método *add_first_party_caveat()* para criar as restrições (*caveats*) do macaroon, que é semelhante às *claims* utilizadas no JWT e PASETO. Como nos exemplos anteriores, os parâmetros de universidade, estudante, número de matrícula e expiração do token foram usados como limitadores de autorização. Para cada restrição adicionada ao macaroon, a assinatura de verificação do token é alterada. A validação do macaroon é realizada pela aplicação usando o método *verify()*, que requer como argumento o macaroon a ser validado e a chave privada estabelecida.

5. RESULTADOS

Para a correta arquitetura de sistemas e seus métodos utilizados para autorização, é essencial que o planejamento do projeto estabeleça critérios definidos que busquem orientar a equipe de desenvolvimento sobre qual alternativa seguir quando se trata da concessão de permissão por meio de chamadas de API. Nas seções seguintes encontra-se expostos os critérios de comparação estabelecidos e seus respectivos resultados, ressaltando os tokens mais adequados levando em consideração cada contexto em destaque.

5.1 Requisito de Criptografia de Tokens

Em algumas aplicações, tokens estruturados podem conter informações consideradas confidenciais pelos usuários que os utilizam. No caso de implantações que exigem transmissão de tokens criptografados, é recomendado a utilização do PASETO com *purpose* configurado como “local”, que possui criptografia por padrão. Apesar da possibilidade de usar a criptografia em JWTs utilizando JWE (JSON Web Encryption)

(Jones, 2015), a utilização de JWE não é comumente usada por adicionar mais camadas de complexidade à sua implementação, além disso, o JWE conta com opções de criptografia vulneráveis (DEGGES, 2019). Os Macaroons podem ser criptografados usando uma camada de criptografia personalizada, mas assim como o JWT, por não ter criptografia por padrão, adiciona mais complexidade ao código e à implementação, gerando mais riscos ao seu processo de desenvolvimento. É importante destacar que o uso do TLS é essencial para garantir a confidencialidade dos tokens, considerando que eles são transmitidos na camada de aplicação (HTTP).

5.2 Requisito de Confiança Entre o Emissor do Token e o Destinatário

Mesmo que "confiança" seja um conceito considerado inadequado em arquiteturas baseadas em *Zero Trust* (ROSE, 2020), este critério se torna relevante para escolha do token de autorização, levando em consideração a necessidade de uma arquitetura depender de uma infraestrutura de verificação de token emitidos por terceiros. Portanto, para estes casos, não é recomendado o compartilhamento de chaves secretas, evitando o risco de terceiros emitirem intencionalmente tokens de autorização com permissões elevadas. Nestes casos, é recomendado o uso de uma arquitetura baseada em chave pública. Portanto, são recomendadas as arquiteturas de JSON Web Token assimétrico, PASETO com *purpose* público e macaroons configurados com *third-party caveats*.

5.3 Requisito de Número de Endpoints que Realizam a Validação de Token

Em um ambiente extenso de microsserviços, pode haver muitas APIs que precisam compartilhar a mesma chave secreta para validação de tokens, tornando o gerenciamento de chaves secretas mais desafiador especialmente quando estes serviços são adicionados ou removidos do sistema dinamicamente, o que pode prejudicar a escalabilidade de novos projetos (XIUYU, 2017). Para esses casos, também são recomendados algoritmos que têm um princípio de chave pública, como JWT assimétrico, PASETO com *purpose* público e macaroons com *third-party caveats* (ressalvas).

5.4 Requisito de Verificação de Tokens por Terceiros

Em casos em que aplicações necessitem de muitas verificações de serviços de terceiros, o método mais recomendado para implementação de tokens de autorização é o uso de macaroons com *third-party caveats*, devido à possibilidade de serviços de terceiros adicionarem ressalvas específicas ao token, tornando-o mais restrito para uso com cada ressalva adicionada.

5.5 Requisito de Reutilização de Tokens

Embora os tokens preferencialmente devam ser utilizados com um período de validade curto, algumas arquiteturas de aplicações exigem a reutilização de tokens de forma extensiva. Nestes casos, é necessário antecipar situações em que os tokens podem ser comprometidos e escolher métodos que suportem a invalidação de tokens comprometidos de maneira oportuna. Os Tokens PASETO não possuem um mecanismo embutido para a prevenção de *replay-attacks*, o que infere que o projeto da arquitetura deve respeitar a premissa de usar o token PASETO somente uma vez. Para o caso de JSON Web Tokens, é utilizado o conceito de tokens de acesso e tokens de atualização (*refresh*), que possuem mecanismos de defesa em caso de identificação de um token comprometido, limitando um atacante a usar o token comprometido enquanto o token de acesso não estiver expirado. Quanto mais curto o tempo de expiração do token, menos risco é exposta a aplicação. No caso de macaroons, *replay-attacks* podem ser contidos devido à necessidade do destinatário validar todos os macaroons descarregados (*discharge macaroon*) durante a adição de ressalvas (*caveats*), o que significa que todos os macaroons descarregados são necessários para serem vinculados ao macaroon autorizador antes de serem enviados juntamente com uma solicitação para o destino.

5.6 Requisito de Nível de Escassez da Equipe de Desenvolvimento

Mesmo que se espere que equipes especializadas em segurança desenvolvam sistemas e aplicativos que requerem autenticação, essa expectativa não corresponde à realidade de muitas empresas e *startups*. Portanto, é essencial saber como medir o nível de habilidade dos desenvolvedores de uma equipe para escolher o método de autorização a ser adotado em um projeto. O uso do token de autorização PASETO em sua versão local ou pública é o mais recomendado para equipes sobrecarregadas ou sem expertise técnica em segurança devido à sua baixa complexidade nas escolhas do algoritmo de criptografia e implementação.

5.7 Quadro Resumo dos Resultados

Na Tabela 3 encontra-se evidenciado o quadro resumo dos tokens de autorização mais adequados de acordo com os requisitos levados em consideração. É indicado que, antes da escolha do token de autorização a ser utilizado, a equipe de desenvolvimento da aplicação em questão leve em consideração um ou mais requisitos e os ordene de acordo com a sua relevância para o projeto que está sendo concebido, realizando a escolha do token, considerando a ordem e o número de vezes que ele surge com os requisitos escolhidos.

Tabela 3. Quadro Resumo dos Resultados

Requisito	Tokens Recomendados
Criptografia de tokens	PASETO Local
Confiança emissor/destinatario	JWT ass, PASETO pub, Macaroons c/ 3rd-caveats
Nº de endpoints p/ validação	JWT ass, PASETO pub, Macaroons c/ 3rd-caveats
Validação de terceiros	Macaroons c/ 3rd-caveats
Reutilização de tokens	JWTs e Macaroons
Escassez de desenv.	PASETO

6. CONCLUSÃO

Com base nos estudos e experimentos realizados utilizando os três métodos para tokens autorização, foi possível identificar seus principais prós e contras, complexidade de implementação e particularidades em relação à criação e validação de seus tokens. A Tabela 1 mostra as propriedades de cada método, como os algoritmos criptográficos usados, complexidade, padrão de transferência e estrutura. É possível observar a partir da tabela de comparação que, dos métodos discutidos, o JSON Web Token é o mais flexível e complexo em sua implementação, sendo possível implementar 12 variações criptográficas divididas em metodologias de chave simétrica e assimétrica. Quanto à complexidade de implementação, o PASETO mostrou-se a metodologia mais simples para a criação e validação de tokens, removendo a responsabilidade do desenvolvedor de escolher algoritmos de criptografia considerados inseguros. Devido à natureza do experimento realizado, não foi possível mostrar o método que apresente maior performance em termos de tempo de validação de token; portanto, é recomendado para trabalhos futuros implementar arquiteturas que visem emular um ambiente computacional sobrecarregado com conexões concorrentes, testando os limites computacionais do servidor e suas consequências no tempo de validação dos tokens de acordo com o método de autorização e número de *claims* implementados.

Outro aspecto possível para trabalhos futuros seria o foco na implementação de IAM em plataformas de gerenciamento de identidade e controles de política de autorização dinâmica, destacando a complexidade de gerenciamento efetivo integrado a processos de monitoramento ao vivo e análise de riscos.

Por fim, também é importante ressaltar que o melhor método de autorização usado com HTTP e REST APIs depende da complexidade da arquitetura estabelecida na infraestrutura em questão. Levando em conta elementos como a granularidade das permissões de aplicação, o número de serviços expostos aos usuários e a maturidade em segurança da equipe que governa o desenvolvimento de aplicativos a serem consumidos.

AGRADECIMENTOS

Agradeço o apoio técnico e computacional do Laboratório de Tecnologias para Tomada de Decisão - LATITUDE, da Universidade de Brasília, que conta com apoio do CNPq - Conselho Nacional de Pesquisa (Outorgas 312180/2019-5 PQ-2 e 465741/2014-2 INCT em Cibersegurança), da Advocacia Geral da União (Outorga AGU 697.935/2019), da Procuradoria Geral da Fazenda Nacional (Outorga PGFN 23106.148934/2019-67), da Polícia Federal (Outorga PF 03/2020), do Mestrado Profissional em Engenharia Elétrica, na área de concentração: Segurança Cibernética – 1ª Turma para Profissionais do Setor de Inteligência (Outorga ABIN 01/2019) ao Decanatos de Pesquisa e Inovação e de Pós-Graduação da Universidade de Brasília (Outorga 7129 FUB/EMENDA/DPI/COPEI/AMORIS) e do Projeto SISTER City –Sistemas Inteligentes Seguros e em Tempo Efetivo Real para Cidades Inteligentes (Outorga 625/2022).

REFERÊNCIAS

Rose, S., Borchert, O., Mitchell, S., & Connelly, S. (2020). Zero trust architecture (No. NIST Special Publication (SP) 800-207). National Institute of Standards and Technology.

Wijayarathna, C., & Arachchilage, N. A. (2019). An empirical usability analysis of the google authentication api. In Proceedings of the evaluation and assessment on software engineering (pp. 268-274).

Wilson, Y., & Hingnikar, A. (2019). *Solving identity management in modern applications: Demystifying OAuth 2.0, OpenID connect, and SAML 2.0*. Apress.

dajiaji /pyseto (2023) https://github.com/dajiaji/pyseto [Online; accessed 14-October-2022].

Cao, R., Lu, S., Wang, X., Xiao, H., & Chi, X. (2018). Unified Account Management for High Performance Computing as a Service with Microservice Architecture. Proceedings of the Unified Account Management for High Performance Computing as a Service with Microservice Architecture, Taipei, Taiwan, 16-23.

ShuLin, Y., & JiePing, H. (2020, October). Research on unified authentication and authorization in microservice architecture. In 2020 IEEE 20th international conference on communication technology (ICCT) (pp. 1169-1173). IEEE.

Lee, S., Jo, J. Y., & Kim, Y. (2017). Authentication system for stateless RESTful Web service. Journal of Computational Methods in Sciences and Engineering, 17(S1), S21-S34.

OWASP Top Ten, Top 10 Web Application Security Risks (2021) https://owasp.org/www-project-top-ten/ [Online; accessed 14-October-2022].

Birgisson, A., Politz, J. G., Erlingsson, U., Taly, A., Vrable, M., & Lentczner, M. (2014). Macaroons: Cookies with contextual caveats for decentralized authorization in the cloud.

PASETO: Platform-Agnostic Security Tokens. paseto-standard/paseto-spec (2023) https://github.com/paseto-standard/paseto-spec [Online; accessed 14-October-2022].

Li, L., & Chou, W. (2011). Design and describe REST API without violating REST: A Petri net based approach. In 2011 IEEE International Conference on Web Services (pp. 508-515). IEEE.

Jones, M., Bradley, J., & Sakimura, N. (2015). Json web token (jwt) (No. rfc7519).

R. Terjesen, S. Haussmann, S. A. (2022). Paseto (platform-agnostic security tokens). https://www.ietf.org/id/draft-paragon-paseto-rfc-01 html. [Online; accessed 14-October-2022].

Randall Degges, A Thorough Introduction to PASETO, 2019 https://developer.okta.com/blog/2019/10/17/a-thorough-introduction-to-paseto [Online; accessed 14-October-2022].

Birgisson, A., Politz, J. G., Erlingsson, U., Taly, A., Vrable, M., & Lentczner, M. (2014). Macaroons: Cookies with contextual caveats for decentralized authorization in the cloud.

He, Xiuyu, and Xudong Yang. (2017) Authentication and authorization of end user in microservice architecture." Journal of Physics: Conference Series. Vol. 910. No. 1. IOP Publishing.

Permit.io (2022). Welcome to permit.io. https://docs.permit.io/. [Online; accessed 17-February-2023].

SuperTokens (2021). User guides. https://supertokens.com/docs/guides. [Online; accessed 13-February-2023].

Jones, M., Campbell, B., and Mortimore, C. (2015). Json web token (jwt) profile for oauth 2.0 client authentication and authorization grants. Technical report.