Instituto Superior de Engenharia de Lisboa

Cibersegurança, Módulo 2

**Trabalho Prático 2**

Mestrado em Engenharia Informática de Multimédia

Pedro Gonçalves, 45890

Rodrigo Dias, 45881

Rúben Santos, 49063

Semestre de Inverno, 2021/2022

**1)**

Uma imagem com texto

Descrição gerada automaticamenteUma imagem com texto, monitor, interior, preto

Descrição gerada automaticamenteO repositório para o grupo **11** foi criado com sucesso.

**2)**

Foi gerado o seguinte ficheiro de ***workflow***:

Uma imagem com texto

Descrição gerada automaticamente

**3)**

Uma imagem com texto, captura de ecrã, monitor, interior

Descrição gerada automaticamenteA vulnerabilidade ***“Database query built from user-controlled sources”*** sobre o ficheiro **routes/login.ts** foi devidamente identificada pela **Github Action CodeQL**.

O **CodeQL** detetou esta linha de código como vulnerável porque existe uma potencial vulnerabilidade a injeções **SQL**. A ***query*****SQL** criada nesta linha de código será concatenada com dados obtidos diretamente do utilizador, sem que exista qualquer tipo de validação (***sanitization***) sobre os mesmos. A informação concatenada com uma ***query***em **SQL**, de forma a evitar injeções, deve ser sempre devidamente validada para que quaisquer valores inesperados sejam manuseados de forma correta e consistente.

A fonte (***source***) da vulnerabilidade são os dados inseridos pelo utilizador. O destino (***sink***) será a ***query*****SQL** efetuada em código que pode retornar potencial informação sensível.

**4)**

No contexto das injeções **SQL**, um exemplo de falso positivo identificado por uma ferramenta de análise de vulnerabilidades, seria uma ***query*****SQL** que utiliza dados inseridos pelo utilizador, apesar de terem sido devidamente e previamente validados. De modo a tratar-se realmente de um falso positivo, o código deve cobrir todas as rotas de ataque possíveis (***sinks***), de modo validar os dados de forma consistente.

Um exemplo de falso negativo seria uma ferramenta de análise de vulnerabilidades determinar que os dados inseridos pelo utilizador são devidamente validados, mas na realidade existem rotas de ataque (***sinks***) que podem ser exploradas.

**5)**

Dado que na função onde a ***query*****SQL** está explicitada, antes da linha de código crítica (onde é efetuada a ***query***), é chamada a função **verifyPreLoginChallenges*()***que, pelo nome e parâmetros que aceita, deveria realizar a validação dos dados inseridos pelo utilizador, conclui-se que a análise que identificou esta vulnerabilidade é de contexto local. Deste ponto de vista, talvez a vulnerabilidade encontrada até se trate de um falso positivo, no caso de a validação dos dados por parte da função **verifyPreLoginChallenges*()***ser feita de forma correta, consistente e cubra todas as rotas de ataque possíveis.