Universidad Nacional de Rosario

Facultad de Ciencias Exactas, Ingeniería y Agrimensura

Licenciatura en Ciencias de la Computación

Trabajo Práctico I

Seguridad Informática

Arroyo Joaquín (A-4294/3)

2024

1. Discuta la diferencia entre datos e información. Encuentre ejemplos en donde controlar el acceso a los datos no necesariamente controla el acceso a la información.

La principal diferencia entre datos e información es el contexto y la utilidad. Los datos son valores, como números o textos, que carecen de significado por sí solos. En cambio, la información es el resultado de procesar esos datos para que tengan un propósito en un contexto específico.

Por ejemplo, los registros de temperatura en Rosario son datos; cuando se analizan para predecir el clima, se convierten en información útil.

Además, controlar el acceso a los datos no siempre controla el acceso a la información. En la bolsa de valores, aunque se restrinjan los datos de transacciones, los analistas pueden usar información pública, como informes financieros, para extraer conclusiones valiosas sobre la empresa. Así, la información puede seguir siendo accesible incluso con restricciones en los datos.

8. De un ejemplo en donde un problema de *security* es también un problema de *safety*; uno donde un problema de *security* NO es problema de *safety*; y uno donde un problema de *safety* NO es problema de *security*.

- Un ataque mediante *Buffer Overflow* es un problema de *security* porque puede permitir la ejecución de código malicioso. También representa un problema de *safety* si la ejecución del código compromete la estabilidad del sistema, poniendo en riesgo su correcto funcionamiento.
- Un robo de contraseña es un problema de *security* ya que compromete la confidencialidad y la integridad de la información del usuario. Sin embargo, no afecta la *safety* del sistema, ya que no implica que algo "malo" suceda durante la ejecución.
- Un programa que no verifica los tipos de los inputs que recibe puede llevar a fallos en la ejecución, constituyendo un problema de *safety*. Sin embargo, no afecta la *security* del sistema, ya que no implica una violación de la confidencialidad o integridad de los datos.

13.

a) Investigue el modelo de Biba para la integridad.

b) ¿Es posible utilizar los modelos de Bell-LaPadula y Biba para modelar simultáneamente la confidencialidad y la integridad? ¿Se pueden emplear las mismas categorías de seguridad para ambas políticas?

El modelo de Biba establece que los sujetos solo pueden escribir hacia niveles de integridad inferiores y leer de niveles superiores, lo que se resume en la regla de *escribir hacia abajo* (write down) y *leer hacia arriba* (read up). Esto contrasta con el modelo de Bell-LaPadula, que prohíbe tanto la escritura hacia abajo como la lectura hacia arriba, centrándose en la protección de la confidencialidad.

En cuanto a la utilización de las mismas categorías de seguridad, sí es posible emplearlas para ambas políticas, pero es esencial implementar controles claros para garantizar que se mantengan las propiedades de confidencialidad e integridad de manera efectiva.

21. Determinar si hay flujo indebido de información en el siguiente programa:

```
x = readH();
writeL("Loading...");
writeH(x);
```

En este programa, se lee una variable `x` de un nivel bajo de seguridad (nivel `L`), y se escribe `Loading...` en un nivel bajo, y luego el valor de `x` que se leyó en un nivel alto.

Este flujo de información puede ser considerado indebido porque se permite que se escriba `Loading...` en un nivel bajo, cuando anteriormente se leyó información de nivel alto. Esta primer lectura debería restringir la escritura de toda información en nivel bajo. Luego, con la segunda escritura no habría problema.

Por lo tanto, este programa presenta un flujo indebido de información.

22. Para el programa del ejercicio anterior, explique cómo funcionaría la ejecución bajo el sistema de seguridad por multi-ejecución.

1. El proceso P comienza clasificado como L.
2. Luego del primer `readH` se realiza un fork del proceso P, obteniendo P_L y P_H.
3. El `writeL` solo puede ser ejecutado por el proceso P_L, para no permitir el *write-down* de parte de P_H.
4. Por último, el `writeH` es ejecutado por ambos procesos.

23. Resuelva los desafíos propuestos en el siguiente link: ifc-challenge.appspot.com. Para el que elija resolver, explique las conclusiones a las que arribe, por ejemplo:

- Qué tipo de ataque logró llevar adelante?
- Qué limitación tiene el sistema de reglas?
- Qué cambio habría que hacerle para impedir el ataque?

Resolví el desafío número cuatro con el siguiente código:

```
let (x = h) in l = x;
```

- **Tipo de ataque:** Se trata de un ataque de `flujo directo`, donde el valor de la variable de alto nivel (h) se filtra a la variable de bajo nivel (l) a través de una asignación indirecta utilizando una variable intermedia (x).
- **Limitación del sistema de reglas:** Aunque se prohíbe la asignación directa de h a l, en los `let in` no se restringe la propagación de valores de variables de alto nivel hacia variables de bajo nivel. Esto permite asignar `l = x`, filtrando el valor de h a l de forma indirecta.
- **Cambio para evitar el ataque:** Se debería imponer restricciones en los `let in`, de forma que variables de alto nivel no puedan ser usadas para modificar variables de bajo nivel, tal como ocurre en otras expresiones.