TRABALHO SEGURANÇA DA INFORMAÇÃO

EM APLICAÇÕES WEB MODELO PHP.

1. Política de Segurança de Conteúdo (CSP) Ausente ou Insegura.

O que é a Política de Segurança de Conteúdo (CSP)?

A CSP é um padrão de segurança da web que define uma lista de fontes confiáveis de conteúdo que um navegador pode carregar em uma página da web. Isso inclui scripts, estilos, imagens, fontes e outros recursos. Ao restringir as fontes de conteúdo, a CSP ajuda a mitigar ataques como o Cross-Site Scripting (XSS), que exploram vulnerabilidades para injetar scripts maliciosos em páginas da web.

Riscos da Ausência ou Implementação Insegura da CSP:

Ataques XSS (Cross-Site Scripting):

- Sem a CSP, uma aplicação fica altamente vulnerável a ataques XSS, nos quais invasores podem injetar scripts maliciosos em páginas da web visualizadas por outros usuários.
- Esses scripts podem roubar dados confidenciais, sequestrar sessões de usuários, adulterar o conteúdo da página ou redirecionar os usuários para sites maliciosos.

Injeção de Código Malicioso:

- A falta da CSP permite que invasores injetem outros tipos de código malicioso, como iframes maliciosos ou conteúdo externo não confiável, comprometendo a integridade da aplicação e a segurança dos usuários.

Roubo de Dados:

- Ataques bem-sucedidos podem resultar no roubo de dados confidenciais, como credenciais de login, informações de cartão de crédito ou dados pessoais dos usuários.

Como corrigir?

Análise das diretivas:

- Examine as diretivas CSP existentes, como script-src, style-src, img-src e default-src, para identificar possíveis brechas de segurança.
- Procure por diretivas permissivas, como unsafe-inline, unsafe-eval e *, que podem enfraquecer a CSP.

Diretivas restritivas:

- Substitua diretivas permissivas por diretivas mais restritivas, como self, https: e listas de fontes específicas.

- Evite o uso de unsafe-inline e unsafe-eval, que permitem a execução de scripts inline e avaliações de código dinâmico.

Teste da CSP:

- Utilize ferramentas de análise de segurança para testar a CSP em um ambiente de teste.
- Monitore os relatórios de violação da CSP para identificar possíveis problemas e ajustar a política.

2. Cookies sem HttpOnly, Secure e SameSite.

Cookies sem os atributos HttpOnly, Secure e SameSite representam uma falha de segurança significativa em aplicações web. Cada um desses atributos desempenha um papel crucial na proteção de cookies contra diferentes tipos de ataques.

O que são os atributos HttpOnly, Secure e SameSite?

HttpOnly:

- Este atributo impede que scripts do lado do cliente (como JavaScript) acessem o cookie.
- Sua função principal é mitigar ataques de Cross-Site Scripting (XSS), que podem roubar cookies se não forem protegidos.

Secure:

- Este atributo garante que o cookie seja transmitido apenas por conexões HTTPS criptografadas.
- Ele protege o cookie contra interceptação em trânsito, impedindo que invasores capturem informações confidenciais.

SameSite:

- Este atributo controla quando o cookie é enviado em solicitações de origem cruzada.
- Ele ajuda a prevenir ataques de Cross-Site Request Forgery (CSRF), que exploram a confiança de um navegador em um site para realizar ações não autorizadas.
- Os valores comuns para SameSite são "Strict", "Lax" e "None".

Riscos de cookies sem esses atributos:

Roubo de cookies (XSS):

- Sem HttpOnly, invasores podem usar scripts maliciosos para roubar cookies, comprometendo sessões de usuários e dados confidenciais.

Interceptação de cookies (Man-in-the-Middle):

- Sem Secure, cookies podem ser interceptados em conexões não criptografadas, expondo informações confidenciais a invasores.

Ataques CSRF:

- Sem SameSite, invasores podem explorar a confiança do navegador para realizar ações não autorizadas em nome do usuário, como alterar senhas ou realizar transações.

Como corrigir?

- **Sempre use HTTPS**: Se você está usando cookies com Secure, é essencial garantir que sua aplicação esteja usando HTTPS. Isso protegerá não apenas os cookies, mas também todos os dados trafegados entre o cliente e o servidor.

- **Evite armazenar dados sensíveis em cookies**: Cookies devem ser usados para manter informações não sensíveis, como identificadores de sessão. Nunca armazene senhas, tokens de autenticação ou outros dados críticos em cookies.

- **Monitore a expiração de cookies**: Defina a expiração dos cookies com o tempo adequado, e garanta que os cookies sejam excluídos quando o usuário sair (se for o caso).

- **Reveja as configurações periodicamente**: Certifique-se de que suas configurações de cookies estejam sempre atualizadas e atendam às melhores práticas de segurança.

3. Vazamento de Informações no Cabeçalho HTTP (X-Powered-By, Server).

O que é vazamento de Informações no Cabeçalho HTTP?

O vazamento de informações no cabeçalho HTTP, especialmente nos campos como X-Powered-By e Server, refere-se à exposição de detalhes sobre as tecnologias usadas pelo servidor web. Esses cabeçalhos podem fornecer pistas valiosas para um atacante sobre como direcionar seus ataques.

Riscos de vazamentos:

1. Reconhecimento e engenharia de ataques

- Atacantes usam ferramentas de varredura (como Nmap, Wappalyzer, Nikto, etc.) para descobrir quais tecnologias estão rodando.

- Com essa informação, eles sabem exatamente que tipos de ataques tentar — por exemplo:
    - SQL Injection para aplicações em PHP com MySQL.
    - Exploração de falhas conhecidas em versões antigas de frameworks como Laravel, ASP.NET, etc.

2. Exploração de vulnerabilidades conhecidas (CVEs)

- Se o cabeçalho indicar uma versão específica (ex: PHP/7.2.0), um atacante pode buscar por CVEs (Common Vulnerabilities and Exposures) conhecidas dessa versão.
- Quanto mais específico o cabeçalho, maior o risco de uma exploração direta.

3. Automação de ataques

- Bots de ataque na internet escaneiam e atacam automaticamente servidores com base nesses cabeçalhos.
- Um servidor que diga "Powered-By: WordPress" pode receber ataques automatizados explorando falhas comuns do WordPress, mesmo que ele esteja atualizado.

4. Phishing ou engenharia social

- Informações reveladas nesses cabeçalhos podem ser usadas para enganar desenvolvedores ou administradores com e-mails falsos:
    - “Seu servidor Apache 2.4.41 tem uma vulnerabilidade crítica. Baixe esse patch agora.”

5. Mapeamento de infraestrutura

- Para ataques mais sofisticados (ex: APTs ou pentests corporativos), saber a tecnologia usada ajuda a mapear toda a pilha da aplicação e até entender como o time de desenvolvimento trabalha.

Como corrigir?

- *Remover ou ocultar cabeçalhos como Server e X-Powered-By dificulta a vida de atacantes.*

- *Isso não substitui outras medidas de segurança, como manter o software atualizado, usar WAF, validar entradas, etc.*
- *Mas ajuda muito a evitar ataques automáticos e direcionados, que dependem de saber o que está rodando no servidor.*

4. Falta de Tokens Anti-CSRF.

O que é Falta de Tokens Anti-CSRF?

É uma falha de segurança que acontece quando um site não utiliza tokens de proteção contra ataques CSRF (Cross-Site Request Forgery — falsificação de requisições entre sites).

Essa vulnerabilidade permite que um atacante engane um usuário autenticado para que ele execute ações indesejadas em uma aplicação sem perceber.

Riscos de Falta de Tokens Anti-CSRF:

1. Execução de ações maliciosas em nome do usuário

- Um atacante pode fazer com que um usuário autenticado **realize ações sem perceber**, como:
    - Transferir dinheiro.
    - Alterar seu e-mail ou senha.
    - Excluir dados ou usuários.
    - Alterar configurações administrativas.

2. Comprometimento de contas

- Se um atacante induzir o usuário a executar uma requisição maliciosa (com clique ou até sem interação), pode **tomar controle parcial ou total da conta**.
- Exemplo: troca de senha via CSRF = sequestro de conta.

3. Escalada de privilégios

- Em aplicações com painéis de admin, CSRF pode permitir que um usuário comum execute ações de administrador **se a checagem de permissões for mal feita**.

4. Exclusão ou alteração de dados críticos

- Imagine um sistema de gestão (ERP, CRM, etc.) onde um clique malicioso pode deletar uma tabela inteira ou alterar informações financeiras.

5. Quebra da confiança do usuário

- O ataque ocorre **sem que o usuário saiba**, o que compromete a confiança na aplicação.

Como corrigir?

- Implemente tokens CSRF nos formulários HTML.
- Use bibliotecas ou frameworks que já ofereçam proteção.
- Django, Laravel, Rails, Spring, ASP.NET, etc. já têm isso embutido.
- Valide os tokens no servidor.
- Use o método POST (ou PUT/DELETE) para ações críticas — nunca GET.

5. Ausência de X-Frame-Options (Clickjacking).

É uma falha de segurança que acontece quando o site não impede que seu conteúdo seja carregado dentro de um <iframe> em outro site. Isso abre espaço para um ataque chamado Clickjacking**.**

O Que é Clickjacking?

"Clickjacking" é quando um atacante esconde seu site malicioso por trás do seu site legítimo (ou vice-versa) dentro de um <iframe>, para enganar o usuário a clicar em algo que parece inofensivo, mas que realiza uma ação perigosa.

Exemplo:

- O atacante carrega sua página dentro de um <iframe> invisível ou com opacidade 0.
- Sobrepõe esse iframe com botões falsos.
- O usuário tenta clicar em um botão inocente (tipo "Play", "Fechar", etc.), mas na verdade está clicando em algo da sua aplicação (tipo "Excluir conta", "Transferir dinheiro", etc.).

Riscos do Clickjacking:

1. **Enganar o usuário para clicar em algo invisível**

Esse é o **clickjacking clássico**. O atacante cria uma página falsa que carrega seu site real em segundo plano, invisível, com botões reais.

- O usuário pensa que está clicando em algo inofensivo (tipo "Play", "Curtir", "Fechar").

- Mas na verdade está clicando em um botão da sua aplicação — como "Excluir conta", "Alterar senha", "Confirmar compra", etc.

O clique é **real**, e **executa a ação na sua aplicação**.

2. Sequestro de conta ou mudança de configurações

Se a vítima estiver **logada** na sua aplicação:

- O atacante pode enganar ela para **mudar seu e-mail**, **trocar a senha**, ou **desativar a conta**.
- Tudo isso pode acontecer **sem a vítima perceber**.

3. Perda de dados ou ações não autorizadas

Se sua aplicação tem botões sensíveis (como excluir, atualizar, transferir valores ou aprovar ações):

- O clickjacking pode causar **perda de dados** ou **ações indesejadas**, feitas por usuários reais, mas sem intenção.

---

4. Violações de segurança e privacidade

Clickjacking pode ser usado para:

- Acessar informações privadas do usuário.
- Espionar interações do usuário com a interface.
- Forçar interações com publicidade ou aceitar termos de forma falsa.

5. Comprometimento da integridade da sua aplicação

Mesmo que os dados não sejam roubados, **a confiança no seu sistema é quebrada**:

- O usuário pode não entender o que aconteceu.
- Pode achar que o problema é no seu site.
- Isso prejudica sua reputação e pode causar reclamações ou processos, especialmente se envolver dados sensíveis.

Como corrigir?

- Escolha uma política de iframe adequada (DENY ou SAMEORIGIN);
- Adicione o cabeçalho no seu servidor ou aplicação;
- Teste se seu site está protegido, tentando carregá-lo dentro de um iframe em outro domínio.