PROGRAMACIÓN DE SERVICIOS Y PROCESOS

TÉCNICO EN DESARROLLO DE APLICACIONES MULTIPLATAFORMA

La programación segura
Amenazas y ataques
Vulnerabilidades
Expresiones regulares.

Introducción a la seguridad

Podemos definir seguridad como el hecho de estar **libre de todo daño**. En informática, la seguridad es **imposible** de conseguir, es decir, ningún programa software va a estar al 100% seguro ante amenazas, ya que día a día surgen **nuevos tipos de riesgos** que desconocemos.

No obstante, en los sistemas informáticos, sean cual sean su composición (programas, aplicaciones, webs, sistemas operativos...), podemos decir que son seguros, o que se pueden acercar a disponer de la máxima seguridad, siempre y cuando cumplan las siguientes **características**:

1. Confidencialidad: Esta característica va a requerir que únicamente las personas **autorizadas** accedan al sistema.
2. Integridad: Requerirá que únicamente las personas autorizadas puedan **modificar** la información existente en el sistema, entendiendo por modificación de la información la **escritura, lectura, modificación, creación y envío** de mensajes.
3. No repudio: Esta cuestión hará que un usuario **no pueda negar que ha enviado** un mensaje. Con el no repudio se va a **proteger al receptor del mensaje** si el emisor niega que ha enviado dicho mensaje. Una forma de no repudio son las **firmas** y los **certificados digitales**.
4. Disponibilidad: Requerirá que todos los recursos del sistema estén **siempre** disponibles para el uso de los usuarios **autorizados**.

Para poder tratar cualquier problema con la seguridad, todas las empresas deben tener obligatoriamente una **política de seguridad** firmemente establecida. La función de la política no es otra que la de dejar claro cuáles son las **responsabilidades y reglas** a seguir **para evitar amenazas**.

Existen una serie de **organismos oficiales** a nivel mundial, que son los encargados de asegurar los servicios de prevención de riesgos y de la asistencia a los tratamientos de cualquier posible incidencia. Uno de ellos es el Computer Emergency Response Team Coordination Center del Software Engineering Institute de la Universidad Carnegie Mellon, el cual es un **centro de alerta y reacción** frente a los ataques informáticos.

En España tenemos el Instituto Nacional de Ciberguridad, o por sus siglas, INCIBE.

Elementos susceptibles de amenazas

Desde el punto de vista de la informática, existen tres tipos de elementos que son los que pueden sufrir amenazas, los cuales son el **hardware**, el **software** y los **datos**. Para cada uno de estos elementos se deben crear medidas de seguridad para no ponerlos en riesgo.

Amenazas de seguridad

Una amenaza de seguridad es una **condición del entorno** de un sistema de información, que una vez dada una oportunidad, **puede producir una violación** de seguridad en el sistema.

Las condiciones del entorno pueden referirse tanto a **personas**, como a **máquinas** que realicen un ataque.

Existen los siguientes **tipos de amenazas**:

Amenazas por el **origen**:

> Estas se dan por el hecho de **conectar un sistema a cualquier entorno**, ya que esto propicia que haya atacantes que puedan entrar en nuestro sistema o alterar el funcionamiento normal de la red. Hay que remarcar que no por no estar conectados a Internet signifique que estamos a salvo. La amenaza puede provenir de **cualquier punto de la red aún sin tener salida** a internet.

Amenazas por el **efecto**:

> Este tipo de amenazas pueden ser:
>
> - **robos** de información,
> - **anulación** de los sistemas informáticos,
> - **suplantación** de identidad,
> - robo de **dinero**,
> - **venta** de datos personales,
> - **destrucción** de información,
> - **estafas**,
> - etc.

Amenazas por el **medio utilizado**:

> Estas pueden ser:
>
> - **virus**,
> - ingeniería **social**,
> - ataques de **denegación** de servicio,
> - **phishing**,
> - etc.

Todas estas amenazas las podemos clasificar principalmente en **cuatro grandes grupos**:

1. Interrupción: En este grupo tenemos todas las amenazas que consiguen **destruir recursos** del sistema informático, pudiendo dejarlo totalmente inservible, provocando así grandes **pérdidas** de información y, posiblemente, de dinero.

2. Intercepción: En este grupo tenemos todas las amenazas que consiguen **acceder a un recurso** de otra persona, pudiendo lucrarse del mismo pidiendo un **rescate** por los datos robados.

3. Modificación: En este grupo tenemos todas las amenazas que intentan **acceder a un recurso** de otra persona, pudiendo llegar a **modificarlo**. (Según un test: aquí se clasifica un programa para escuchar el tráfico de la red).

4. Fabricación: En este grupo tenemos todas las amenazas que impliquen un ataque **contra la autenticidad** de los datos, insertados **datos faltos** en los originales.

Tipos de virus

Un virus informático es un sistema de software dañino, escrito intencionadamente para entrar en una computadora sin permiso o conocimiento del usuario. Tiene la capacidad de replicarse a si mismo, continuando así su propagación.

Algunos virus no hacen mucho mas que replicarse, mientras que otros pueden causar graves danos o afectar negativamente el rendimiento de un sistema.

Un virus nunca debe ser considerado como inofensivo y dejarlo en un sistema sin tomar medidas.

1. Virus de **acción directa**:

   El objetivo principal de estos tipos de virus informáticos es replicarse y actuar cuando son ejecutados. Cuando se cumple una condición especifica, el virus se pondrá en acción para infectar a los ficheros en el directorio o carpeta que se especifica en el archivo autoexec.bat.

2. Virus de **sobreescritura**: Estos tipos de virus informáticos se caracterizan por el hecho de que borran la información contenida en los ficheros que infectan, haciéndolos parcial o totalmente inútiles. Una vez infectados, el virus reemplaza el contenido del fichero sin cambiar su tamaño.

3. Virus de **sector de arranque**: Este tipo de virus afecta al sector de arranque del disco duro. Se trata de una parte crucial del disco en la que se encuentra la información que hace posible arrancar el ordenador desde disco.

4. Virus **polimórfico**: Estos tipos de virus informáticos se encriptan o codifican de una manera diferente, utilizando **diferentes algoritmos y claves de cifrado** cada vez que infectan un sistema. Esto hace **imposible** que el software **antivirus** los encuentre utilizando búsquedas de cadena o firma porque **son diferentes cada vez**.

Técnicas seguras en Java

Un concepto propio de la seguridad en los sistemas es, por ejemplo, el **tratamiento de excepciones** mediante los **bloques try-catch**. Si repasamos para qué servían estos bloques recordaremos que su finalidad era la de controlar las posibles excepciones, errores o **ejecuciones no deseadas** que pudiesen ocurrir durante la ejecución de un programa.

Ya estudiamos que una excepción era un evento que podía ocurrir en tiempo de ejecución de una aplicación y que interrumpiría el flujo normal de las instrucciones de la misma, provocando que la aplicación se detuviera. Si analizamos esto, podemos ver que para luchar contra las ejecuciones no deseadas podemos utilizar un bloque try-catch, lanzando una **excepción que pare el proceso no deseado**. Como sabemos, en el lenguaje de programación Java podemos encontrar varios tipos de excepciones:

- Error. Estas son excepciones que van a indicar **problemas muy graves**, los cuales por norma general **no suelen ser recuperables** y que no deben casi nunca ser capturadas.
- Exception. Estas son excepciones que no son definitivas, pero que vamos a poder **detectar en tiempo de codificación**.
- RuntimeException. Estas son excepciones que se dan durante el **tiempo de ejecución** del programa o aplicación.

Otro ejemplo de técnica de seguridad es la de las **validaciones** de entradas de datos mediante **expresiones regulares**.

Ataques a un sistema informático

Ya sabemos que todos los sistemas informáticos están expuestos a ser atacados de una forma u otra, y que no hay una forma de estar libre al 100% de esta posibilidad, aunque si cumplimos con ciertos requisitos de seguridad, como los vistos anteriormente, podremos **interponer obstáculos** y que a los atacantes **no les resulte fácil atacarnos**.

Los ataques a los sistemas informáticos los podemos definir como ciberataques. Es muy importante tener en cuenta que hoy en día, algunos ciberataques, dependiendo de dónde se realice, a quién o cuándo, pueden llegar a formar parte de una **guerra informática** o de un ataque de **ciberterrorismo**.

Los ataques que podemos recibir se pueden clasificar en **dos grandes grupos** principalmente:

1. Ataques **pasivos**:

   Esta categoría engloba a los tipos de ataques en los que el atacante **no necesita alterar la comunicación**, este únicamente se dedicará **escuchar o monitorizar** el tráfico de información en la red para intentar **obtener información** que está siendo transmitida. A este tipo de ataques se les suele dar uso para intentar ver el **tráfico de red**, pudiendo conseguir de esta forma **credenciales** de acceso, como usuarios y contraseñas, **páginas web** que están siendo visitadas, etc.

2. Ataques **activos**:

   Esta categoría engloba a los tipos de ataques en los que el atacante realiza algún tipo de **modificación de los datos** que están siendo transmitidos en la red, o incluso pudiendo llegar a crear un **flujo de datos falso** que se transmitirá por la red como si de un verdadero se tratase.

   Los objetivos de estos ataques son:

   a. Intentar una suplantación de identidad, haciéndose pasar el atacante por el usuario suplantado.

   b. Una reactuación, pudiendo **reenviar una serie de mensajes** determinados para producir un efecto no deseado.

   c. Ataques de denegación de servicio, pudiendo de esta forma **apagar sitios** web.

Vulnerabilidades en el software

Las vulnerabilidades del software surgieron en el mismo momento en el que surgió Internet, y aún a día de hoy sigue siendo uno de los mayores problemas a los que nos debemos enfrentar.

Éstas son un **fallo o hueco en la seguridad** de un software o sistema informático, que ha sido **detectado** por algún otro sistema, o persona que monitoriza ese sistema malicioso, el cual, puede ser utilizado para **entrar** en el sistema destino **de forma no autorizada**, pudiendo realizar operaciones indeseadas.

El mayor inconveniente de las vulnerabilidades de software reside en la **dificultad en la forma en la que son detectadas**. De hecho, cuando se trata de un programa con una gran **envergadura**, las **auditorías de vulnerabilidades** normalmente son encargadas a **empresas externas** que se dedican a buscar **fallos** de software, las cuales, cuentan con **expertos** en la materia.

Cuando se crea una aplicación software, **antes de lanzarla al mercado** se hace un estudio meticuloso en el que se intentan descubrir todos los **fallos de seguridad** de la misma, pudiendo **identificarlos y solucionarlos** sin haber puesto la aplicación en producción. En este proceso se pueden descubrir una gran cantidad de fallos de seguridad, pero desgraciadamente, estos no serán todos los que surgirán, por lo que aún con la aplicación en el mercado, se deberá seguir con este proceso ininterrumpidamente, monitoreando de forma continua el programa, y lanzando **actualizaciones y parches** que solventen los nuevos **fallos** de seguridad **encontrados**.

Como en este módulo estamos centrados en el **lenguaje de programación Java**, vamos a ver los **dos pilares de la seguridad** en los que se basa éste:

1. Seguridad interna de la aplicación.

   Esto se refiere a que cuando se programe una aplicación han de seguirse unos **criterios de tratamiento de errores**.

   Un ejemplo serían los **bloques try-catch-finally** para el tratamiento de excepciones.

2. Políticas de acceso.

   Las políticas de acceso se refieren a las **acciones** que puede realizar la **aplicación en nuestro equipo**, es decir, el hecho de dar o no **permiso** a la aplicación para que pueda **utilizar ciertos recursos** del sistema.

Mecanismos de control de acceso

En el ámbito de la seguridad informática existen lo que se denominan políticas de seguridad:

> Las podemos definir como una serie de **documentos de muy alto nivel** que van a ser **fundamentales** para llevar a cabo el compromiso con la **ciberseguridad**.
>
> Estos documentos contienen:
>
> - la definición de la seguridad de la información,
> - las medidas a adoptar tanto por la **empresa**, **trabajadores**, así como el **departamento** de sistemas, para aplicar todos los procedimientos necesarios que garanticen que el trabajo se desarrolla en un entorno seguro.

Las políticas de seguridad por sí solas no son suficientes y deben de ser utilizadas **con otras medidas** que van a depender de las mismas, como pueden ser:

- los objetivos de seguridad,
- los procesos.

Las políticas de seguridad deben ser **fácilmente accesibles** para que **todo el personal** de la empresa **esté al tanto** de su existencia y **entiendan** su contenido.

Seguridad en la aplicación mediante tokens

Si debemos hacer que las peticiones de realización de las operaciones que los usuarios mandan al servidor de la empresa sean seguras, intentando **validar a los usuarios** de alguna forma, una de las formas más sencillas y efectivas de hacer que sea seguro es mediante una 'tokenización' de los usuarios.

La 'tokenización' consiste en asignar un **token único** a cada uno de los **usuarios**, teniendo que **verificarlo en todas las peticiones** que realice al servidor de la empresa.

En un primer momento, esto puede parecer que va a ralentizar el sistema, pero, al fin y al cabo, no va a ser más que un bloque if-else, lo cual, no va a influir demasiado en el tiempo de las operaciones.

En caso de que el **token enviado no sea correcto** o no exista, se **lanzará un error** desconocido.

Validación de entradas

Una forma de entrada de errores en nuestras aplicaciones que puede afectar a la seguridad, son las entradas de datos que introducen los usuarios, por ejemplo, en el momento del control de acceso a los sistemas, o simplemente, cuando nos encontramos desarrollando un nuevo programa. Uno de los fallos de seguridad más comunes se trata de los conocidos como buffer overflow, los cuales, se producen cuando se **desborda el tamaño de una determinada variable**.

Para evitar parte de estos errores podemos usar la **validación de datos mediante expresiones regulares**, la cual nos va a **permitir**:

1. Mantener la consistencia de los datos, pudiendo comprobar que los datos insertados cumplen ciertos **requisitos**.
2. Evitar desbordamientos de memoria buffer overflow, pudiendo **comprobar la longitud** de los campos.

Para poder usar las expresiones regulares, Java usa la biblioteca java.util.regex, la cual nos va a permitir usar la clase Patter, que permite la **definición de expresiones regulares**. Para la validación de entrada podemos utilizar los siguientes elementos y operadores:

Elementos y operadores de expresiones regulares

Elementos:

- x → El carácter x.
- [abc] → Los caracteres a, b o c.
- [a-z] → Una letra minúscula.
- [A-Z] → Una letra mayúscula.
- [A-Za-z] → Una letra minúscula o mayúscula.
- [0-9] → Un número entre 0 y 9.
- [A-Za-z0-9] → Una letra minúscula, mayúscula o un número.

Operadores:

- [a-z]{2} → Dos letras minúsculas.
- [a-z]{2,5} → Entre dos y cinco letras minúsculas.
- [a-z]{2,} → Más de 2 letras minúsculas.
- hola|adios → Solo se pueden introducir las palabras hola o adiós. | (barra vertical) funciona como la operación OR.
- XY → Solo se pueden introducir dos expresiones. Esta es la operación AND.
- e(n|l)campo → Los **delimitadores** ( ) permiten hacer expresiones más complejas. En el ejemplo se debe introducir el texto “en campo” o “el campo”.

Ejemplo de validación de datos

Validación de DNI mediante Expresiones Regulares en Java

Exploraremos cómo validar la entrada de un Documento Nacional de Identidad (DNI) mediante expresiones regulares en Java.

En primer lugar, hemos creado **dos expresiones regulares** que nos permitirán validar un DNI:

1. La primera expresión regular indica que necesitamos dígitos en una **cantidad de 8**, seguidos por una **letra mayúscula**.
2. En la segunda expresión regular, lo único que **cambia** es la **forma de indicar que necesitamos dígitos**.

Ambas expresiones regulares **funcionan de la misma manera**.

Posteriormente, hemos creado **cuatro variables** con cuatro posibles **DNIs a comprobar**.

- El único DNI **válido es el primero**, ya que tiene ocho dígitos y una letra mayúscula.
- En el segundo, la letra es minúscula, lo cual debería dar falso.
- En el tercero, falta un dígito,
- y, en el cuarto, hay dos letras.

Para comprobar la expresión regular, utilizamos la clase `Pattern`, que debemos importar del paquete `java.util.regex`. En el método `matches`, podemos verificar si una expresión regular coincide con una cadena. En este caso, le pasaremos la expresión regular 1 y luego podremos probar los otros tres DNIs.

Si ejecutamos nuestro programa, observamos que nos indica que el DNI es correcto. Si cambiamos la expresión 1 por la 2, el DNI sigue siendo correcto. Ahora, al probar el DNI 2, muestra que no es correcto, el DNI 3 tampoco es correcto y el DNI 4 tampoco es correcto.

```
    public static void main(String[] args) {
            // Expresiones regulares para el DNI
            String dniexpresionreg1 = "\\d{8} [A-Z] {1}";
            String dniexpresionreg2 = "(0-9] {8} [A-Z]{1}";

            // DNIs para comprobar
            String dni1 = "25841796T"; // true
            String dni2 = "25841796t"; // false, la letra es minúscula
            String dni3 = "2584179T"; // false, falta un digito
            String dni4 = "25841796TT"; // false, tiene dos letras

            // Comprobamos si es correcto el DNI a comprobar
```

```
        if (Pattern.matches(dniexpresionreg1, dni1))
        {
              System.out.println("El DNI es correcto.");
        }
        else
        {
              System.out.println("El DNI no es correcto.");
        }
  }
```

Reflexión

El hecho de que se revisen aspectos de seguridad en un momento determinado, no exime de volver a revisar antes de entregar la aplicación al cliente, o lanzarla a producción. Ya que, por ejemplo, durante el periodo en el que no se ha revisado, un usuario puede ejecutar un programa de terceros e instalar un software malicioso en el PC del trabajo, sin ser siquiera consciente de ello.

Por esta razón y por muchas otras, cuando creamos una aplicación debemos tener revisado en todo momento, el aspecto de la seguridad, sobre todo, en la transmisión de los datos, en el almacenamiento de los datos, en las entradas de información a nuestra aplicación, en las entradas al sistema de los usuarios, etc.

Si el tiempo no permite hacer las revisiones oportunas sobre seguridad, habría que negociar con el cliente la fecha de entrega de los trabajos, ya que será más perjudicial entregar un proyecto a tiempo inseguro, que tarde, pero totalmente seguro.