

# Curso de Extensão em Tecnologias Microsoft

## INF0998 Programação segura (segurança de software)

### Atividades práticas de testes de segurança automatizados com OWASP ZAP

Testes de segurança manuais são demorados e, quando realizados em grande quantidade, podem causar atrasos em projetos de desenvolvimento de software, ou mesmo não estarem disponíveis em tempo hábil para que correções sejam efetuadas.

Testes de segurança auxiliados por ferramentas aceleram a geração de resultados que podem ser tratados rapidamente. Porém, vale lembrar que ferramentas automáticas de verificações de segurança não substituem o testador experiente.

Esta aula usa a ferramenta de testes de intrusão **OWASP Zed Attack Proxy (ZAP)** para auxiliar a descoberta e a exploração de vulnerabilidades presentes nas aplicações estudadas nas aulas anteriores.

# OWASP ZAP

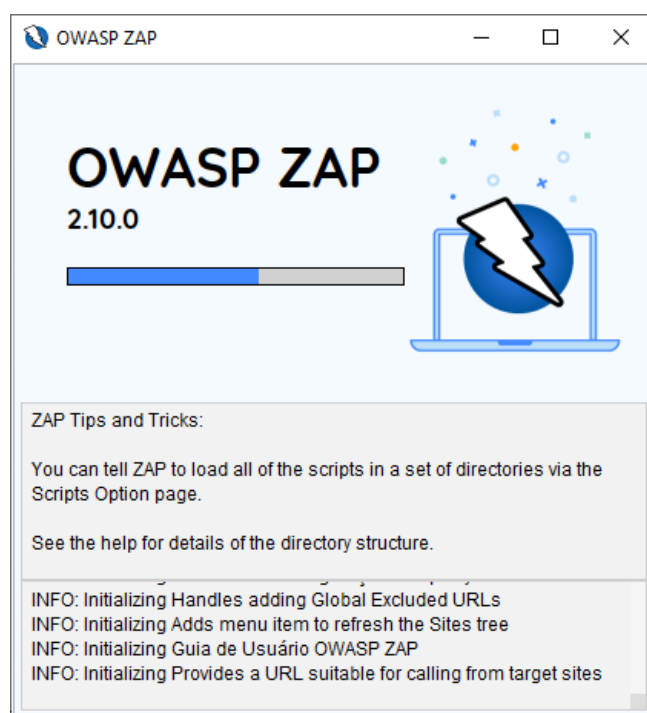
## Apresentando o OWASP ZAP

- Site do ZAP (Zed Attack Proxy) no OWASP no <https://owasp.org/www-project-zap/>
- Site oficial do OWASP ZAP <https://www.zaproxy.org/>.
- Download <https://www.zaproxy.org/download/>.
- Repositório de Código fonte <https://www.zaproxy.org/download/>.
- Tutoriais em vídeos curtos <https://www.zaproxy.org/zap-in-ten/>.
- Este treinamento usa a versão 2.10.0.

## Início e configurações iniciais

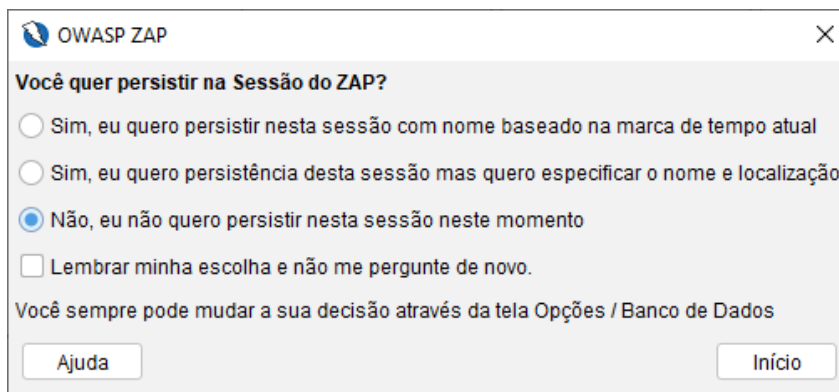
O ZAP tem uma CLI e uma GUI.

A GUI do ZAP inicia com uma janela de abertura, progresso e anúncios (para a versão 2.10.0).



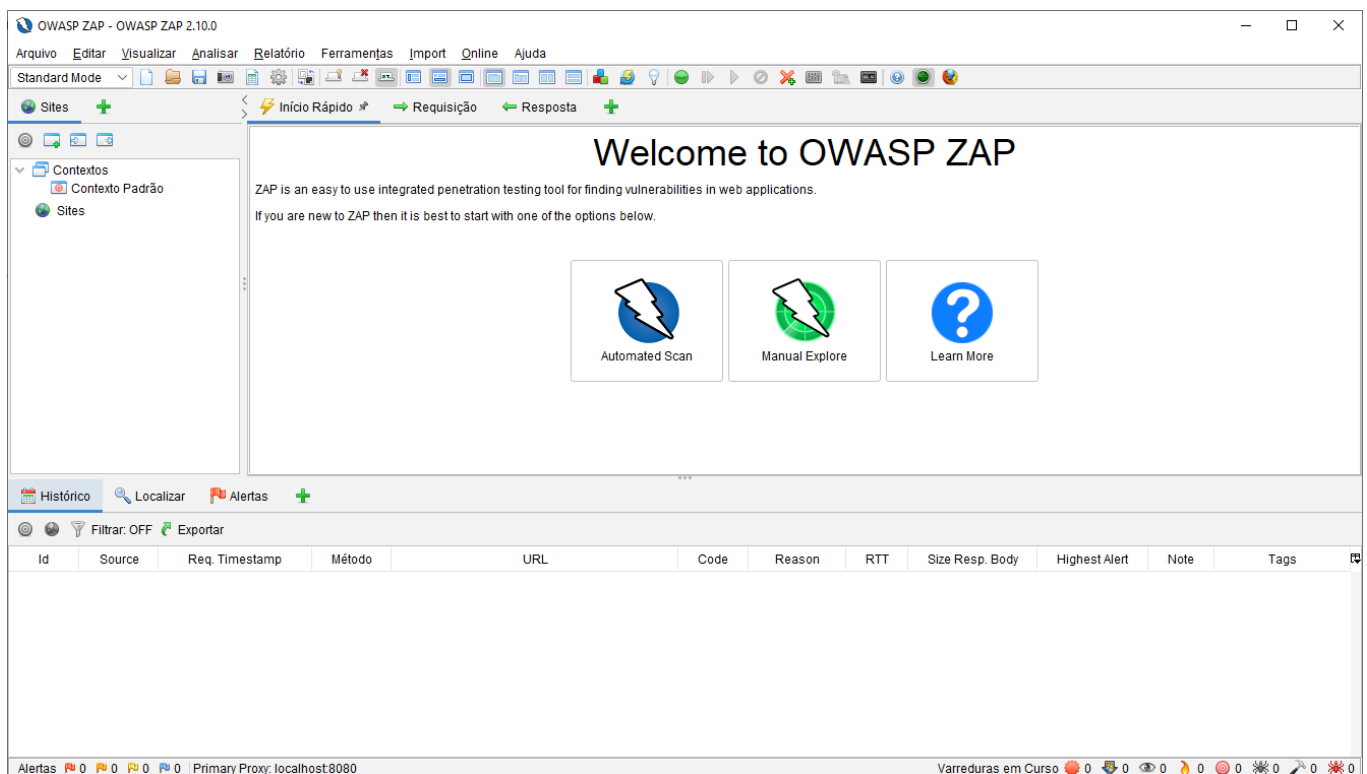
Em seguida, o ZAP solicita que o usuário escolha como deseja persistir a sessão.

- Neste treinamento, sempre escolha a opção **Não**.
- Se quiser, marque a caixa para lembrar a escolha..
- Clicar no botão de **Início**.



A tela inicial (início rápido) do ZAP oferece três maneiras de iniciar:

- Varredura automatizada (Automated Scan).
- Exploração manual (Manual Explore).
- Documentação (Learn more).

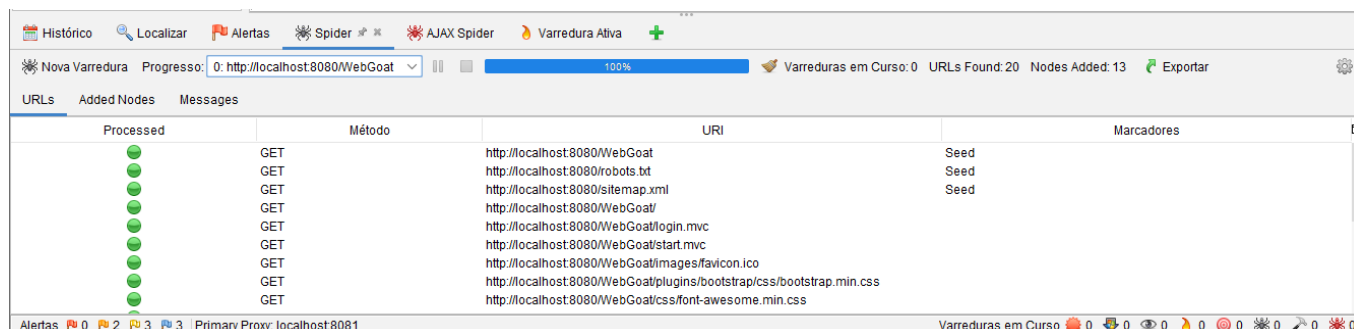


As funcionalidades Automated Scan e Manual Explore, assim como todas as explorações de vulnerabilidades desta aula, serão demonstradas sobre as aplicações vulneráveis para aprendizado Juice Shop e WebGoat 7.1 (Ambas foram apresentadas anteriormente).

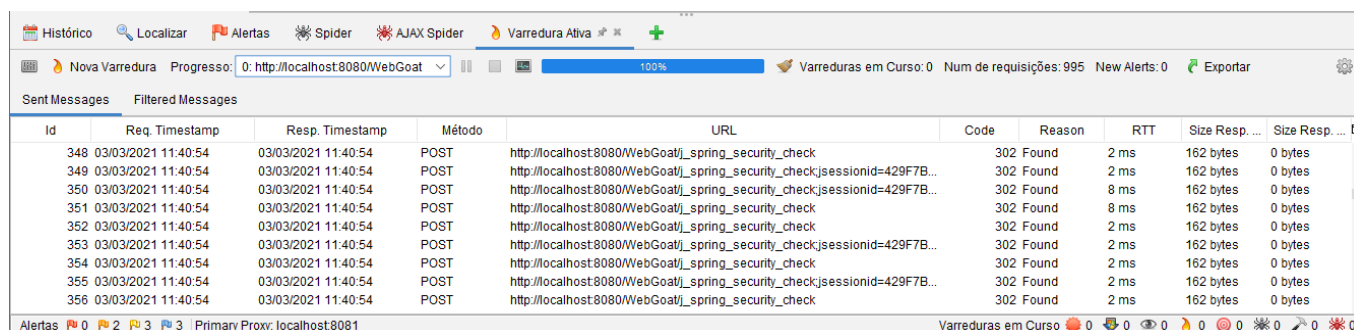
- As aplicações vulneráveis devem ser iniciadas antes de prosseguir com os exercícios.

## Automated Scan

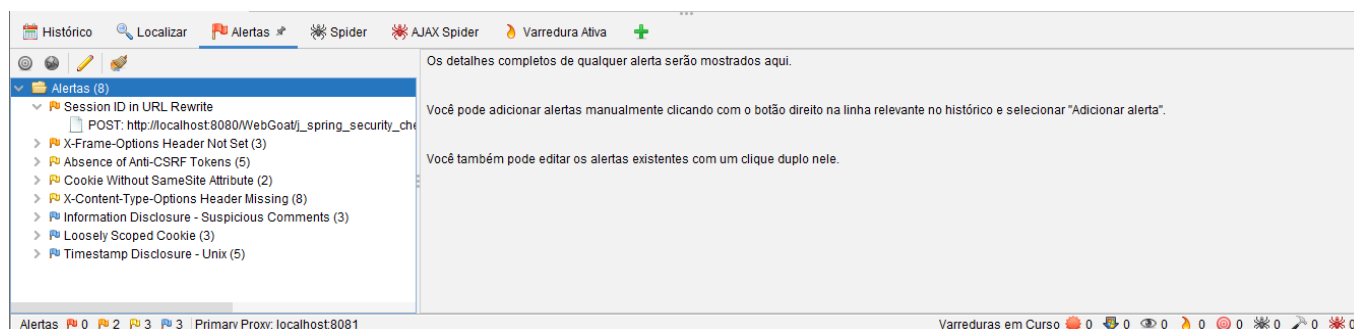
- Clicar no botão de Automated Scan.
- Na Janela Automated Scan.
  - Fornecer a URL que será varrida, p.ex., <http://localhost:8080/WebGoat>
  - Usar **Spider**, **Ajax Spider** com **Firefox headless**.
  - Clicar no botão **Attack/Ataque**.
- O progresso do Scan é mostrado na barra de progresso do Spider/AjaxSpider.



- Os ataques automáticos são mostrados na aba Varredura Ativa.



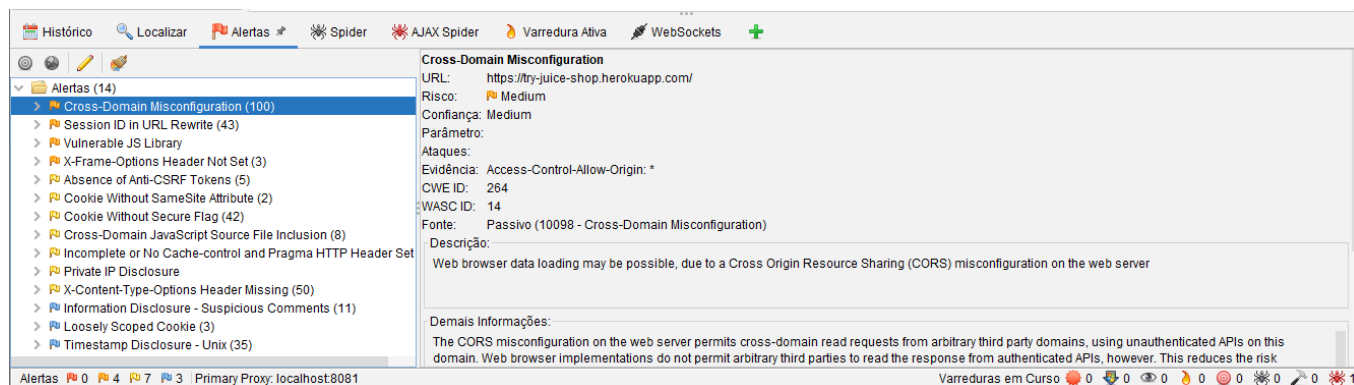
- O resultado do ataque é mostrado na aba Alertas.



- A varredura detectou problemas em 3 níveis de alertas: Médio, baixo e informativo (vazamento de informação).

O teste automático de outra aplicação mostra resultado diferente.

- Na Janela Automated Scan.
  - Fornecer a URL que será varrida, p.ex., <https://try-juice-shop.herokuapp.com/#/>
  - Usar **Spider**, **Ajax Spider** com **Firefox headless**.
  - Clicar no botão **Ataque**.



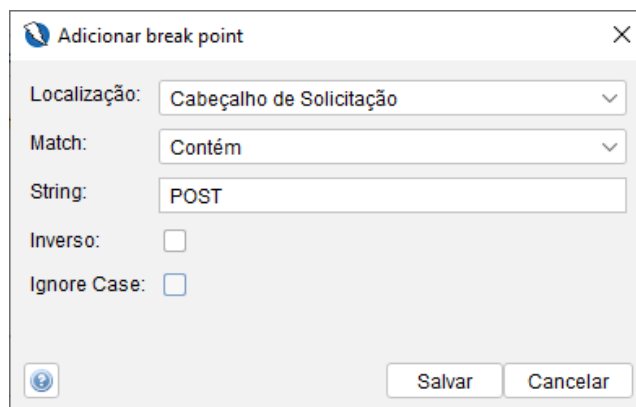
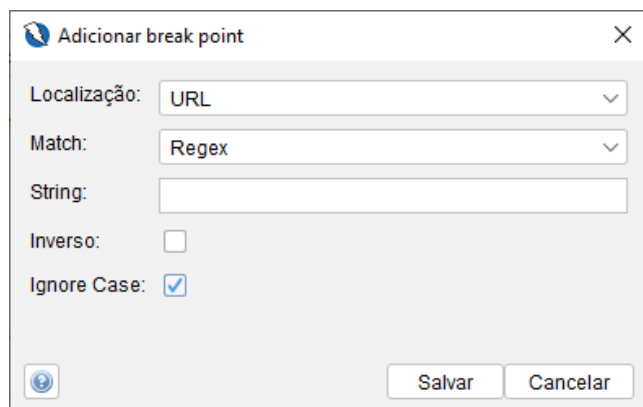
## Manual Explore



- Na Janela Início Rápido, clique no botão de **Manual Explore**.
- Na janela **Manual Explore**.
  - Fornecer a URL da aplicação que será explorada, p.ex., <http://localhost:8080/WebGoat>.
  - Desmarcar a caixa Enable HUD (HUD é explicado adiante).
  - Escolher o browser a ser aberto: Firefox ou Chrome.
  - Ativar o botão Launch Browser.

## Criação de Breakpoints e Filtros

Para a interceptação apenas das requisições úteis para o trabalho de teste.

Por exemplo, para criar um breakpoint para as requisições HTTP do tipo POST.

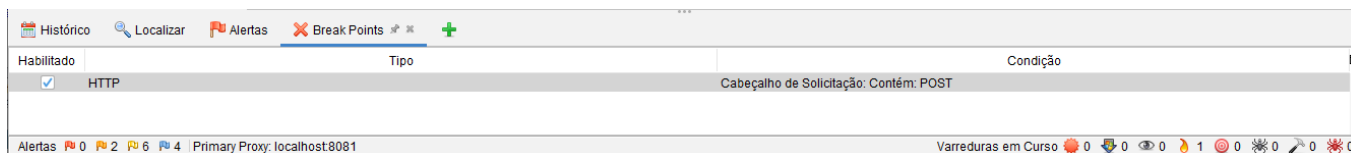


Clicar no botão  na barra de botões de controle de breakpoint do ZAP , a janela aparece:

Preencher os campos:

- **String:** POST
- **Localização:** Cabeçalho da Solicitação (Request Header)
- **Match:** Contém / Contain
- Clicar no botão **Salvar**.


Os breakpoints criados aparecem na barra e podem ser ativados/desativados pelo checkbox. Assim o controle manual de breakpoint (botões verde/vermelho) não é mais necessário.



Filtro de requisições. Na guia **Histórico**, clicar no botão de Filtro, inserir no campo URL Inc Regex a string `.*.WebGoat.*`. (incluindo os pontos) para mostrar apenas as requisições relacionadas ao WebGoat.

## Intercepção e modificação de requisições

Com o Breakpoint ativado no ZAP

- No WebGoat 7.1, ir para o menu **General** → **Http basics**.
- No campo de formulário **Enter your name**, digite a string “teste” e ative o botão **Go!**.
- O ZAP indicará que interceptou a requisição.
  - No corpo da requisição, modificar o valor do parâmetro como quiser.
  - Por exemplo, adicionando “123” ao final da string “teste”.
  - Clicar em um dos botões Play  para dar andamento à requisição.
  - (Lembrar de desativar o breakpoint!)

## Edição e reenvio de requisições


No Histórico de requisições do ZAP, selecionar a requisição mais recente (aquela da tarefa anterior).

- Clicar com botão direito do mouse, escolher a opção **Reenviar**.
- O ZAP abre outra janela **Editor Manual de Requisições** para edição e reenvio de requisição.
- Na Janela de edição de requisição
  - Na guia **Requisição** modificar o parâmetro no corpo.
  - Na guia Resposta, que está vazia ainda, clicar no botão **Enviar**.
    - A resposta é enviada
  - Lembrar de fechar o **Editor Manual de Requisições**.

## Opções de proxy

O ZAP pode servir de proxy intermediário entre o browser e a aplicação web.

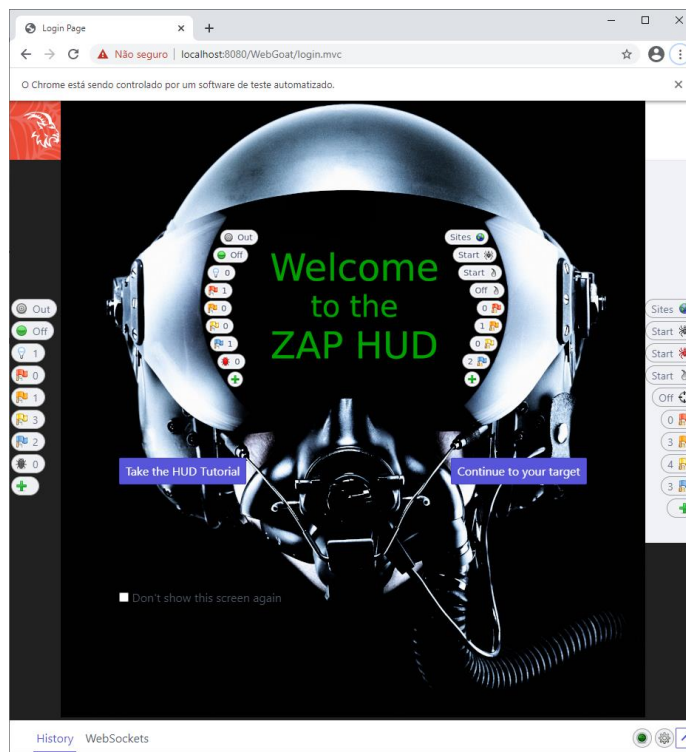
Há três maneiras de fazer isto:

- A maneira mais antiga de configurar manualmente o ZAP como proxy de aplicação no browser. Esta opção não será usada neste treinamento.
- Outra maneira é por meio da opção de início rápido Manual Explore, como visto anteriormente.
- A terceira maneira é com o botão de ativação do navegador . O ícone do browser abre um navegador intermediado e controlado remotamente pelo ZAP Proxy. Já o ícone verde de “mira em alvo” liga o HUD, explicado a seguir.

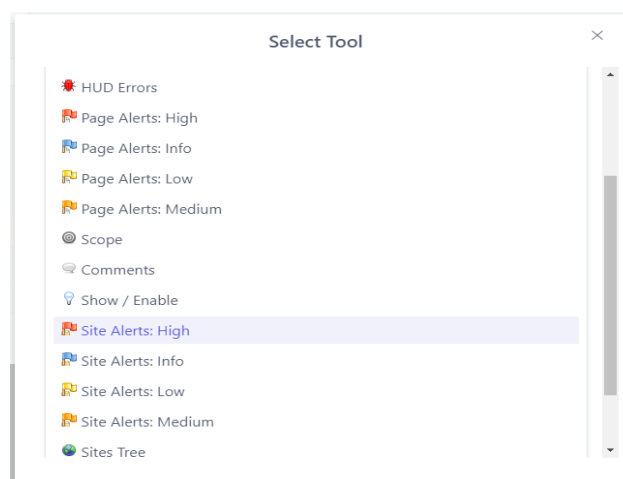
## HUD Heads-On-Display

HUD é uma GUI avançada que coloca no navegador os controles da interface do ZAP desktop. Com o HUD, o usuário não precisa deixar a tela no browser para inspecionar o ZAP, pois (quase) todos os controles, comandos e alertas mais usados estão no navegador, sobre a interface da aplicação.

- A metáfora dos controles em realidade aumentada presentes no visor do capacete do piloto.
- Ou, como eu gosto de dizer, o capacete do homem-de-ferro.



- O usuário pode fazer o tutorial ou partir para a exploração da aplicação.
- É importante passar o mouse por cada ícone do HUD para conhecer seus nomes.
- Os significados dos ícones são simples e autoexplicativos.
- **IMPORTANTE: Spider, Ajax Spider, Active Scan, Attack mode e Breakpoint estão no HUD.**
- A barra inferior no browser tem histórico de requisições, botão on/off do HUD e configurações.
- Os ícones de alertas são parecidos, mas têm diferenças, página e site (vide figura).



# Exploração de CSRF com OWASP ZAP

CSRF (Cross Site Request Forgery) é a falsificação de requisições válidas que são enviadas de maneira oculta (às escondidas) em nome de um usuário legitimamente autenticado (e em uma sessão ativa) em um site.

A requisição forjada reutiliza parâmetros válidos da requisição legítima.

O roteiro simula uma transferência bancária oculta.

- No WebGoat 7.1 via ZAP Proxy
  - Menu **Cross Site Scripting (XSS)** → **Cross Site Request Forgery (CSRF)**.
  - Digitar um título e uma mensagem no formulário indicado.
    - Por exemplo, Title = “Olá” e Message = “Tentativa”.
    - Clicar no botão **Submit**.
  - Rolar a página para baixo, clicar no link de mesmo nome da mensagem (Olá) para visualizar a mensagem.
- No ZAP, Histórico de navegação, identificar a requisição
  - GET `http://localhost:8080/WebGoat/attack?Screen=2078372&menu=900`
  - Guarda os dados `Screen=2078372` e `menu=900`.
- Voltar ao formulário e digitar um título e uma mensagem.
  - Title = “Test”
  - Mensagem  
`Hacked! <img  
src=http://localhost:8080/WebGoat/attack?Screen=2078372&menu=900&t  
ransferFunds=5000 width=1 height=1 />`
  - Clicar no botão **Submit**.
- Rolar a página para baixo, clicar no link de mesmo nome da mensagem (Test) para visualizar a mensagem.
- No ZAP, Histórico de navegação, verificar que a requisição oculta foi enviada às escondidas.

GET

`http://localhost:8080/WebGoat/attack?Screen=2078372&menu=900&transferFunds=5000`



## Exploração de tratamento incorreto de erro com OWASP ZAP

- No WebGoat 7.1 via ZAP Proxy
  - Menu ***Improper Error Handling*** → ***Fail Open Authentication Scheme***.
  - No formulário, digitar user/password administrativos
    - webgoat/webgoat
    - clicar no botão ***Login***.
  - Logou como admin? Fazer logout.
- No ZAP, ativar o breakpoint de interceptação de requisições.
- No WebGoat, fazer novamente o login webgoat/webgoat no mesmo formulário.
- O ZAP indica que interceptou a requisição, remover o parâmetro senha do corpo da requisição.
  - Era assim: Username=webgoat&Password=webgoat&SUBMIT>Login
  - Fica assim: Username=webgoat&SUBMIT>Login
- Enviar a requisição (Botão ***Play*** dos controles de breakpoint)
  - Terá logado como admin sem fornecer a senha!
  - Erro de projeto de programa! WebGoat falha em aberto em caso de erro no processo de autenticação.

# Bypass Client-Side Validation com OWASP ZAP

## Exercício 1

- No ZAP, ativar breakpoint de interceptação de requisições.
- No WebGoat 7.1
  - Tarefa/Menu **Parameter Tampering** → **Bypass Client Side JavaScript Validation**
  - O formulário já está com valores preenchidos.
  - Clicar no botão **Submit**.
- No ZAP, requisição foi interceptada
  - Inserir os caracteres @@@@ em cada um dos parâmetros no corpo da requisição.
  - Liberar/enviar a requisição.
  - Verificar a resposta devolvida pelo servidor.
    - O servidor alerta que a requisição contém os caracteres @@@@
    - Isto indica que a validação foi feita pelo servidor
- O WebGoat indica que a adulteração do cliente foi detectada pelo servidor.

## Exercício 2

- Entrar no seu deploy do WebGoat <http://localhost:8080/WebGoat/>
- Logar na aplicação como usuário/senha: guest/guest
- Selecionar a opção de menu **Parameter Tampering** → **Exploit Hidden Fields**
- Testar a funcionalidade de Shopping Cart (Carrinho de Compras)
  - Editar quantidade, campo *Quantity*;
  - Atualizar o carrinho, botão *UpdateCart*;
  - Realizar a compra, botão *Purchase*.
- Qual o comportamento do valor total da compra? Será possível editá-lo?
- Abrir as ferramentas de desenvolvedor do seu navegador.
- Ativar ferramenta de seleção (ponteiro do mouse) e selecionar o FORM do carrinho de compras.
- Encontrar no FORM a campo “*Price*” do tipo HIDDEN.
- No código do FORM para o campo “*Price*”, editar o valor 2999.99 para 0.99
- Na aplicação, ativar o botão *UpdateCart*.
- O que aconteceu? Qual o valor final da compra?
- Isto é uma vulnerabilidade de código ou uma falha de projeto?

# Bypass de autenticação com OWASP ZAP

No Webgoat 7.1

- **Session Management Flaws** → **Spoof an Authentication Cookie**
- Digitar login/senha **webgoat/webgoat**. Clicar no botão **Login**.
  - Autenticação com usuário legítimo
- Clicar no botão **Refresh**

No ZAP, identificar no histórico a requisição POST para este login

- login/senha são parâmetros do POST (ignorar isso!)
- Na resposta da requisição há um cookie com o token de autenticação
  - por exemplo, Set-Cookie: AuthCookie=65432ubphcfx
- Observar que o `AuthCookie` está presente em todas as requisições posteriores.

No WebGoat 7.1

- Digitar login/senha `guest/guest`. Clicar no botão **Login**.
  - Autenticação com usuário **ilegítimo/malicioso**
  - Autenticação mal sucedida!!

Há duas maneiras de executar o teste:

- método 1
  - No ZAP, no histórico, identificar a requisição de autenticação `guest/guest`
  - Clicar com botão direito do mouse na opção reenviar
  - Inserir o `AuthCookie` no cabeçalho Cookie e enviar a requisição.
    - Cookie: JSESSIONID=8243B04E5FF6913CAFA41EF9507B2390; AuthCookie=65432ubphcfx
    - Botão **Send** da resposta.
  - A resposta recebida indica que a autenticação por cookie funcionou.
- método 2
  - No ZAP, ativar breakpoint de interceptação de requisições (Lembra dele?).
  - No WebGoat 7.1, digitar login/senha `guest/guest`. Clicar no botão **Login**.
  - No ZAP, requisição foi interceptada
  - Inserir o `AuthCookie` no cabeçalho Cookie e enviar a requisição.
  - O WebGoat indica que a autenticação por cookie funcionou.

Pergunta: Será que o ataque funciona se o usuário webgoat fizer logout?