<u>Definição das vulnerabilidades</u>

No seguinte documento iremos fazer uma descrição de vulnerabilidades existentes no nosso código, iremos fazer uma breve introdução da vulnerabilidade inclusive alguns riscos que estas possam ter. Para referencia o nossos projeto consiste num sistema simples de troca de mensagens entre utilizadores. Estes criam conta com um nome e palavra passe associada ou entram numa conta já existente. O username e password são guardadas numa base de dados MySQL. Cada cliente pode enviar mensagens para um certo user indicando o seu username ou ler mesnagens que foram enviadas para este.

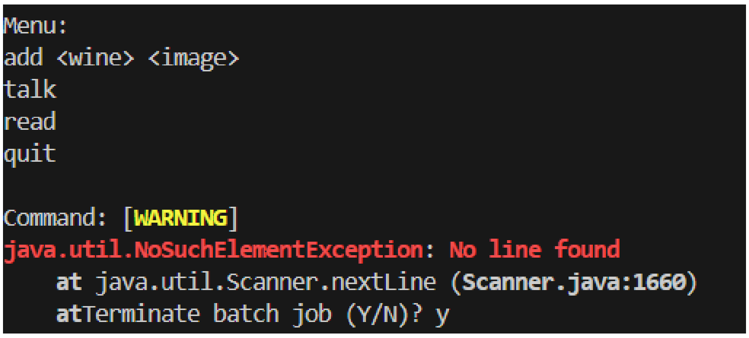
1. **<u>Suscetível a um ataque DDOS</u>**

   A nossa aplicação pode ser sobrecarregado por um volume elevado e simultâneo de pedidos TCP ao servido. Impedindo este de fazer login a novos clientes e de clientes já com o log in feito de executarem os comandos durante o ocorrer do ataque. Esta vulnerabilidade é possível sendo que não temos no nosso código uma vez que não implementamos “rate limiting” para os pedidos que o servidor pode receber. Exploramos esta vulnerabilidade ao enviar um grande volume de pedidos TCP para o nosso servidor a partir de múltiplos users virtuais de modo a simular uma grande quantidade de tráfico.

   Estas 3 imagens mostram o estado do servidor durante o ataque e <u>tentativas</u> falhas de login e de enviar uma mensagem no lado do cliente durante a duração do ataque.

```
        at java.base/java.io.ObjectInputStream$BlockDataInputStream.readShort(ObjectInputStream.java:3424)
        at java.base/java.io.ObjectInputStream.readStreamHeader(ObjectInputStream.java:983)
        at java.base/java.io.ObjectInputStream.<init>(ObjectInputStream.java:414)
        at group10.mavenproject.TintolmarketServer$ServerThread.run(TintolmarketServer.java:144)
java.io.EOFException
        at java.base/java.io.ObjectInputStream$PeekInputStream.readFully(ObjectInputStream.java:2929)
        at java.base/java.io.ObjectInputStream$BlockDataInputStream.readShort(ObjectInputStream.java:3424)
        at java.base/java.io.ObjectInputStream.readStreamHeader(ObjectInputStream.java:983)
        at java.base/java.io.ObjectInputStream.<init>(ObjectInputStream.java:414)
        at group10.mavenproject.TintolmarketServer$ServerThread.run(TintolmarketServer.java:144)
java.io.EOFException
        at java.base/java.io.ObjectInputStream$PeekInputStream.readFully(ObjectInputStream.java:2929)
        at java.base/java.io.ObjectInputStream$BlockDataInputStream.readShort(ObjectInputStream.java:3424)
        at java.base/java.io.ObjectInputStream.readStreamHeader(ObjectInputStream.java:983)
        at java.base/java.io.ObjectInputStream.<init>(ObjectInputStream.java:414)
        at group10.mavenproject.TintolmarketServer$ServerThread.run(TintolmarketServer.java:144)
java.io.EOFException
        at java.base/java.io.ObjectInputStream$PeekInputStream.readFully(ObjectInputStream.java:2929)
        at java.base/java.io.ObjectInputStream$BlockDataInputStream.readShort(ObjectInputStream.java:3424)
        at java.base/java.io.ObjectInputStream.readStreamHeader(ObjectInputStream.java:983)
        at java.base/java.io.ObjectInputStream.<init>(ObjectInputStream.java:414)
        at group10.mavenproject.TintolmarketServer$ServerThread.run(TintolmarketServer.java:144)
```

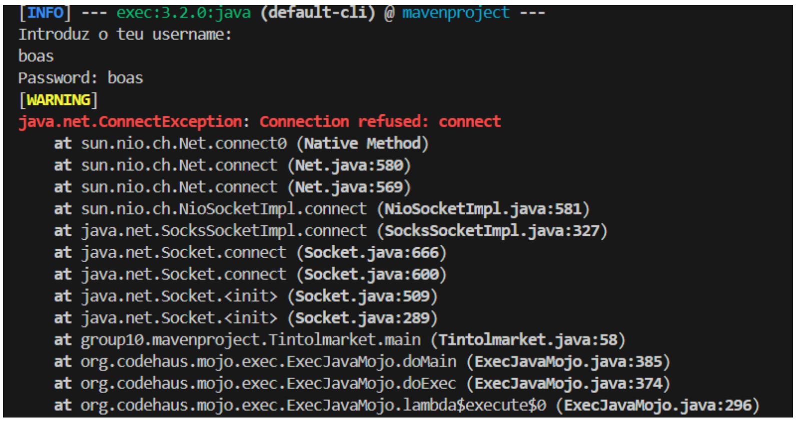
Figura 1servidro durante o ataque



```
Menu:
add <wine> <image>
talk
read
quit

Command: [WARNING]
java.util.NoSuchElementException: No line found
    at java.util.Scanner.nextLine (Scanner.java:1660)
    atTerminate batch job (Y/N)? y
```

Figura 2Tentativa de executar o read durante um ataque



```
[INFO] --- exec:3.2.0:java (default-cli) @ mavenproject ---
Introduz o teu username:
boas
Password: boas
[WARNING]
java.net.ConnectException: Connection refused: connect
    at sun.nio.ch.Net.connect0 (Native Method)
    at sun.nio.ch.Net.connect (Net.java:580)
    at sun.nio.ch.Net.connect (Net.java:569)
    at sun.nio.ch.NioSocketImpl.connect (NioSocketImpl.java:581)
    at java.net.SocksSocketImpl.connect (SocksSocketImpl.java:327)
    at java.net.Socket.connect (Socket.java:666)
    at java.net.Socket.connect (Socket.java:600)
    at java.net.Socket.<init> (Socket.java:509)
    at java.net.Socket.<init> (Socket.java:289)
    at group10.mavenproject.Tintolmarket.main (Tintolmarket.java:58)
    at org.codehaus.mojo.exec.ExecJavaMojo.doMain (ExecJavaMojo.java:385)
    at org.codehaus.mojo.exec.ExecJavaMojo.doExec (ExecJavaMojo.java:374)
    at org.codehaus.mojo.exec.ExecJavaMojo.lambda$execute$0 (ExecJavaMojo.java:296)
```

Figura 3 tentativa de login durante o ataque

2. **SQL Injection:**

   Durante o desenvolvimento do nosso projeto, implementamos vulnerabilidades como SQL Injection e ocorre quando um invasor tenta manipular as consultas SQL enviadas a base de dados através dos comandos do sistema da aplicação, permitindo contornar processos e aceder informações confidenciais ou realizar ações não autorizadas.
   Um exemplo de como a vulnerabilidade foi introduzida em nosso sistema foi a mudança de consultas prepared statement para consultas statement e o uso de concatenações para formar as consultas SQL que procuram as tabelas. Isso permitiu que os invasores fizessem ações maliciosas, manipulando as consultas SQL para alcançar acesso não autorizado ao sistema.

3. **Transmissão de mensagens não criptografadas:**
   A nossa aplicação transmite data entre o server e os seus clientes por sockets utilizando o protocolo TCP não utilizado qualquer tipo de encriptação nas mensagens transmitidas. Isto pode levar ao acesso não autorizado de data como por exemplo as credenciais de inicio de sessão de um utilizador no momento que este se conecta ao server e faz log in.
   De modo a verificar esta vulnerabilidade analisamos o tráfego da rede enquanto executávamos a aplicação e eventualmente encontramos algumas das mensagens a serem transmitidas entre o servidor e os clientes e que era possível ver os conteúdos das mensgans.