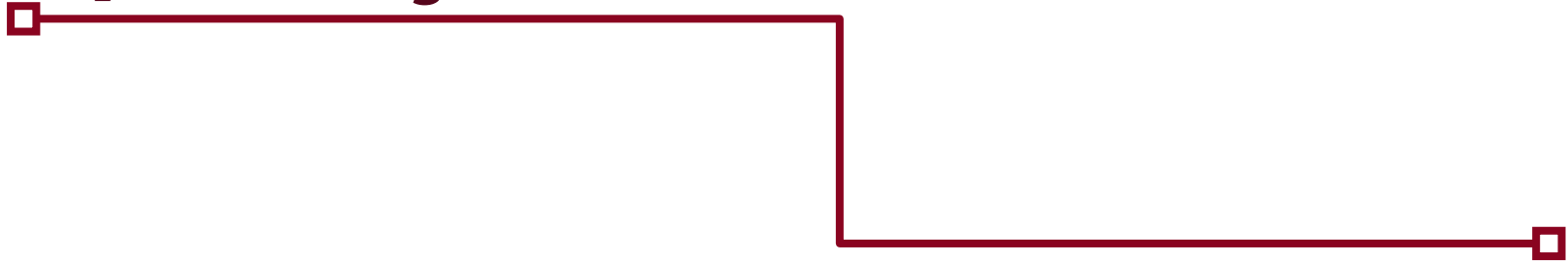


Segurança no desenvolvimento de aplicações



INTRODUÇÃO A SEGURANÇA DA INFORMAÇÃO

- ❑ Compreender o pilar de **Segurança da Informação** e empregar **técnicas de programação** segura para o **desenvolvimento de aplicações Web**, na **proteção os dados de entrada dos usuários**.
- ❑ Compreender e utilizar conceitos de **SQL Injection**, para testar as **vulnerabilidades das aplicações**.
- ❑ Aplicar técnicas de **validação** ou **codificação**, para assegurar as mensagens enviadas ao navegar.
- ❑ Realizar **armazenamento seguro das informações**, com a utilização de autenticidade e criptografia.

INTRODUÇÃO A SEGURANÇA DA INFORMAÇÃO



CAPÍTULO 1

INTRODUÇÃO

- ❑ Nos anos 1950-60, os computadores eram **máquinas enormes e isoladas**. Como não havia redes, a troca de dados exigia o transporte físico de mídias como **fitas magnéticas** ou **cartões perfurados**, o que era lento e ineficiente. As primeiras redes de computadores surgiram para resolver esse problema, permitindo a **comunicação direta entre máquinas distantes**, sem depender de dispositivos físicos.

INTRODUÇÃO

- ❑ Quando a **mobilidade de dados tornou requisito** para a comunicação entre centro de pesquisas, empresas e o governos Surgiram agências ou **provedores de serviços de rede** (como a Telenet nos EUA, nos anos 1970) para intermediar o tráfego.
- ❑ Essas agências ofereciam infraestrutura dedicada para que as **organizações compartilhassem informações** sem precisar construir suas próprias redes.

INTRODUÇÃO

- ❑ A **internet** surgiu em **1969** com a **ARPANET** (Advanced Research Projects Agency Network), rede financiada pelo Departamento de Defesa dos EUA (DARPA) para conectar institutos militares com e centros de pesquisas. Seu objetivo era **compartilhar recursos computacionais e informações** para garantir comunicação mesmo em cenários de guerra (como um ataque nuclear). E então foram surgindo padrões de comunicação.

INTRODUÇÃO

- ❑ Essa nova era da rede colocou em **risco os dados privados e sigilosos de entidades** que agora precisavam trafegar entre computadores de terceiros ou nós(roteadores).
- ❑ Então surgiram os **Hackers** que tinham o objetivo de **interferir nas comunicações** entre computadores sem permissão de maneira maliciosa.
- ❑ Para resolver problemas de segurança surgiram **protocolos abertos, políticas de redes** para manter o tráfego seguro, e os dados começaram a serem **criptografados**.

INTRODUÇÃO

- ❑ **Por que é importante entender o contexto da internet?**

- ❑ Muitos dos problemas de segurança atuais são **heranças diretas de decisões técnicas e arquiteturais** tomadas nas primeiras décadas da Internet, quando a rede era restrita a ambientes acadêmicos e militares – um ecossistema fechado e considerado *confiável*.
- ❑ Naquela época, não se imaginava que a Internet se tornaria uma infraestrutura global crítica, sujeita a ataques sofisticados e exploração em massa. Muitos **protocolos foram criados sem criptografia, sistemas foram projetados sem autenticação** robusta e a **comunicação entre redes assumia não tinha políticas para manter a confiabilidade das redes**.

SEGURANÇA DA INFORMAÇÃO

“Segurança da informação (InfoSec) é a proteção de informações importantes contra acesso não autorizado, divulgação, uso, alteração ou interrupção. Ajuda a garantir que os dados organizacionais confidenciais estejam disponíveis para usuários autorizados, permaneçam confidenciais e mantenham sua integridade.” (IBM,2024)

DIFERENÇA ENTRE CIBERSEGURANÇA E SEGURANÇA

- ❑ **Segurança da informação:** é um campo amplo que envolve a **proteção de todos os tipos de dados**, independentemente do formato ou do meio em que são armazenados ou transmitidos. (Sans)
- ❑ **Cybersecurity:** um ramo da segurança da informação focada em proteger ativos digitais e ambientes conectado, como redes, sistemas, aplicativos e plataformas em nuvem, de ameaças cibernéticas.(Sans)

PILARES DA SEGURANÇA DA INFORMAÇÃO

*Quais são os princípios que definem se uma rede está segura ou não?
Os Principais pilares são conhecidas como Tríade(CID)*



1. **Confidencialidade:** Garantir que dados sejam acessíveis apenas por autorizados.
2. **Integridade:** Manter dados precisos e sem alterações não autorizadas.
3. **Autenticidade:** Verificar a origem e legitimidade dos dados.

PILARES DE UMA REDE SEGURA



Autenticidade

- Por fim, a autenticidade é um dos pilares da segurança da informação focado em assegurar que a informação foi produzida, expedida, modificada ou destruída por uma determinada pessoa física, equipamento, sistema, órgão ou entidade(LNCC,2024);

PILARES DE UMA REDE SEGURA

As Fake News são quebras de autenticidade

<https://www.gov.br/capes/pt-br/assuntos/noticias/pesquisa-analisa-os-impactos-das-fake-news-na-pandemia>



Ministério da Educação

Órgãos do Governo

Acesso à Informação

Legislação

Acessibilidade



Entrar com gov.br

CAPES

O que você procura?



> Assuntos > Notícias > Pesquisa analisa os impactos das fake news na pandemia

BOLSISTA EM DESTAQUE

Pesquisa analisa os impactos das fake news na pandemia

Com bolsa da CAPES, José Elderson de Souza Santos foi um dos 12 pesquisadores selecionados mundialmente para apresentar sua tese na Bélgica

Publicado em 09/04/2025 10h07 | Atualizado em 09/04/2025 10h12


Compartilhe: [f](#) [in](#) [wh](#) [ln](#)

PILARES DE UMA REDE SEGURA

- ❑ Serviços disponíveis para manter a informação autêntica
 - ❑ Controle de versão em Banco de Dados
 - ❑ Assinatura digital em equipamentos
 - ❑ Certificados SSL/TLS em Sites
 - ❑ Token JWT (JSON Web Token) em APIs

PILARES DE UMA REDE SEGURA

Integridade

-  Garantia de que as informações não foram alteradas de forma indevida por partes não autorizadas, ou de acidentalmente. A integridade assegura que qualquer modificação, seja acidental ou maliciosa, possa ser detectada.

QUEBRA DE INTEGRIDADE

<https://g1.globo.com/tecnologia/noticia/equifax-empresa-de-credito-dos-eua-sofre-ataque-hacker-e-dados-de-143-milhoes-de-pessoas-sao-expostos.ghtml>



The screenshot shows the top of a G1 news article. The header is red with the G1 logo on the left, the word 'ECONOMIA' in the center, and a search icon on the right. Below the header is a dark red bar with the word 'TECNOLOGIA' in white. A purple banner with white text reads 'Garanta sua viagem em LATAM.com'. The main headline is in bold black text: 'Equifax, empresa de crédito dos EUA, sofre ataque hacker e dados de 143 milhões de pessoas são expostos'. Below the headline is a sub-headline in smaller black text: 'Empresa sofreu ataque no fim de julho e admitiu nesta quinta-feira vazamentos de informações dos usuários.'. At the bottom left of the article preview is the EFE logo, followed by the text 'Por Agência EFE' and '07/09/2017 21h05 · Atualizado há 7 anos'. A red button with a white document icon is visible on the right side of the article preview.

ECONOMIA

TECNOLOGIA

Garanta sua viagem em LATAM.com

Equifax, empresa de crédito dos EUA, sofre ataque hacker e dados de 143 milhões de pessoas são expostos

Empresa sofreu ataque no fim de julho e admitiu nesta quinta-feira vazamentos de informações dos usuários.

Por Agência EFE

07/09/2017 21h05 · Atualizado há 7 anos

QUEBRA DE INTEGRIDADE

<https://noticias.uol.com.br/internacional/ultimas-noticias/2024/03/09/passageiros-se-recusam-a-voar-em-boeing-737-max.htm#:~:text=J%C3%A1%20a%20China%20suspendeu%20definitivamente,falhas%20por%20parte%20da%20FAA.>



Jogos

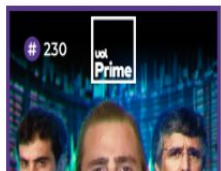
Assine UOL

Internacional

346 mortes: por que passageiros estão se recusando a voar de Boeing 737 Max

Do UOL*, em São Paulo

09/03/2024 04h00 ⓘ Atualizada em 09/03/2024 11h45



PILARES DE UMA REDE SEGURA

- ❑ Serviços disponíveis para manter a informação íntegra:
 - ❑ Hash
 - ❑ Assinatura Digital
 - ❑ Checksum
 - ❑ Controle de versão

PILARES DE UMA REDE SEGURA

- ❑ Confidencialidade:
 - ❑ garantia de que o acesso a informações seja revelada apenas para aos usuários, entidades ou processos autorizadas.

QUEBRA DE CONFIDENCIALIDADE

[Irish cyber-attack: Hackers bail out Irish health service for free](#)

BBC

Home News Sport Business Innovation Culture Arts Travel Earth Audio Video Live

ADVERTISEMENT

BBC NEWS



Irish cyber-attack: Hackers bail out Irish health service for free

21 May 2021

Share Save



ECONOMIA

TECNOLOGIA



QUEBRA DE CONFIDENCIALIDADE

[Entenda o escândalo de uso político de dados que derrubou valor do Facebook e o colocou na mira de autoridades | Tecnologia | G1](#)

Entenda o escândalo de uso político de dados que derrubou valor do Facebook e o colocou na mira de autoridades

Vazamento sem precedentes expôs dados de 50 milhões de usuários e mergulhou empresa em nova crise, pouco tempo depois de comoção sobre disseminação de notícias falsas

PILARES DE UMA REDE SEGURA

❑ Serviços disponíveis para manter a informação confidencial:

- ❑ Criptografia
- ❑ Senha
- ❑ Autenticação de dois fatores
- ❑ Biometria
- ❑ Tokens

PILARES DE UMA REDE SEGURA

- ❑ Disponibilidade
- ❑ Garantir acesso contínuo aos dados/sistemas.
 - ❑ o sistema deve assegurar a disponibilidade das informações quando necessário, garantindo que os recursos estejam acessíveis e sob demanda por entidades autorizadas, sem impedimentos indevidos. Para isso deve manter um hardware e software em funcionamento correto e fornecer largura de banda adequada.(FreeCodeCamp, 2020)

QUEBRA DE DISPONIBILIDADE

<https://www.kaspersky.com.br/resource-center/threats/ransomware-wannacry>

kaspersky

Pessoal

Empresas

Parceiros

Sobre a Kaspersky



A minha conta

Início > Produtos de uso doméstico > Centro de recursos > Ameaças > O que é o ransomware WannaCry?

O que é o ransomware WannaCry?



Seu computador está vulnerável a ataques do ransomware WannaCry? Continue lendo e descubra.

Vamos explorar tudo o que há para saber sobre o ataque do ransomware WannaCry.



Anatomia da Segurança de Dispositivos Móveis da Kaspersky

Curso gratuito sobre ameaças móveis no Brasil

Aprenda a proteger seu celular e seus dados dos maiores golpes e roubos do Brasil.

Email *

☐ Eu confirmo que autorizo a AO Kaspersky a usar meu endereço de email, para entrar em contato comigo por email sobre ofertas especiais personalizadas, notícias relevantes e eventos. Você pode, a qualquer momento, retirar seu consentimento clicando no link de cancelamento de inscrição fornecido nos emails relacionados. Eu confirmo que fui informado sobre esta Política de Privacidade para Sites da Web. *

Inscriver

PILARES DE UMA REDE SEGURA

- ❑ As Medidas para mitigar ameaças às informações disponíveis:
 - ❑ Off-site backups
 - ❑ Recuperação a desastres
 - ❑ redundância
 - ❑ Failover
 - ❑ RAID
 - ❑ Alta disponibilidade das nuvens

PRINCÍPIOS DE UMA REDE SEGURA (ANATEL)

- ❑ autenticidade;
- ❑ confidencialidade;
- ❑ disponibilidade;
- ❑ diversidade;
- ❑ integridade;
- ❑ interoperabilidade;
- ❑ prioridade;
- ❑ responsabilidade e
- ❑ transparência.

DIRETRIZES DE UMA REDE SEGURA (ANATEL)

- ❑ adoção de boas práticas e normas internacionais referentes à segurança cibernética;
- ❑ disseminação da cultura de segurança cibernética;
- ❑ a utilização segura e sustentável das redes e serviços de telecomunicações;
- ❑ identificação, proteção, diagnóstico, resposta e recuperação de incidentes de segurança cibernética;
- ❑ cooperação entre os diversos agentes envolvidos com fins de mitigação dos riscos cibernéticos;
- ❑ respeito e promoção dos direitos humanos e das garantias fundamentais, em especial a liberdade de expressão, a proteção de dados pessoais, a proteção da privacidade e o acesso à informação do usuário dos serviços de telecomunicações; e
- ❑ incentivo à adoção de conceitos de *security by design* e *privacy by design* no desenvolvimento e aquisição de produtos e serviços no setor de telecomunicações.

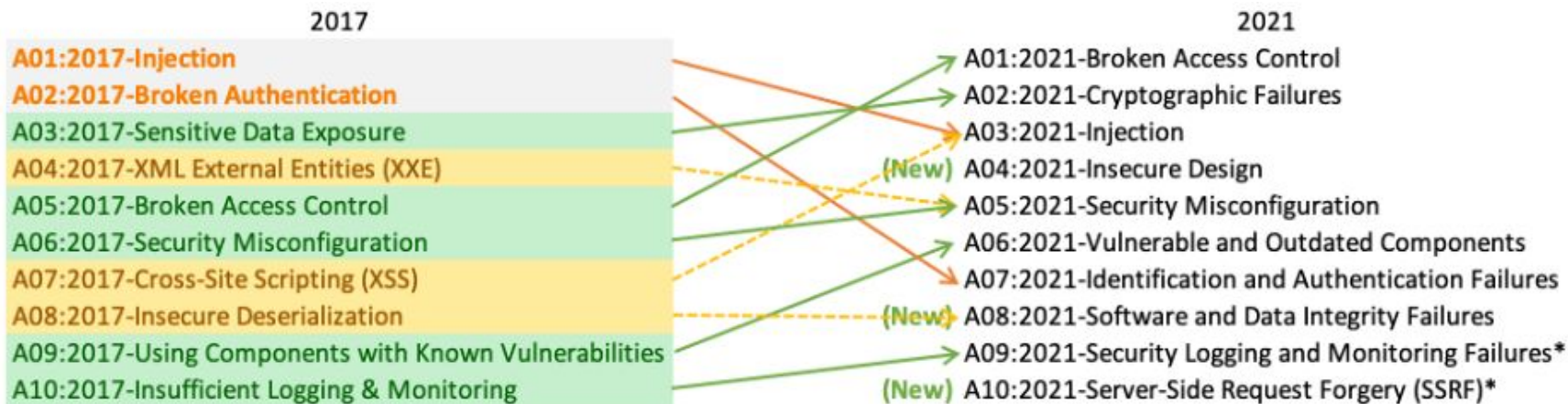
WEB APPLICATION SECURITY

- ❑ A Web Application Security é a prática de proteger web sites, aplicações, API, nuvem e IoT de ataques maliciosos
- ❑ A **W3C (World Wide Web Consortium)** é a organização responsável pelo desenvolvimento de padrões abertos, diretrizes técnicas e boas práticas voltadas à web.
- ❑ Entre suas iniciativas, destaca-se o **Web Application Security Working Group** (Grupo de Trabalho em Segurança de Aplicações Web), que se dedica à criação de especificações como **CSP (Content Security Policy)**, **CORS (Cross-Origin Resource Sharing)** e **Web Authentication**. Essas tecnologias têm como objetivo mitigar vulnerabilidades comuns, como **XSS (Cross-Site Scripting)**, **CSRF (Cross-Site Request Forgery)** e ataques de injeção.
- ❑ Acesso as Informações <https://www.w3.org/groups/wg/webappsec/>

OWASP

A OWASP (Open Worldwide Application Security Project) é uma comunidade open source e sem fins lucrativos de desenvolvedores, pesquisadores e evangelistas dedicada a melhorar a segurança de aplicações web, aplicativos, API, nuvem e IoT através de ferramentas, códigos, documentação e padrões abertos.

AS 10 PRINCIPAIS VULNERABILIDADES DE SEGURANÇA EM APLICAÇÕES WEB DE ACORDO COM A OWASP



* From the Survey

AS 10 PRINCIPAIS VULNERABILIDADES DE SEGURANÇA EM APLICAÇÕES WEB DE ACORDO COM A OWASP

- ❑ **A01:2021-Broken Access Control** - Mecanismos de autorização falhos que permitem que usuários acessem dados ou funcionalidades indevidos além do permitido;
- ❑ **A02:2021-Cryptographic Failures** - Exposição no uso de dados sensíveis que estão deixando de serem criptografados;
- ❑ **A03:2021-Injection** - Inserção de comandos maliciosos nos sistemas através de SQL Injection, NoSQL Injection, Command Injection, XSS;
- ❑ **A04:2021-Insecure Design** - Vulnerabilidades intrínsecas à arquiteturas/design do software, sem considerar os riscos de cada módulo;
- ❑ **A05:2021-Security Misconfiguration** - Configurações incorretas ou não protegidas;
- ❑ **A06:2021-Vulnerable and Outdated Components** - uso de bibliotecas, frameworks, software desatualizados;

AS 10 PRINCIPAIS VULNERABILIDADES DE SEGURANÇA EM APLICAÇÕES WEB DE ACORDO COM A OWASP

- ❑ **A07:2021-Identification and Authentication Failures** - problemas na autenticação de usuários;
- ❑ **A08:2021-Software and Data Integrity Failures** - O Software ou os dados não tem integridade garantida;
- ❑ **A09:2021-Security Logging and Monitoring Failures** - ausência de logs e alertas dificultando a detecção de ataques, e a não conformidade com as regulamentações de segurança(GDPR, LGPD);
- ❑ **A10:2021-Server-Side Request Forgery** - O servidor é enganado para realizar requisições a destino internos ou externos maliciosos;

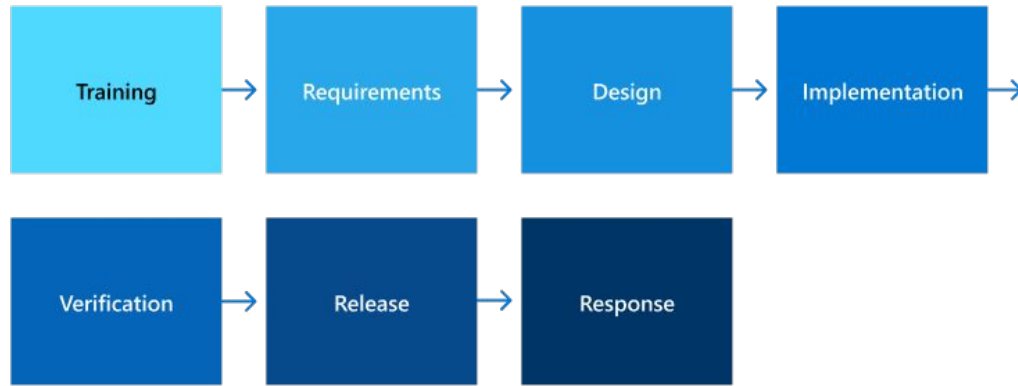
SDL - CICLO DE VIDA DE DESENVOLVIMENTO DE SEGURANÇA

- ❑ O SDL é uma abordagem estruturada e contínua que incorpora a segurança e a privacidade em todas as etapas do ciclo de vida do software, desde a definição dos requisitos até o pós-lançamento, com o objetivo de produzir software mais seguro e com menos vulnerabilidades, a um custo reduzido(**Microsoft,2024**).
- ❑ Proposto pela Microsoft em janeiro de 2004.

COMPONENTES DO SDL

O SDL é composto por sete componentes principais:

- Cinco fases principais: Requisitos, Design, Implementação, Verificação e Lançamento.
- Duas atividades de suporte: Preparação (antes das fases) e Resposta (após o lançamento).



COMPONENTES PRINCIPAIS DO SDL

- Fases principais:
 - Requisitos: Definição clara dos requisitos de segurança e privacidade, baseados em dados, ameaças conhecidas, regulamentos e melhores práticas.
 - Design: Criação de modelos de ameaças (usando ferramentas como o Threat Modeling Tool), diagramas de fluxo de dados e análise de riscos.
 - Implementação: Programadores desenvolvem o código seguindo os requisitos e utilizando ferramentas seguras fornecidas pela Microsoft.
 - Verificação: Revisões manuais e automáticas do código, incluindo análise estática, análise binária, scanners de credenciais, validação de criptografia, testes fuzz, validação de configuração e governança de componentes open source. Testes de penetração também são realizados regularmente.
 - Lançamento: O software é liberado gradualmente em “anéis” (grupos de usuários), permitindo monitoramento e correção antes da liberação total.

COMPONENTES DE SUPORTE DO SDL

- Atividades de suporte:
 - Preparação: Antes das fases principais, para garantir que tudo esteja pronto para o desenvolvimento seguro.
 - Resposta: Após o lançamento, para garantir que o software continue seguro ao longo do tempo.

REFERÊNCIAS

- ❑ **University System of Georgia.** *A brief history of the Internet.* Disponível em: https://www.usg.edu/galileo/skills/unit07/internet07_02.phtml. Acesso em 11 de Abril de 2024.
- ❑ **IBM.** (2024). *Segurança da informação: o que é e por que ela é importante.* Disponível em: <https://www.ibm.com/br-pt/topics/information-security>. Acesso em 11 de abril de 2025.
- ❑ **SANS Institute.** *What is Information Security?* Disponível em: <https://www.sans.org/security-resources/glossary-of-terms/information-security/>. Acesso em 11 de abril de 2025.
- ❑ **Agência Nacional de Telecomunicações** (2024). *Regulamentação.* Disponível em: <https://www.gov.br/anatel/pt-br/assuntos/seguranca-cibernetica/regulamentacao>. Acesso em 13 de abril de 2025.
- ❑ **FreeCodeCamp** (2020). *The CIA Triad — Confidentiality, Integrity, and Availability Explained* <https://www.freecodecamp.org/news/the-cia-triad-confidentiality-integrity-and-availability-explained/>. Acesso em 13 de abril de 2025.

REFERÊNCIAS



LABORATÓRIO NACIONAL DE COMPUTAÇÃO CIENTÍFICA (LNCC). *Os quatro pilares da segurança da informação – Confidencialidade, Disponibilidade, Integridade e Autenticidade.* Disponível em: <https://www.gov.br/lncc/pt-br/centrais-de-conteudo/campanhas-de-conscientizacao/gestao-de-seguranca-da-informacao/os-quatro-pilares-da-seguranca-da-informacao-2013-confidencialidade-disponibilidade-integridade-e-autenticidade>. Acesso em: 20 abr. 2025.



Microsoft. *Security Development Lifecycle (SDL).* Disponível em: <https://learn.microsoft.com/pt-br/compliance/assurance/assurance-microsoft-security-development-lifecycle>. Acesso em: 20 abr. 2025.

REFERÊNCIAS

- ❏ **Commonwealth of Learning.** (2023). *The CIA triad*. In *Cybersecurity training for teachers (CTT)*. Disponível em: <https://openbooks.col.org/ctft/chapter/the-cia-triad-3/>. Acesso em 12 de abril de 2025.

APÊNDICE



[RFC 4949: Internet Security Glossary, Version 2](#)

← ↻ 🔒 <https://www.rfc-editor.org/rfc/rfc4949>

[\[RFC Home\]](#) [\[TEXT\]](#) [\[PDF\]](#) [\[HTML\]](#) [\[Tracker\]](#) [\[IPR\]](#) [\[Info page\]](#)

INFORMATIONAL

Network Working Group
Request for Comments: 4949
FYI: 36
Obsoletes: [2828](#)
Category: Informational

R. Shirey
August 2007

Internet Security Glossary, Version 2

Status of This Memo

This memo provides information for the Internet community. It does not specify an Internet standard of any kind. Distribution of this memo is unlimited.

Copyright Notice

Copyright (C) The IETF Trust (2007).

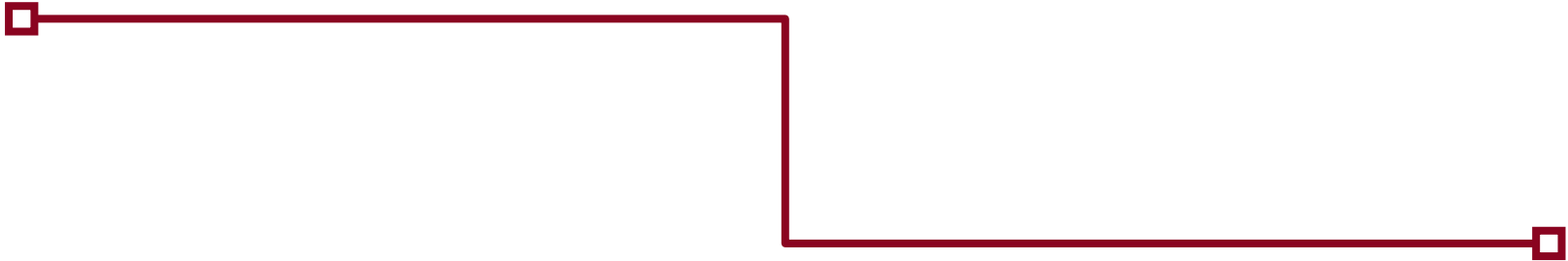
RFC Editor Note

This document is both a major revision and a major expansion of the Security Glossary in [RFC 2828](#). This revised Glossary is an extensive reference that should help the Internet community to improve the clarity of documentation and discussion in an important area of Internet technology. However, readers should be aware of the following:

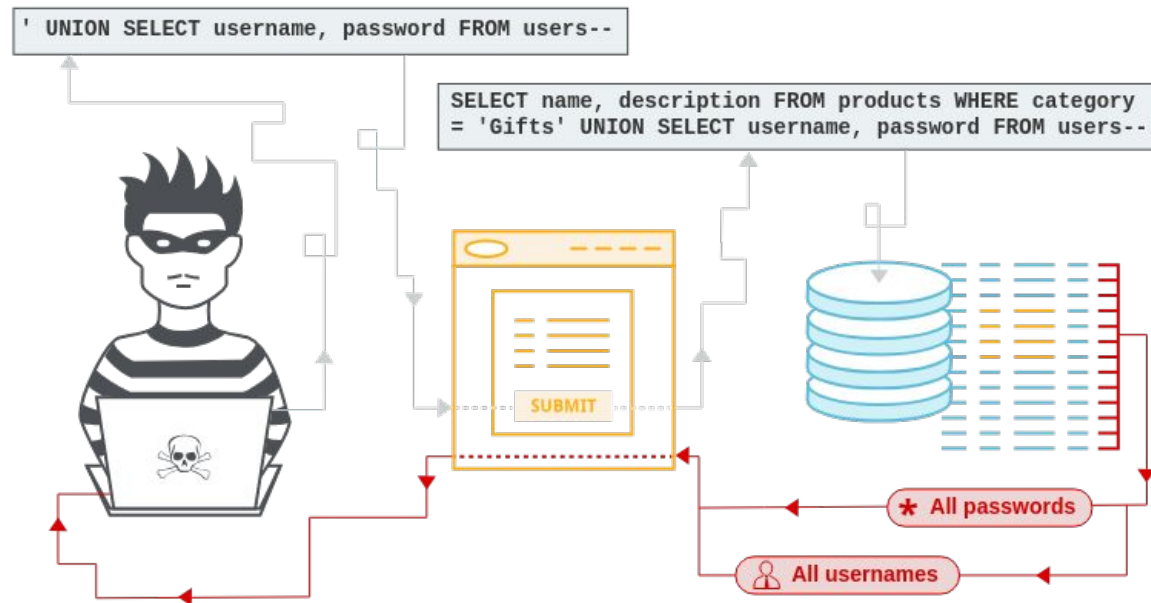
(1) The recommendations and some particular interpretations in definitions are those of the author, not an official IETF position. The IETF has not taken a formal position either for or against recommendations made by this Glossary, and the use of [RFC 2119](#) language (e.g., SHOULD NOT) in the Glossary must be understood as unofficial. In other words, the usage rules, wording interpretations, and other recommendations that the Glossary offers are personal opinions of the Glossary's author. Readers must judge for themselves whether or not to follow his recommendations, based on their own knowledge combined with the reasoning presented in the Glossary.

(2) The glossary is rich in the history of early network security work, but it may be somewhat incomplete in describing recent security work, which has been developing rapidly.

SQL Injection



INTERAÇÃO DO HACKER COM O DATABASE



INTRODUÇÃO



- SQL Injection, ou Injeção de SQL, é uma vulnerabilidade de segurança que ocorre quando comandos SQL são inseridos de forma maliciosa em campos de entrada de dados da aplicação, com o objetivo de manipular as consultas feitas ao BD.
- Essa falha é uma das mais conhecidas e exploradas na área de segurança da informação e frequentemente ocupa posições de destaque em relatórios da OWASP (Open Worldwide Application Security Project).

INTRODUÇÃO



- A origem da vulnerabilidade está na construção dinâmica de comandos SQL a partir de dados fornecidos pelo usuário, sem o devido tratamento ou validação.
- Quando a aplicação insere diretamente os dados recebidos nas consultas, o atacante pode injetar trechos de comandos SQL arbitrários que alteram a lógica original da consulta.

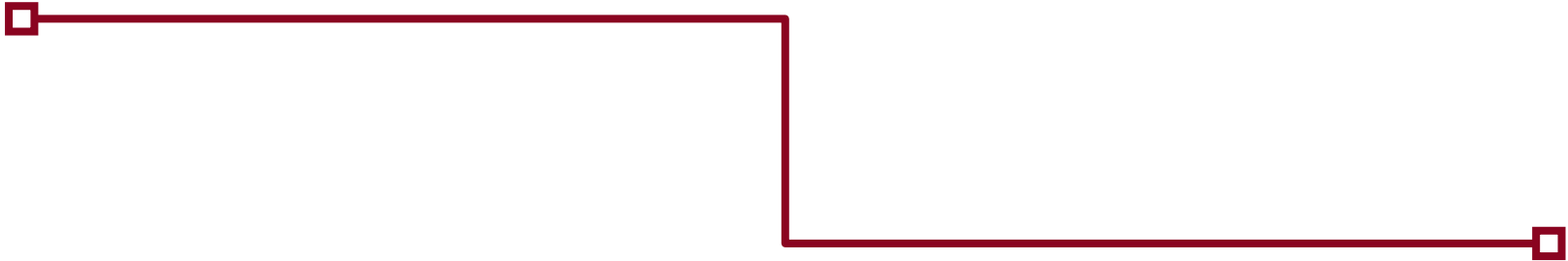
INTRODUÇÃO

- Exemplos:
- Imagine um sistema de login com a seguinte consulta:
 - `const query = 'SELECT * FROM users WHERE username = '${username}' AND password = '${password}';`
- Se um atacante inserir `admin' OR '1'='1` como nome de usuário, a consulta se torna:
 - `SELECT * FROM users WHERE username = 'admin' OR '1'='1' AND password =`
- Como `'1'='1'` é sempre verdadeiro, o atacante pode acessar o sistema sem credenciais válidas.

CONSEQUÊNCIAS DO SQL Injection

- Isso pode permitir que um invasor visualize dados que normalmente não consegue recuperar. Isso pode incluir dados pertencentes a outros usuários ou quaisquer outros dados que o aplicativo possa acessar.
- Em muitos casos, um invasor pode modificar ou excluir esses dados, causando alterações persistentes no conteúdo ou comportamento do aplicativo.
- Em algumas situações, um invasor pode escalar um ataque de injeção de SQL para comprometer o servidor subjacente.

XSS - Cross Site Scripting



INTRODUÇÃO

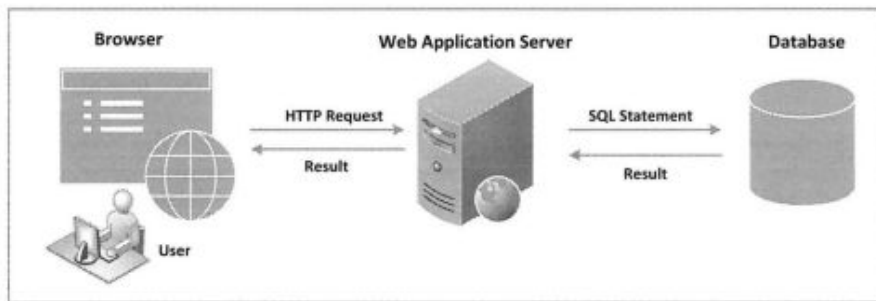
- ❑ O XSS é uma das vulnerabilidades mais antigas documentadas na história da segurança da informação. Desde o surgimento da Web 2.0 e a popularização de aplicações dinâmicas, o problema do XSS tornou-se particularmente crítico, dado o aumento das interações personalizadas baseadas em dados do usuário.
- ❑ O XSS é conhecido desde os anos 1990, quando a web começou a se tornar mais dinâmica. Com o aumento de aplicações Single-Page Applications (SPAs) e o uso massivo de JavaScript, o XSS continua sendo uma das vulnerabilidades mais críticas, aparecendo no Top 10 da OWASP há anos (<https://owasp.org/www-project-top-ten>).
- ❑ Frameworks modernos (React, Angular e Vue) incluem proteções contra XSS, mas falhas de configuração ou uso incorreto ainda permitem ataques.

INTRODUÇÃO

- ❑ É importante distinguir o XSS de outras vulnerabilidades conhecidas, como a Injeção de SQL (SQL Injection). Enquanto o SQL Injection explora falhas em comandos de banco de dados visando manipular ou roubar informações armazenadas no servidor, **o XSS explora a forma como dados são apresentados ao navegador**, afetando diretamente a experiência do usuário final.
- ❑ Diferença entre o SQL Injection e o XSS:
 - ❑ SQL Injection: visa o servidor e os dados;
 - ❑ XSS: visa o navegador e o comportamento do usuário.
- ❑ Assim, ainda que ambas envolvem a manipulação maliciosa de entradas de dados, os seus alvos e impactos são substancialmente diferentes.

INTERAÇÃO DO BROWSER COM APLICAÇÃO WEB

- ❑ O navegador comunica com o servidor web por meio do protocolo HTTP (Hypertext Transfer Protocol), responsável pela transmissão de dados na web
- ❑ Os servidores interagem com os banco de dados usando a linguagem SQL (Structured Query Language), empregada para consultar, inserir, atualizar e gerenciar dados armazenados.

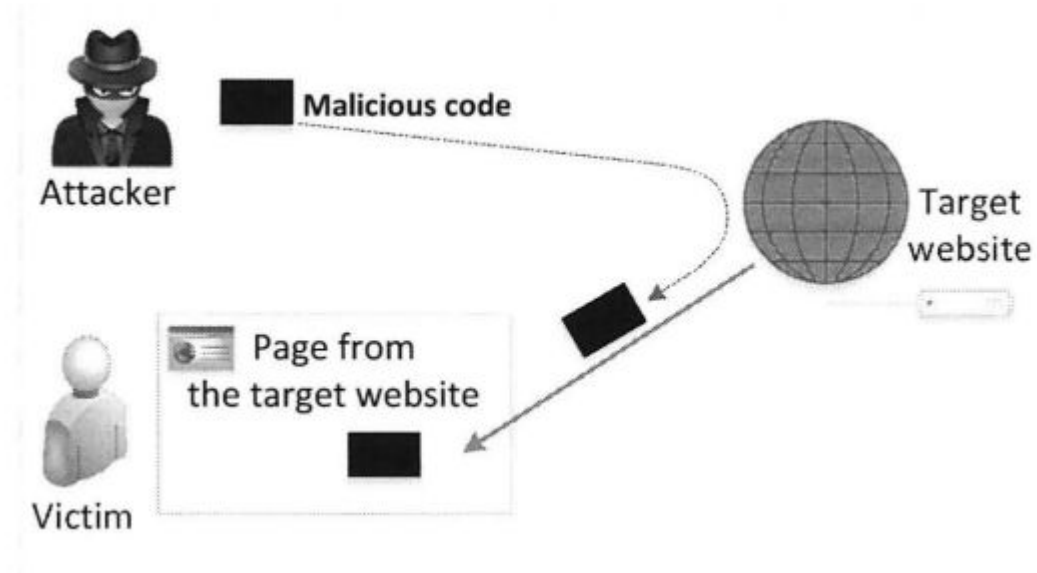


Observação: atualmente protocolo HTTPS foi amplamente substituído pelo HTTP, devido a sua capacidade de adicionar uma camada de segurança conhecida como criptografia, para a proteção dos dados entre o cliente e servidor.

Cross-Site Scripting (XSS)

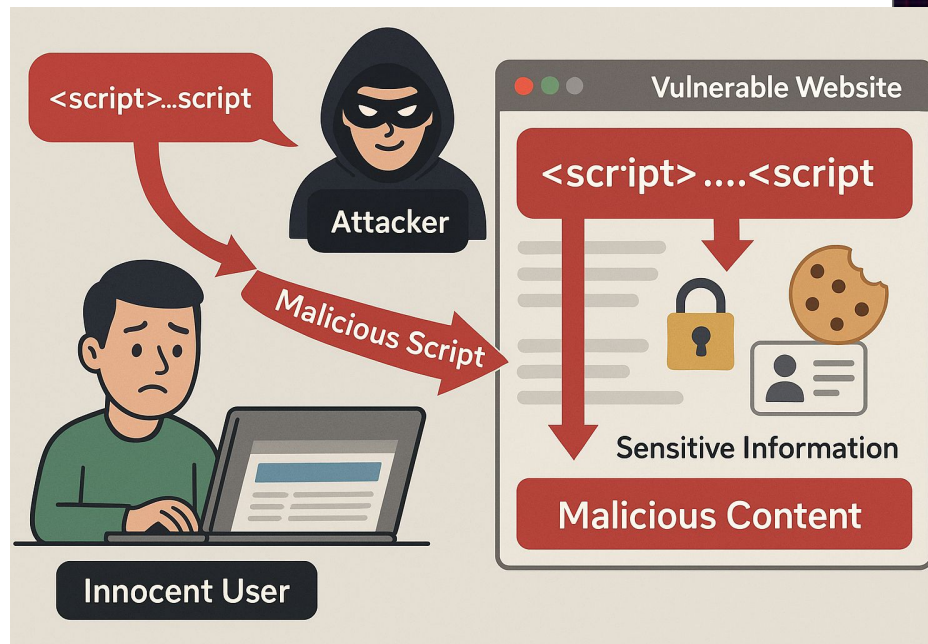
- Ataques de **Cross-Site Scripting** (XSS) são um tipo de injeção na qual scripts maliciosos são injetados em sites inofensivos e confiáveis.
- Ocorrem quando um invasor usa uma aplicação web para enviar código malicioso, geralmente na forma de um script do lado do navegador, para um usuário final diferente.
- As falhas que permitem que esses ataques sejam bem-sucedidos são bastante comuns e ocorrem em qualquer lugar onde uma aplicação web utiliza a entrada de um usuário na saída gerada sem validá-la ou codificá-la.

Cross-Site Scripting (XSS)



Cross-Site Scripting (XSS)

1. Um invasor pode usar XSS para enviar um script malicioso a um usuário desavisado.
2. O navegador do usuário final não tem como saber que o script não é confiável e o executará.
3. Como ele acredita que o script veio de uma fonte confiável, o script malicioso pode acessar quaisquer cookies, tokens de sessão ou outras informações confidenciais retidas pelo navegador e usadas com aquele site.
4. Esses scripts podem até mesmo reescrever o conteúdo da página HTML.



CARACTERÍSTICAS DOS ATAQUES XSS

- Os dados entram em uma aplicação web por meio de uma fonte não confiável, mais frequentemente uma solicitação web.
- Os dados são incluídos em conteúdo dinâmico enviado a um usuário web sem serem validados quanto a conteúdo malicioso.

CARACTERÍSTICAS DOS ATAQUES XSS

- O conteúdo malicioso enviado ao navegador web geralmente assume a forma de um segmento de JavaScript, mas também pode incluir HTML ou qualquer outro tipo de código que o navegador possa executar.
- A variedade de ataques baseados em XSS é quase ilimitada, mas eles geralmente incluem a transmissão de dados privados, como cookies ou outras informações de sessão, para o invasor, o redirecionamento da vítima para conteúdo web controlado pelo invasor ou a execução de outras operações maliciosas na máquina do usuário sob o disfarce do site vulnerável.

IMPACTOS DOS ATAQUES XSS

- **Roubo de cookies:** permitindo que um atacante se aproprie da sessão autenticada da vítima;
- **Sequestro de sessões (session hijacking):** assumindo o controle de contas em sistemas protegidos;
- **Ataques de phishing:** exibindo conteúdo falso dentro de páginas legítimas;
- **Redirecionamentos maliciosos:** encaminhando a vítima para sites controlados pelo atacante, que podem aplicar golpes ou instalar malwares;
- **Keylogging:** captura de teclas digitadas pelo usuário.

TIPOS DE ATAQUES XSS

- **Reflected XSS (Não Persistente ou Type I):** o script malicioso vem da requisição HTTP atual.
- **Stored XSS (Persistente ou Type II):** o script malicioso vem do banco de dados de uma página web
- **DOM Based XSS (Type-0):** Aqui, a vulnerabilidade existe no código do lado do cliente em vez do código do lado do servidor.
 - Definido por Amit Klein em 2005

REFLECTED XSS (Type I)

1. O XSS refletido ocorre quando a entrada do usuário é retornada imediatamente por uma aplicação web em uma mensagem de erro, resultado de pesquisa ou qualquer outra resposta que inclua parte ou toda a entrada fornecida pelo usuário como parte da solicitação.

STORED XSS (Type II)

- O XSS armazenado geralmente ocorre quando a entrada do usuário é armazenada no servidor alvo, como em um banco de dados, em um fórum de mensagens, no registro de visitantes, no campo de comentários, etc. A vítima consegue recuperar os dados armazenados do aplicativo web sem que esses dados sejam processados com segurança no navegador.

DOM Based XSS (Type-0)

- XSS baseado em DOM (ou, como é chamado em alguns textos, "XSS tipo 0") é um ataque XSS em que o payload do ataque é executado como resultado da modificação do "ambiente" DOM no navegador da vítima, usado pelo script original do lado do cliente, de forma que o código do lado do cliente seja executado de maneira "inesperada".

FILOSOFIA DE DEFESA DO XSS

- Para que um ataque XSS seja bem-sucedido, um invasor deve ser capaz de inserir e executar conteúdo malicioso em uma página web.
- Garantir que todas as variáveis passem pela validação e, em seguida, escapem ou sejam sanitizadas é conhecido como resistência perfeita à injeção.
- Qualquer variável que não passe por esse processo é uma fraqueza em potencial. Frameworks facilitam a garantia de que as variáveis sejam validadas, escapem ou sanitizadas corretamente.
- No entanto, nenhum framework é perfeito e ainda existem lacunas de segurança em frameworks populares como React e Angular. A codificação de saída e a sanitização de HTML ajudam a suprir essas lacunas.

COMO EVITAR O XSS

- **Filtragem na Entrada:**

Valide e filtre rigorosamente todos os dados recebidos do usuário, conforme o esperado.

- **Codificação na Saída:**

Ao exibir dados controlados pelo usuário, codifique conforme o contexto (HTML, URL, JS, CSS) para evitar execução maliciosa.

- **Cabeçalhos de Resposta Seguros:**

Use Content-Type e X-Content-Type-Options para forçar o navegador a interpretar os dados corretamente.

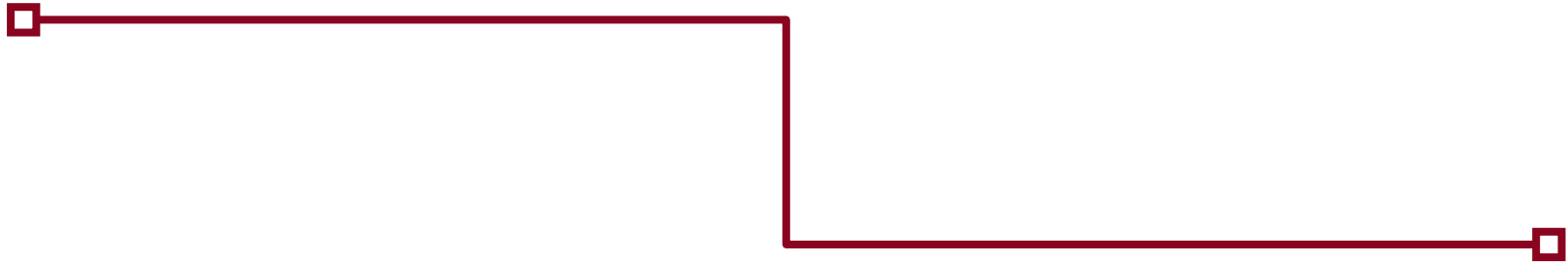
- **Política de Segurança de Conteúdo (CSP):**

Adicione uma camada extra de proteção contra XSS limitando recursos executáveis nas páginas.

REFERÊNCIAS

- ❑ OWASP Foundation. *Cross Site Scripting (XSS)*. OWASP, [s.d.]. Disponível em: <https://owasp.org/www-community/attacks/xss/>. Acesso em: 12 maio 2025.
- ❑ DU, Wenliang. *Computer Security: A Hands-on Approach*. 2. ed. [S.l.]: Wenliang Du, 2019. 688 p. ISBN 978-1733003926.
- ❑ OWASP FOUNDATION. *Cross Site Scripting Prevention Cheat Sheet*. Disponível em: https://cheatsheetseries.owasp.org/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.html. Acesso em: 11 maio 2025.
- ❑ PORTSWIGGER. *Reflected cross-site scripting (reflected XSS)*. Disponível em: <https://portswigger.net/web-security/cross-site-scripting/reflected>. Acesso em: 12 maio 2025.

Validação e Sanitização de entradas



INTRODUÇÃO

No universo digital contemporâneo, onde sistemas de software processam volumes imensos de dados provenientes de diversas fontes, a integridade e segurança dessas informações tornaram-se preocupações primordiais. Assim como um **porteiro verifica cuidadosamente quem entra em um edifício**, os sistemas de software necessitam de mecanismos robustos para examinar os dados que recebem antes de processá-los ou armazená-los. Neste contexto a análise de entrada dos dados (verificação) que estão entrando para dentro do software é importante.



VALIDAÇÃO E SANITIZAÇÃO

A validação de entrada (input validation) e a sanitização de entrada (input sanitization) são técnicas para a verificação e limpeza dos dados recebidos por um sistema. Essas práticas estão intimamente ligadas a princípios fundamentais de segurança e pretendem impedir que dados inválidos ou scripts maliciosos sejam processados, ou armazenados pelo sistema.

A validação é o processo de garantir se os dados estão em conformidade com formatos, tipos, restrições antes de ser processado pelo sistema.

A sanitização é o processo de examinar dados de entrada para identificar conteúdo que poderia ser interpretado como código malicioso ou comandos indesejados, e assim removendo-os ou modificando-os para tornar o sistema seguro.

VALIDAÇÃO E SANITIZAÇÃO


- **Exemplo:** tipos estão corretos, campo vazio, quantidade de caracteres, formato da entrada.
- **Exemplo:** Remover tags HTML de um comentário, Scripts Java Script e sentenças SQL.

Cadastro

Nome Completo

E-mail

E-mail inválido

 Please fill out this field.

CPF

CPF inválido

Senha

Cadastrar

Login

E-mail

⚠ VULNERÁVEL: Código XSS injetado (permite execução de scripts maliciosos)

CPF

⚠ VULNERÁVEL SQL INJECTION: SELECT * FROM usuarios WHERE username = 'Admin' OR '1'='1' --' AND password = [qualquer coisa];

Senha

Entrar

PRINCÍPIO DA DESCONFIANÇA

A análise de entrada dos dados é essencial para garantir a segurança e a integridade do sistema, pelos seguintes motivos:

1. **Verificação de formato:** assegura que os dados estejam em um padrão válido, como um e-mail no formato correto (ex: usuario@dominio.com);
2. **Análise semântica:** considera o contexto do dado inserido, como um CPF que, apesar de estar no formato correto, pode conter dígitos verificadores inválidos;
3. **Prevenção de código malicioso:** elimina caracteres perigosos e comandos que podem explorar vulnerabilidades, como **SQL Injection** e **XSS**;
4. **Preservação da integridade dos dados:** evita que dados inconsistentes ou corrompidos sejam armazenados;
5. **Prevenção de falhas na aplicação:** reduz o risco de comportamentos inesperados ou falhas no funcionamento do sistema.

ESTRATÉGIAS DE VALIDAÇÃO E SANITIZAÇÃO



VALIDAÇÃO SINTÁTICA

A **VALIDAÇÃO SINTÁTICA** deve aplicar a sintaxe correta dos campos estruturados.

- CPF (Cadastro de Pessoa Física)
 - **Correto:** “123.456.789.-10”
 - **Incorreto:** “12345678910”, “ABC.456.789.-10”
- Formato de email
 - **Correto:** “mail@mail.com”
 - **Incorreto:** “mail.mail.com”, “usuario.com.br”
- Telefone correto
 - **Correto:** “(11) 2321 - 2321”
 - **Incorreto:** “1123212321”, “(011)2321 2321”
- Data (formato brasileiro)
 - **Correto:** “02/03/2023”
 - **Incorreto:** “11-12-2021”, “23 21 2001”

VALIDAÇÃO SEMÂNTICA

- **A VALIDAÇÃO SEMÂNTICA** deve aplicar a correção de seus valores no contexto comercial específico (por exemplo, a data de início é antes da data de término, o preço está no intervalo esperado). Exemplos:
 - Os dois dígitos verificadores do CPF têm que corresponder aos 9 primeiros dígitos.
 - 529.982.247-25
 - A data de início deve preceder a data de término
 - Data de Início: 03/01/2000
 - Data de Término: 03/09/2005
 - O preço do produto deve estar no intervalo permitido pela plataforma.
 - Mínimo: R\$ 1,00
 - Máximo: R\$ 100.000,00



Allowlist e Denylist



O *Allowlist* usa os princípios fundamentais de "*Zero Trust*" para negar o acesso por padrão e permite somente fontes explicitamente permitidas para acessar um ativo. A lista de permissões pode ser aplicada a qualquer ativo (rede, endpoint, aplicativo, etc.) para permitir acesso específico a qualquer tipo de fonte (usuários, dispositivos, aplicativos, endereços IP, etc).

A lista negra, ou *Denylist*, é uma medida de segurança que bloqueia os dados maliciosos conhecidos, endereços IP, sites, máquinas ou programas de acesso aos recursos de uma organização. Muitas soluções de segurança serão construídas em uma lista negra como parte de um recurso de segurança anti-malware ou bloqueio de ataque e uma organização pode adicionar manualmente a algumas listas. A lista negra não satisfaz os princípios da confiança zero, porque a condição padrão para o acesso permitirá geralmente o acesso, a menos que seja a lista negra.



Allowlist e DenyList



Vantagens:

- Mais seguro por padrão
- Rejeita tudo que não é explicitamente permitido
- Menor chance de bypass
- Mais fácil de auditar

Desvantagens:

- Pode ser mais trabalhoso de implementar
- Pode rejeitar entradas válidas não previstas
- Requer manutenção ao adicionar novos casos válidos

Vantagens:

- Geralmente mais fácil de implementar
- Mais flexível com entradas não previstas
- Menor impacto na experiência do usuário

Desvantagens:

- Menos seguro por padrão
- Vulnerável a técnicas de bypass
- Difícil prever todas as variações de ataques
- Requer constante atualização da lista de bloqueio



EXEMPLOS: Allowlist e DenyList



Endereço de e-mail na lista de permissões para garantir a entrega adequada de e-mails de remetentes confiáveis.

Endereço IP da lista de permissões em um firewall para filiais

Endereço da Web na lista de permissões em um servidor para reduzir o potencial conexões externas para um ativo vulnerável

Endereço Dispositivo MAC da lista de permissões para permitir o acesso a dispositivos corporativos em uma rede.

Aplicativos corporativos para acessar um banco de dados interno

Lista de Usuário de permissões para um aplicativo corporativo interno

Endereço de e-mail de spam ou malware conhecido em um programa de segurança de e-mail.

Endereço IP da fonte de ataques maliciosos em um firewall.

Endereço da web de sites de pornografia em um servidor DNS.

Endereço Dispositivo MAC conhecidos, como botnets conhecidas.

Lista de aplicações, como assinaturas de malware em um programa antivírus.

Lista de usuários que violavam as regras da comunidade em um fórum de discussão.

Allowlist e Denylist

[DICA]

Sempre prefira o uso de *allowlist* (lista de permissões) em vez de *denylist* (lista de bloqueios), pois a *allowlist* segue o princípio de segurança "**deny by default**" (*negar por padrão*). Essa abordagem é mais segura, pois somente entradas explicitamente autorizadas são aceitas. Já a *denylist* deve ser usada somente como uma **camada complementar de defesa**, pois ela identifica somente um subconjunto conhecido de dados potencialmente maliciosos — o que pode deixar brechas para ataques não mapeados.

EXPRESSÃO REGULAR

- **A EXPRESSÃO REGULAR** (regex) consiste em uma notação formal utilizada para descrever padrões em cadeias de caracteres. Trata-se de uma ferramenta para a análise e manipulação de textos, com aplicações que incluem a **validação de formatos**, **filtragem de entradas**, **extração de informações** e **substituição de substrings**. Sua sintaxe é baseada em regras precisas que permitem representar, de forma compacta, conjuntos complexos de strings.

[DICA]

Ao construir **expressões regulares complexas**, desenvolva-as incrementalmente, testando cada parte separadamente. Use ferramentas online como *Regex101.com* para **visualizar** e **depurar** suas expressões em tempo real.

REGEX: CASOS DE USO NA VALIDAÇÃO

- **Validação de formato**
 - Email, telefone, CEP, CPF/CNPJ
 - URLs, nomes de domínio
 - Datas, horas, números
- **Validação de conteúdo**
 - Força de senha
 - Detecção de conteúdo malicioso
 - Validação de nomes de usuário
- **Sanitização**
 - Identificação de padrões para remoção
 - Extração de partes específicas de texto
 - Substituição de conteúdo problemático

REGEX: EXEMPLOS

- Para visualizar e depurar regex acesse <https://regex101.com/>
-
- **E-MAIL**
 - `/^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$/`
 - `user@mail.com`, `@user.com`, `@mail.com`
- **CPF**
 - `/^\d{3}\.\d{3}\.\d{3}\-\d{2}$/`
 - `123.456.789-10`, `123-456-789-10`
- **Detecção de XSS Básico**
 - `<script>[\s\S]*?</script>`
 - `<script>alert('XSS')</script>`, `<script src="index.js"></script>`

REGEX: LIMITAÇÕES

- Não são adequadas para análise de HTML/XML (use parsers específicos).
- Podem se tornar complexas e difíceis de manter para padrões muito elaborados.
- Expressões mal escritas podem ter problemas de desempenho (ReDoS).
- Não substituem validações semânticas (ex: verificar se um CPF é válido, não somente seu formato).
- **Exemplo formato data DD/MM/AAAA**

`^\d{2}\d{2}\d{4}$`

Aceitar datas como 99/99/9999

`^(0[1-9]|[12][0-9]|3[01])\V(0[1-9]|1[0-2])\V(19|20)\d{2}$`

O problema é não se o ano é bissexto ou não

VALIDAÇÃO CLIENTE VS SERVIDOR

No cliente (frontend) faça a validação com JavaScript, para melhorar a **experiência do usuário** (como avisos rápidos e campos obrigatórios).

No servidor (backend) faça a validação para garantir a **segurança** nos dados de forma confiável antes de serem processados internamente pelo sistema e antes de serem armazenados no banco de dados.

[DICA]

A **validação de entrada** deve ser feita **no lado do servidor** antes que qualquer dado seja processado pelas funções da aplicação. Isso porque a validação feita apenas com **JavaScript no lado do cliente** (no navegador) **pode ser facilmente burlada** por um atacante que desative o JavaScript ou utilize um proxy web para manipular os dados enviados.

SANITIZANDO HTML

- **Sanitização na web** é a prática de filtrar, limpar ou transformar dados de entrada visando remover ou neutralizar qualquer conteúdo malicioso que comprometa a segurança, a integridade ou a interpretação do sistema.
- Envolve adaptar os dados para não poderem ser interpretados como comandos, scripts ou instruções perigosas no contexto específico em que serão utilizados — como HTML, SQL ou comandos de sistema.
- Essa técnica é essencial para prevenir ataques como *Cross-Site Scripting* (XSS) e injeções de código.

SANITIZANDO HTML

- Utilizado a biblioteca `sanitize.html`

```
import sanitizeHtml from 'sanitize-html';
```

```
const html = "<strong>hello world</strong>";
```

```
console.log(sanitizeHtml(html));
```

```
<strong>hello world</strong>
```

```
console.log(sanitizeHtml("<img src=x onerror=alert('img') />"));
```

```
<img src=x />
```

```
console.log(sanitizeHtml("console.log('hello world')"));
```

```
console.log('hello world');
```

```
console.log(sanitizeHtml("<script>alert('hello world')</script>"));
```

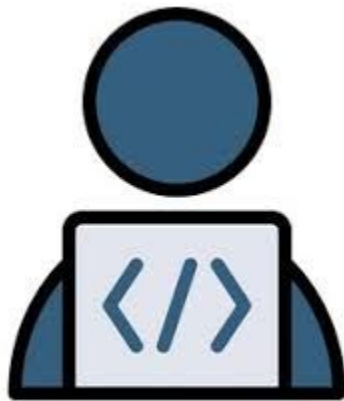
```
alert('hello world')
```

ATENÇÃO

A validação de autenticidade e veracidade refere-se ao processo de verificar se uma informação, documento ou identidade é genuína e corresponde à realidade, utilizando métodos específicos para confirmar sua origem e precisão. Este processo envolve a aplicação de técnicas como verificação de assinaturas digitais, certificados, tokens de autenticação, análise forense de documentos, verificação cruzada com fontes confiáveis e uso de tecnologias como blockchain ou criptografia para garantir que o conteúdo não foi adulterado. Em contextos digitais, essa validação é fundamental para estabelecer confiança em transações online, comunicações e documentos eletrônicos, protegendo contra falsificações, fraudes e desinformação.

LABORATÓRIO PARA ACESSAR OS EXEMPLOS

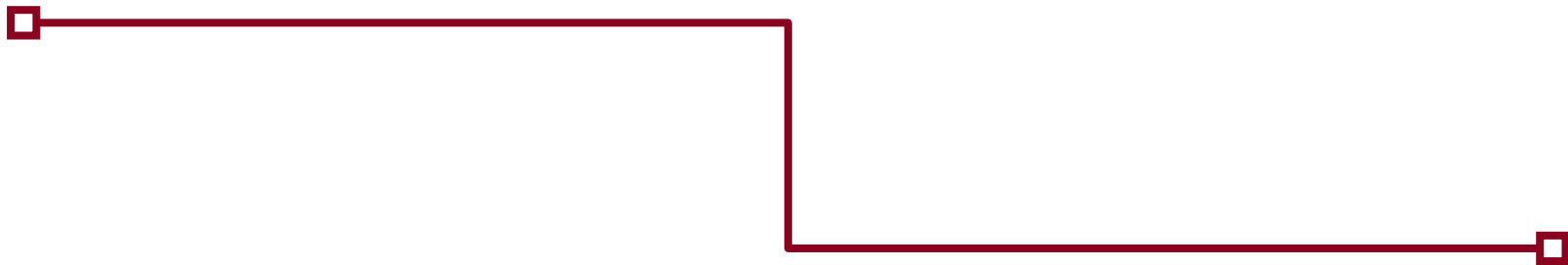
https://github.com/danynazaretech/WebSecurityApplicationFatecJacarei/tree/main/validacao_sanitizacao



REFERÊNCIAS

- OWASP. OWASP Top 10 Proactive Controls – 2024 Archive. Disponível em: <https://top10proactive.owasp.org/archive/2024/#how-to-use-this-document>. Acesso em: 6 jun. 2025.¹
- GOYVAERTS, Jan. Regular Expressions Quick Start. Disponível em: <https://www.regular-expressions.info/quickstart.html>. Acesso em: 6 jun. 2025.²
- OWASP. OWASP Validation Regex Repository. Disponível em: https://owasp.org/www-community/OWASP_Validation_Regex_Repository. Acesso em: 6 jun. 2025.
- KOTSUBO, Henrique. Posso usar regex para validar datas? Disponível em: <https://hkotsubo.github.io/blog/2019-04-05/posso-usar-regex-para-validar-datas>. Acesso em: 6 jun. 2025.
- ESECURITY PLANET. Whitelisting vs. Blacklisting: Which Is Better? Disponível em: <https://www.esecurityplanet.com/applications/whitelisting-vs-blacklisting-which-is-better/>. Acesso em: 6 jun. 2025.
- PORTSWIGGER. XSS: Beating HTML sanitizing filters. Disponível em: <https://portswigger.net/support/xss-beating-html-sanitizing-filters>. Acesso em: 6 jun. 2025.¹

Autenticação e Autorização



AUTENTICAÇÃO VS AUTORIZAÇÃO

A **Autenticação** é “O processo de estabelecer confiança de autenticidade; Nesse caso, a validade da identidade de uma pessoa e um autenticador” (NIST)

- Verifica a identidade do usuário antes da permissão de acesso ao sistema
- confirma a identidade do usuário
- compara as credenciais com dados armazenados
- Username/Password, questões de segurança

A **Autorização** é “O processo de verificar se uma ação ou serviço solicitado é aprovado para uma entidade específica” (NIST)

- Verifica o nível do acesso do usuário ao sistema
- Garante que os usuários podem somente acessar os recursos necessários
- Permite ou recusa acesso baseado nas permissões
- Regras baseadas em controle de acesso(RBAC), permissões

O NIST (National Institute of Standards and Technology) é uma agência do governo dos EUA que cria padrões técnicos e métricas científicas para tecnologia, segurança e indústria.

AUTENTICAÇÃO VS AUTORIZAÇÃO

Define “Quem pode entrar?”.

Login

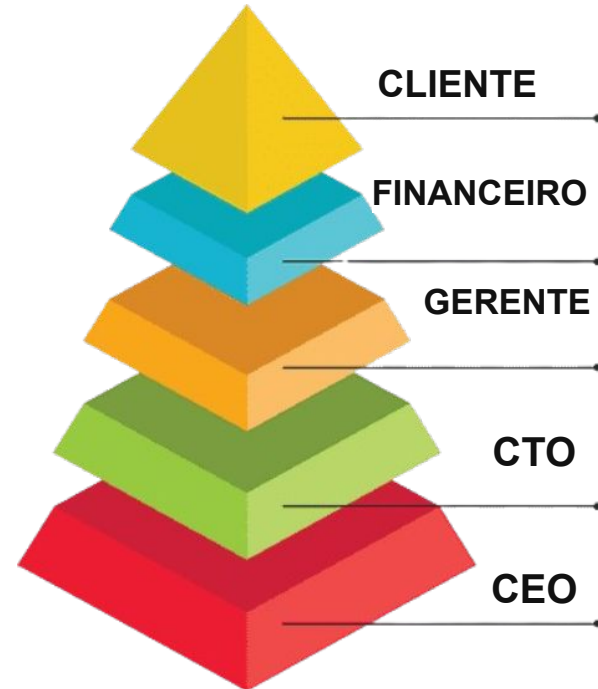
Usuário

Senha

Entrar

[Esqueceu sua senha?](#)

Define “O que o usuário pode fazer na aplicação?”.



AUTENTICAÇÃO COM SENHAS

Quais os métodos utilizados para armazenar a senha seguramente, de forma que só o usuário saiba a senha?

[DICA]

A **OWASP** aconselha o desenvolvedor a propor métodos de armazenamento de senha seguro, para proteger esse dado sensível de ataques externos e acesso de quem tem maior controle do sistema (desenvolvedores, CEO, empregados da empresa).

BOAS PRÁTICAS AUTENTICAÇÃO COM SENHAS

CLIENTE

- Tenha pelo menos 12 caracteres e senha complexa com números, caracteres especiais, letras maiúsculas e minúsculas.
- Não permita que o usuário reutilize senhas.
- Não armazene senhas em texto.

SERVIDOR

- Utilize Hash para armazenar senha (bcrypt, argon2), e criptografia para adicionar mais uma camada de segurança.
- Aplique limites de tentativas para evitar ataques de força bruta.
- Usa a API do Have I Been Pwned para verificar senhas Vazadas.
- Implementar autenticação multifator.
- Mensagens de erro para confundir o usuário sobre o que ocorreu.
- Solicite atualizações periódicas ou quando o sistema for comprometido.

AUTENTICAÇÃO MULTIFATOR

A autenticação multifator (MFA) é um método de segurança no qual o usuário precisa comprovar sua identidade mais de uma vez para acessar um sistema — por exemplo, digitando uma senha e depois inserindo um código enviado para o celular. A MFA se diferencia da autenticação de dois fatores (2FA) porque, enquanto a 2FA exige exatamente dois fatores, a MFA pode envolver três ou mais etapas de verificação.

[DICA]

As camadas de fatores devem ser usadas de forma independente uma com a outra, pois se um for comprometido, os demais permanecerão protegidos.

FATORES DE AUTENTICAÇÃO

Fator de Autenticação	Exemplos
O que você sabe?	Senhas, PINs, perguntas de segurança
O que possui?	Tokens, certificados digitais, e-mail, SMS, chamadas telefônicas
Algo que você é?	Impressão digital, reconhecimento facial, escaneamento de íris
Onde você está?	Endereço IP de origem, geolocalização, georreferenciamento
o que você faz?	Perfil comportamental, dinâmica de digitação, movimento do mouse, forma de andar, pressão da assinatura digital

O QUE É JWT?

- O JWT, ou JSON Web Token, é um padrão aberto (RFC 7519) para envio de dados Json assinados entre sistemas. Ele é usado para compartilhar com segurança dados JSON entre cliente e servidor.
- Em vez de enviar dados brutos (como informações do usuário), que podem ser facilmente manipulados, os tokens oferecem um mecanismo seguro de validação. São comumente usados em mecanismos de autenticação, gerenciamento de sessões e controle de acesso.
- São comumente usados em mecanismos de autenticação, gerenciamento de sessões e controle de acesso.
- Se um invasor puder modificar com sucesso um JWT, poderá escalar seus próprios privilégios ou se passar por outros usuários.

JWT (JSON Web Token)

Quais os métodos utilizados para armazenar a senha seguramente, de forma que só o usuário saiba a senha?

1. **Cabeçalho:** O cabeçalho contém metadados sobre o token, incluindo o algoritmo de assinatura e o tipo de token aqui metadados significa dados sobre dados.
2. **Carga útil:** A carga útil contém as informações sobre o usuário também chamadas de reclamação e algumas informações adicionais, incluindo o registro de data e hora em que foi emitido e o tempo de validade do token.
3. **Assinatura:** Garante a integridade e a autenticidade do token.

O QUE É JWT?

URL Encoded Header

```
{
  "alg": "HS256",
  "typ": "JWT"
}
```

Base64Url Encoded Payload

```
{
  "userId": 123,
  "role": "admin",
  "exp": 1672531199
}
```

Signature

```

HMACSHA256(
    base64UrlEncode(header) +
    "." +
    base64UrlEncode(payload),
    secret)

```

Após todas essas etapas, o token JWT é criado unindo o cabeçalho, segue a carga e a assinaturas concatenando os seguintes pontos: **URL Encoded Header + Base64Url Encoded Payload + Signature**.

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9

eyJzdWliOilxMjM0NTY3ODkwlwibmFtZSI6IkpvaG4gRG9lliwiaWF0IjoxNzA4MzQ1MTIzLCJleHAiOjE3M
DgzNTUxMjN9

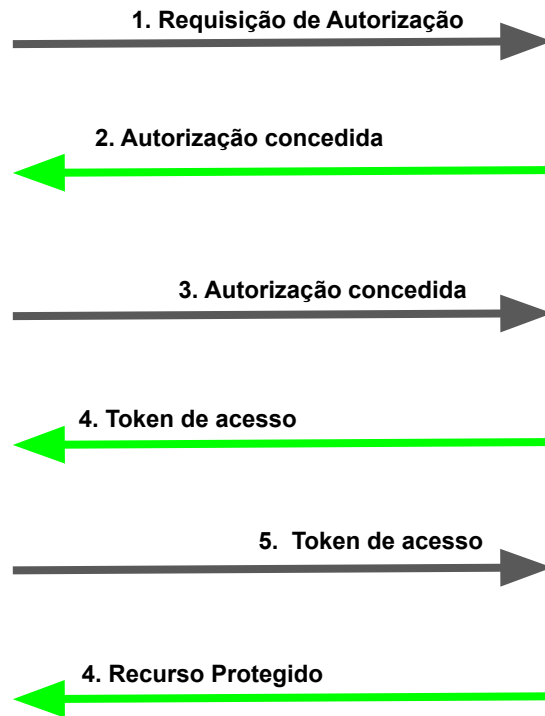
SflKxwRJSMeKKF2QT4fwpMeJf36POk6yJV_adQssw5c

OAuth 2.0: AUTORIZAÇÃO SEGURA

O protocolo OAuth 2.0 estabelece um modelo de autorização que permite a aplicações de terceiros o acesso restrito a recursos hospedados em serviços HTTP, mediante o consentimento do titular dos dados. Essa abordagem substitui o protocolo OAuth 1.0, conforme especificado no RFC 5849, oferecendo maior flexibilidade e segurança no controle de acessos (RFC 5849). Principais Vantagens são:

- **Delegação de acesso:** Permite que aplicativos acessem recursos em nome do usuário, sem que o usuário forneça suas credenciais ao aplicativo;
- **Escalabilidade:** Pode ser usado em diferentes dispositivos e plataformas, desde aplicativos móveis até sistemas web.
- **Controle granular:** Oferece escopos de acesso, permitindo que o usuário autorize somente partes específicas de seus dados;
- **Experiência do usuário:** Usuários se autenticam diretamente com provedores confiáveis;
- **Expiração e revogação:** Tokens podem ter tempo de vida limitado e serem revogados a qualquer momento, aumentando o controle;
- **Compatibilidade com OpenID Connect:** Pode ser estendido para autenticação, não somente autorização, permitindo login seguro.

OAuth 2.0: AUTORIZAÇÃO SEGURA



Resource Owner

Servidor de
Autorização

Resource Server

OAuth 2.0: AUTORIZAÇÃO SEGURA

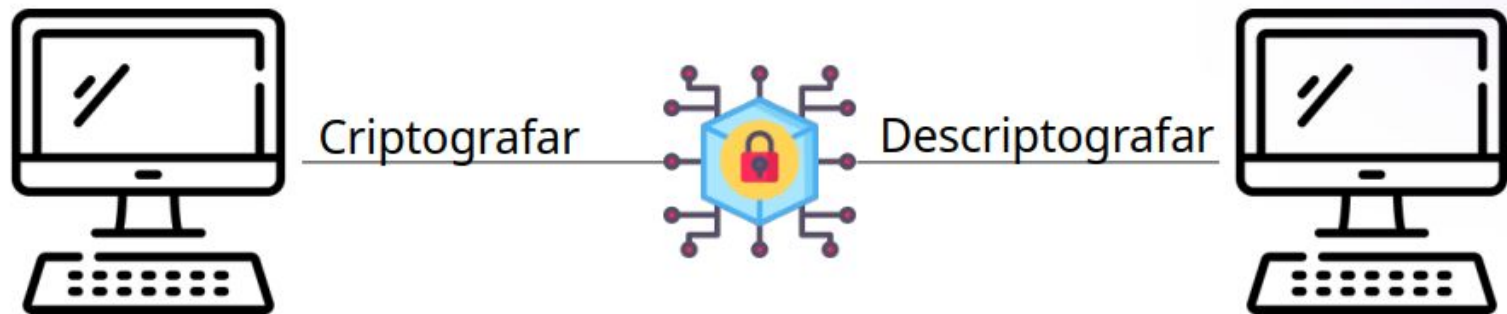
1. **Solicitação de Autorização:** O **cliente** (aplicativo) solicita autorização ao **proprietário do recurso** (usuário)
2. **Concessão de Autorização:** O cliente recebe um “**código de autorização**”, o qual é uma **prova de que o usuário permitiu o acesso**.
3. **Solicitação de Token de Acesso:** O cliente envia o **código de autorização** para o **servidor de autorização**, se identificando, e solicita um **token de acesso**.
4. **Emissão do Token:** O servidor de autorização **verifica a identidade do cliente e valida o código de autorização**. Se tudo estiver correto, ele gera e envia um **token de acesso**.
5. **Acesso ao Recurso Protegido:** O cliente utiliza o **token de acesso** para solicitar o recurso protegido ao **servidor de recurso**.
6. **Validação e Resposta:** O servidor de recurso **verifica se o token é válido**, e então **libera o recurso solicitado**.

APÊNDICE: CRIPTOGRAFIA

- A criptografia utiliza técnicas matemáticas para transformar dados e evitar que sejam lidos ou alterados por terceiros não autorizados(NIST).
- Dados Criptografados fornecem uma comunicação segura.
- Tipos de criptografia:
 1. Simétrica
 2. Assimétrica
 3. Hash

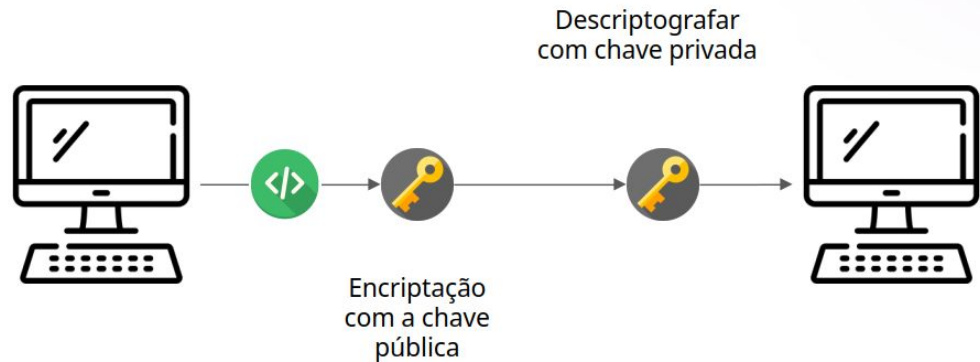
APÊNDICE: CRIPTOGRAFIA

A **criptografia simétrica** é uma técnica onde a mesma chave usada para criptografar é a mesma para descriptografar.



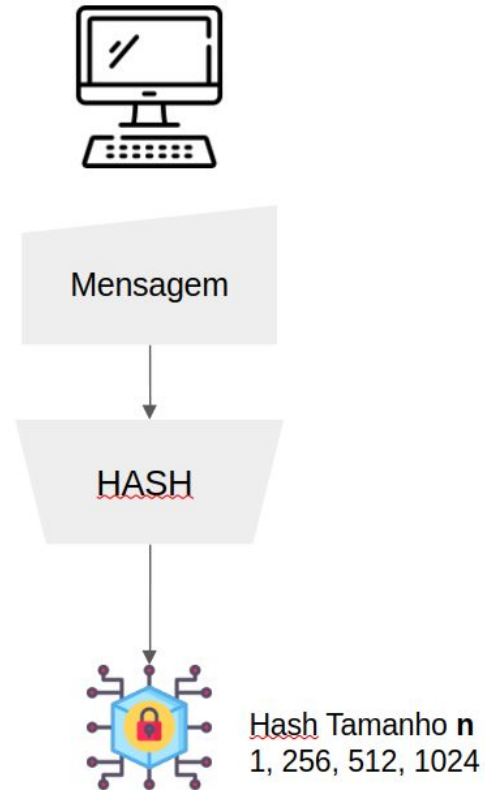
APÊNDICE: CRIPTOGRAFIA

- **A criptografia assimétrica** utiliza dois pares de chaves distintas para criptografar e descriptografar dados:
 1. A chave pública é usada para criptografar e ela permanece aberta para o público;
 2. A chave privada usada para descriptografar por quem recebe a mensagem



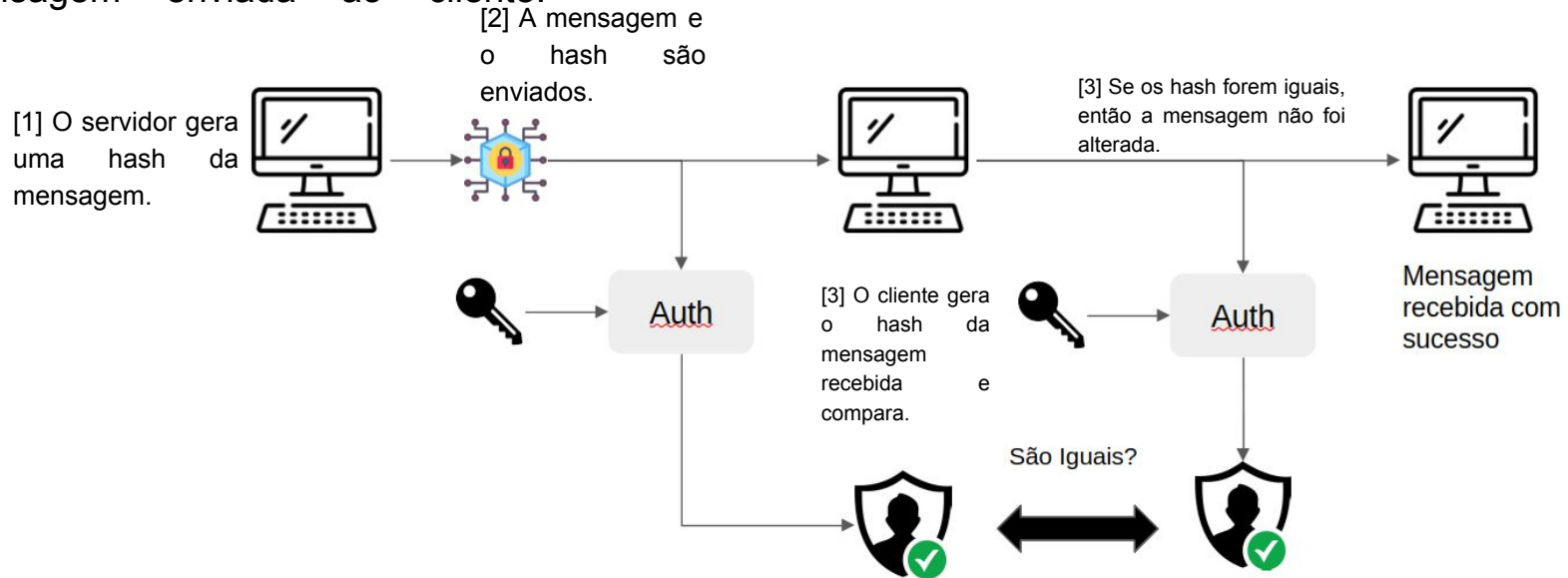
APÊNDICE: CRIPTOGRAFIA

A Hash é uma função que transforma uma entrada (como um texto, arquivo ou senha) em uma sequência fixa de caracteres, geralmente uma sequência de números e letras. Essa sequência é chamada de **valor hash** ou **digest**.



APÊNDICE: CRIPTOGRAFIA

O processo de autenticação da Hash verifica a integridade da mensagem enviada ao cliente.



APÊNDICE: CRIPTOGRAFIA

[DICA]

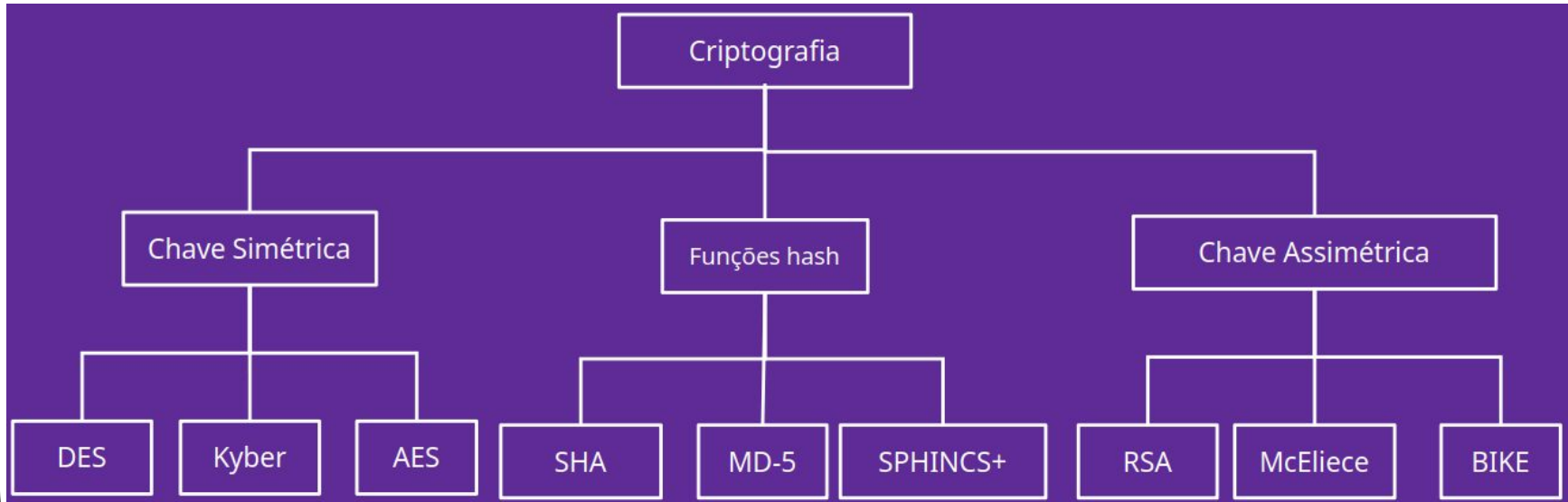
O hash não garante o sigilo dos dados; para isso, é necessário adicionar uma camada de criptografia, como fazem os protocolos HTTPS e TLS.

APÊNDICE: CRIPTOGRAFIA

As Ferramentas em Java Script:

1. `crypto`
2. `bcrypt`
3. `argon2`
4. `crypto-js`

APÊNDICE: CRIPTOGRAFIA



REFERÊNCIAS

NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY. Glossary. Disponível em:

<https://csrc.nist.gov/glossary?index=A>. Acesso em: 9 jun. 2025.

FORTINET. Authentication vs Authorization: Key Differences. Disponível em:

<https://www.fortinet.com/resources/cyberglossary/authentication-vs-authorization>. Acesso em: 9 jun. 2025.

OWASP. Authorization Cheat Sheet. Disponível em:

https://cheatsheetseries.owasp.org/cheatsheets/Authorization_Cheat_Sheet.html. Acesso em: 9 jun. 2025.

PORTSWIGGER. Password-based authentication. Disponível em:

<https://portswigger.net/web-security/authentication/password-based>. Acesso em: 9 jun. 2025.

OWASP. Authentication Cheat Sheet. Disponível em:

https://cheatsheetseries.owasp.org/cheatsheets/Authentication_Cheat_Sheet.html. Acesso em: 9 jun. 2025.

POSTMAN. What is JWT? Disponível em: <https://blog.postman.com/what-is-jwt/>. Acesso em: 9 jun. 2025.

LODDERSTEDT, T.; BRADLEY, J.; LABUNETS, A. OAuth Security Topics. Internet Engineering Task Force (IETF), Internet-Draft, 9 set. 2019. Disponível em: <http://datatracker.ietf.org/doc/html/draft-ietf-oauth-security-topics>. Acesso em: 9 jun. 2025.