Estudo Comparativo de Mecanismos de Segurança Aplicados à Autenticação e Autorização em Sistemas Web

Rafael Strack[1], **Adriano Ferrasa**[1]

[1]Departamento de Informática – Universidade Estadual de Ponta Grossa (UEPG)
84.030-900 – Ponta Grossa – PR – Brasil

rafa_strack@hotmail.com, ferrasa@uepg.br

Abstract. *This article presents a comparative study of authentication and authorization mechanisms in web systems, aiming to provide an in-depth analysis of the available options and their characteristics. The study describes the most commonly used methods such as passwords, tokens, multifactor authentication, and OAuth, analyzing their advantages and disadvantages. Additionally, sequence diagrams are provided to illustrate the usage flow of each method. Finally, a comparison of the studied methods is conducted, evaluating their effectiveness in terms of security. Proper understanding of these mechanisms is crucial to ensure the security of web systems and guide the correct choice in future projects.*

Resumo. *Este artigo apresenta um estudo comparativo dos mecanismos de autenticação e autorização em sistemas web, visando fornecer uma análise aprofundada das opções disponíveis e suas características. O estudo descreve os métodos mais utilizados, como senhas, tokens, autenticação multifator e OAuth, analisando suas vantagens e desvantagens. Além disso, são apresentados diagramas de sequência para ilustrar o fluxo de utilização de cada método. Ao final, é realizada uma comparação dos métodos estudados, avaliando sua eficácia em termos de segurança. A compreensão adequada desses mecanismos é fundamental para garantir a segurança dos sistemas web e orientar a escolha correta em projetos futuros.*

1. Introdução

Com a expansão da internet, os sistemas web assumiram um papel crucial no cotidiano de bilhões de pessoas em todo o mundo. Desde o uso de redes sociais até o gerenciamento de negócios online, essas ferramentas se tornaram indispensáveis para diversas atividades. Tanto pessoas físicas quanto empresas dependem desses sistemas para garantir a eficiência e produtividade de suas operações. No entanto, a segurança desses sistemas é uma preocupação constante para desenvolvedores e usuários, pois há uma série de ameaças e vulnerabilidades que podem comprometer sua integridade.

A fundação OWASP (*Open Worldwide Application Security Project*) atualiza regularmente um relatório chamado OWASP Top 10, onde são descritos os 10 riscos de segurança mais críticos em sistemas web. Na última edição, realizada em 2021, a categoria que ficou em primeira colocação foi a quebra de controle de acesso. Em sétima colocação, ficou a categoria de falhas de identificação e autenticação [OWASP 2021]. Esses problemas são diretamente relacionados aos processos de autenticação e autorização de usuários, os quais são essenciais para garantir a proteção adequada dos sistemas.

De modo geral, a autenticação é o processo de validação de usuários, enquanto a autorização é o método que fornece as permissões de acesso corretas aos recursos para usuários previamente autenticados [Tumin and Encheva 2012]. Atualmente, existem diversos mecanismos de autenticação e autorização de usuários disponíveis, como senhas, *tokens*, autenticação multifator, OAuth, OpenID, entre outros. Cada um desses mecanismos apresenta características distintas, pontos positivos e negativos, sendo fundamental garantir a correta implementação dos mecanismos escolhidos, de forma a assegurar a efetividade da segurança dos sistemas web.

Diante desse contexto, o presente trabalho tem como objetivo realizar um estudo comparativo dos diferentes mecanismos de autenticação e autorização, com o propósito de fornecer uma análise aprofundada que auxilie na escolha adequada desses mecanismos em projetos de sistemas web. O estudo visa oferecer uma compreensão ampla das características, pontos fortes e limitações de cada mecanismo, permitindo a seleção correta e a implementação eficiente das medidas de segurança necessárias.

2. Autenticação e Autorização

Na maioria dos sistemas web, é necessário realizar um controle de acesso, para que somente certos usuários possam acessar recursos protegidos. Para isso, o mecanismo de controle de acesso depende de dois processos relacionados: a autenticação e a autorização [Sullivan and Liu 2011].

A autenticação pode ser definida como o processo de confirmação de identidade. Em sistemas web, devido a falta de conhecimento do mundo real, este processo pode não ser simples [Chapman and Chapman 2012]. Existem três grupos de fatores amplamente utilizados para confirmar a identidade de um usuário: algo que o usuário sabe, algo que o usuário é e algo que o usuário possui. No primeiro grupo, inclui-se as senhas, PINs (*Personal Identification Number*) e frases secretas. No segundo grupo, inclui-se certificados digitais, *smart cards* e *tokens* de segurança. O terceiro grupo inclui técnicas biométricas, como como impressões digitais, reconhecimento facial ou de voz, entre outras [Sullivan and Liu 2011].

A autorização é o processo pelo qual o sistema decide se um usuário previamente autenticado possui permissão para acessar um recurso ou executar uma determinada ação [Spilca 2020]. Ela pode ser realizada de várias formas, porém as mais comuns são as baseadas em usuários, garantindo acesso aos recursos do sistema web para cada usuário de forma distinta, baseada em perfis (*roles*), em que cada usuário possui um perfil, e cada perfil tem diferentes níveis de acesso aos recursos do sistema e com o protocolo OAuth [Chapman and Chapman 2012]. Alguns dos métodos de autenticação e autorização mais utilizados e conhecidos serão apresentados a seguir.

2.1. Autenticação Básica HTTP

A Autenticação Básica HTTP (*Hypertext Transfer Protocol*) foi definida na especificação HTTP/1.0 [Nielsen et al. 1996], porém foi realocada para a RFC 2617 [Franks et al. 1999]. Neste tipo de autenticação, o servidor web recusa uma transação caso o cliente não esteja autenticado, desafiando-o para obter um nome de usuário e senha válidos. Este desafio de autenticação é iniciado retornando o status HTTP 401 (não autorizado) e especificando o domínio de segurança (*security realm*) a ser acessado, com

o cabeçalho `WWW-Authenticate`. Ao receber o desafio, o cliente abre uma caixa de diálogo para que o usuário insira as credenciais para acesso ao domínio. O cliente então junta as informações de usuário e senha, colocando dois pontos entre eles, e os codifica usando o método de codificação base-64. Estas credenciais codificadas são colocadas no cabeçalho `Authorization`, e então a requisição é enviada para o servidor, que fará a validação das credenciais e, caso validadas, retorna-se o status HTTP 200 (OK) [Gourley and Totty 2002]. Um exemplo do funcionamento deste método de autenticação é mostrado na Figura 1.

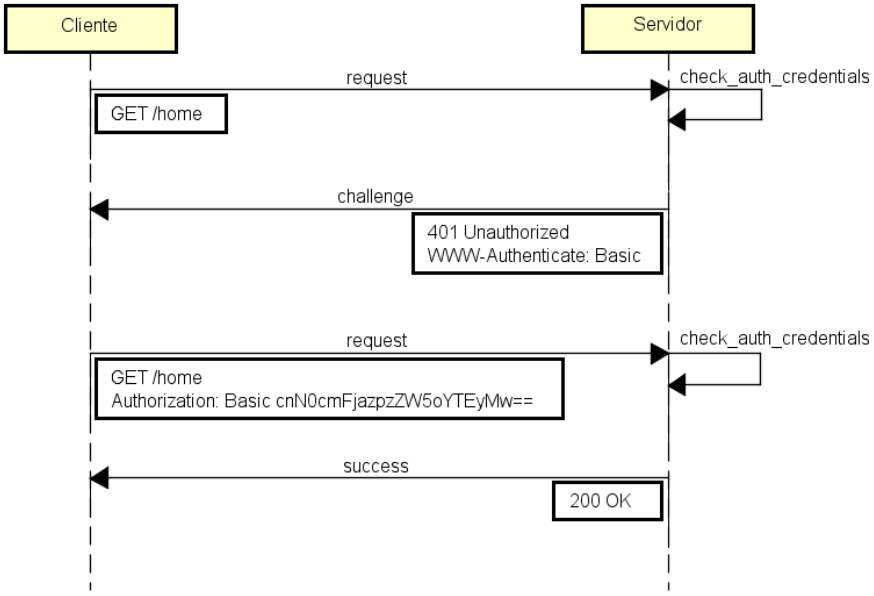

Figura 1. Exemplo de Autenticação Básica HTTP

A diretiva de domínio (*realm*) utilizada nas autenticações HTTP define os espaços de proteção do sistema web. Esses domínios permitem que os recursos protegidos sejam particionados, cada um com seu próprio esquema de autenticação e/ou autorização [Franks et al. 1999].

2.2. Autenticação Digest

A autenticação *Digest* foi especificada na RFC 2019 [Hallam-Baker et al. 1997], porém também foi realocada para a RFC 2617. Foi desenvolvida para ser uma alternativa mais compatível e segura para a autenticação básica, corrigindo as falhas mais graves da mesma, como a falta de criptografia de senhas, vulnerabilidade a captura e reprodução de pacotes e proteção contra vários outros tipos comuns de ataques [Gourley and Totty 2002].

Assim como a autenticação básica HTTP, a *Digest* é baseada no paradigma desafio-resposta [Shekh-Yusef et al. 2015]. A diferença é que foram adicionados diversos parâmetros nos cabeçalhos, para identificação única de desafios, nível de qualidade de proteção, especificação de algoritmo de hashing utilizado entre outros recursos [Chapman and Chapman 2012]. Por padrão, o algoritmo utilizado é o MD5, porém na RFC 7616 foram adicionados e recomendados os algoritmos SHA-256 e SHA-512/256 [Shekh-Yusef et al. 2015].

O parâmetro `response` é a principal parte do cabeçalho `Authorization`: ele contém uma concatenação criptografada de dados da requisição, como nome do usuário, `realm`, senha, método HTTP, URL, entre outros parâmetros, todos separados por dois pontos. [Chapman and Chapman 2012]. O cliente realiza o cálculo resultante no valor de `response`. O servidor também o faz e compara com o valor recebido. Caso as credenciais sejam válidas, retorna-se o status HTTP 200 (OK) e o cabeçalho `Authentication-Info`, que contém parâmetros utilizados para uma futura autenticação, autenticação mútua e reenvio de parâmetros para confirmação de legitimidade. Um exemplo do funcionamento deste método de autenticação é mostrado na Figura 2.

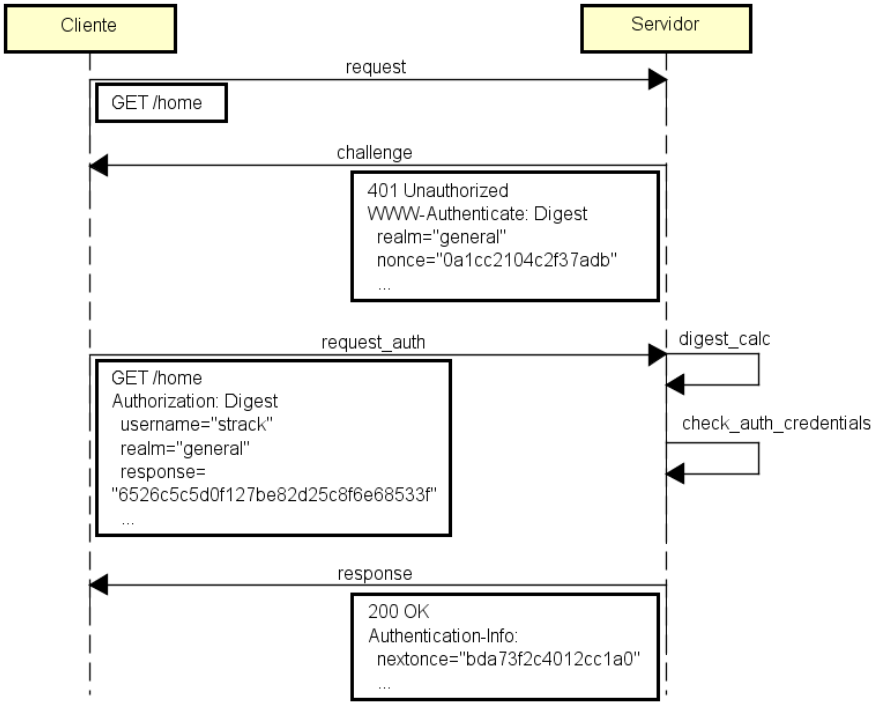

Figura 2. Exemplo de autenticação Digest

2.3. Autenticação Baseada em Sessão

Uma sessão web é uma troca de informações semipermanente entre um cliente e um servidor web [Calzavara et al. 2018]. O mecanismo de gerenciamento de estados para o HTTP, baseado em sessões, foi especificado na RFC 2109 [Montulli and Kristol 1997] com sua versão mais atual especificada na RFC 6265 [Barth 2011]. Este mecanismo utiliza o termo *cookie* para se referir às informações de estados que são passadas entre o servidor e o cliente, e salvas no cliente [Montulli and Kristol 1997].

A autenticação baseada em sessão é o método mais comum de autenticação em sistemas web. Neste método, após o envio de credenciais de acesso e validação do usuário por meio de uma requisição HTTP, o servidor gera um *cookie*, armazena-o e envia-o pelo cabeçalho `Set-Cookie` da resposta para o cliente. O cliente salva o valor, que é enviado no cabeçalho `Cookie` em toda requisição para o mesmo servidor de origem [Papathanasaki et al. 2022]. Um exemplo do funcionamento deste método é mostrado na figura 3

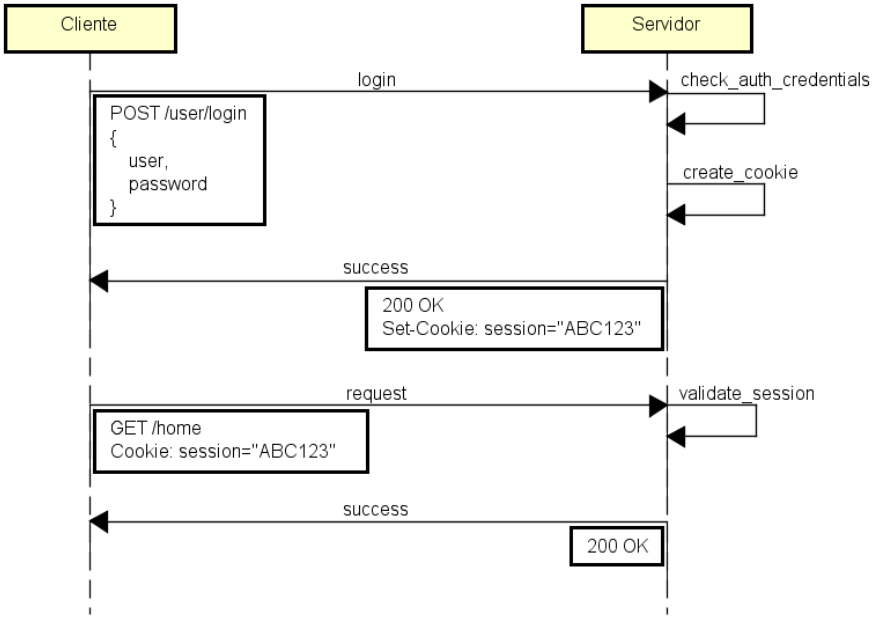

Figura 3. Exemplo de Session-Based Authentication

2.4. OAuth e Oauth 2.0

O protocolo OAuth foi especificado na RFC 5849, fornecendo um método para clientes acessarem recursos de um servidor em nome de um proprietário de recurso e também um processo para que os usuários finais autorizem o acesso de terceiros aos seus recursos de servidor sem compartilhar suas credenciais [Hammer-Lahav 2010]. Este método de autenticação funciona seguindo as seguintes etapas:

- O usuário solicita o serviço de um sistema, chamado de cliente;
- O cliente, tendo previamente configurado acesso ao servidor de recursos, possuindo um identificador e um segredo compartilhado, envia uma solicitação a este servidor para receber um *token* de solicitação;
- O servidor valida a solicitação, enviando o *token* de solicitação não-autorizado no corpo da resposta HTTP;
- O cliente redireciona o agente do usuário para o servidor, que solicita o *login* do usuário e depois a autorização para o cliente acessar o recurso;
- O cliente recebe um *token* de solicitação, que utiliza em uma nova solicitação por *token* de acesso. Estas requisições são realizadas por meio de um canal TLS;
- O servidor valida o *token* de solicitação e envia o *token* de acesso ao cliente.
- O cliente utiliza o *token* de acesso para solicitar os recursos do servidor

Na RFC 6749 foi especificado o OAuth 2.0, tornando obsoleta sua versão anterior. Esta versão possui poucas semelhanças com a versão anterior, utilizando os mesmos princípios mas com fluxo diferente e abrangendo mais casos de uso. Por este motivo as duas não são compatíveis, coexistindo e podendo ambas serem suportadas pelos sistemas [Hardt 2012]. Seu funcionamento é dado pelas seguintes etapas:

- O cliente solicita permissão de acesso ao usuário;
- O cliente recebe a permissão e solicita um *token* de acesso ao servidor de autenticação;

- O servidor de autenticação autentica o cliente e valida suas permissões, enviando a ele o *token* de acesso e opcionalmente um *token* de atualização;
- O cliente solicita o recurso protegido ao servidor de recursos, que valida o *token* de acesso e atende a solicitação.

Um exemplo de funcionamento de uma autenticação utilizando OAuth 2.0 é mostrado na figura 4

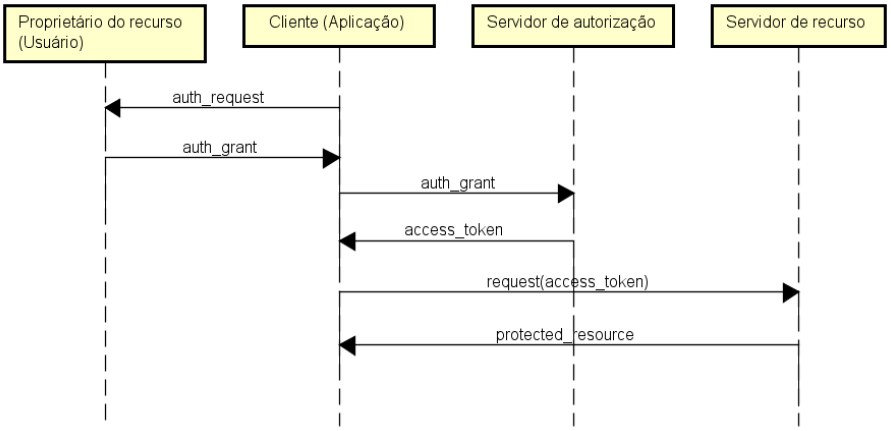

Figura 4. Exemplo de autenticação utilizando OAuth 2.0

3. Materiais e Métodos

4. Resultados e Discussão

5. Conclusão

Referências

Barth, A. (2011). HTTP State Management Mechanism. RFC 6265.

Calzavara, S., Focardi, R., Squarcina, M., and Tempesta, M. (2018). Surviving the web: A journey into web session security. *WWW '18: Companion Proceedings of the The Web Conference 2018*, page 451–455.

Chapman, N. and Chapman, J. (2012). *Authentication and Authorization on the Web*. MacAvon Media, Escócia, Reino Unido.

Franks, P. J., Hallam-Baker, P., Stewart, L. C., Hostetler, J. L., Lawrence, S., Leach, P. J., and Luotonen, A. (1999). HTTP Authentication: Basic and Digest Access Authentication. RFC 2617.

Gourley, D. and Totty, B. (2002). *HTTP: The Definitive Guide*. O'Reilly Media, Califórnia, Estados Unidos.

Hallam-Baker, P., Franks, P. J., Stewart, L. C., Sink, E. W., Hostetler, J. L., Leach, P. J., and Luotonen, A. (1997). An Extension to HTTP : Digest Access Authentication. RFC 2069.

Hammer-Lahav, E. (2010). The OAuth 1.0 Protocol. RFC 5849.

Hardt, D. (2012). The OAuth 2.0 Authorization Framework. RFC 6749.

Montulli, L. and Kristol, D. M. (1997). HTTP State Management Mechanism. RFC 2109.

Nielsen, H., Fielding, R. T., and Berners-Lee, T. (1996). Hypertext Transfer Protocol – HTTP/1.0. RFC 1945.

OWASP (2021). Owasp top 10:2021. `https://owasp.org/Top10/`. Acesso em: 26 abr. 2023.

Papathanasaki, M., Maglaras, L., and Ayres, N. (2022). Modern authentication methods: A comprehensive survey. *AI, ComputerScience and Robotics Technology*, pages 1–24.

Shekh-Yusef, R., Ahrens, D., and Bremer, S. (2015). HTTP Digest Access Authentication. RFC 7616.

Spilca, L. (2020). *Spring Security in Action*. Manning Publications, Nova Iorque, Estados Unidos.

Sullivan, B. and Liu, V. (2011). *Web Application Security, A Beginner's Guide*. McGraw-Hill, Estados Unidos.

Tumin, S. and Encheva, S. (2012). A closer look at authentication and authorization mechanisms for web-based applications. In *Recent Researches in Applied Information Science*, pages 100–105, Faro, Portugal.