

PROGRAMACIÓN DE SERVICIOS Y PROCESOS
TÉCNICO EN DESARROLLO DE APLICACIONES
MULTIPLATAFORMA

**La programación
segura**

ÍNDICE

/ 1. Introducción y contextualización práctica	3
/ 2. Introducción a la seguridad	4
/ 3. Amenazas de seguridad	4
/ 4. Caso práctico 1: “Técnicas seguras en Java”	5
/ 5. Ataques a un sistema informático	6
/ 6. Vulnerabilidades en el software	7
/ 7. Mecanismos de control de acceso	7
/ 8. Caso práctico 2: “Seguridad en la aplicación”	8
/ 9. Validación de entradas	8
/ 10. Resumen y resolución del caso práctico de la unidad	9
/ 11. Bibliografía	10

OBJETIVOS



Aprender el concepto de seguridad.

Identificar las amenazas y ataques más comunes que pueden existir en un sistema informático.

Conocer las vulnerabilidades del software.

Validar entradas con expresiones regulares.



/ 1. Introducción y contextualización práctica

En esta unidad vamos a tratar el concepto de seguridad en un sistema informático.

Vamos a ver las amenazas más comunes que puede sufrir un sistema, para entender así la necesidad de la seguridad en los sistemas de comunicación.

También vamos a conocer los tipos de ataques más comunes que puede sufrir un sistema informático.

Por último, aprenderemos a validar las entradas de datos mediante expresiones regulares, medida que puede ayudar a prevenir ciertos ataques a nuestros sistemas.

Escucha el siguiente audio en el que planteamos el caso práctico que iremos resolviendo a lo largo de esta unidad.



Fig. 1. Seguridad.



Audio Intro. "La importancia de la seguridad"

<https://bit.ly/32XU029>





/ 2. 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 Ciberseguridad, o por sus siglas, INCIBE.




Fig. 2. Logo INCIBE.



Audio 1. "Elementos susceptibles de amenazas"

<https://bit.ly/36Tt0BQ>



/ 3. 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**:

- a. 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.
- b. 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.
- c. 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.
- 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.



Fig. 3. Amenazas.



Vídeo 1. "Tipos de virus"

<https://bit.ly/2UEuiv5>



/ 4. Caso práctico 1: "Técnicas seguras en Java"

Planteamiento: Pilar y José están repasando los apuntes de la asignatura de Programación para ver qué pueden hacer para controlar la seguridad de sus aplicaciones.

"No encuentro nada para realizar programación de aplicaciones seguras en ninguna de las unidades de Programación, ¿tú has encontrado algo?", le dice Pilar a José. Este le responde que todavía no ha terminado, pero que tampoco ha encontrado nada.

Nudo: ¿Crees que Pilar y José están realizando una búsqueda exhaustiva?

Desenlace: Cuando empezamos a programar aplicaciones, como es el caso de nuestros amigos al estudiar el módulo de Programación, es normal que no se entre en detalles de seguridad de los sistemas, ya que lo que se busca en esos momentos es aprender los conceptos básicos de la programación.

No obstante, aun así, sí que se estudian algunos conceptos propios de la seguridad en los sistemas. Un ejemplo de ello es 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.

```
1  try
2  {
3      // Código a ejecutar
4      // de forma segura
5  }
6  catch(Exception error)
7  {
8      // Tratamiento seguro de una excepción
9  }
```

Fig. 4. Bloque try-catch.

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

/ 5. 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, éste 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

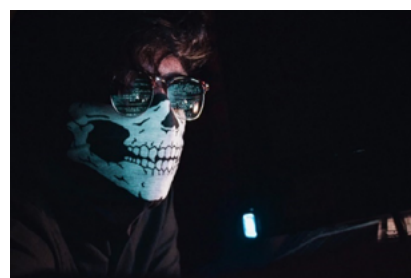


Fig. 5. Hacker.



/ 6. 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.



Fig. 6. Aplicación no segura.

/ 7. Mecanismos de control de acceso

En el ámbito de la seguridad informática existen lo que se denominan: **políticas de seguridad**, las cuales, 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 contienen la definición de la seguridad de la información, y las medidas a adoptar tanto por la empresa, trabajadores, así como el departamento de sistemas, para aplicar todos los procedimientos necesarios que garantizan 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 y 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.

/ 8. Caso práctico 2: “Seguridad en la aplicación”

Planteamiento: Nuestros amigos Pilar y José acaban de recibir un encargo de una empresa bastante importante. Este consiste en que deben diseñar un sistema de seguridad para garantizar que únicamente sean los usuarios de la empresa los que puedan realizar las operaciones en el sistema. El desarrollo consiste en una aplicación móvil que se conecta mediante una pasarela al servidor de la empresa, pudiendo los usuarios acceder así a todas las funcionalidades del mismo.

Nudo: ¿Cómo crees que podrían Pilar y José hacer que el sistema sea más seguro para los usuarios de la empresa?

Desenlace: Antes de nada, cabe destacar que en seguridad ocurre lo mismo que cuando se plantean cuestiones de programación en sí, y es que para un mismo problema pueden diseñarse varias soluciones, y todas ser aptas.

Para comenzar con el trabajo, en primer lugar, habría que analizar el sistema y buscar opciones para sea un poco más seguro.

Según les ha trasladado la empresa a nuestros amigos, deberían 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.

Este problema se da en multitud aplicaciones, y 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.

```
1 // Comprobamos el token del usuario
2 si tokenenviado = tokenusuario
3     // realizamos la operación solicitada
4 sino
5     // lanzamos un error desconocido
```

Fig. 7. Pseudocódigo de comprobación

/ 9. 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 **Pattern**, que permite la definición de expresiones regulares. Para la validación de entrada podemos utilizar los siguientes elementos y operadores:

Elemento	Comentario	Operador	Comentario
x	El carácter x.	<code>[a-z]{2}</code>	Dos letras minúsculas.
[abc]	Los caracteres a, b o c.	<code>[a-z]{2,5}</code>	Entre dos y cinco letras minúsculas.
[a-z]	Una letra minúscula.	<code>[a-z]{2,}</code>	Más de 2 letras minúsculas.
[A-Z]	Una letra mayúscula.	<code>hola adios</code>	Solo se pueden introducir las palabras hola o adiós. funciona como la operación OR.
[a-Za-z]	Una letra minúscula o mayúscula.	<code>XY</code>	Solo se pueden introducir dos expresiones. Esta es la operación AND.
[0-9]	Un número entre 0 y 9.	<code>e(n l)campo</code>	Los delimitadores () permiten hacer expresiones más complejas. En el ejemplo se debe introducir el texto "en campo" o "el campo".
[A-Za-z0-9]	Una letra minúscula, mayúscula o un número.		

Tabla 1. Elementos y operadores de expresiones regulares.



Vídeo 2. "Ejemplo de validación de datos"
<https://bit.ly/3pKrquH>



/ 10. Resumen y resolución del caso práctico de la unidad

A lo largo de esta unidad hemos visto el concepto de seguridad en un sistema informático.

Hemos visto las amenazas más comunes que puede sufrir un sistema informático, para así poder justificar la necesidad intrínseca de la seguridad en los sistemas de comunicación.

También hemos visto los tipos de ataques más comunes que puede sufrir un sistema informático, aunque puede haber otros muchos tipos.

Por último, hemos visto cómo podemos validar las entradas de datos mediante expresiones regulares, medida que puede ayudarnos a prevenir ciertos ataques a nuestros sistemas.



Resolución del caso práctico de la unidad

Como hemos visto a lo largo de la unidad, una parte vital de las aplicaciones, sean cuales sean las plataformas a las que van destinadas, es la seguridad en las mismas.

El hecho de que se revisaran aspectos de seguridad en un momento determinado, no exige de volver a revisarla 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.

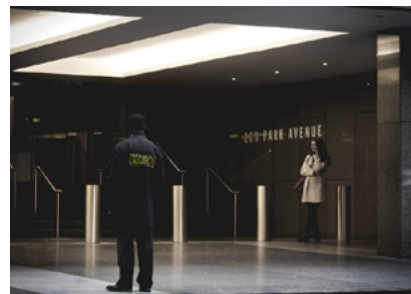


Fig. 8. Seguridad en el sistema.

/ 11. Bibliografía

Colaboradores de Wikipedia. (2020a, julio 7). *Ciberataque*. Wikipedia, la enciclopedia libre. <https://es.wikipedia.org/wiki/Ciberataque>

Colaboradores de Wikipedia. (2020, 17 agosto). *Seguridad informática*. Wikipedia, la enciclopedia libre. https://es.wikipedia.org/wiki/Seguridad_inform%C3%A1tica

Expresiones Regulares en Java - ChuWiki. (s. f.). *chuidiang*. Recuperado 31 de agosto de 2020, de http://chuwiki.chuidiang.org/index.php?title=Expresiones_Regulares_en_Java

Java Exceptions (Try...Catch). (s. f.). *w3schools*. Recuperado 31 de agosto de 2020, de https://www.w3schools.com/java/java_try_catch.asp

Colaboradores de Wikipedia. (2019, 12 julio). *Política de seguridad informática*. Wikipedia, la enciclopedia libre. https://es.wikipedia.org/wiki/Pol%C3%ADtica_de_seguridad_inform%C3%A1tica#:~:text=La%20pol%C3%ADtica%20de%20seguridad%20es,de%20vista%20de%20cierta%20entidad.&text=Puede%20ser%20tambi%C3%A9n%20un%20documento,en%20un%20manual%20de%20seguridad.

Control de Acceso Basado en Roles. (s. f.). *SG Buzz*. Recuperado 17 de octubre de 2020, de <https://sg.com.mx/content/view/707>

I. (2020, 15 julio). *Seguridad basada en roles*. Microsoft Docs. <https://docs.microsoft.com/es-es/dotnet/standard/security/role-based-security>