

## Práctica 5: Condiciones de carrera

---

En esta práctica se considera la condición de carrera TOCTOU (Time of Check Time of Use) y sus implicaciones en la seguridad del software. Se proporcionan API-Java vulnerables a TOCTOU. Se pide escribir un programa que utilice las API, identificar y explotar la vulnerabilidad, y finalmente proponer una solución para las API que elimina dicha vulnerabilidad. En la bola extra, opcional, se proporciona un programa en C++ también vulnerable a TOCTOU. Se pide identificar la ventana de vulnerabilidad, realizar una explotación e implementar una versión parcheada del programa.

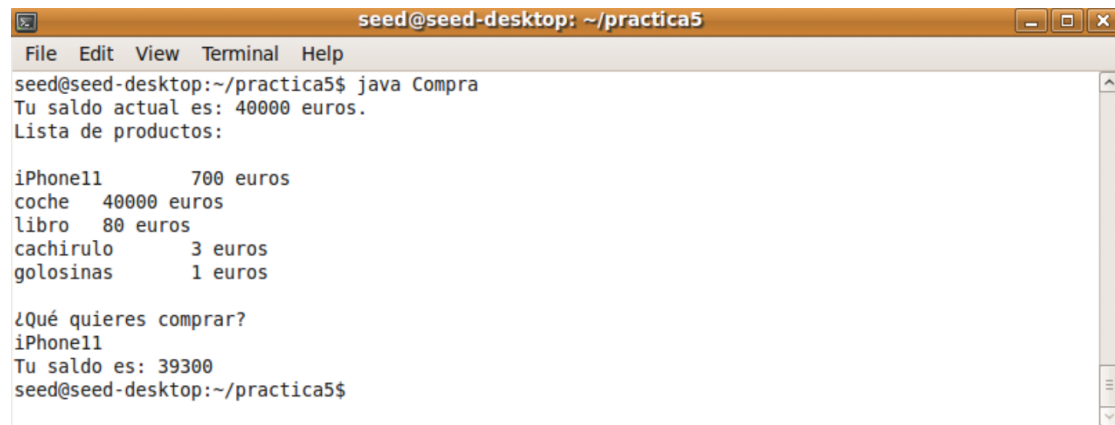
### 1. ENTORNO DE PRÁCTICA

La práctica se lleva a cabo utilizando la misma máquina virtual de la práctica anterior (enlace disponible en Moodle).

### 2. IMPLEMENTACIÓN DE UN PROGRAMA VULNERABLE

La primera tarea es implementar un programa en Java (`Compra.java`) que utilice las API proporcionadas en la sub-carpeta *api* de la carpeta *practica5*, para la compra de productos en una tienda virtual. El escenario de ejecución del programa es el siguiente (véase ejemplo de ejecución en la Figura 2.1):

1. Escribe en pantalla el saldo de la cuenta.
2. Escribe en pantalla la lista de productos con los precios.
3. Pide el producto que se quiere comprar.
4. Controla si el saldo de la cuenta es suficiente para comprar el producto (introducido por teclado). Si el saldo no es suficiente, termina la ejecución.
5. Si el saldo es suficiente, actualiza la cuenta (resta de la cuenta la cantidad correspondiente al precio del producto).
6. Escribe el saldo actualizado de la cuenta y termina la ejecución.



```
seed@seed-desktop: ~/practica5
File Edit View Terminal Help
seed@seed-desktop:~/practica5$ java Compra
Tu saldo actual es: 40000 euros.
Lista de productos:

iPhone11      700 euros
coche  40000 euros
libro   80 euros
cachirulo    3 euros
golosinas    1 euros

¿Qué quieres comprar?
iPhone11
Tu saldo es: 39300
seed@seed-desktop:~/practica5$
```

Figura 2.1: Ejemplo de ejecución del programa.

En particular, las API (documentadas en el código fuente) incluyen:

- `Cuenta.java`: interacciona con la cuenta.
- `Carrito.java`: interacciona con el carrito.
- `Tienda.java`: interacciona con los productos y precios.

## 2.1. EXPLOTACIÓN DE LA VULNERABILIDAD

Se recuerda que una condición de carrera ocurre cuando múltiples procesos (o hilos) concurrentes acceden y manipulan un mismo recurso, y el resultado de la ejecución depende del orden en el que el acceso se efectúa. El objetivo es explotar la vulnerabilidad TOCTOU en el programa. ¿Se puede conseguir un coche pero pagando menos de su valor? Obviamente no está permitido modificar los ficheros `cuenta.txt` o `carrito.txt` que no sea a través de las API.

**Pregunta 1.** ¿Cuáles son los recursos compartidos? ¿Quién los comparte? ¿Cuál es la ventana temporal de vulnerabilidad? Indica las líneas en el código que marcan la ventana. Explica en detalle los pasos a hacer para conseguir el objetivo, por ejemplo, comprar dos coches con un saldo de 40.000 euros en la cuenta.

Sugerencia. Puedes añadir acciones de retraso en el código implementado para que el ataque tenga éxito.

## 3. CORRECCIÓN DE LAS API

En esta tarea se pide corregir las API para evitar la vulnerabilidad de condición de carrera que ha sido explotada en la tarea anterior. En particular, hay que implementar un nuevo método para `Cuenta.java`:

```
public void sacarDinero(int cantidad) throws Exception
```

que incluya las protecciones necesarias.

**Pregunta 2.** ¿Hay otras API vulnerables a la condición de carrera? Si las hay, explica como puede ocurrir la condición de carrera y actualiza las API para eliminarlas. ¿Las protecciones incluidas en las API modificadas son suficientes?

Finalmente, se pide implementar una nueva versión `CompraSegura.java` para verificar el funcionamiento correcto de las nuevas API.

#### 4. BOLA EXTRA: OTRO PROGRAMA VULNERABLE

**Este apartado no es guiado y es opcional.**

En la sub-carpeta *extra* de la carpeta *practica5* hay un programa en C++ vulnerable a TOCTOU. Supongamos que el programa es un programa que se ejecuta con privilegios de root (`set-UID root`).

TAREA 1. Identifica la ventana temporal de vulnerabilidad e indica las líneas en el código que marcan la ventana. ¿Qué objetivo podría conseguir un atacante explotando la vulnerabilidad? Explica los pasos a realizar para su explotación.

NOTA IMPORTANTE. Para evitar problemas de autenticación en el siguiente inicio de sesión en la máquina virtual, es recomendable realizar una copia de la máquina virtual antes de realizar el ataque.

TAREA 2. Implementa una nueva versión del programa `vulnPatched.cpp` para eliminar la condición de carrera del programa original o para hacer más difícil su explotación.

#### 5. PUNTUACIÓN

La valoración máxima de esta práctica es **10 puntos** así repartidos:

- Parte obligatoria (Secciones 1,2,3): **8 puntos**
- Bola extra (Sección 4): **2 puntos**

#### 6. ENTREGA

Se pide entregar el siguiente material en Moodle:

- Nombre, apellidos y NIA de cada miembro del grupo de práctica, así como el correspondiente grupo de prácticas (*inf1 ... inf6*).
- Un informe donde se explique el procedimiento seguido para la realización del ataque. En particular, contestar a las preguntas, proporcionar los resultados de ejecución del programa vulnerable, y añadir observaciones sobre los resultados obtenidos. Bola extra opcional: de forma similar a la parte común, informe detallado de las tareas 1 y 2.
- Una carpeta comprimida con: el programa vulnerable `Compra.java`, las API modificadas, el programa `CompraSegura.java` que usa las API modificadas, un fichero `README` con las instrucciones para la compilación y ejecución de los programas. Bola extra opcional: el código utilizado para la explotación del programa vulnerable, el código revisado `vulnPatched.cpp` con comentarios.

La fecha límite de entrega de esta práctica es **tres semanas desde la realización de la práctica.**