

Aspectos de seguridad de sistemas informáticos

Obligatorio

Ximena Moure – 175660

Ismael Puricelli – 187086

Índice

Como ejecutar la Parte 1	3
Como ejecutar la Parte 2	3
Diseño de la solución	3
SecureSystem.....	5
ReferenceMonitor.....	5
ObjectManager	5
InstructionObject	5
Subject y Object	5
Validadores y Filtros.....	5
CovertChannel	5
Evidencia de las corridas	6
Parte 1.....	6
Parte 2.....	11

Como ejecutar la Parte 1

Para que el programa pueda ejecutarse, debe existir la carpeta llamada *Entrada* en la misma ubicación donde se encuentra el ejecutable **SeguridadParte1.jar**.

Dentro de la carpeta *Entrada* debe existir el archivo llamado *parte1.txt* el cual contiene las instrucciones a ejecutar.

Para ejecutar el programa, en la consola de comandos (*cmd* en Windows) nos dirigimos a la carpeta donde se encuentra el ejecutable **SeguridadParte1.jar**.

Luego ejecutamos el siguiente comando:

```
java -jar SeguridadParte1.jar
```

Nos mostrará por consola el resultado de la ejecución.

Como ejecutar la Parte 2

Para que el programa pueda ejecutarse, debe existir la carpeta llamada *Entrada* en la misma ubicación donde se encuentra el ejecutable **SeguridadParte2.jar**.

Para ejecutar el programa, en la consola de comandos (*cmd* en Windows) nos dirigimos a la carpeta donde se encuentra el ejecutable **SeguridadParte2.jar**.

Luego ejecutamos el siguiente comando:

```
java -jar SeguridadParte2.jar <RUTA_ARCHIVO>
```

Donde RUTA_ARCHIVO es la ubicación donde se encuentra el archivo a transferirse por el canal encubierto.

Si queremos que se genere un log con las instrucciones debemos agregar el parámetro *v* al comando:

```
java -jar SeguridadParte2.jar <RUTA_ARCHIVO> v
```

El resultado de la ejecución lo vemos en la carpeta *Entrada* en el archivo *output.txt*, el cual contiene el mensaje trasferido. Si lo ejecutamos con el parámetro *v*, se tiene que haber generado también un archivo llamado *log.txt* el cual contiene las instrucciones que se generan al transmitir el mensaje por el canal encubierto.

Diseño de la solución

A continuación se muestra el diagrama UML de la solución con las relaciones más importantes.

SecureSystem

Esta clase es la encargada de administrar los sujetos y el *ReferenceMonitor* del sistema, a su vez, también se encarga de leer y ejecutar las instrucciones del archivo. Por otra parte, muestra por consola el estado del sistema una vez ejecutadas las instrucciones.

ReferenceMonitor

Una vez que recibe las instrucciones validadas en un objeto *InstructionObject*, esta clase es responsable de realizar el control de acceso de lectura y escritura de las mismas por parte de los sujetos a los objetos según sus etiquetas en base a las propiedades de Bell LaPadula (Seguridad simple y Propiedad *).

En caso de recibir una *BadInstruction* el mismo la guarda localmente.

En la Parte 2, también se encarga de ejecutar la acción *run* de cada sujeto.

ObjectManager

Esta clase se encarga de administrar los objetos del sistema, provee mecanismos de acceso de lectura y escritura por parte de los sujetos a los objetos.

InstructionObject

Esta clase se encarga de validar la sintaxis y la semántica de las instrucciones del archivo a través de los validadores mencionados en la sección [Validadores y Filtros](#). A su vez representa una instrucción en el sistema.

Subject y Object

Son las clases responsables de representar los sujetos y objetos del sistema.

Validadores y Filtros

Son un conjunto de clases encargadas de validar que las instrucciones leídas por el programa sean válidas.

Dichas validaciones se implementaron utilizando el patrón *Pipes & Filters*.

CovertChannel

Esta clase tiene como objetivo transferir los bits de un archivo dado a través del canal encubierto para el sistema y generar las instrucciones necesarias para su transferencia.

Evidencia de las corridas

Parte 1

Entrada:

```
write hal hobj  
read hal  
write lyle lobj 10  
read hal lobj  
write lyle hobj 20  
write hal lobj 200  
read hal hobj  
read lyle lobj  
read lyle hobj  
foo lyle lobj  
hi lyle, This is hal  
The missile launch code is 1234567
```

Salida:

```
Bad Instruction
The current state is:
    Lobj has value: 0
    Hobj has value: 0
    lyle has recently read: 0
    hal has recently read: 0
```

```
Bad Instruction
The current state is:
    Lobj has value: 0
    Hobj has value: 0
    lyle has recently read: 0
    hal has recently read: 0
```

```
lyle writes value 10 to lobj
The current state is:
    Lobj has value: 10
    Hobj has value: 0
    lyle has recently read: 0
    hal has recently read: 0
```

```
hal reads lobj
The current state is:
    Lobj has value: 10
    Hobj has value: 0
    lyle has recently read: 0
    hal has recently read: 10
```

```
lyle writes value 20 to hobj
The current state is:
    Lobj has value: 10
    Hobj has value: 20
    lyle has recently read: 0
    hal has recently read: 10
```

```
hal writes value 200 to lobj
The current state is:
    Lobj has value: 10
    Hobj has value: 20
    lyle has recently read: 0
    hal has recently read: 10
```

```
hal reads hobj
The current state is:
    Lobj has value: 10
    Hobj has value: 20
    lyle has recently read: 0
    hal has recently read: 20
```

```
lyle reads lobj
The current state is:
    Lobj has value: 10
    Hobj has value: 20
    lyle has recently read: 10
    hal has recently read: 20
```

```
lyle reads hobj
The current state is:
    Lobj has value: 10
    Hobj has value: 20
    lyle has recently read: 0
    hal has recently read: 20
```

```
Bad Instruction
The current state is:
    Lobj has value: 10
    Hobj has value: 20
    lyle has recently read: 0
    hal has recently read: 20
```

```
Bad Instruction
The current state is:
    Lobj has value: 10
    Hobj has value: 20
    lyle has recently read: 0
    hal has recently read: 20
```

```
Bad Instruction
The current state is:
    Lobj has value: 10
    Hobj has value: 20
    lyle has recently read: 0
    hal has recently read: 20
```

Entrada:

```
hola  
write hal hobj  
read hal  
write lyle lobj 10  
read hal lobj  
write lyle hobj 20  
write hal lobj 200  
writes hal lobjf 300  
read hal hobj  
reads lyle lobj  
bye
```


Salida:

```
Bad Instruction
The current state is:
    Lobj has value: 0
    Hobj has value: 0
    lyle has recently read: 0
    hal has recently read: 0
```

```
Bad Instruction
The current state is:
    Lobj has value: 0
    Hobj has value: 0
    lyle has recently read: 0
    hal has recently read: 0
```

```
Bad Instruction
The current state is:
    Lobj has value: 0
    Hobj has value: 0
    lyle has recently read: 0
    hal has recently read: 0
```

```
lyle writes value 10 to lobj
The current state is:
    Lobj has value: 10
    Hobj has value: 0
    lyle has recently read: 0
    hal has recently read: 0
```

```
hal reads lobj
The current state is:
    Lobj has value: 10
    Hobj has value: 0
    lyle has recently read: 0
    hal has recently read: 10
```

```
lyle writes value 20 to hobj
The current state is:
    Lobj has value: 10
    Hobj has value: 20
    lyle has recently read: 0
    hal has recently read: 10
```

```
hal writes value 200 to lobj
The current state is:
    Lobj has value: 10
    Hobj has value: 20
    lyle has recently read: 0
    hal has recently read: 10
```

```
Bad Instruction
The current state is:
    Lobj has value: 10
    Hobj has value: 20
    lyle has recently read: 0
    hal has recently read: 10
```

```
hal reads hobj
The current state is:
    Lobj has value: 10
    Hobj has value: 20
    lyle has recently read: 0
    hal has recently read: 20
```

```
Bad Instruction
The current state is:
    Lobj has value: 10
    Hobj has value: 20
    lyle has recently read: 0
    hal has recently read: 20
```

```
Bad Instruction
The current state is:
    Lobj has value: 10
    Hobj has value: 20
    lyle has recently read: 0
    hal has recently read: 20
```

Entrada:

```
writ lyle lobj 10
write ismael hobj 30
read hal lobj
write lyle hobj 54
bye
```

Salida:

```
Bad Instruction
The current state is:
    Lobj has value: 0
    Hobj has value: 0
    lyle has recently read: 0
    hal has recently read: 0

Bad Instruction
The current state is:
    Lobj has value: 0
    Hobj has value: 0
    lyle has recently read: 0
    hal has recently read: 0

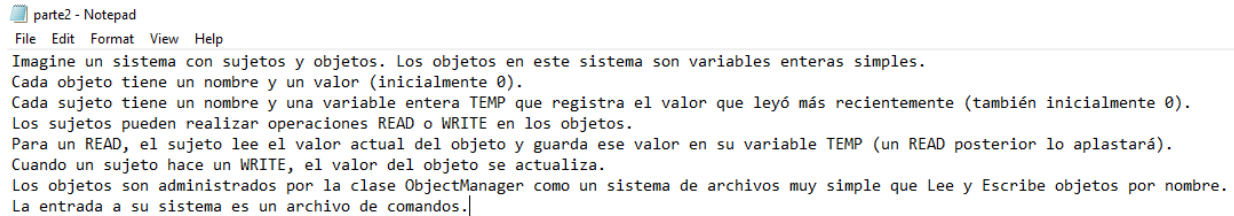
hal reads lobj
The current state is:
    Lobj has value: 0
    Hobj has value: 0
    lyle has recently read: 0
    hal has recently read: 0

lyle writes value 54 to hobj
The current state is:
    Lobj has value: 0
    Hobj has value: 54
    lyle has recently read: 0
    hal has recently read: 0

Bad Instruction
The current state is:
    Lobj has value: 0
    Hobj has value: 54
    lyle has recently read: 0
    hal has recently read: 0
```

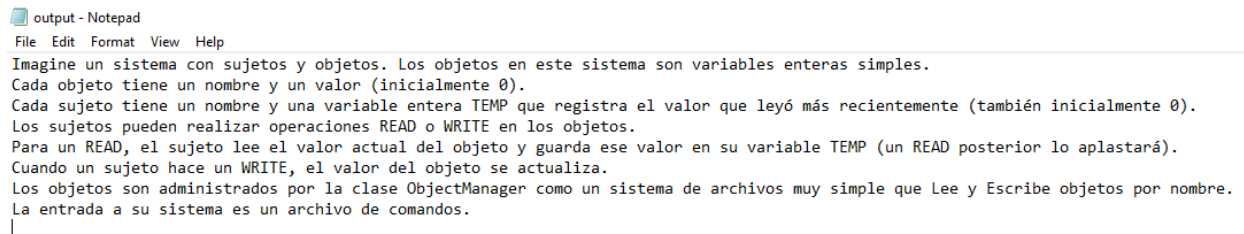
Parte 2

Entrada:



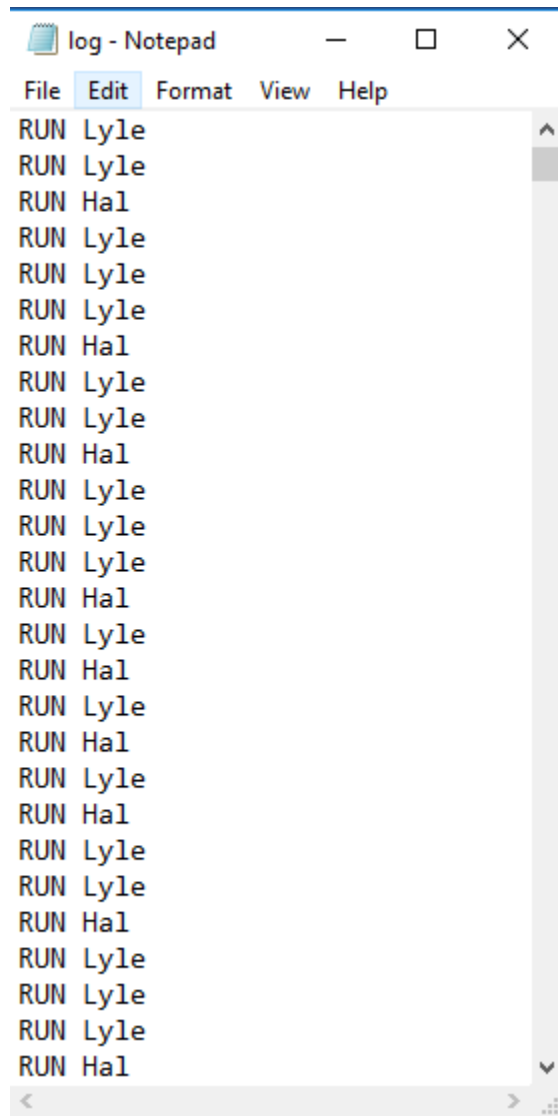
parte2 - Notepad
File Edit Format View Help
Imagine un sistema con sujetos y objetos. Los objetos en este sistema son variables enteras simples.
Cada objeto tiene un nombre y un valor (inicialmente 0).
Cada sujeto tiene un nombre y una variable entera TEMP que registra el valor que leyó más recientemente (también inicialmente 0).
Los sujetos pueden realizar operaciones READ o WRITE en los objetos.
Para un READ, el sujeto lee el valor actual del objeto y guarda ese valor en su variable TEMP (un READ posterior lo aplastará).
Cuando un sujeto hace un WRITE, el valor del objeto se actualiza.
Los objetos son administrados por la clase ObjectManager como un sistema de archivos muy simple que Lee y Escribe objetos por nombre.
La entrada a su sistema es un archivo de comandos.

Salida:



output - Notepad
File Edit Format View Help
Imagine un sistema con sujetos y objetos. Los objetos en este sistema son variables enteras simples.
Cada objeto tiene un nombre y un valor (inicialmente 0).
Cada sujeto tiene un nombre y una variable entera TEMP que registra el valor que leyó más recientemente (también inicialmente 0).
Los sujetos pueden realizar operaciones READ o WRITE en los objetos.
Para un READ, el sujeto lee el valor actual del objeto y guarda ese valor en su variable TEMP (un READ posterior lo aplastará).
Cuando un sujeto hace un WRITE, el valor del objeto se actualiza.
Los objetos son administrados por la clase ObjectManager como un sistema de archivos muy simple que Lee y Escribe objetos por nombre.
La entrada a su sistema es un archivo de comandos.
|

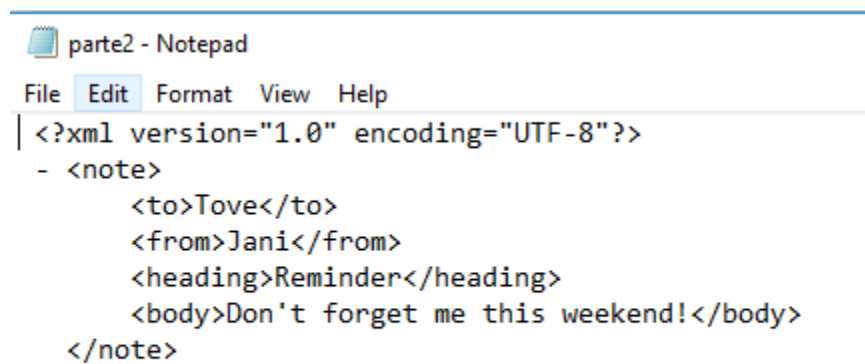
Se muestra una parte del log ya que es muy extenso.



The image shows a Notepad window with the title bar 'log - Notepad'. The menu bar includes 'File', 'Edit', 'Format', 'View', and 'Help'. The text area contains a list of log entries, each starting with 'RUN' followed by a name. The names alternate between 'Lyle' and 'Hal'. There are 25 lines of text in total. A vertical scrollbar is on the right side of the text area, and a horizontal scrollbar is at the bottom.

```
log - Notepad
File Edit Format View Help
RUN Lyle
RUN Lyle
RUN Hal
RUN Lyle
RUN Lyle
RUN Lyle
RUN Hal
RUN Lyle
RUN Lyle
RUN Hal
RUN Lyle
RUN Lyle
RUN Lyle
RUN Hal
RUN Lyle
RUN Hal
RUN Lyle
RUN Hal
RUN Lyle
RUN Hal
RUN Lyle
RUN Lyle
RUN Hal
RUN Lyle
RUN Lyle
RUN Hal
```

Entrada:

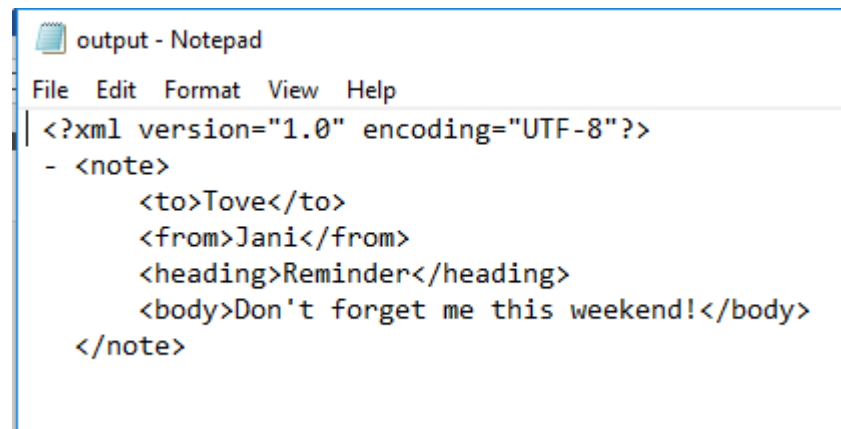


parte2 - Notepad

File Edit Format View Help

```
<?xml version="1.0" encoding="UTF-8"?>
- <note>
    <to>Tove</to>
    <from>Jani</from>
    <heading>Reminder</heading>
    <body>Don't forget me this weekend!</body>
</note>
```

Salida:

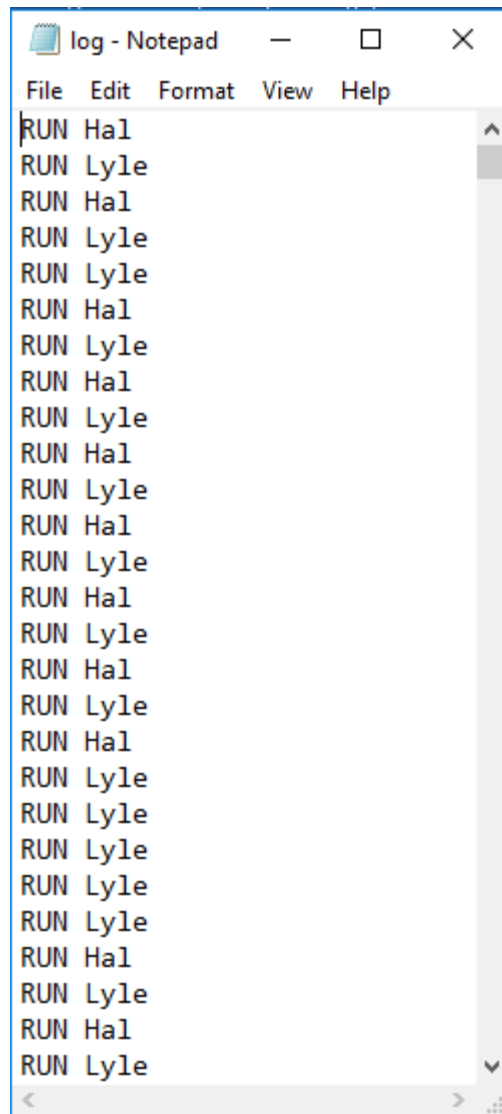


output - Notepad

File Edit Format View Help

```
<?xml version="1.0" encoding="UTF-8"?>
- <note>
    <to>Tove</to>
    <from>Jani</from>
    <heading>Reminder</heading>
    <body>Don't forget me this weekend!</body>
</note>
```

Se muestra una parte del log ya que es muy extenso.



A screenshot of a Notepad window titled "log - Notepad". The window has a menu bar with "File", "Edit", "Format", "View", and "Help". The text area contains a list of 30 lines, each starting with "RUN" followed by either "Hal" or "Lyle". The names alternate in a roughly regular pattern. A vertical scrollbar is on the right, and a horizontal scrollbar is at the bottom.

```
log - Notepad
File Edit Format View Help
RUN Hal
RUN Lyle
RUN Hal
RUN Lyle
RUN Lyle
RUN Hal
RUN Lyle
RUN Hal
RUN Lyle
RUN Hal
RUN Lyle
RUN Hal
RUN Lyle
RUN Hal
RUN Lyle
RUN Hal
RUN Lyle
RUN Hal
RUN Lyle
RUN Hal
RUN Lyle
RUN Lyle
RUN Lyle
RUN Lyle
RUN Lyle
RUN Lyle
RUN Lyle
RUN Hal
RUN Lyle
RUN Hal
RUN Lyle
```