3º Projeto Prático - Segurança da Informação

JOÃO LUCAS GOMES PINHEIRO

Utilizaremos neste projeto a aplicação OWASP Juice Shop. Uma aplicação que autoriza terceiros a testarem vulnerabilidades.

Neste projeto seguiremos algumas etapas, como, preparar o ambiente OWASP Juice Shop, analisar a aplicação, explorar as vulnerabilidades e emitir um relatório com as vulnerabilidades encontradas e possíveis soluções

1. Preparando o ambiente:

1.1 No meu sistema Kali Linux, utilizei o comando definido na documentação da aplicação(OWASP Juice Shop) para baixar a imagem da aplicação (docker pull bkimminich/juice-shop).

1.2 Em seguida, utilizei o comando “sudo docker run -d -p 3000:3000 bkimminich/juice-shop”, para rodar a aplicação na porta 3000.

2. Etapas de exploração e testes.

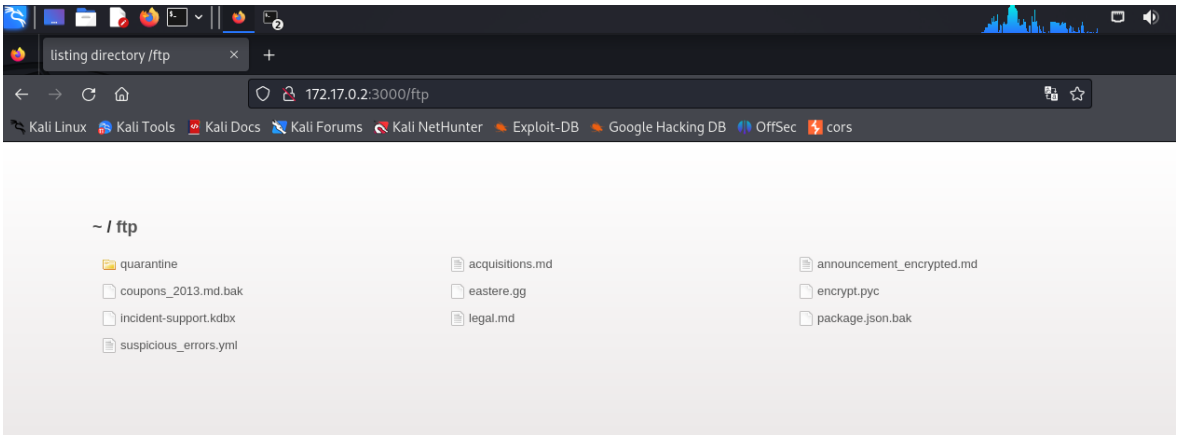
2.1 Coletar informações e reconhecimento:

- Abrir e explorar a aplicação manualmente.
- Utilizar as ferramentas Nmap, dirb e gobuster, para detectar as portas e fazer uma varredura na rede para descobrir possíveis tecnologias sendo utilizadas na aplicação, diretórios de fácil acesso e outros:

“nmap -p 3000 “ip em que a aplicação está rodando””

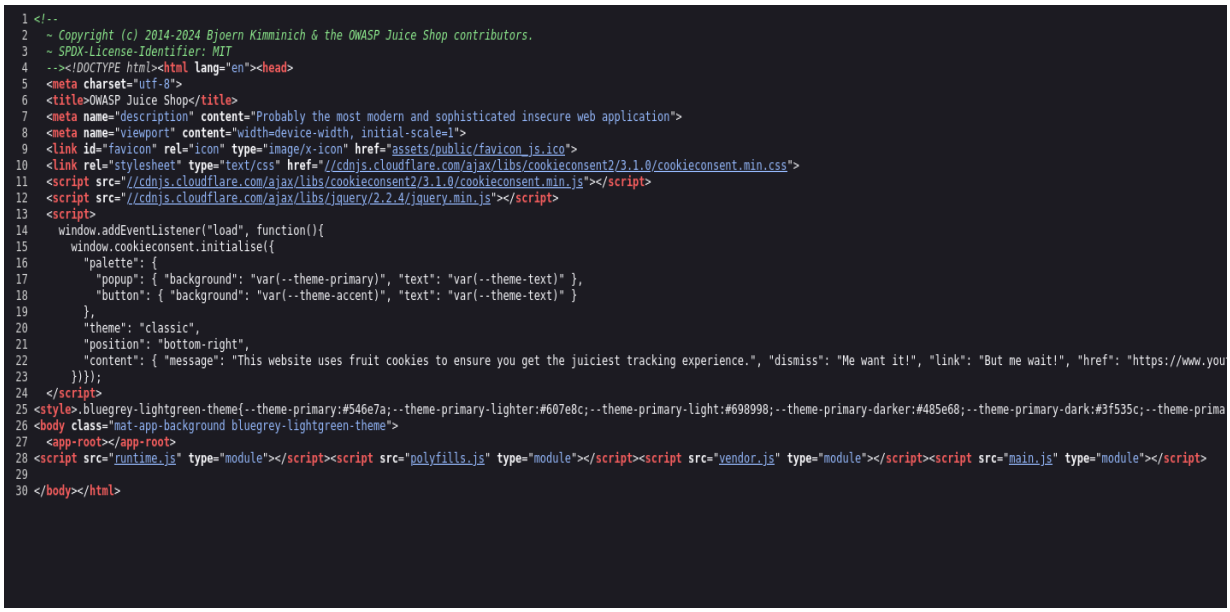
“gobuster dir -u http://172.17.0.2:3000 -w /usr/share/wordlists/dirb/common.txt”

“dirb http://172.17.0.2:3000 /usr/share/wordlists/dirb/common.txt”



*Esse foi um dos diretórios confidenciais encontrados pelo dirb

- Inspecionar a página(também podemos encontrar informações importantes ao inspecionar a página, por exemplo, rastros da própria linguagem de programação utilizada no desenvolvimento da aplicação.



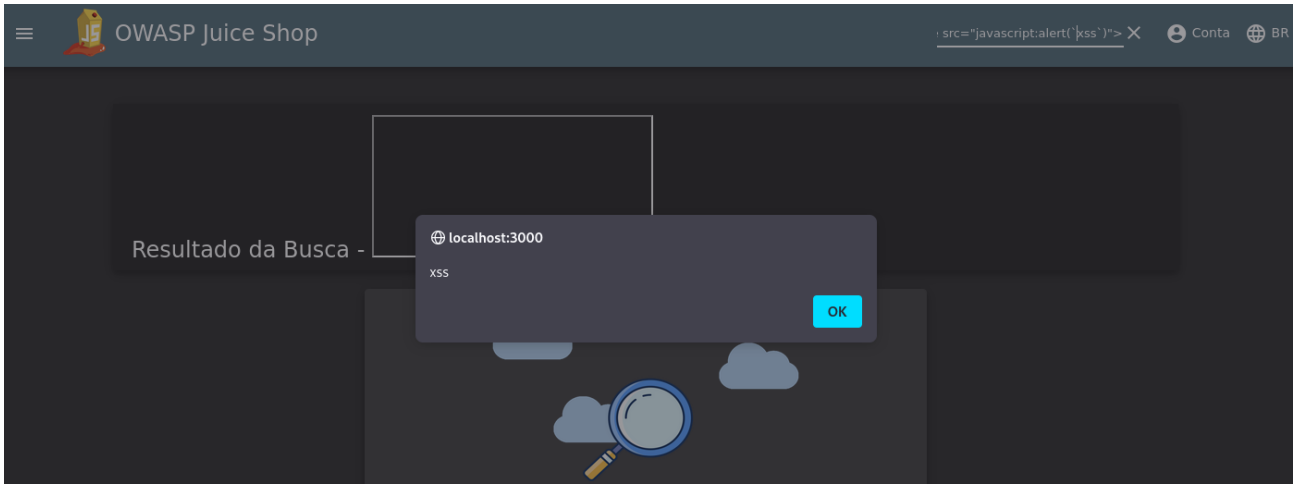
3. Testes de entrada de dados

3.1 XSS:

Na aba de serviços de ‘comentários dos clientes’, vamos inserir um script(XSS) para verificar como o aplicação irá lidar.

<script>alert('XSS Test');</script>

<iframe src="javascript:alert(`xss`)">

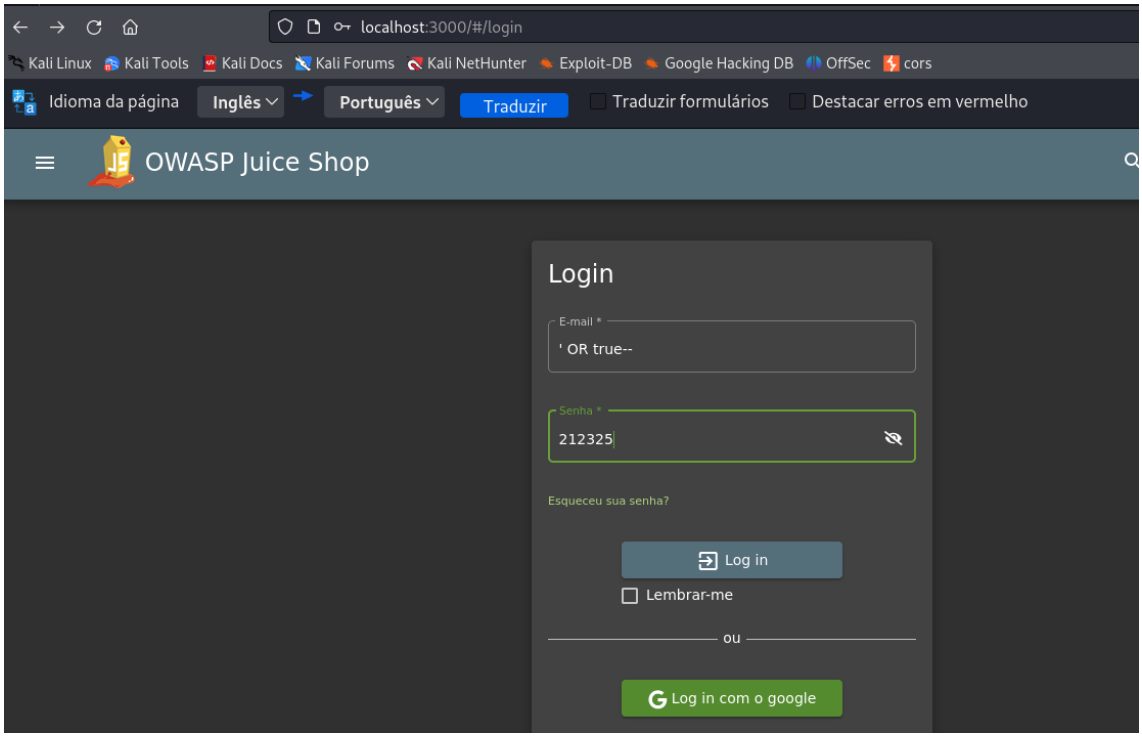


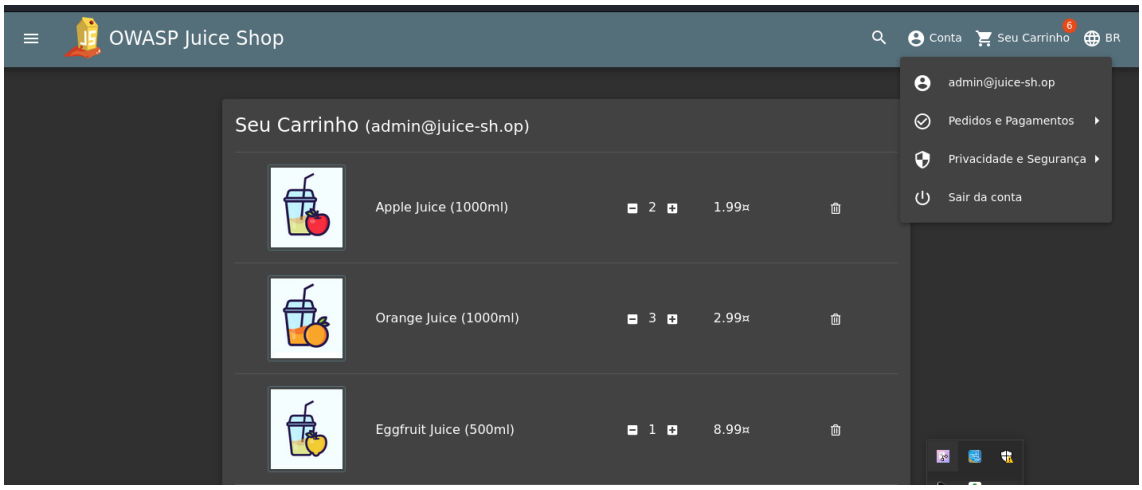
*Segundo código realizado com sucesso

3.2 SQL injection:

Agora, faremos outro teste de entrada de dados, tentando acessar uma área restrita(apenas para administradores), utilizando SQLinjection, inserindo na área de login e no campo ‘e-mail’ a sintaxe ' OR true-- e uma senha aleatória no campo ‘senha’.

Obs: Também podemos antes mesmo de tentar utilizar ataques aleatórios de SQLinjection, tentar mapear essas vulnerabilidades com o SQLmap.





*Entramos.

4. Testes de autenticação e autorização

4.1 Brute force com Hydra:

- Precisaremos pegar a url na aba de serviços de login(http://localhost:3000/#/login).

- Em seguida interceptaremos a requisição de login, pelo Burpsuite, inserindo qualquer e-mail e senha(apenas para interceptar e não logar) para podermos encontrar os campos necessários(variáveis) para logar.

-Logo, inserimos o comando hydra para ele utilizar de uma lista de e-mails e senhas para testar várias combinações (sudo hydra -L ~/Downloads/email-top-100-domains.txt -P ~/Downloads/rockyou.txt 172.17.0.2 -s 3000 http-get /rest/user/login)

*E o resultado será que a aplicação cairá, pois provavelmente a aplicação possui em seu código uma ação

para impedir que um número muito alto de requisições, ou, tentativas de login sejam feitas.

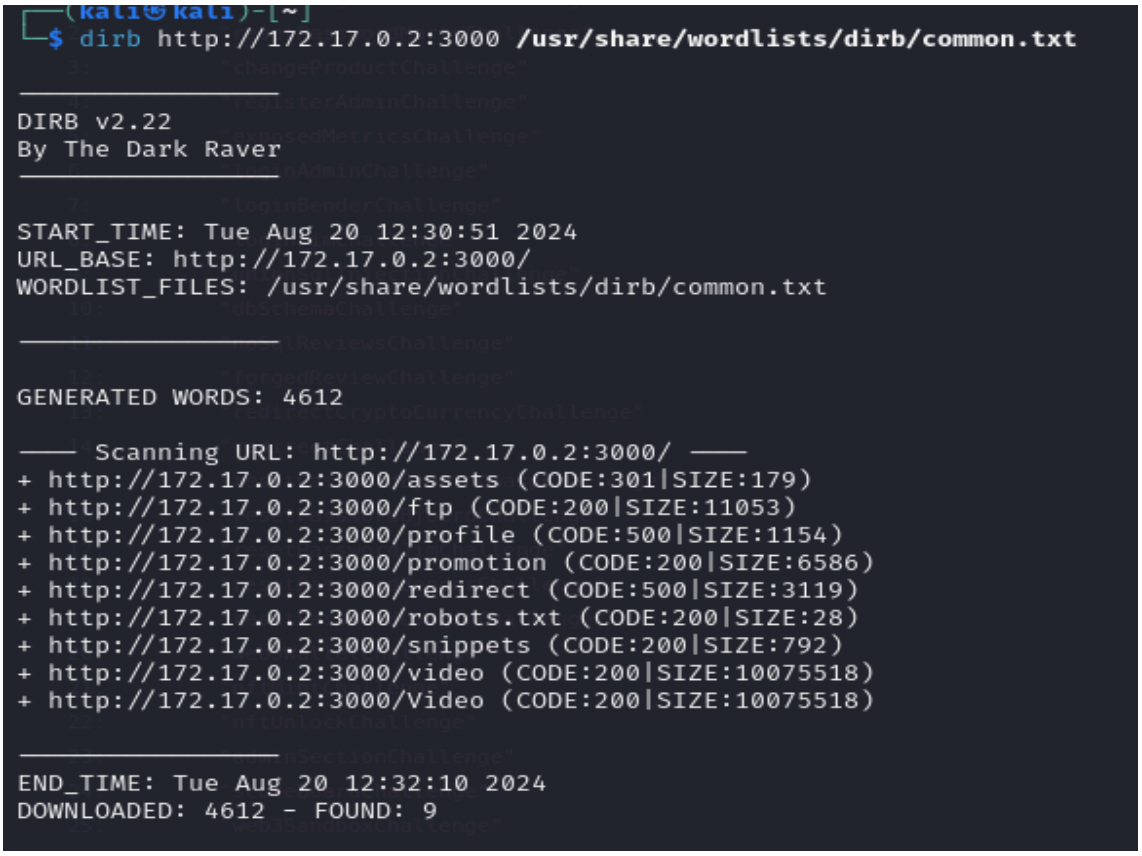
5. Vulnerabilidades Encontradas e Soluções:

5.1 Diretório Exposto

Na fase de exploração e mapeamento da aplicação, através do comando:

dirb http://172.17.0.2:3000 /usr/share/wordlists/dirb/common.txt

Foram encontrados diversos diretórios confidenciais:



Solução recomendada: Configurar permissões adequadas para o diretório para impedir acesso não autorizado.

5.2 Cross-Site Scripting (XSS)

A aplicação permite a execução de scripts maliciosos injetados pelos usuários.

Script testado:

```
<script>alert('XSS Test');</script>
<iframe src="javascript:alert(`xss`)">
```

Solução Recomendada: Implementar validação e sanitização de entradas de dados, utilizando técnicas como escaping de HTML e frameworks que protejam contra XSS.

5.3 SQL injection

A aplicação é vulnerável a injeções SQL através de campos de entrada.

Sintaxe realizada no teste: ' OR true--

Solução Recomendada: Validar e sanitizar entradas de dados.

5.4 BruteForce(Hydra)

O sistema caiu ao tentar realizar um ataque de brute force, indicando que existe uma proteção.

Solução Recomendada: Melhorar a proteção contra brute force com medidas como CAPTCHA e bloqueio de IP após múltiplas tentativas falhas.

Conclusão:

O projeto demonstrou a eficácia de ferramentas e técnicas para explorar vulnerabilidades em aplicações web. As principais vulnerabilidades identificadas foram a exposição de diretórios confidenciais, XSS e SQL Injection. Medidas de mitigação foram sugeridas para cada vulnerabilidade, focando na segurança e integridade da aplicação. As proteções contra brute force também foram observadas como uma medida positiva.

João Lucas Gomes Pinheiro

Turma 1 – Segurança da Informação

2024