

Aspectos de seguridad de sistemas informáticos

Trabajo Obligatorio

Diego Franggi - 210434
Cono Rodríguez - 144425

8 de diciembre de 2020

[Link al repositorio](#)

Índice general

1.	Guía del repositorio y proyecto	2
2.	Primera parte	2
2.1.	Resultados	4
3.	Segunda parte	4
3.1.	Sobre canal encubierto	4
3.2.	Sobre concurrencia	5
3.3.	Resultados	6
4.	Anexo	7
4.1.	Parte 1	7

1. Guía del repositorio y proyecto

Esta documentación se entrega en conjunto al contenido del repositorio de git donde se puede ver la evolución del proyecto. Dentro del repositorio se encuentran dos liberaciones que marcan la parte uno y dos del proyecto.

Una vez con el proyecto veremos 4 carpetas, la primera contiene elementos pertinentes a la documentación, como textos de salida o el log para la segunda parte. Luego encontramos las carpetas referentes a las parte uno y dos de la entrega. Para acceder al contenido de forma cómoda utilizamos el IDE **Apache Netbeans**, el cual nos permite abrir directamente los proyectos y correrlos en su propio ambiente. Ahora si se desea correr los archivos java directamente desde consola las clases sin compilar se encuentran siempre dentro de `\Parte 2 seguridad \obligatorioSeguridad \Parte 2 terminada \src \obligatorio`

Finalmente la carpeta de obligatorio refiere a lo que es el ambiente de producción y posee el contenido de la segunda parte ya que es la ultima en ser liberada.

- Para ejecutar la primera parte se ha de inicializar en startup y pasarle la ruta al archivo de comandos a leer.
- Para ejecutar segunda parte se ha de inicializar en startup y pasarle la ruta al archivo a pasar de forma encubierta.

2. Primera parte

La primera parte de la entrega se trata de un sistema seguro acorde a las reglas de Bell-LaPadula cumpliendo seguridad simple, propiedad * y la tranquilidad fuerte. Se maneja un entorno con sujetos y objetos con etiquetas de seguridad asociadas, para esta iteración no se requiere el uso de *necesidad del saber*.

Esta sección de la entrega consta principalmente de la creación del sistema de manejo de variables, se valida la correcta sintaxis de las instrucciones y su validez a nivel del modelo de seguridad planteado arriba. A continuación se muestra un ejemplo de las validaciones durante producción.

```

C:\Users\Diego\Desktop\seguridad.txt
write lyle lobj 10 is a valid command
write hal hobj 90 is a valid command
Invalid command como venimos por ahora?
read lyle lobj is a valid command
read hal lobj is a valid command
write lyle lobj 20 is a valid command
write hal lobj 200 is a valid command
read hal hobj is a valid command
write lyle hobj 150 is a valid command
read hal hobj is a valid command
read lyle lobj is a valid command
read lyle hobj is a valid command
write lyle hobj 50 is a valid command
Invalid command leer lyle lobj
write lyle lucas 70 is a valid command
read hal hobj is a valid command
write hal lobj is a valid command
write hal hobj 200 is a valid command
Invalid command Final de las instrucciones

```

Figura 1: Validación de comandos en produccion

Acorde a lo solicitado en la letra nuestra aplicación sigue el siguiente flujo para realizar validaciones y ejecutar (de ser permitido) alguna operación de lectura o escritura.

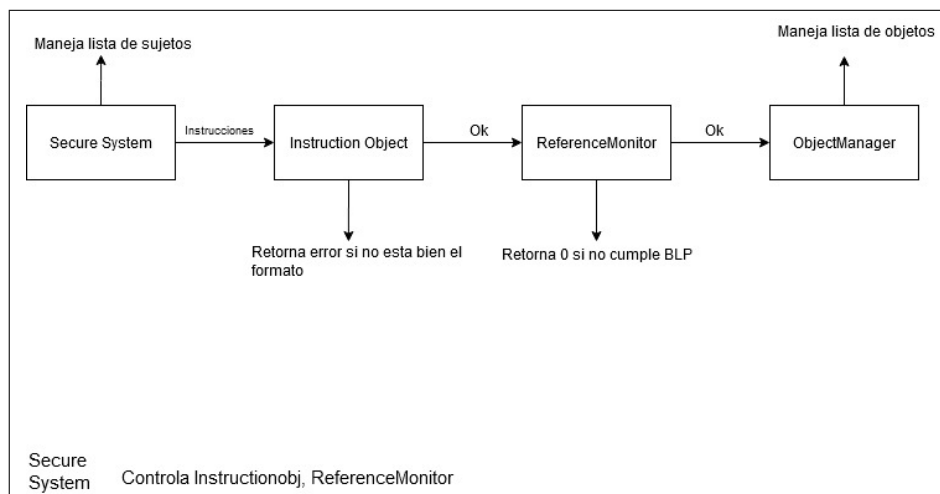


Figura 2: Flujo simple de la aplicación

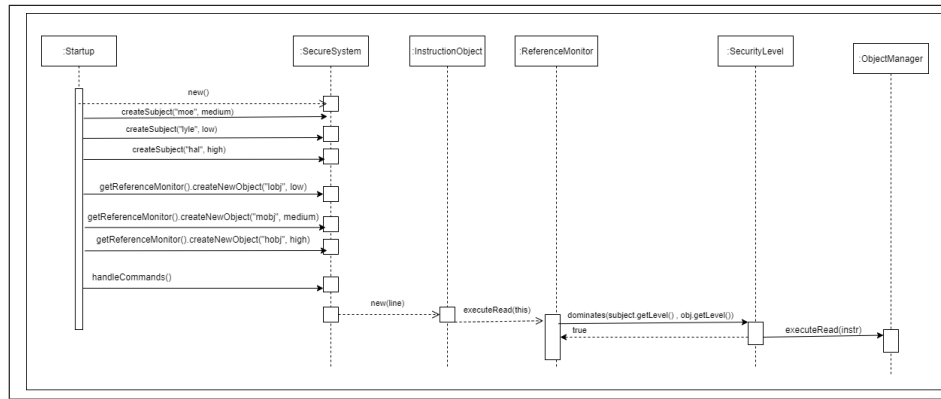


Figura 3: Diagrama de secuencia para la lectura

2.1. Resultados

Se adjuntan resultados dentro de transcripciones parte 1 para su lectura mas cómoda pero se incluye en el anexo el resultado de correr InstructionList brindado en aulas. Se corrieron las instrucciones brindadas en la letra del obligatorio llegando al resultado deseado.

3. Segunda parte

Para esta segunda iteración se nos solicita la mejora del sistema de la primera parte en conjunto a la incorporación de un canal encubierto. Para la mejora del sistema se implementaron las operaciones CREATE y DESTROY en conjunto a sus validaciones sintácticas y lógicas (de si existe o no el objeto).

Seguido a esto se incorpora el canal encubierto solicitado por medio de la incorporación del comando RUN para un sujeto dado.

Reconocemos no haber logrado cumplir los requerimientos solicitados para la segunda parte ya que **no manejamos un sistema de concurrencia** para el orden de ejecucion de las instrucciones a los sujetos. El no cumplimiento del requerimiento es detallado mas adelante pero se enfatiza que se incorporó un canal encubierto real sin transferir el archivo por fuera.

3.1. Sobre canal encubierto

Para transferir un archivo basta con enviar su dirección al startup y el mismo se encargará de convertirlo en bytes, luego por cada bit de los mismos ejecutara las instrucciones apropiadas para enviarlo secretamente, ya sea un 0 o un 1 el bit. Las instrucciones bajan hasta el reference monitor el cual mantiene un

hashmap de acceso único para cada sujeto, es decir lyle solo puede acceder a su variable dentro de este, lo mismo para los demás. Es el equivalente a tenerlo en una variable de sujeto pero ubicada ahí por prolijidad de acceso.

Ahora si el contenido de la variable del sujeto es *temp* implica que estamos frente al inicio de un byte a escribir; esta es substituida por lo ultimo leído por el sujeto. En caso de tener un byte incompleto se van concatenando apropiadamente los 0 o 1 requeridos; y así finalmente al llegar a los 8 bit (un byte completo) se habilita el retorno de una variable representante del carácter transferido de forma secreta. Finalmente desde el startup se solicita ese carácter y se manda al mensaje al archivo out de la ejecucion. Todo esto mientras se hace un log de cada ejecucion solicitada.

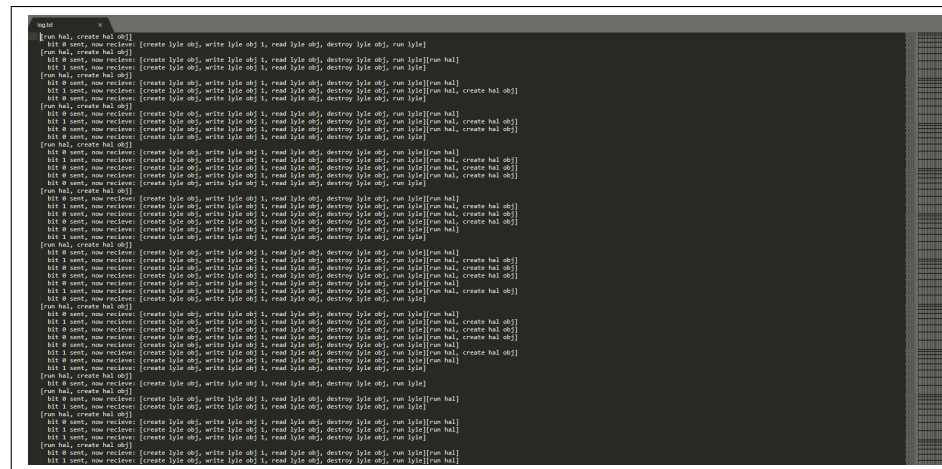


Figura 4: Ejemplo de log.txt generado tras ejecutar el canal encubierto

Nota: Tanto log.txt como el archivo generado se crean en la raíz del proyecto de netbeans.

3.2. Sobre concurrencia

No se implementó el requerimiento referente a la concurrencia, se reconoce que no se calculó acorde el tiempo requerido para la misma tras la implementación del canal. La infraestructura para la lectura y manejo del archivo de secuencia está presente, se requeriría implementar un condicional acorde a la letra a leer para ejecutar las llamadas pertinentes al envío de un 1 o un 0 y manejar que de no ser un 0 o un 1 el valor leído por lyle sea descartado ya que seria obra de moe y no de hal.

3.3. Resultados

Se obtiene correctamente el mensaje a transferir en *nombrearchivoOriginal.txt.out*, el mismo se lee con un editor de texto. El mensaje se transfiere utilizando el metodo mencionado en la sección anterior.

4. Anexo

4.1. Parte 1

Se adjunta el resultado de correr InstructionList brindada en aulas.

```
Reading from file: C:\\Users\\Diego\\Desktop\\seguridad.txt
lyle writes value 10 to lobj
The current state is:
lobj has value: 10
mobj has value: 0
hobj has value: 0
moe has recently read: 0
hal has recently read: 0
lyle has recently read: 0
-----

hal writes value 90 to hobj
The current state is:
lobj has value: 10
mobj has value: 0
hobj has value: 90
moe has recently read: 0
hal has recently read: 0
lyle has recently read: 0
-----

Bad Instruction
The current state is:
lobj has value: 10
mobj has value: 0
hobj has value: 90
moe has recently read: 0
hal has recently read: 0
lyle has recently read: 0
-----

lyle reads lobj
The current state is:
lobj has value: 10
mobj has value: 0
hobj has value: 90
moe has recently read: 0
hal has recently read: 0
lyle has recently read: 10
-----

hal reads lobj
The current state is:
lobj has value: 10
mobj has value: 0
hobj has value: 90
moe has recently read: 0
hal has recently read: 10
```


lyle has recently read: 10

lyle writes value 20 to lobj
The current state is:
lobj has value: 20
mobj has value: 0
hobj has value: 90
moe has recently read: 0
hal has recently read: 10
lyle has recently read: 10

hal writes value 200 to lobj
The current state is:
lobj has value: 20
mobj has value: 0
hobj has value: 90
moe has recently read: 0
hal has recently read: 10
lyle has recently read: 10

hal reads hobj
The current state is:
lobj has value: 20
mobj has value: 0
hobj has value: 90
moe has recently read: 0
hal has recently read: 90
lyle has recently read: 10

lyle writes value 150 to hobj
The current state is:
lobj has value: 20
mobj has value: 0
hobj has value: 150
moe has recently read: 0
hal has recently read: 90
lyle has recently read: 10

hal reads hobj
The current state is:
lobj has value: 20
mobj has value: 0
hobj has value: 150
moe has recently read: 0
hal has recently read: 150
lyle has recently read: 10

lyle reads lobj
The current state is:
lobj has value: 20

mobj has value: 0
hobj has value: 150
moe has recently read: 0
hal has recently read: 150
lyle has recently read: 20

lyle reads hobj
The current state is:
lobj has value: 20
mobj has value: 0
hobj has value: 150
moe has recently read: 0
hal has recently read: 150
lyle has recently read: 0

lyle writes value 50 to hobj
The current state is:
lobj has value: 20
mobj has value: 0
hobj has value: 50
moe has recently read: 0
hal has recently read: 150
lyle has recently read: 0

Bad Instruction
The current state is:
lobj has value: 20
mobj has value: 0
hobj has value: 50
moe has recently read: 0
hal has recently read: 150
lyle has recently read: 0

Bad Instruction
The current state is:
lobj has value: 20
mobj has value: 0
hobj has value: 50
moe has recently read: 0
hal has recently read: 150
lyle has recently read: 0

hal reads hobj
The current state is:
lobj has value: 20
mobj has value: 0
hobj has value: 50
moe has recently read: 0
hal has recently read: 50
lyle has recently read: 0

Bad Instruction
The current state is:
lobj has value: 20
mobj has value: 0
hobj has value: 50
moe has recently read: 0
hal has recently read: 50
lyle has recently read: 0

hal writes value 200 to hobj
The current state is:
lobj has value: 20
mobj has value: 0
hobj has value: 200
moe has recently read: 0
hal has recently read: 50
lyle has recently read: 0

Bad Instruction
The current state is:
lobj has value: 20
mobj has value: 0
hobj has value: 200
moe has recently read: 0
hal has recently read: 50
lyle has recently read: 0

