Aula 2 – Arquitetura Web e Pontos de Ataque

Imagine que você está prestes a construir uma casa. Antes de erguer paredes, você precisa entender o terreno, os materiais disponíveis e, crucialmente, onde essa casa pode ser mais vulnerável a intempéries ou invasores. No universo da segurança em aplicações web, a lógica é a mesma. Antes de proteger, precisamos compreender a "arquitetura" da nossa "casa digital": como ela é construída, quais são seus alicerces, seus cômodos e, principalmente, suas portas e janelas que podem ser pontos de entrada para ataques.

Esta aula é o seu guia para desvendar a estrutura fundamental da web e identificar os locais onde os atacantes costumam focar suas energias. Ao final, você não apenas entenderá como as aplicações web funcionam por trás das cenas, mas também será capaz de visualizar os potenciais pontos fracos que exigem atenção redobrada. Nosso objetivo é que você desenvolva uma percepção aguçada para a segurança, transformando o conhecimento da arquitetura web em uma ferramenta poderosa para a defesa digital.

Vamos explorar desde os protocolos básicos que regem a comunicação na internet até as ferramentas que nos permitem "espiar" o tráfego de dados, passando pelos componentes essenciais que formam a espinha dorsal de qualquer aplicação web. Prepare-se para uma jornada que conectará conceitos teóricos a aplicações práticas, preparando você para os desafios de segurança que o aguardam.

A Web em Ação: Entendendo o Diálogo Digital

A Web como Biblioteca

Cada página é um livro, você é o leitor, e o servidor é o bibliotecário que busca e entrega o conteúdo solicitado.

Protocolo HTTP

O conjunto de regras que define como navegadores e servidores trocam informações de forma organizada.

Segurança HTTPS

Adiciona criptografia à comunicação, protegendo dados sensíveis contra interceptação por terceiros.

No nosso dia a dia, a web parece mágica: clicamos em um link e, instantaneamente, uma página aparece. Mas por trás dessa simplicidade, há um diálogo complexo e bem orquestrado. Pense na web como uma vasta biblioteca global, onde cada livro é uma página ou recurso, e você é o leitor. Para pegar um livro, você precisa pedir ao bibliotecário, que o busca e o entrega. Esse "pedido" e "entrega" são a essência da comunicação web.

> 💡 **Ponto-chave:** O coração dessa comunicação é o **protocolo HTTP** (Hypertext Transfer Protocol). Ele é, em termos simples, o conjunto de regras que define como os navegadores (seus "clientes") e os servidores web (os "bibliotecários") trocam informações. É como a etiqueta de um jantar formal: todos sabem como se comportar, o que dizer e como responder, garantindo que a conversa flua sem mal-entendidos. Sem o HTTP, a internet como a conhecemos simplesmente não existiria.

No entanto, um jantar formal pode ser ouvido por bisbilhoteiros. É aí que entra o **HTTPS** (Hypertext Transfer Protocol Secure). Ele adiciona uma camada de segurança, criptografando a comunicação entre o seu navegador e o servidor. Imagine que, em vez de falar abertamente, você e o bibliotecário agora sussurram em um código secreto que só vocês dois entendem. Isso impede que terceiros interceptem e leiam suas informações, protegendo dados sensíveis como senhas e detalhes de cartão de crédito. A presença do "S" no HTTPS é um pequeno detalhe com um impacto gigantesco na sua segurança online.

Clientes e Servidores: Os Pilares da Interação Web

Cliente

Toda interação na web envolve, no mínimo, dois participantes principais: o **cliente** e o **servidor**. O cliente é, na maioria das vezes, o seu navegador web (Chrome, Firefox, Edge, Safari) ou um aplicativo no seu celular. Ele é o "solicitante" da informação, o ponto de partida da sua jornada online. Quando você digita um endereço ou clica em um link, é o cliente que formula o pedido e o envia para a internet.

- Navegadores web
- Aplicativos mobile
- Ferramentas de linha de comando
- APIs consumidoras

Servidor

Por outro lado, o **servidor** é a máquina poderosa que "serve" o conteúdo. Ele hospeda os sites, os bancos de dados e as aplicações que você acessa. Pense no servidor como um grande armazém de informações, sempre pronto para receber pedidos dos clientes, processá-los e enviar as respostas adequadas. Quando seu navegador pede uma página, o servidor a encontra, a prepara e a envia de volta.

- Hospeda aplicações web
- Gerencia bancos de dados
- Processa requisições
- Envia respostas

Essa divisão de papéis é fundamental para a escalabilidade e a organização da internet. Um servidor pode atender a milhares de clientes simultaneamente, e um cliente pode se conectar a inúmeros servidores diferentes ao longo do dia. Entender essa dinâmica é o primeiro passo para identificar onde as vulnerabilidades podem surgir, pois cada ponto de interação é um potencial vetor de ataque.

A Anatomia de uma Requisição e Resposta HTTP

O Diálogo Detalhado

Quando seu navegador (o cliente) solicita uma página, ele não apenas grita "Quero a página X!". Ele envia uma mensagem estruturada, uma **requisição HTTP**, que é como uma carta com remetente, destinatário, assunto e conteúdo. Essa requisição é composta por várias partes cruciais que os atacantes frequentemente exploram.

01

Linha de Requisição

Especifica o **método HTTP** (como GET, POST, PUT, DELETE – que indicam a ação desejada, como "obter" uma página ou "enviar" dados de um formulário) e o caminho do recurso que você quer acessar.

02

Cabeçalhos (Headers)

Metadados sobre a requisição: quem está fazendo o pedido (User-Agent), qual tipo de conteúdo é aceito, cookies de sessão, e muitas outras informações que ajudam o servidor a entender o contexto.

03

Corpo (Body)

Para métodos como POST, há o corpo da requisição, que contém os dados que estão sendo enviados, como as informações de um formulário de login.

A Resposta do Servidor

Após o servidor processar a requisição, ele envia de volta uma **resposta HTTP**, que também é uma mensagem estruturada.

1

Linha de Status

Inclui um **código de status HTTP** (como 200 OK para sucesso, 404 Not Found para página não encontrada, 500 Internal Server Error para um problema no servidor) e uma mensagem descritiva.

2

Cabeçalhos de Resposta

Fornecem metadados sobre o servidor, o tipo de conteúdo da resposta, e outras informações importantes.

3

Corpo da Resposta

Contém o conteúdo real que o cliente solicitou, como o código HTML da página web, uma imagem ou dados JSON.

Métodos e Códigos de Status HTTP

Entendendo as Intenções e os Resultados

Métodos HTTP

Os **métodos HTTP** são verbos que indicam a intenção da requisição.

GET

Usado para solicitar dados de um recurso especificado, como carregar uma página web ou uma imagem. É como pedir um livro na biblioteca: você apenas quer lê-lo.

POST

Utilizado para enviar dados ao servidor, como preencher um formulário de cadastro ou fazer um login. Neste caso, você está "depositando" informações no sistema.

PUT

Para atualizar um recurso existente, substituindo-o completamente com novos dados.

DELETE

Para remover um recurso do servidor, mais comum em APIs.

Códigos de Status

Os **códigos de status HTTP** são a forma do servidor de comunicar o resultado da requisição.

1xx - Informacionais

A requisição foi recebida e o processo continua.

2xx - Sucesso

A requisição foi recebida, entendida e aceita com sucesso (ex: 200 OK).

3xx - Redirecionamento

É preciso tomar uma ação adicional para completar a requisição (ex: 301 Moved Permanently).

4xx - Erro do Cliente

A requisição contém sintaxe incorreta ou não pode ser atendida (ex: 404 Not Found, 403 Forbidden).

5xx - Erro do Servidor

O servidor falhou ao cumprir uma requisição aparentemente válida (ex: 500 Internal Server Error).

⚠️ **Atenção de Segurança:** Entender esses métodos é vital para a segurança, pois um atacante pode tentar usar um método inadequado para uma ação, ou manipular os dados enviados em um POST para explorar vulnerabilidades. A análise dos códigos de status é crucial para identificar comportamentos anômalos. Um atacante pode buscar respostas 200 OK em requisições que deveriam falhar (indicando uma falha de autorização, por exemplo) ou usar erros 4xx/5xx para inferir informações sobre a arquitetura ou vulnerabilidades do servidor.

Modelo Cliente-Servidor e Seus Pontos de Vulnerabilidade

O modelo Cliente-Servidor, embora eficiente, introduz uma série de pontos de vulnerabilidade inerentes à sua natureza distribuída e à constante troca de informações. Pense nisso como uma ponte: ela conecta dois lados, mas cada pilar e cada viga podem ser um ponto fraco se não forem bem construídos ou mantidos. A comunicação entre o cliente e o servidor é um terreno fértil para ataques, pois os dados trafegam por uma rede que não é totalmente controlada por nenhuma das partes.

Validação de Dados

Um dos principais pontos de vulnerabilidade reside na **validação de dados**. O cliente envia dados, e o servidor os processa. Se o servidor confia cegamente nos dados enviados pelo cliente, sem validação rigorosa, ele se torna suscetível a injeções de código (SQL Injection, XSS), manipulação de parâmetros e outras formas de ataque.

Autenticação e Autorização

Outro ponto crítico é a **autenticação e autorização**. Como o servidor sabe quem você é e o que você tem permissão para fazer? Falhas nesses mecanismos podem permitir que um atacante se passe por outro usuário ou acesse recursos que não deveria.

Exposição de Informações

A **exposição de informações sensíveis** nos cabeçalhos ou no corpo das respostas, ou mesmo em mensagens de erro detalhadas, pode fornecer pistas valiosas para um atacante sobre a arquitetura interna do sistema.

🎯 **Lembre-se:** É como se o bibliotecário aceitasse qualquer pedido, mesmo que mal formulado ou com intenções maliciosas, sem verificar a identidade do solicitante ou a validade do pedido. Cada interação, cada dado trocado, é um potencial ponto de exploração se não for devidamente protegido e validado.

O Papel dos Proxies: Intermediários Estratégicos

No fluxo de comunicação entre cliente e servidor, nem sempre a conexão é direta. Muitas vezes, um intermediário entra em cena: o **proxy**. Pense no proxy como um assistente pessoal que fica entre você (o cliente) e o mundo exterior (o servidor). Em vez de você enviar suas requisições diretamente, você as envia para o assistente, que as retransmite. A resposta do servidor também passa pelo assistente antes de chegar a você.

Proxy Forward

Um **proxy forward** (ou proxy de encaminhamento) é usado por clientes para acessar a internet. Ele pode ser configurado em uma rede corporativa para:

- Controlar o acesso a sites
- Filtrar conteúdo
- Armazenar em cache páginas frequentemente acessadas
- Acelerar a navegação

Proxy Reverso

Já um **proxy reverso** fica na frente de um ou mais servidores web. Ele recebe as requisições dos clientes e as encaminha para o servidor apropriado, agindo como:

- Balanceador de carga
- Cache de conteúdo
- Camada de segurança
- Proteção contra ataques

🔐 **Perspectiva de Segurança:** Do ponto de vista da segurança, os proxies são ferramentas poderosas. Eles podem inspecionar, modificar e bloquear tráfego malicioso antes que ele chegue ao servidor ou ao cliente. Um atacante, por outro lado, pode usar um proxy para anonimizar sua origem ou para manipular requisições e respostas em seu próprio benefício, testando as defesas de uma aplicação. Compreender o funcionamento dos proxies é essencial tanto para defender quanto para atacar.

Firewalls: A Primeira Linha de Defesa da Rede

Se o proxy é um assistente, o **firewall** é o porteiro rigoroso de um edifício. Sua função principal é monitorar e controlar o tráfego de rede de entrada e saída com base em regras de segurança predefinidas. Ele atua como uma barreira entre uma rede interna confiável e redes externas não confiáveis, como a internet. É a primeira linha de defesa, decidindo quem pode entrar e quem pode sair.

Firewalls de Pacotes

Inspecionam cabeçalhos de pacotes de dados, verificando endereços IP de origem e destino, portas e protocolos. São rápidos, mas não entendem o contexto da comunicação.

Firewalls de Estado

Mais inteligentes, pois acompanham o estado das conexões ativas, permitindo que o tráfego de resposta de uma conexão legítima passe, mesmo que a porta de destino esteja fechada para novas conexões.

Importância para Aplicações Web

Para a segurança de aplicações web, o firewall é crucial para proteger a infraestrutura subjacente. Ele pode:

- Bloquear tentativas de acesso a portas não utilizadas
- Impedir ataques de negação de serviço (DoS) básicos
- Isolar segmentos da rede

> ⚠️ **Limitação Importante:** No entanto, um firewall tradicional não entende o "idioma" das aplicações web (HTTP/HTTPS) em um nível profundo. Ele pode permitir o tráfego na porta 80 ou 443, mas não consegue discernir se o conteúdo desse tráfego é uma requisição legítima ou um ataque de injeção SQL. É aí que entram os WAFs.

WAFs (Web Application Firewalls)

A Guarda Costeira das Aplicações

Enquanto um firewall de rede é como um porteiro que verifica a identidade de quem entra no prédio, um **WAF (Web Application Firewall)** é como um segurança especializado que inspeciona o conteúdo das malas e mochilas de cada pessoa que tenta acessar uma área restrita. Ele opera na camada de aplicação (camada 7 do modelo OSI), entendendo profundamente o protocolo HTTP/HTTPS e o conteúdo das requisições e respostas.

Proteção Especializada

O WAF é projetado especificamente para proteger aplicações web contra uma variedade de ataques, incluindo aqueles listados no OWASP Top 10.

Análise Profunda

Ele pode detectar e bloquear ataques como SQL Injection, Cross-Site Scripting (XSS), inclusão de arquivos, manipulação de sessão e muitos outros.

Detecção Inteligente

Faz isso analisando padrões de tráfego, assinaturas de ataques conhecidos e comportamentos anômalos.

💡 **Importância Crescente:** A importância dos WAFs cresceu exponencialmente com a complexidade das aplicações web. Eles atuam como uma camada de segurança adicional, filtrando o tráfego malicioso antes que ele chegue à aplicação propriamente dita. Embora não sejam uma solução mágica para todos os problemas de segurança (a aplicação ainda precisa ser bem codificada), eles fornecem uma defesa robusta contra muitas ameaças comuns, comprando tempo e reduzindo a superfície de ataque.

OWASP Top 10: O Mapa das Vulnerabilidades Mais Críticas

2021 e Tendências 2024

A **OWASP (Open Web Application Security Project)** é uma comunidade global sem fins lucrativos dedicada a melhorar a segurança de software. Sua lista **OWASP Top 10** é um documento de referência crucial, atualizado periodicamente, que destaca as dez vulnerabilidades de segurança mais críticas e prevalentes em aplicações web. É como um boletim meteorológico para desenvolvedores e profissionais de segurança, alertando sobre as tempestades mais prováveis. A versão de 2021 trouxe atualizações significativas, e as tendências para 2024 continuam a evoluir.

A04:2021 – Insecure Design

Design Inseguro

Esta categoria enfatiza que a segurança deve ser pensada desde as fases iniciais do projeto, não apenas como um "patch" no final. Falhas de design podem levar a vulnerabilidades sistêmicas que são difíceis de corrigir posteriormente. É como construir uma casa com uma planta falha: não importa quão bem você reforce as paredes, a fundação está comprometida.

A08:2021 – Software and Data Integrity Failures

Falhas de Integridade

Esta categoria aborda vulnerabilidades relacionadas à confiança em atualizações de software, dados críticos e pipelines de CI/CD sem verificações de integridade. Um atacante pode injetar código malicioso em uma atualização de software ou manipular dados críticos se não houver mecanismos robustos para garantir sua integridade. Isso ressalta a importância de verificar a autenticidade de tudo que entra e sai do seu ambiente de desenvolvimento e produção.

📊 **Contexto Importante:** A lista de 2021 introduziu novas categorias que refletem a mudança no cenário de ameaças, reconhecendo que a segurança moderna vai além de simplesmente corrigir bugs de código.

Conectando o OWASP Top 10 à Arquitetura Web

As vulnerabilidades listadas no OWASP Top 10 não são conceitos abstratos; elas se manifestam diretamente nos pontos de ataque que discutimos na arquitetura web.

A01:2021 – Broken Access Control

Quebra de Controle de Acesso, que será o tema da nossa próxima aula, ocorre quando as restrições sobre o que usuários autenticados podem fazer não são aplicadas corretamente. Isso se relaciona diretamente com a fase de "autorização" no modelo Cliente-Servidor, onde o servidor falha em verificar se o cliente tem permissão para acessar um recurso específico.

A03:2021 – Injection

Como SQL Injection ou Command Injection, explora a falta de validação de entrada de dados no servidor. Um atacante envia dados maliciosos no corpo ou nos parâmetros de uma requisição HTTP, e o servidor os executa sem sanitização adequada, comprometendo o banco de dados ou o sistema operacional.

Arquiteturas Complexas

As tendências para 2024 continuam a focar na complexidade crescente das aplicações. Com a proliferação de microsserviços e APIs, a superfície de ataque se expande. Vulnerabilidades como A05:2021 – Security Misconfiguration e A06:2021 – Vulnerable and Outdated Components se tornam ainda mais críticas.

🎯 **Conclusão Fundamental:** Isso reforça a ideia de que cada ponto onde o cliente interage com o servidor é um potencial vetor de ataque. Um único componente mal configurado ou desatualizado em uma arquitetura complexa pode comprometer todo o sistema. A compreensão da arquitetura é, portanto, a base para aplicar as mitigações do OWASP Top 10 de forma eficaz.

Segurança em APIs: O Novo Campo de Batalha

REST e GraphQL

Com a crescente adoção de arquiteturas baseadas em microsserviços e a necessidade de aplicações se comunicarem de forma programática, as **APIs (Application Programming Interfaces)** se tornaram a espinha dorsal da web moderna. Elas são, essencialmente, os "contratos" que definem como diferentes softwares podem interagir. As duas arquiteturas de API mais populares são **REST (Representational State Transfer)** e **GraphQL**.

APIs REST

APIs REST são baseadas no protocolo HTTP e utilizam os métodos HTTP (GET, POST, PUT, DELETE) para realizar operações em recursos. Elas são amplamente utilizadas e seguem um modelo de comunicação stateless (sem estado), onde cada requisição do cliente para o servidor contém todas as informações necessárias para entender a requisição.

Focos de Segurança:

- Autenticação (tokens JWT, OAuth)
- Autorização (controle de acesso baseado em papéis)
- Validação de entrada
- Proteção contra ataques comuns

GraphQL

GraphQL é uma linguagem de consulta e manipulação de dados para APIs, desenvolvida pelo Facebook. Diferente do REST, onde o cliente acessa endpoints fixos, no GraphQL o cliente envia uma única requisição para um único endpoint, especificando exatamente os dados que precisa.

Desafios de Segurança:

- Complexidade das consultas (DoS)
- Exposição excessiva de dados
- Validação rigorosa de consultas
- Controle de profundidade de queries

> 🚀 **Tendência 2025:** Isso oferece flexibilidade, mas também introduz novos desafios de segurança. A segurança em APIs, seja REST ou GraphQL, é um campo de batalha crucial para 2025 e além.

Ferramentas Essenciais para Análise de Tráfego

DevTools do Navegador

Para entender a arquitetura web e identificar pontos de ataque, precisamos de ferramentas que nos permitam "espiar" o que está acontecendo por trás das cenas. As **DevTools (Ferramentas de Desenvolvedor)**, presentes em todos os navegadores modernos (Chrome, Firefox, Edge, Safari), são um excelente ponto de partida. Elas são como um laboratório de análise embutido no seu próprio navegador, permitindo que você inspecione e manipule a comunicação web.

01

Acessar as DevTools

Ao abrir as DevTools (geralmente com F12 ou botão direito > Inspecionar), você encontrará várias abas úteis.

02

Aba Network (Rede)

A aba mais relevante para a análise de tráfego. Nela, você pode ver todas as requisições HTTP/HTTPS que seu navegador faz e as respostas que recebe.

03

Inspecionar Detalhes

Para cada requisição, você pode inspecionar cabeçalhos, payload/corpo, status e timing.

O que você pode analisar:

Cabeçalhos

Ver os cabeçalhos da requisição e da resposta, incluindo cookies, User-Agent, Content-Type, etc.

Payload/Corpo

Analisar os dados enviados na requisição (se for POST) e os dados recebidos na resposta.

Status

Verificar o código de status HTTP da resposta.

Timing

Entender o tempo que cada requisição levou.

💻 **Prática Essencial:** Usar as DevTools é fundamental para qualquer profissional de segurança. Você pode simular requisições, modificar cabeçalhos (em alguns navegadores com extensões), e observar como a aplicação reage. É uma forma prática e acessível de começar a entender o fluxo de dados e a identificar potenciais vulnerabilidades em tempo real, sem a necessidade de software adicional.

Ferramentas Essenciais para Análise de Tráfego

Burp Suite Community

Enquanto as DevTools do navegador são ótimas para uma análise inicial, para uma inspeção mais profunda e manipulação de tráfego, precisamos de uma ferramenta mais robusta: o **Burp Suite Community Edition**. Esta é a versão gratuita de uma das suítes de segurança web mais populares e poderosas do mercado, essencial para testes de penetração e análise de vulnerabilidades.

> 🔍 **Como Funciona:** O Burp Suite atua como um **proxy interceptador** entre o seu navegador e a aplicação web. Isso significa que todas as requisições e respostas HTTP/HTTPS que passam entre eles podem ser capturadas, inspecionadas e, crucialmente, modificadas antes de serem enviadas ao seu destino. Imagine que você está enviando uma carta, mas ela passa por um "posto de controle" onde você pode abri-la, mudar o conteúdo e depois selá-la novamente antes que ela chegue ao destinatário.

Funcionalidades Principais:

Proxy

Intercepta e permite modificar requisições e respostas em tempo real.

Repeater

Permite enviar requisições repetidamente, modificando parâmetros para testar diferentes cenários de ataque.

Intruder

Automatiza o envio de múltiplas requisições com payloads variados para encontrar vulnerabilidades (como força bruta ou injeção).

Decoder

Codifica e decodifica dados em vários formatos.

> 🎓 **Divisor de Águas:** Dominar o Burp Suite é um divisor de águas para quem quer se aprofundar em segurança de aplicações web. Ele permite que você vá além da observação passiva e comece a interagir ativamente com a aplicação de uma forma que revela suas fraquezas, simulando o comportamento de um atacante.

Consolidação e Próximos Passos

Nesta aula, desvendamos a complexa, mas fascinante, arquitetura da web. Começamos com os fundamentos do HTTP/HTTPS, entendemos a dinâmica entre clientes e servidores, e mergulhamos na anatomia detalhada das requisições e respostas. Exploramos os pontos de vulnerabilidade inerentes a esse modelo e a importância de intermediários como proxies, firewalls e WAFs na defesa. Finalmente, conectamos esses conceitos com as vulnerabilidades críticas do OWASP Top 10 e a crescente importância da segurança em APIs, além de apresentar ferramentas essenciais como DevTools e Burp Suite.

Em prática:

- Sempre valide as entradas do usuário no lado do servidor.
- Implemente autenticação e autorização robustas para cada recurso.
- Monitore os logs do servidor para detectar atividades anômalas.
- Mantenha-se atualizado com as últimas recomendações do OWASP.
- Utilize ferramentas como Burp Suite para testar suas próprias aplicações.

Autoavaliação

1. **Qual a principal diferença de segurança entre HTTP e HTTPS?**
   a) HTTPS é mais rápido que HTTP.
   b) HTTPS criptografa a comunicação, protegendo contra interceptação.
   c) HTTP usa portas diferentes de HTTPS.
   d) HTTPS é exclusivo para servidores Linux.
2. **Um código de status HTTP 403 Forbidden indica que:**
   a) A requisição foi bem-sucedida.
   b) O servidor não pôde encontrar o recurso solicitado.
   c) O cliente não tem permissão para acessar o recurso.
   d) Houve um erro interno no servidor.
3. **Qual das seguintes ferramentas atua como um proxy interceptador, permitindo a modificação de requisições e respostas HTTP/HTTPS?**
   a) Firewall de rede
   b) WAF (Web Application Firewall)
   c) Burp Suite Community
   d) DevTools do navegador (aba Console)
4. **A categoria "Insecure Design" (A04:2021) do OWASP Top 10 enfatiza a importância de:**
   a) Realizar testes de penetração apenas no final do ciclo de desenvolvimento.
   b) Garantir que a segurança seja considerada desde as fases iniciais do projeto.
   c) Utilizar apenas componentes de software de código aberto.
   d) Focar exclusivamente na correção de vulnerabilidades de injeção.

Gabarito: 1. b) | 2. c) | 3. c) | 4. b)

Questão Dissertativa

> Explique como a falta de validação de entrada de dados no servidor pode levar a diferentes tipos de ataques, citando exemplos e relacionando-os com a arquitetura Cliente-Servidor.

Próxima Aula

Na Aula 3, mergulharemos em uma das vulnerabilidades mais críticas e prevalentes: **A01:2021 - Quebra de Controle de Acesso**. Entenderemos como ela funciona, seus impactos e as melhores práticas para preveni-la.

Recursos Adicionais

- **OWASP Top 10 (site oficial):** Para aprofundar-se nas vulnerabilidades e suas mitigações.
- **Documentação do Burp Suite:** Para explorar as funcionalidades da ferramenta em detalhes.
- **MDN Web Docs (HTTP):** Para uma referência técnica completa sobre o protocolo HTTP.

NOTA IMPORTANTE: As informações regulatórias/legais/técnicas desta aula estão atualizadas até 2025. Consulte sempre fontes oficiais para verificar alterações.