



Relatório de Avaliação de Segurança Cibernética - Teste de Penetração Whitebox

21 de Setembro de 2023

Sumário Executivo

Este sumário executivo apresenta as principais conclusões e recomendações resultantes do teste de segurança conduzido na aplicação web da Conviso Application Security. O objetivo do teste foi avaliar a segurança da aplicação e identificar possíveis vulnerabilidades e riscos de segurança.

Principais Conclusões:

1. Content Security Policy (CSP) Não Definido:

Foi identificado que o cabeçalho CSP não está adequadamente configurado no ambiente digital. Isso significa que não se tem uma política estrita de controle de fontes de scripts, o que pode tornar o sistema vulnerável a ataques de injeção de código.

A falta de uma política CSP bem definida pode permitir que atacantes explorem vulnerabilidades de cross-site scripting (XSS) com maior facilidade.

2. Falta de Cabeçalho Anti-clickjacking:

Constatou-se que a proteção contra ataques de clickjacking não está implementada no site. Isso expõe os usuários a riscos de engenharia social e ataques de fraude, onde um atacante pode manipular a interface do usuário de forma enganosa.

A ausência de um cabeçalho anti-clickjacking pode comprometer a confiabilidade e a segurança da plataforma.

3. Vazamento de Informações de Versão do Servidor:

Foi observado que o servidor está vazando informações de versão por meio do campo de cabeçalho de resposta HTTP "Server".

Essa exposição da versão do servidor facilita a identificação de possíveis vulnerabilidades e configurações desatualizadas. Atacantes podem explorar essas informações para lançar ataques específicos, visando versões conhecidas com vulnerabilidades.

Recomendações:

Com base nas conclusões acima, recomendo as seguintes ações imediatas:

1. Configuração do CSP:

- Implementar e configurar uma política de Content Security Policy (CSP) estrita que restrinja as fontes de scripts permitidas.
- Realizar testes regulares de segurança para garantir que a política CSP esteja efetivamente protegendo contra ataques XSS.

2. Implementação do Cabeçalho Anti-clickjacking:

- Configurar e habilitar um cabeçalho anti-clickjacking (por exemplo, X-Frame-Options) na aplicação web.
- Garantir que os controles de segurança estejam em vigor para prevenir ataques de manipulação de UI.

3. Ocultação das Informações de Versão do Servidor:

- Ocultar ou mascarar as informações de versão do servidor nos cabeçalhos de resposta HTTP.
- Monitorar continuamente a exposição de informações de versão do servidor e aplicar patches de segurança quando necessário.

Resumo Geral:

O teste de segurança revelou várias áreas de preocupação que exigem atenção imediata. A correção das vulnerabilidades e a implementação das recomendações de segurança são essenciais para proteger a aplicação web da Conviso Application Security contra ameaças e garantir a integridade dos dados dos usuários.

Este sumário executivo fornece uma visão geral das descobertas, mas é essencial revisar o relatório completo para obter detalhes adicionais sobre as vulnerabilidades e recomendações específicas.

Sumário

Informações Gerais	5
Escopo do Pentest	5
1. Análise da Aplicação Web Principal	5
2. Ambiente de Hospedagem	5
3. Exclusões do Escopo	6
4. Metodologia Utilizada	7
5. Tecnologias Avaliadas	7
6. Requisitos de Acesso	8
Vulnerabilidades Identificadas	9
Boas práticas e qualidade de código	13

Informações Gerais

Nome do Cliente: Conviso Application Security

Data de Início do Pentest: 14 de Setembro de 2023

Data de Término do Pentest: 21 de Setembro de 2023

Equipe de Teste de Segurança:

Anne Caroline - Analista de Cibersegurança

Introdução:

O propósito central do teste de penetração (pentest) conduzido foi avaliar integralmente a segurança da aplicação web da Conviso Application Security. Esse pentest abordou uma série de pontos cruciais em relação à segurança, compreendendo a análise minuciosa do código-fonte, as políticas de autenticação e autorização, além da configuração do servidor web.

Escopo do Pentest

O escopo do pentest abrange os seguintes aspectos da aplicação e infraestrutura:

1. Análise da Aplicação Web Principal

O objetivo desta seção é fornecer uma visão geral da análise de segurança da aplicação web da Conviso Application Security, com acesso total às informações internas, incluindo código-fonte e arquitetura. As áreas avaliadas incluem:

- **Revisão de Código-Fonte:** Uma análise detalhada do código-fonte da aplicação web, com foco na identificação de vulnerabilidades de segurança, como injeção de SQL, XSS, CSRF e outras.
- **Avaliação da Política de Autenticação e Autorização:** Uma análise crítica das políticas de autenticação e autorização da aplicação, incluindo revisão de permissões e controles de acesso.
- **Verificação da Configuração do Servidor Web:** Examinar e validar as configurações de segurança do servidor web para garantir que estejam alinhadas com as melhores práticas de segurança.

2. Ambiente de Hospedagem

O ambiente de hospedagem utilizado para executar a aplicação web principal em um ambiente Docker local também estará no escopo do pentest. Esta seção se concentra na avaliação das configurações de segurança específicas do ambiente Docker, bem como em possíveis vulnerabilidades relacionadas à infraestrutura local, incluindo:

- **Configurações de Segurança do Docker:** Avaliação das configurações de segurança do ambiente Docker, incluindo políticas de controle de acesso, configurações de rede e permissões do contêiner.

- **Isolamento de Contêineres:** Verificação do isolamento adequado entre contêineres e a segurança das configurações de isolamento.
- **Configuração da Rede:** Análise das configurações de rede Docker para garantir que a comunicação entre contêineres e com o host seja segura e adequada.
- **Possíveis Vulnerabilidades:** Identificação de possíveis vulnerabilidades relacionadas à infraestrutura, como configurações de firewall inadequadas, serviços expostos desnecessariamente e configurações de segurança do sistema operacional do host.
- **Políticas de Atualização e Manutenção:** Avaliação das políticas de atualização e manutenção do ambiente Docker para garantir que as atualizações de segurança sejam aplicadas regularmente.
- **Gerenciamento de Imagens:** Verificação do processo de criação, distribuição e gerenciamento de imagens Docker para garantir que apenas imagens confiáveis e seguras sejam usadas.

3. Exclusões do Escopo

Esta seção destina-se a esclarecer quais aspectos ou áreas específicas não estão incluídos no escopo do pentest devido ao cenário de teste da aplicação localmente em um ambiente Docker. As seguintes exclusões do escopo são aplicáveis:

- **Hospedagem em Ambientes de Produção:** Como a aplicação está sendo testada em um ambiente local de desenvolvimento baseado em Docker, quaisquer aspectos relacionados à segurança da infraestrutura de produção, como servidores em nuvem ou infraestrutura física, não estão no escopo deste pentest.
- **Configurações de Infraestrutura de Terceiros:** Quaisquer configurações de segurança específicas de provedores de serviços em nuvem ou de terceiros que não se apliquem ao ambiente local de Docker também estão excluídas deste escopo.
- **Interação com Redes Externas:** Os testes de segurança relacionados à interação da aplicação com redes externas, como acesso à internet ou redes de produção, estão fora do escopo deste pentest, pois a aplicação está atualmente isolada em um ambiente local.
- **Gestão de Redes e Infraestrutura:** A gestão de redes, configurações de firewall, política de acesso à rede e outras responsabilidades relacionadas à infraestrutura de rede não estão incluídas no escopo, uma vez que esses aspectos não são controlados diretamente pelo cenário de teste em Docker.
- **Proteção Física:** Questões de segurança física, como o acesso físico a servidores ou hardware, não estão dentro do escopo desta avaliação, uma vez que a aplicação é executada em um ambiente local.

4. Metodologia Utilizada

Neste relatório de pentest, detalhamos a metodologia usada para avaliar a segurança da aplicação web executada em um ambiente Docker local. A abordagem adotada é baseada nas diretrizes do OWASP (Open Web Application Security Project), um padrão reconhecido para testes de segurança de aplicações web.

A metodologia de teste de segurança utilizada abrange uma combinação de testes manuais e automatizados para garantir uma avaliação abrangente da segurança da aplicação. Esta abordagem é adaptada para atender às necessidades específicas deste ambiente de teste.

Os principais objetivos desta metodologia de teste de segurança são:

- Identificar e relatar vulnerabilidades críticas que possam comprometer a segurança da aplicação.
- Avaliar a postura geral de segurança da aplicação, incluindo a eficácia de controles de acesso e políticas de segurança.
- Fornecer recomendações claras e ações corretivas para mitigar qualquer risco de segurança identificado.

Esta metodologia foi projetada para garantir uma avaliação completa e precisa da segurança da aplicação, permitindo que a equipe de desenvolvimento tome medidas apropriadas para melhorar a segurança onde necessário.

5. Tecnologias Avaliadas

Durante o pentest, foram avaliadas as seguintes tecnologias utilizadas na aplicação web da Conviso Application Security:

- JavaScript Frameworks: Utilização de frameworks JavaScript para o desenvolvimento da aplicação.
 - Handlebars 4.7.7
- Font Scripts: Scripts de fontes usadas no design da aplicação.
 - Font Awesome 4.7.0
- JavaScript Graphics: Biblioteca gráfica em JavaScript para renderização de gráficos interativos.
 - Chart.js
- Linguagens de Programação:
 - PHP
- CDN (Content Delivery Network):
 - cdnjs
- Serviço de Proteção de CDN:
 - Cloudflare

- JavaScript Libraries: Bibliotecas JavaScript utilizadas na aplicação.
 - DataTables 1.13.1
 - jQuery 3.5.1
 - jQuery UI 1.13.2
 - Moment.js 2.29.4
 - Select2

Essas tecnologias foram avaliadas quanto à sua segurança e potenciais vulnerabilidades durante o pentest, com o objetivo de garantir a integridade e segurança geral da aplicação web da Conviso Application Security. As conclusões e recomendações específicas resultantes deste pentest devem ser implementadas para fortalecer ainda mais a postura de segurança da aplicação. Recomenda-se a revisão do relatório completo para obter detalhes adicionais sobre as vulnerabilidades identificadas e as recomendações específicas para mitigá-las.

6. Requisitos de Acesso

Para a execução deste pentest, foi necessário configurar um ambiente local de desenvolvimento. Abaixo estão os requisitos e etapas necessários para criar esse ambiente:

Ambiente de Desenvolvimento Local

A aplicação web foi adquirida a partir do repositório oficial do GitHub, onde segui com a clonagem do projeto em minha máquina local.

- Comando: ``git clone https://github.com/convisolabs/CVWA.git``

Após o clone, naveguei até o diretório do projeto:

- Comando: ``cd CVWA``

O Docker foi empregado na criação e inicialização do ambiente de desenvolvimento local, utilizando um Dockerfile que incorpora tanto o servidor web quanto o banco de dados.

- Comando para levantar o Docker: ``docker build -t cvwa .``

O ambiente Docker foi inicializado através de um contêiner baseado na imagem "cvwa" com interação interativa. Além disso, foi configurado o mapeamento da porta 8080 do host para a porta 80 do contêiner, viabilizando o acesso ao serviço dentro deste último por meio da porta 8080 no host.

- Comando: `docker container run -ti -p 8080:80 cvwa`

Inicialização da Aplicação

Para iniciar a aplicação, acessamos a URL localhost:8080 no navegador da web após o ambiente Docker ter sido levantado com sucesso na porta 8080. A partir desse ponto, a aplicação web estará disponível para testes e avaliação de segurança.

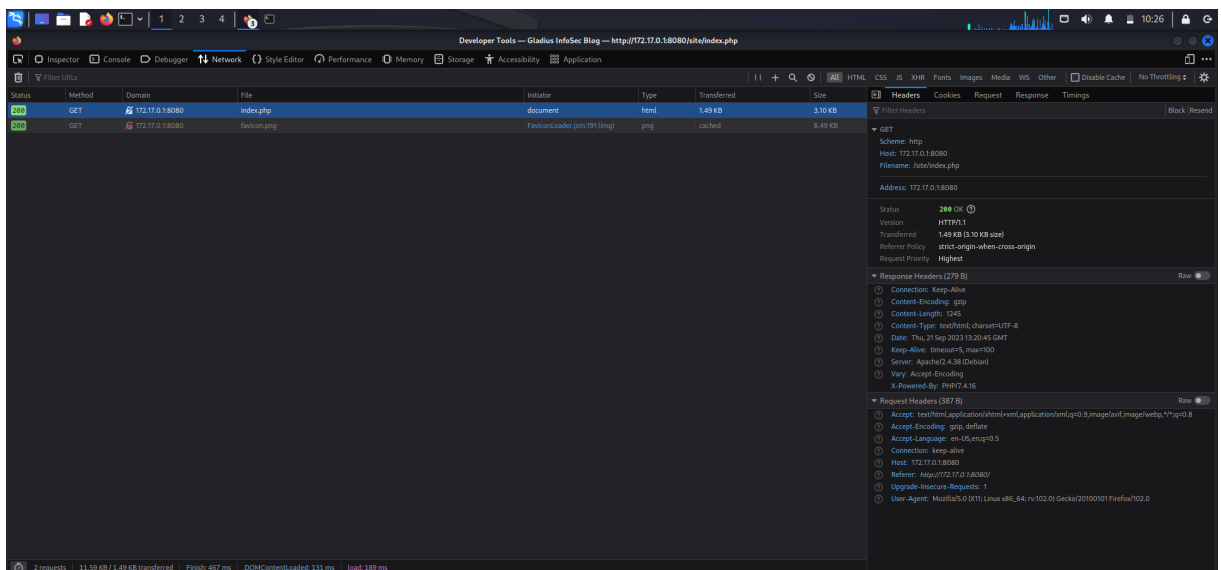
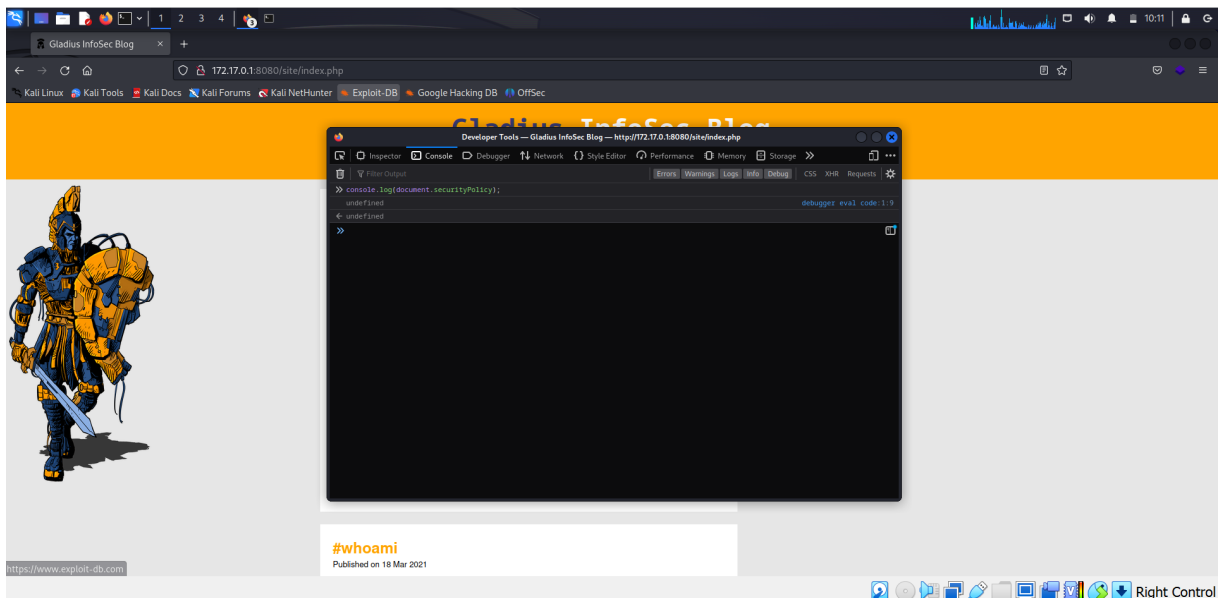
Vulnerabilidades Identificadas

Content Security Policy (Cabeçalho da Política de Segurança de Conteúdo - CSP) não definido

URL: <http://localhost:8080>

Risco: Médio

Descrição: A Política de Segurança de Conteúdo (CSP) é uma camada adicional de segurança que ajuda a detectar e mitigar certos tipos de ataques, incluindo Cross Site Scripting (XSS) e ataques de injeção de dados. Esses ataques são usados para tudo, desde roubo de dados até destruição de sites ou distribuição de malware. O CSP fornece um conjunto de cabeçalhos HTTP padrão que permitem aos proprietários de sites declarar fontes aprovadas de conteúdo que os navegadores devem ter permissão para carregar naquela página - os tipos cobertos são JavaScript, CSS, quadros HTML, fontes, imagens e objetos incorporáveis, como miniaplicativos Java, ActiveX, arquivos de áudio e vídeo.



Solução: Certifique-se de que seu servidor web, servidor de aplicativos, balanceador de carga, etc. esteja configurado para definir o cabeçalho Content-Security-Policy. Configurar a Política de Segurança de Conteúdo envolve adicionar o cabeçalho HTTP Content-Security-Policy a uma página da web e fornecer valores para controlar quais recursos o agente do usuário tem permissão para carregar para essa página.

Exemplo de adição de cabeçalho CSP (Content Security Policy):

```
<!DOCTYPE html>
<html>

<head>
  <!-- Configurando o cabeçalho HTTP X-Content-Type-Options como nosniff -->
  <meta http-equiv="X-Content-Type-Options" content="nosniff">
  <title>Exemplo de CSP</title>
  <!-- Resto do código HTML -->
</head>

<body>
  <!-- Conteúdo da página -->
</body>

</html>
```

Referência:

OWASP_2021_A05 https://owasp.org/Top10/A05_2021-Security_Misconfiguration/

Missing Anti-clickjacking Header

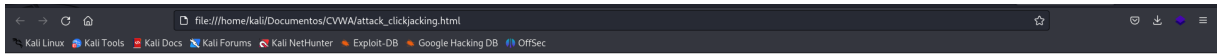
URL: <http://localhost:8080>

Risco: Médio

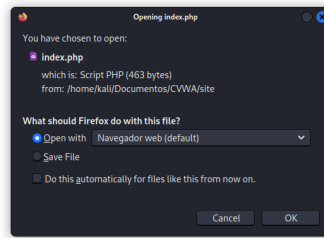
Descrição: A resposta não contém a política de cabeçalho X-Frame-Options, que é essencial para a proteção contra ataques de 'ClickJacking'.

Exemplo de ataque ClickJacking:

Após a execução de um arquivo HTML que chama o frame da página principal do sistema, o redirecionamento para o arquivo index.php ocorre sem problemas. No entanto, é importante observar que o navegador não é capaz de renderizar diretamente arquivos PHP. Em vez disso, ele apresenta ao usuário a opção de escolher como deseja manipular o arquivo, seja fazendo o download ou abrindo-o com o programa adequado.



Conteúdo da Página Externa



Solução: Navegadores da Web modernos suportam os cabeçalhos HTTP X-Frame-Options. É importante garantir que eles estejam definidos em todas as páginas da web retornadas pelo site.

Se a intenção é permitir que a página seja enquadrada apenas por páginas do servidor (por exemplo, como parte de um FRAMESET), a opção recomendada é usar 'SAMEORIGIN'. Por outro lado, se não se espera que a página seja enquadrada em nenhum contexto, a opção adequada é 'DENY'.

Recomendo também reforçar a segurança dos cabeçalhos HTTP com as seguintes configurações:

X-XSS-Protection: Essa medida visa proteger contra ataques de script entre sites (XSS).

Strict-Transport-Security: Esta configuração é fundamental para forçar o uso de HTTPS e proteger contra ataques de interceptação de tráfego.

Exemplo de adição de cabeçalho X-Frame-Options:

```
<!DOCTYPE html>
<html>

<head>
  <!-- Configurando o cabeçalho HTTP X-Frame-Options como DENY -->
  <meta http-equiv="X-Frame-Options" content="DENY">
  <title>Exemplo de CSP</title>
  <!-- Resto do código HTML -->
</head>

<body>
  <!-- Conteúdo da página -->
</body>

</html>
```

Referências:

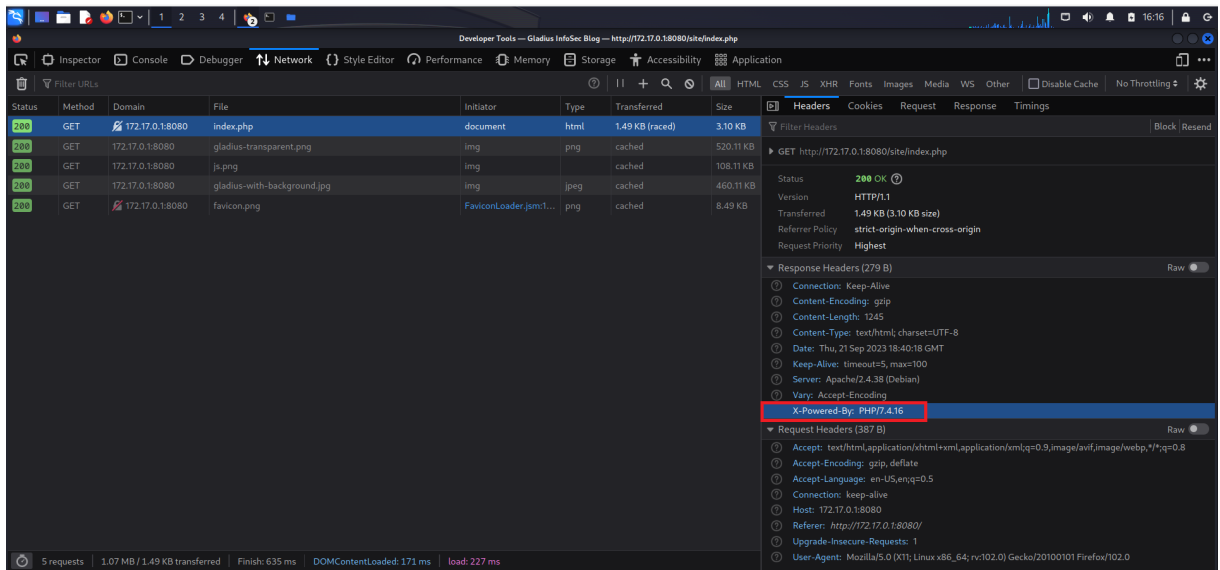
OWASP_2021_A05	https://owasp.org/Top10/A05_2021-Security_Misconfiguration/
WSTG-v42-CLNT-09	https://owasp.org/www-project-web-security-testing-guide/v42/4-Web_Application_Security_Testing/11-Client-side_Testing/09-Testing_for_Clickjacking

Servidor vazando informações de versão por meio do campo de cabeçalho de resposta HTTP "Server"

URL: <http://localhost:8080>

Risco: Baixo

Descrição: O servidor web/aplicativo está vazando informações de versão por meio do cabeçalho de resposta HTTP "Server". O acesso a essas informações pode facilitar que invasores identifiquem outras vulnerabilidades às quais servidor web/aplicativo está sujeito.



Solução: Certifique-se de que a configuração do servidor web, servidor de aplicativos, balanceador de carga, etc., esteja definida para ocultar o cabeçalho 'Server' ou para fornecer informações genéricas.

A configuração para controlar a divulgação de informações de versão no cabeçalho "Server" de um servidor web pode variar de acordo com o software do servidor que está sendo utilizado, como o Apache ou o Nginx. Essas configurações são específicas para cada servidor e devem ser ajustadas diretamente nas configurações do servidor web escolhido.

Referências:

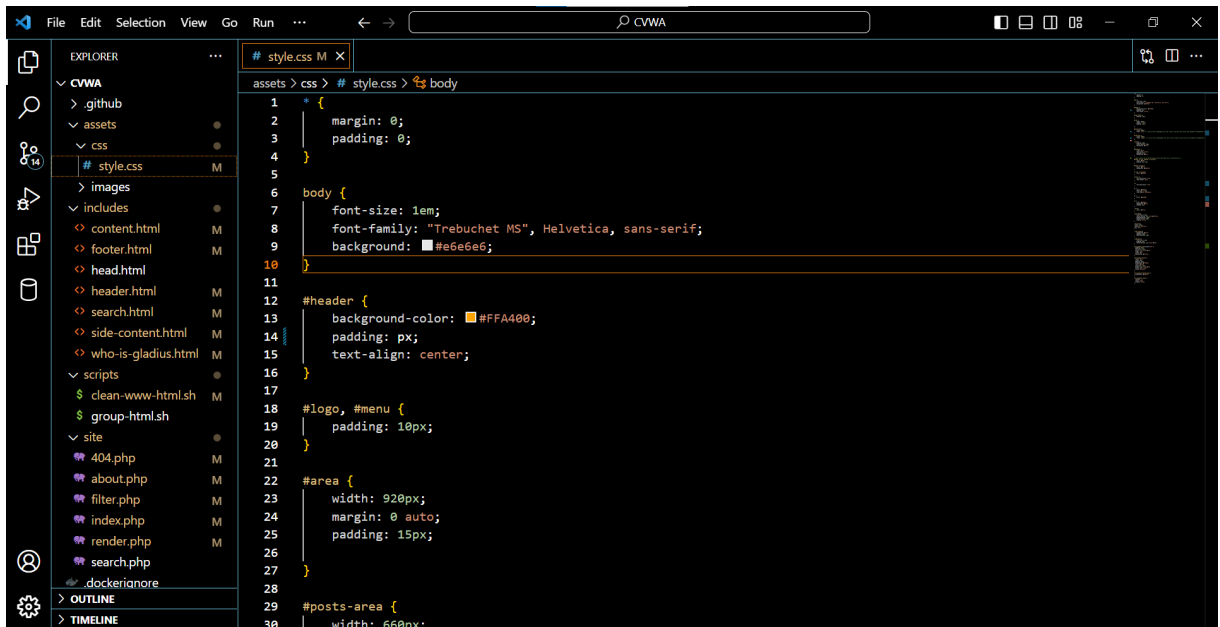
- OWASP_2021_A05 https://owasp.org/Top10/A05_2021-Security_Misconfiguration/
- WSTG-v42-INFO-02 https://owasp.org/www-project-web-security-testing-guide/v42/4-Web_Application_Security_Testing/01-Information_Gathering/02-Fingerprint_Web_Server

Boas práticas e qualidade de código

A seguir, apresentam-se algumas sugestões para aprimorar o código, garantindo o funcionamento adequado do sistema e reforçando sua segurança. Essas sugestões derivam de uma análise manual do código-fonte.

Arquivo *style.css*

Atributo 'padding' inadequado: O valor "px" no atributo de padding do elemento #header na linha 12, não está configurado de maneira adequada. Para definir o espaçamento corretamente, é necessário fornecer um valor numérico, como por exemplo padding: 10px;.



Utilização de valores float para #posts-area e #side-area: Para seguir as melhores práticas e evitar que elementos seguintes afetem o layout devido aos floats, é recomendado empregar uma técnica de limpeza, conhecida como "clearfix". Uma maneira eficaz de fazer isso é adicionar uma classe "clearfix" ao elemento pai que contenha tanto #posts-area(linha 29) quanto #side-area(linha 34), como exemplificado abaixo:

```
/* Adicione uma classe clearfix para resolver problemas de flutuação */
.clearfix::after {
  content: "";
  display: table;
  clear: both;
}
```

Isso garantirá que o layout do conteúdo não seja afetado devido aos flutuadores utilizados.

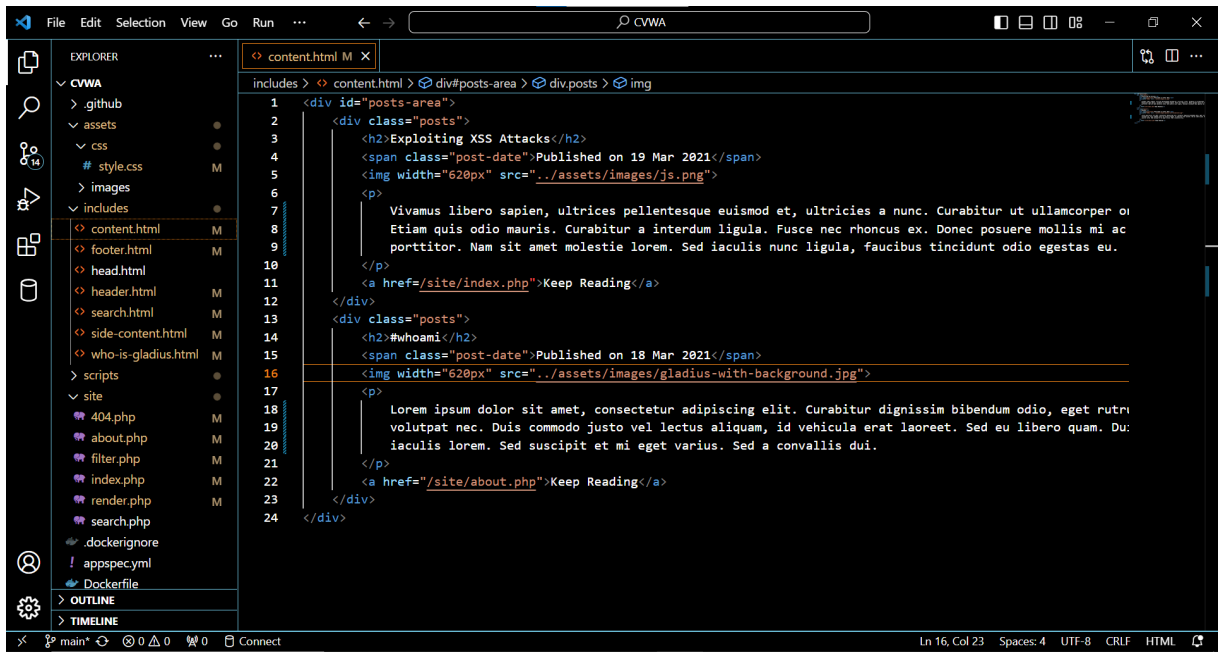


Uso de classes: Prefira o uso de classes em vez de IDs para estilos mais genéricos e reutilizáveis (linhas 52 e 57).



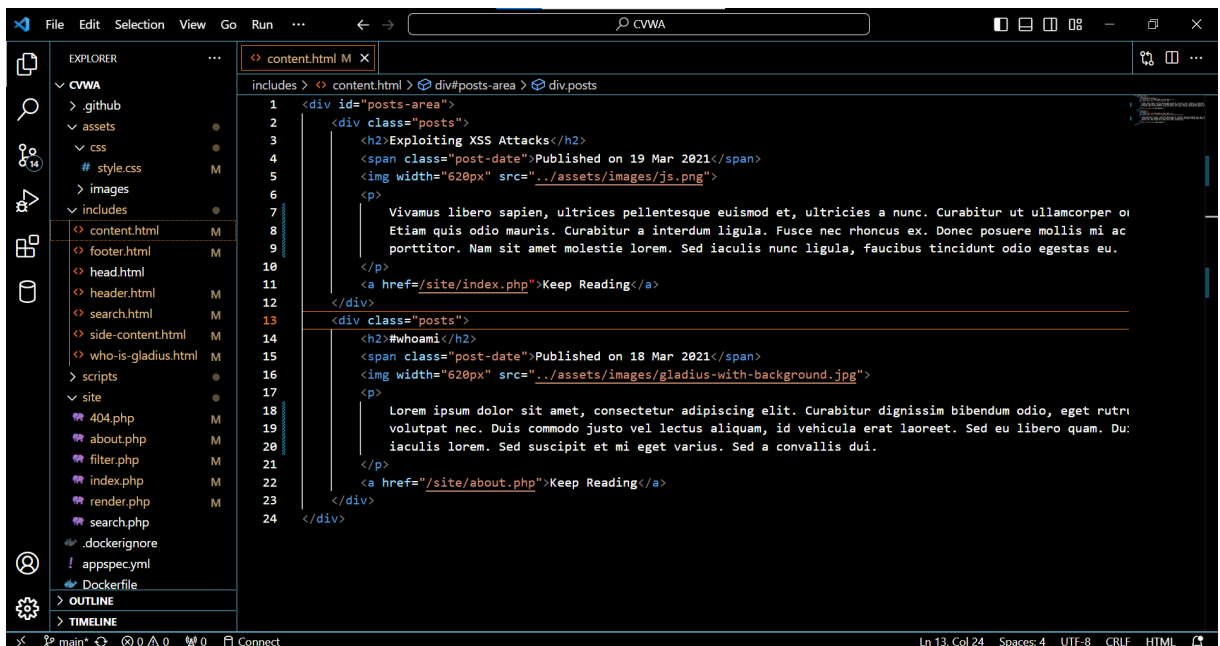
Arquivo content.html

Propriedade width: A propriedade "width" da tag está definida como "620px" nas linhas 5 e 16. A unidade "px" não deve ser usada ao especificar a largura da imagem dentro do atributo HTML. Basta definir a largura como um valor numérico, como .



```
1 <div id="posts-area">
2   <div class="posts">
3     <h2>Exploiting XSS Attacks</h2>
4     <span class="post-date">Published on 19 Mar 2021</span>
5     
6     <p>
7       Vivamus libero sapien, ultrices pellentesque euismod et, ultricies a nunc. Curabitur ut ullamcorper or
8       Etiam quis odio mauris. Curabitur a interdum ligula. Fusce nec rhoncus ex. Donec posuere mollis mi ac
9       porttitor. Nam sit amet molestie lorem. Sed iaculis nunc ligula, faucibus tincidunt odio egestas eu.
10    </p>
11    <a href="/site/index.php">Keep Reading</a>
12  </div>
13  <div class="posts">
14    <h2>#whoami</h2>
15    <span class="post-date">Published on 18 Mar 2021</span>
16    
17    <p>
18      Lorem ipsum dolor sit amet, consectetur adipiscing elit. Curabitur dignissim bibendum odio, eget rutri
19      volutpat nec. Duis commodo justo vel lectus aliquam, id vehicula erat laoreet. Sed eu libero quam. Du:
20      iaculis lorem. Sed suscipit et mi eget varius. Sed a convallis dui.
21    </p>
22    <a href="/site/about.php">Keep Reading</a>
23  </div>
24 </div>
```

Erro na tag <a>: No segundo elemento <a>(linha 11), o atributo "href" está faltando uma aspa de abertura. O correto é Keep Reading.



```
1 <div id="posts-area">
2   <div class="posts">
3     <h2>Exploiting XSS Attacks</h2>
4     <span class="post-date">Published on 19 Mar 2021</span>
5     
6     <p>
7       Vivamus libero sapien, ultrices pellentesque euismod et, ultricies a nunc. Curabitur ut ullamcorper or
8       Etiam quis odio mauris. Curabitur a interdum ligula. Fusce nec rhoncus ex. Donec posuere mollis mi ac
9       porttitor. Nam sit amet molestie lorem. Sed iaculis nunc ligula, faucibus tincidunt odio egestas eu.
10    </p>
11    <a href="/site/index.php">Keep Reading</a>
12  </div>
13  <div class="posts">
14    <h2>#whoami</h2>
15    <span class="post-date">Published on 18 Mar 2021</span>
16    
17    <p>
18      Lorem ipsum dolor sit amet, consectetur adipiscing elit. Curabitur dignissim bibendum odio, eget rutri
19      volutpat nec. Duis commodo justo vel lectus aliquam, id vehicula erat laoreet. Sed eu libero quam. Du:
20      iaculis lorem. Sed suscipit et mi eget varius. Sed a convallis dui.
21    </p>
22    <a href="/site/about.php">Keep Reading</a>
23  </div>
24 </div>
```

Arquivo footer.html

Acessibilidade: Para tornar o código mais inclusivo, é uma boa prática adicionar um atributo `aria-label` à `<div>` ou ao elemento de texto para descrever seu propósito. Isso pode ser especialmente útil para usuários que dependem de leitores de tela.

Uso da tag `` para formatação: Está sendo empregado a tag `` para aplicar formatação em negrito ao texto "All rights reserved". Contudo, o uso de elementos de formatação como

`` não é considerado uma prática recomendada. Em vez disso, é mais apropriado utilizar CSS para estilizar o texto como negrito.

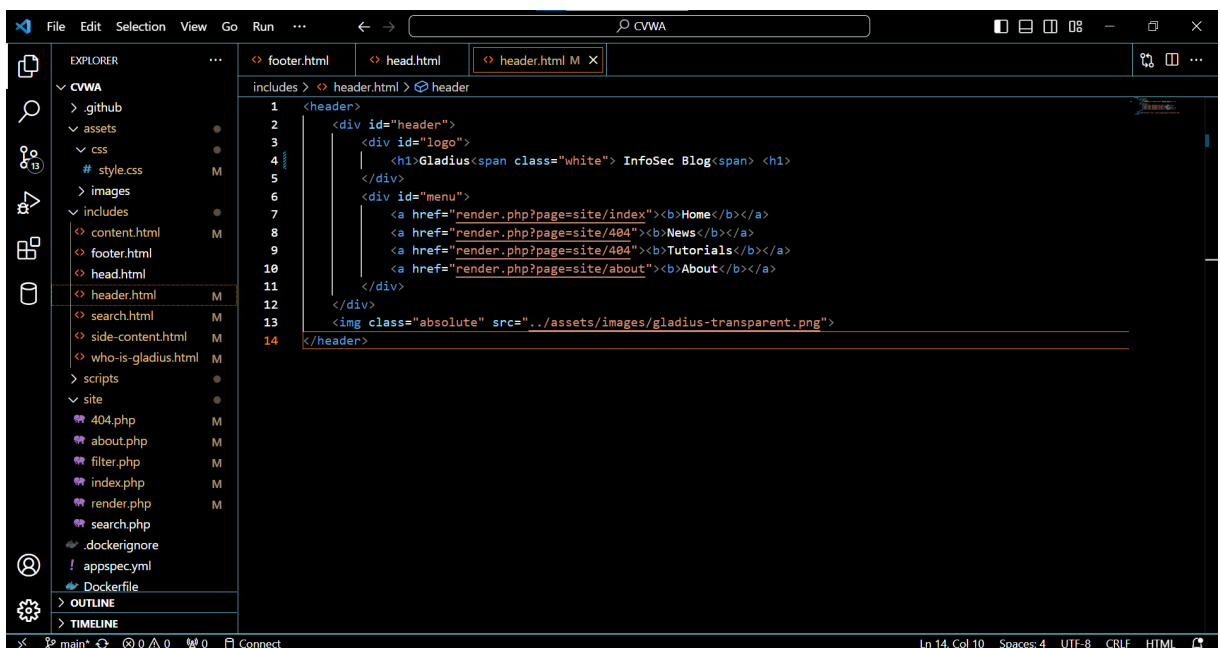


The screenshot shows the Visual Studio Code editor with the CVWA project open. The Explorer sidebar on the left shows the project structure, including folders like .github, assets, css, images, includes, scripts, and site, and files like 404.php, about.php, filter.php, index.php, render.php, search.php, .dockerignore, appspec.yml, Dockerfile, OUTLINE, and TIMELINE. The footer.html file is selected in the Explorer and its content is displayed in the editor. The code in footer.html is as follows:

```
1 <footer>
2   <div id="baseboard">
3     <b>All rights reserved</b>
4   </div>
5 </footer>
```

Arquivo header.html

Elemento `<h1>`: Há um erro de marcação no código HTML. O elemento `<h1>` (linha 4) não está sendo fechado corretamente.



The screenshot shows the Visual Studio Code editor with the CVWA project open. The Explorer sidebar on the left shows the project structure, including folders like .github, assets, css, images, includes, scripts, and site, and files like 404.php, about.php, filter.php, index.php, render.php, search.php, .dockerignore, appspec.yml, Dockerfile, OUTLINE, and TIMELINE. The header.html file is selected in the Explorer and its content is displayed in the editor. The code in header.html is as follows:

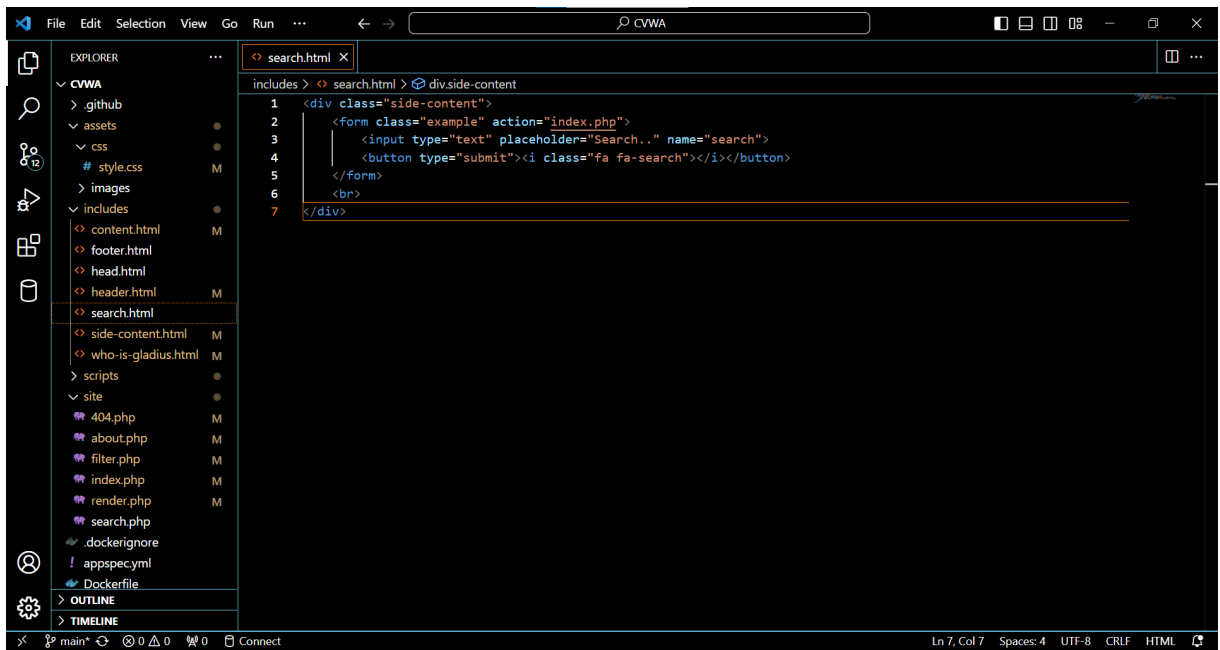
```
1 <header>
2   <div id="header">
3     <div id="logo">
4       <h1>Gladius<span class="white"> InfoSec Blog</span> <h1>
5     </div>
6     <div id="menu">
7       <a href="render.php?page=site/index"><b>Home</b></a>
8       <a href="render.php?page=site/404"><b>News</b></a>
9       <a href="render.php?page=site/404"><b>Tutorials</b></a>
10      <a href="render.php?page=site/about"><b>About</b></a>
11    </div>
12  </div>
13  
14 </header>
```

Arquivo search.html

Acessibilidade: Para aprimorar a acessibilidade do formulário, é recomendável incluir um atributo "label" que descreva o campo de pesquisa. Isso proporciona assistência aos usuários que dependem de leitores de tela. Por exemplo:

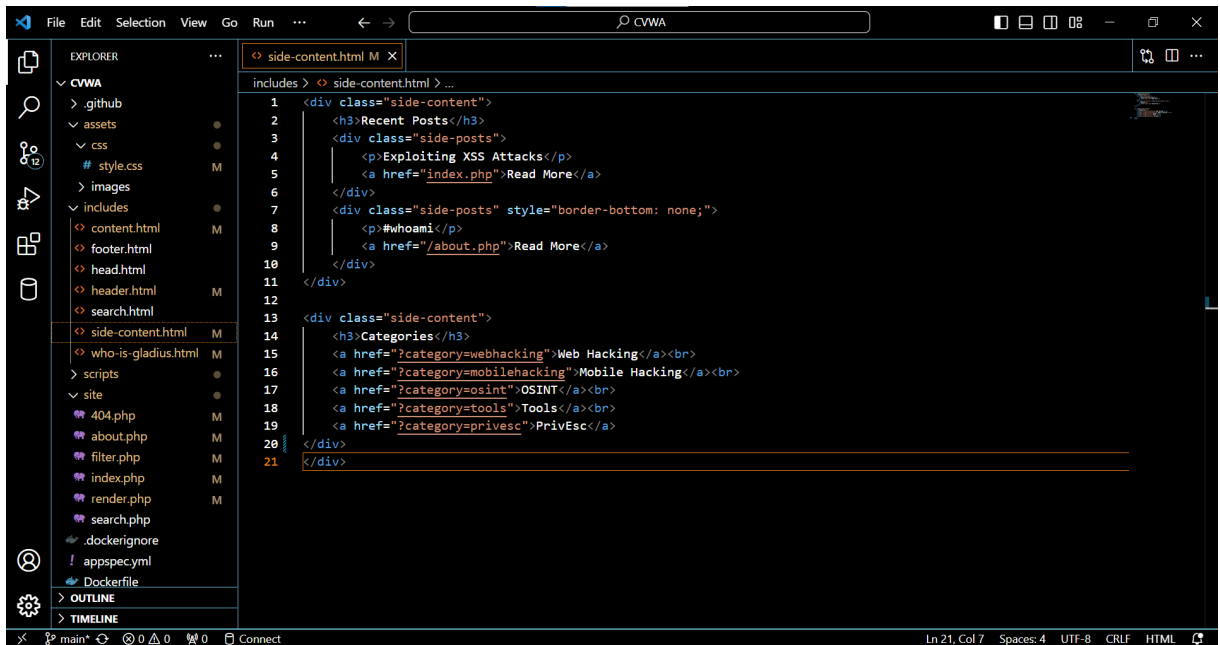
```
<label for="search">Pesquisar:</label>
```

```
<input type="text" id="search" placeholder="Pesquisar..." name="search">
```



Arquivo side-content.html

Fechamento do segundo <div>: Há um fechamento </div> adicional no segundo bloco de código (linha 21). Recomendo remover o </div> excedente para que a estrutura fique corretamente configurada.



Arquivo *who-is-gladius.html*

Uso de `
` para espaçamento: Está sendo empregado várias tags `
` para criar espaçamento vertical entre os elementos. Embora isso possa funcionar para adicionar espaço, não é considerado uma prática ideal. Em vez disso, é recomendável utilizar CSS para controlar o espaçamento, o que separa a formatação do conteúdo de forma mais eficaz. Por exemplo:

```
<style>
    .posts p {
        margin-bottom: 10px;
    }
</style>
```

```
1 <div id="posts-area">
2   <div class="posts">
3     <h2>Who is Gladius?</h2>
4     <br>
5     
6     <p><br>
7       Lorem ipsum dolor sit amet, consectetur adipiscing elit. Curabitur dignissim bibendum odio, eget rutru
8     </p><br>
9     <p>
10      Etiam sed nunc et nisi scelerisque vulputate. Donec eros elit, porttitor in convallis non, luctus qui:
11    </p><br>
12    <p>
13      Suspendisse dapibus ac augue et congue. Etiam quis ante vulputate, finibus eros in, posuere dui. Donec
14    </p>
15  </div>
16 </div>
```

Largura da imagem: A propriedade "width" da tag está definida como "620px" (linha 5). A unidade "px" não deve ser usada ao especificar a largura da imagem dentro do atributo HTML. Basta definir a largura como um valor numérico, como

```
1 <div id="posts-area">
2   <div class="posts">
3     <h2>Who is Gladius?</h2>
4     <br>
5     
6     <p><br>
7       Lorem ipsum dolor sit amet, consectetur adipiscing elit. Curabitur dignissim bibendum odio, eget rutru
8     </p><br>
9     <p>
10      Etiam sed nunc et nisi scelerisque vulputate. Donec eros elit, porttitor in convallis non, luctus qui:
11    </p><br>
12    <p>
13      Suspendisse dapibus ac augue et congue. Etiam quis ante vulputate, finibus eros in, posuere dui. Donec
14    </p>
15  </div>
16 </div>
```

Arquivo clean-www-html.sh

Uso do comando "sudo": O comando "sudo" é empregado para elevar os privilégios e executar comandos com privilégios de superusuário. Isso implica que o comando tem o potencial de causar danos substanciais ao sistema ou à estrutura de diretórios do servidor web. Portanto, é fundamental garantir que você esteja executando esse comando somente

se possuir a permissão adequada e compreender completamente as consequências envolvidas.

Comando "rm -rf": É fundamental estar ciente de que o uso do comando "rm -rf" é extremamente arriscado, uma vez que resulta na exclusão irreversível de todos os dados. Qualquer execução equivocada desse comando pode resultar na perda de informações críticas. Portanto, antes de utilizá-lo, assegure-se de que está especificando o caminho corretamente e realmente deseja remover todos os conteúdos dentro dele.

Antes de executar um comando desse tipo em um ambiente de produção, é recomendável realizar um backup de todos os dados críticos e ponderar cuidadosamente se é realmente necessário prosseguir. É importante considerar alternativas mais seguras como a substituição de arquivos ou diretórios específicos, em vez de optar pela exclusão completa de dados.



```
File Edit Selection View Go Run ...  
CVWA  
EXPLORER  
  > .github  
  > assets  
    > css  
      # style.css M  
    > images  
  > includes  
  > scripts  
    $ clean-www-html.sh  
    $ group-html.sh  
  > site  
    404.php M  
    about.php M  
    filter.php M  
    index.php M  
    render.php M  
    search.php  
    .dockerignore  
    appspec.yml  
    Dockerfile  
    index.html M  
    LICENSE  
    README.md  
  > OUTLINE  
  > TIMELINE  
main* 0 0 0 0 Connect  
Ln 3, Col 28 Spaces: 4 UTF-8 CRLF Shell Script
```

Arquivo 404.php

O código em questão possui dois símbolos de maior (>) consecutivos (linha 6), o que resulta em um erro de sintaxe. É necessário remover um deles para corrigir a situação.



```
1 <!DOCTYPE html>
2 <html>
3     <?php include "../includes/head.html" ?>
4     <body>
5         <?php include "../includes/header.html" ?>
6         <div id="area">
7             <div id="posts-area">
8                 <div class="posts">
9                     <h1>404 - Page not found</h1>
10                    <h3><a href="/">Return to home</a></h3>
11                </div>
12            </div>
13        </div>
14        <?php include "../includes/footer.html" ?>
15    </body>
16 </html>
```

Arquivo *about.php*

Na parte final do código, a tag `</html>` não está sendo fechada. É fundamental assegurar o fechamento adequado da estrutura HTML, pois isso desempenha um papel crucial na validação e no funcionamento apropriado da página.



```
1 <html>
2     <?php include "../includes/head.html" ?>
3     <body>
4         <?php include "../includes/header.html" ?>
5         <div id="area">
6             <?php include "../includes/who-is-gladius.html" ?>
7             <div id="side-area">
8                 <?php include "../includes/search.html" ?>
9                 <?php include "search.php" ?>
10                <?php include "../includes/side-content.html" ?>
11            </div>
12        </div>
13        <?php include "../includes/footer.html" ?>
14    </div>
15 </body>
16 </body>
```

Arquivo *filter.php*

Inclusão de Barras Invertidas: No código em questão, está sendo empregada a função `addslashes` para inserir barras invertidas antes das aspas simples no parâmetro da categoria. Isso é uma medida destinada a mitigar possíveis ataques de injeção SQL.

Entretanto, é fundamental observar que a utilização de recursos de escape automático, como as funções preparadas de banco de dados, representa uma prática mais segura e recomendada.



```
1 <?php
2 if (function_exists('str_contains')) {
3     function str_contains(string $haystack, string $needle): bool {
4         return '' === $needle || false !== strpos($haystack, $needle);
5     }
6 }
7
8 $category= $_GET['category'];
9
10 if (isset($category)) {
11     if (str_contains($category, "")) {
12         if (str_contains($_SERVER["HTTP_USER_AGENT"], "sqlmap")) {
13             http_response_code(500);
14             die();
15         }
16
17         if (substr_count($category, "") % 2 != 0) {
18             echo "<br><br>You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the
19             right syntax to use near '' addslashes($category).'' and STATUS = 'Ativo'' at line 1.<br><br>";
20         }
21
22         else {
23             echo "<br><br><cite><b>Category:</b></cite><br>" . htmlspecialchars(substr($category, 0, strpos($category, ""))) . "<br><br>";
24         }
25
26         else {
27             echo "<br><br><cite><b>Category:</b></cite><br>" . htmlspecialchars($category) . "<br><br>";
28         }
29     }
30 }
```

Arquivo filter.php

Detecção de Ferramenta de Teste de Injeção SQL via HTTP_USER_AGENT: O código atual realiza uma verificação no cabeçalho HTTP_USER_AGENT (linha 12) da solicitação para identificar a presença da string "sqlmap". Esse método visa detectar a ferramenta de teste de injeção SQL conhecida como "sqlmap" e, em caso positivo, retorna um código de resposta HTTP 500 (Internal Server Error). No entanto, é importante observar que essa abordagem pode não ser completamente confiável, uma vez que os cabeçalhos do usuário têm a capacidade de serem falsificados.

Implementar um WAF é uma medida eficaz de segurança para proteger o aplicativo contra ataques de injeção SQL, incluindo os realizados com ferramentas como o "sqlmap". Essa solução é capaz de detectar e bloquear automaticamente tentativas suspeitas de injeção SQL com base em regras de segurança. Além disso, é fundamental validar e sanitizar as entradas do usuário antes de utilizá-las em consultas SQL. Isso engloba a validação dos tipos de dados, o emprego de funções de escape apropriadas para dados não confiáveis e a restrição do escopo das consultas SQL sempre que viável. Quando disponível, a adoção de consultas SQL parametrizadas representa a abordagem mais segura, uma vez que os parâmetros são tratados de maneira segura pelo banco de dados.



```
1 <?php
2 if (!function_exists('str_contains')) {
3     function str_contains(string $haystack, string $needle): bool {
4         return '' === $needle || false !== strpos($haystack, $needle);
5     }
6 }
7
8 $category= $_GET['category'];
9
10 if (isset($category)) {
11     if (str_contains($category, "'")) {
12         if (str_contains($_SERVER['HTTP_USER_AGENT'], "sqlmap")) {
13             http_response_code(500);
14             die();
15         }
16
17         if (substr_count($category, "'") % 2 !== 0) {
18             echo "<br><br>You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the
19             right syntax to use near '' addslashes($category)." and STATUS = 'Ativo' at line 1.<br><br>";
20         }
21
22         else {
23             echo "<br><br><cite><b><cite><br>" . htmlspecialchars(substr($category, 0, strpos($category, "'"))) . "<br><br>";
24         }
25
26         else {
27             echo "<br><br><cite><b><cite><br>" . htmlspecialchars($category) . "<br><br>";
28         }
29     }
30 }
```

Arquivo filter.php

Validação de Categoria para Prevenir Problemas de Sintaxe SQL: Se a categoria conter um número ímpar de aspas simples (') (linha 17), o sistema exibirá uma mensagem de erro que indica a presença de um possível problema de sintaxe SQL. Em contrapartida, se o número de aspas simples for par, o sistema exibirá a categoria normalmente.



```
1 <?php
2 if (!function_exists('str_contains')) {
3     function str_contains(string $haystack, string $needle): bool {
4         return '' === $needle || false !== strpos($haystack, $needle);
5     }
6 }
7
8 $category= $_GET['category'];
9
10 if (isset($category)) {
11     if (str_contains($category, "'")) {
12         if (str_contains($_SERVER['HTTP_USER_AGENT'], "sqlmap")) {
13             http_response_code(500);
14             die();
15         }
16
17         if (substr_count($category, "'") % 2 !== 0) {
18             echo "<br><br>You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the
19             right syntax to use near '' addslashes($category)." and STATUS = 'Ativo' at line 1.<br><br>";
20         }
21
22         else {
23             echo "<br><br><cite><b><cite><br>" . htmlspecialchars(substr($category, 0, strpos($category, "'"))) . "<br><br>";
24         }
25
26         else {
27             echo "<br><br><cite><b><cite><br>" . htmlspecialchars($category) . "<br><br>";
28         }
29     }
30 }
```




Arquivo render.php

Implementação de Validação e Filtragem Adequada no Uso da Variável `$_GET['page']`: Atualmente o código não implementa um processo adequado de validação ou filtragem na variável `$_GET['page']`. Essa falha pode representar um sério problema de segurança, pois permite que os usuários forneçam qualquer valor para `$_GET['page']`, o que, por sua vez, pode resultar em inclusão de arquivo inseguras.

É aconselhável adotar uma abordagem rigorosa de validação e filtragem para os valores recebidos por `$_GET['page']`, a fim de garantir que somente arquivos seguros sejam incluídos no processo.



```
1 <?php
2 $page = $_GET['page'];
3
4 if (isset($page)) {
5     include '/var/www/html/' . $page . ".php";
6 }
7
8 else {
9     include 'index.php';
10 }
11 ?>
```

Foi verificado que não são criados novos arquivos a partir da página.



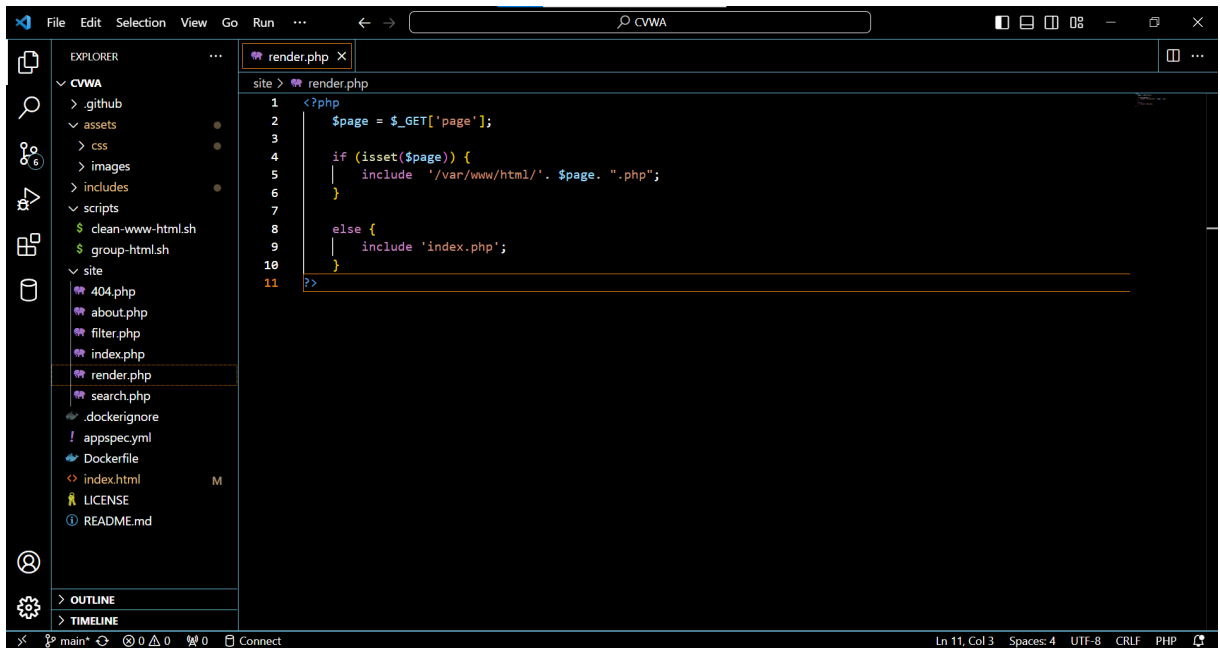
Warning: include(/var/www/html/' OR '1'='1.php): failed to open stream: No such file or directory in /var/www/html/site/render.php on line 5

Warning: include(): Failed opening '/var/www/html/' OR '1'='1.php' for inclusion (include_path='.:usr/local/lib/php') in /var/www/html/site/render.php on line 5

Arquivo render.php

Riscos de Segurança na Inclusão de Arquivos com Caminhos Absolutos no Código: Atualmente o código realiza a inclusão de arquivos com base em caminhos absolutos, o que pode apresentar riscos à segurança. É essencial garantir que os caminhos absolutos especificados sejam seguros e não permitam que um atacante acesse arquivos não autorizados no sistema.

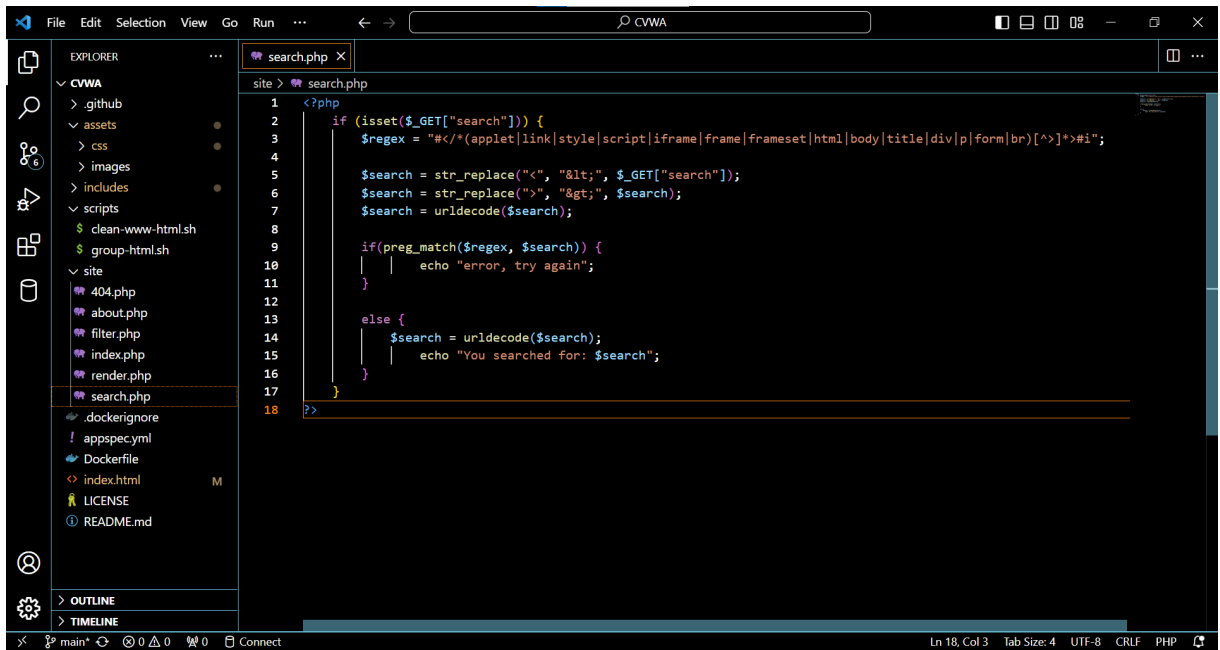
Para uma abordagem mais segura, é recomendável utilizar caminhos relativos ou estabelecer um mapeamento estrito entre os valores de `$_GET['page']` e os arquivos permitidos para inclusão. Isso ajuda a mitigar possíveis vulnerabilidades de segurança relacionadas à localização de arquivos.



Arquivo search.php

Uso Recomendado da Função 'urldecode' para Decodificar Dados de URL: A função “urldecode” é empregada para decodificar a entrada do usuário, uma prática recomendada para lidar com dados de URL. No entanto, ao realizar essa decodificação, está sendo revertido as substituições previamente feitas com str_replace, o que significa que as tags < e > serão restauradas na entrada antes da verificação pelo regex.

Para corrigir essa questão, é aconselhável realizar a verificação após a decodificação da entrada. Dessa forma, a verificação é realizada na entrada após a decodificação, assegurando que as tags HTML sejam detectadas corretamente. É importante complementar essa abordagem com outras medidas de segurança apropriadas, como validação de entrada e proteção contra injeções SQL.



Arquivo index.html

Redirecionamento de página: É importante ressaltar que o redirecionamento por meio de JavaScript pode ser desativado pelo navegador do usuário ou bloqueado por extensões de segurança. Além disso, essa não é a prática recomendada para redirecionar páginas em um site.

Se a intenção for direcionar os visitantes para a página "site/index.php", uma abordagem mais convencional e confiável é fazê-lo diretamente no lado do servidor. Isso pode ser realizado por meio de configurações de redirecionamento em seu servidor web (como o Apache ou Nginx) ou usando código em linguagens do lado do servidor, como PHP.

Exemplo:

```

<?php
    header("Location: site/index.php");
    exit;
?>

```

Essa abordagem garante um redirecionamento mais confiável e é recomendada em vez de usar JavaScript para esse fim.



Conclusão:

O teste de penetração (pentest) realizado na aplicação web da Conviso Application Security proporcionou uma avaliação da sua segurança, abordando diversos aspectos críticos relacionados à proteção dos dados e à integridade do sistema. A análise minuciosa do código-fonte, das políticas de autenticação e autorização, bem como da configuração do servidor web, revelou insights cruciais sobre a postura de segurança da aplicação.

Anexos:

Em anexo, encontram-se os relatórios das ferramentas automatizadas que foram utilizadas: OWASP ZAP e Wapiti.

- [2023-09-20-ZAP-Report-.html](#)
- [localhost_8080_09202023_1728.html](#)