Cibersegurança

Grupo 11

Nota: Encontram-se fotografias de todos os exercícios realizados nas pastas com os nomes correspondentes.

# Exercício 1

## Exercício i)

Na biblioteca marsdb, existe uma vulnerabilidade critíca de “Command Injection”, esta vulnerabilidade enquadra-se nos riscos de segurança OWASP como A1 – Injection, pois encontramo-nos numa situação que é possível ser injetado comandos que causem danos.

A outra vulnerabilidade critíca é “Authorization Bypass”, que implica que é possível passar ao lado de autorizações necessárias para aceder a recursos de alguma forma. Esta vulnerabilidade de acordo com a nossa análise enquadra-se em A5 – Broken Access Control, pois permite a passar por controls de segurança, neste caso de autenticação.

## Exercício ii)

Foram identificadas duas vulnerabilidades de Cross-Site-Scripting (XSS), e ambas se encontram presentes no package sanitize-html.

# Exercício 2

Neste exercício, andámos a procurar como eram feitos os carregamentos de ficheiros de servidor, como imagens, ficheiros pré carregados, etc. Apenas tivemos sorte quando chegámos à página de about us, e tentámos ver os termos de utilização, que eram um download. Encontrámos esse link, que era <https://g11-cs-isel.herokuapp.com/ftp/legal.md>, e alterámos para o link <https://g11-cs-isel.herokuapp.com/ftp/aquisitions.md>, que funcionou bem e fez download do ficheiro.

As primeiras três linhas são :

# Planned Acquisitions

> This document is confidential! Do not distribute!

Our company plans to acquire several competitors within the next year.

# Exercício 3

Este exercício foi bastante simples, pois apenas implicou abrir a developer tool do Firefox, e pesquisar pela palavra chave “scoreboard”, após isso foi apresentado um url que continha essa palavra. Ao testarmos esse url, pareceu ser exatamente o scoreboard. O url é <https://g11-cs-isel.herokuapp.com/api/Challenges/?name=Score%20Board>

# Exercício 4

Na página de resultados o texto do critério de pesquisa é inserido exclusivamente no span com id “searchValue”, que é o sitío que aparece à frente de “Search Results”. A caixa de texto de pesquisa utilizada para pesquisar “lemon” continua com essa mesma string. A partir daqui conseguimos inferir, que o valor da searchBox apenas é utilizado para construir o url, e não atualizada com o valor do mesmo, enquanto que o span “searchValue” é sempre atualizado com o valor mais recente do url no caso de uma pesquisa.

# Exercício 5

Neste exercício começámos por tentar injetar o texto fornecido no único sítio onde conseguíamos inserir texto até ao momento, que era na barra de pesquisa. Para tal começámos por testar aí, pois para efetuar um ataque de XSS temos sempre de injetar código no site de alguma forma. Testando com o texto apresentado no enunciado, após pesquisar apareceu o alerta a dizer “xss” que era o esperado, de acordo com o texto apresentado.

# Exercício 6

Para este exercício era proposto que encontrássemos a password da conta de administrador do serviço, para fazer isto, fizemos uma primeira tentativa com o user [admin@juice-sh.op](mailto:admin@juice-sh.op), e com a password “admin000”, pois ia de encontro ao explicado no enunciado. Após isso fomos ver o pedido no ZAP, e encontrámos, e escolhemos a opção Attack/Fuzzing.

Para efetuar este ataque é necessário escolher que parte será fuzzed, e foi a password, e qual será a regra de fuzzing. No nosso caso optámos por utilizar REGEX para isto, tendo em conta que nos pareceu o mais simples de implementar, a regra utilizada foi “(admin[0-9]{3})”, que basicamente vai verificar todas as passes com admin e 3 digítos à frente, totalizando 1000 pedidos. Após isto esperámos até terminarem todos os pedidos, e procurámos se algum tinha retornado um código HTTP diferente de um da série dos códigos 4XX. Um deles, nomeadamente o com a password admin123, retornou 200 – OK, pelo que percebemos que era a password.

Por fim fomos ao site e testámos esse e-mail com essa password e conseguimos efetuar login.