PRUEBAS DE RENDIMIENTO

Introducción

- El rendimiento, o más bien la eficiencia del desempeño, se define como el desempeño relativo a la cantidad de recursos utilizados en las condiciones establecidas. [ISO25010].
- El rendimiento se relaciona con la duración de la ejecución de las tareas por parte de un sistema de TI y la carga que dicho sistema puede manejar. Dado que existen muchos usos diferentes de los sistemas de TI, las expectativas de rendimiento de los usuarios también serán diferentes. Ejemplos de rendimiento que suelen ser importantes son el tiempo de respuesta de un sistema en tiempo real, la duración de un proceso por lotes y la cantidad de usuarios que se pueden manejar simultáneamente.
- Al probar el rendimiento, intervienen varias capas de un sistema de TI. Cada capa puede tener un efecto específico en el rendimiento general experimentado por el usuario.
- La norma ISO25010 define tres subcaracterísticas de rendimiento:
  - Comportamiento: Grado en el que los tiempos de respuesta y procesamiento y las tasas de rendimiento de un producto o sistema, al realizar sus funciones, cumplen con los requisitos.
  - Utilización de recursos: Grado en el que las cantidades y tipos de recursos utilizados por un producto o sistema, al realizar sus funciones, cumplen con los requisitos.
  - Capacidad: Grado en que los límites máximos de un producto o parámetro del sistema cumplen con los requisitos.

Pruebas de rendimiento

- Las pruebas de rendimiento establecen si un sistema de TI cumple con los requisitos de rendimiento pertinentes. Tratar correctamente los requisitos de rendimiento se vuelve aún más importante cuando la tecnología se integra en todos los aspectos de nuestra vida profesional y personal (es decir, computación basada en la nube, soluciones móviles e "Internet de las cosas").
- En la vida real, los requisitos de desempeño a menudo no están especificados. Para los actores empresariales es evidente que el desempeño debe ser bueno, pero muchas veces no son capaces de especificar lo que para ellos significa "bueno". Durante el refinamiento de las historias de usuario (u otros documentos de requisitos), se deben prestar especial atención a los requisitos no funcionales, como la eficiencia del desempeño. E incluso cuando falten por completo los requisitos de rendimiento, será útil que se organicen y realicen pruebas de rendimiento.
- Los aspectos técnicos de las pruebas de rendimiento deben responder preguntas sobre el comportamiento del tiempo, la utilización de recursos y la capacidad. La implementación técnica específica depende de las herramientas e infraestructura disponibles.

Tipos

Pruebas de carga, estrés y picos

- Pruebas de carga (load test): pruebas para determinar y validar la respuesta de la aplicación cuando es sometida a una carga de usuarios y/o transacciones que se espera en el ambiente de producción. Ejemplo: verificar la correcta respuesta de la aplicación ante el alta de 100 usuarios en forma simultanea. Se compara con el volumen esperado.
- Pruebas de estrés (stress test): pruebas para encontrar el volumen de datos o de tiempo en que la aplicación comienza a fallar o es incapaz de responder a las peticiones. Son pruebas de carga o rendimiento, pero superando los límites esperados en el ambiente de producción y/o determinados en las pruebas. Ejemplo: encontrar la cantidad de usuarios simultáneos, en que la aplicación deja de responder (cuelgue o time out) en forma correcta a todas las peticiones.
- Pruebas de picos (spike testing): es un sub tipo de prueba de estrés que mide el rendimiento del software bajo un «pico» significativo y repentino o una carga de trabajo creciente como la de los usuarios simulados. Indica si el software puede manejar ese aumento abrupto de la carga de trabajo de forma repetida y rápida.

Pruebas de resistencia, volumen y escalabilidad

- Pruebas de resistencia (soak testing, endurance testing): evalúa el rendimiento del software durante un periodo prolongado bajo una carga de trabajo regular y fija, determina cuánto tiempo puede soportar el software una carga de trabajo constante para proporcionar sostenibilidad a largo plazo. Durante estas pruebas, los equipos de pruebas supervisan los KPI como las fugas de memoria, de proceso, etc. Las pruebas de resistencia también analizan los tiempos de respuesta y el rendimiento tras un uso prolongado para mostrar si estas métricas son consistentes.
- Pruebas de volumen (volume testing): comprueban la eficacia del software cuando se somete a grandes volúmenes de datos. Comprueba la pérdida de datos, el tiempo de respuesta del sistema, la fiabilidad del almacenamiento de datos, etc.
- Pruebas de escalabilidad (scalability testing): miden la eficacia del software a la hora de manejar una cantidad creciente de carga de trabajo, añadiendo volumen de datos o usuarios de forma gradual mientras se supervisa el rendimiento del software. La prueba informará sobre el comportamiento cuando aumenten o disminuyan los atributos de rendimiento del software.

Pasos para la supervisión

1. Definir los objetivos concretos de supervisión.
2. Seleccionar la herramienta apropiada.
3. Identificar los componentes que se desea supervisar.
4. Seleccionar métricas para dichos componentes.
5. Tomar medidas.
6. Analizar las medidas obtenidas.
7. Solucionar los problemas detectados o implementar las posibles mejoras.

Componentes a supervisar

- Las áreas siguientes afectan al rendimiento:
  - Recursos del sistema (hardware)
  - Sistema operativo
  - Arquitectura de red
  - Configuración de servidores e infraestructuras
  - Aplicaciones de bases de datos
  - Aplicaciones cliente
- Dentro de los Recursos del sistema, los componentes principales en los que se debe concentrar inicialmente son:
  - Actividad del disco
  - Uso de la memoria
  - Uso del procesador
  - Uso de la red

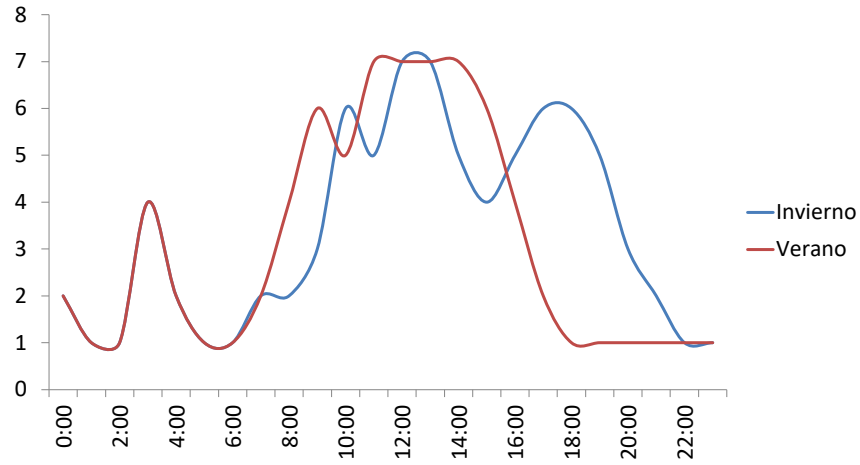
Cuellos de botella

- Se denomina cuello de botella al fenómeno en donde el rendimiento o capacidad de un sistema completo es severamente limitado por componentes individuales.
- Entre las causas de estos cuellos de botella se incluyen:
  - Recursos que funcionan incorrectamente.
  - Recursos mal configurados.
  - Recursos del mismo tipo que no distribuyen de forma equilibrada las cargas de trabajo; por ejemplo, cuando un recurso monopoliza un disco.
  - Recursos insuficientes que requieren componentes adicionales o actualizados.
- Para detectar los cuellos de botella es necesario realizar un proceso de toma de medidas y su análisis ulterior.

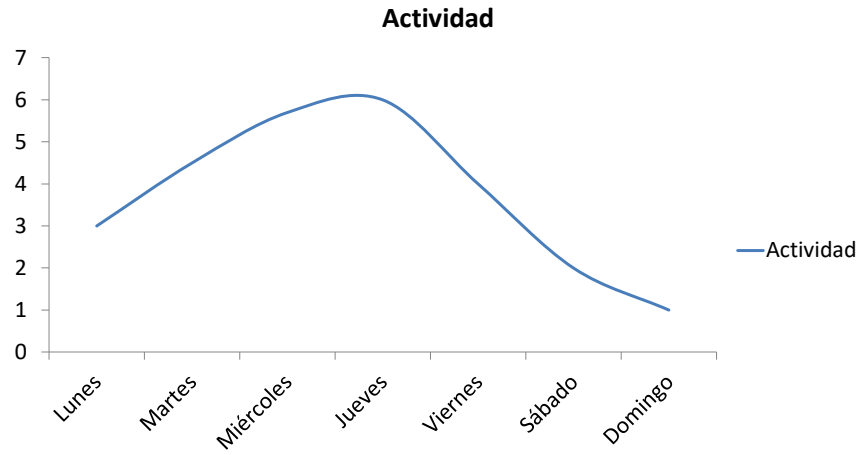
Toma de medidas

- Se denomina toma de medidas al proceso de captura de valores cuantitativos sobre los elementos representativos en diferentes momentos. Teniendo en cuenta las diferentes idiosincrasias de los sistemas y de los recursos humanos, es necesario tomar medidas:
  - A diferentes horas del día.
  - Los diferentes días de la semana.
  - A lo largo de todo el mes
  - En diferentes meses
- La toma de medidas es un proceso continuo acumulativo que deben representar las cargas de trabajo habituales que se ejecutan en las bases de datos o sistemas.
- La calidad de las medidas tomadas determina la calidad del análisis.

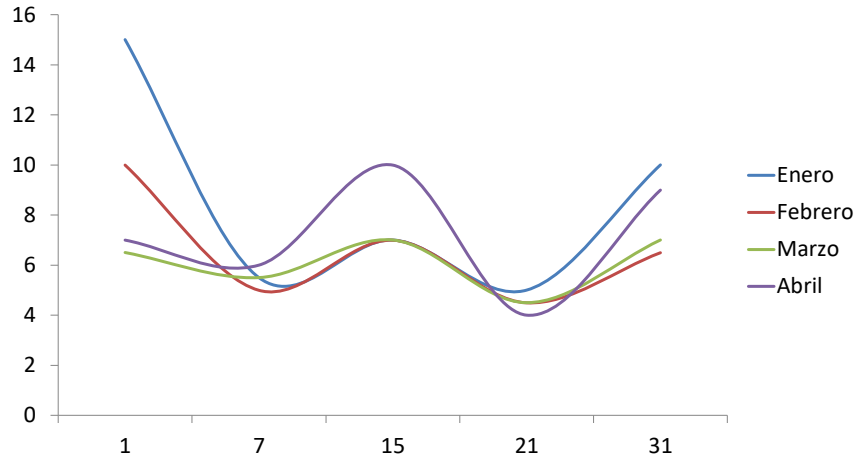
Gráficas diaria



Gráfica semanal



Graficas mensuales



Requisitos

- Herramientas de captura y reproducción de tráfico de red (proxy o record & replay):
  - diseño, construcción y mantenimiento del entorno de prueba (tipo producción) y uso de las herramientas disponibles para crear scripts de prueba de rendimiento que coincidan con los perfiles de usuario designados.
- Configuración del generador de carga múltiple:
  - diseño e implementación del escenario de rendimiento (combinación de scripts de prueba) para simular la carga en el sistema como se define en el modelo de carga a niveles similares a los de producción.
- Configuración del laboratorio de pruebas:
  - diseño y mantenimiento de un laboratorio con suficiente capacidad o flexibilidad para ejecutar múltiples aplicaciones en un escenario de prueba (se necesitan habilidades adicionales en componentes de red, administración de máquinas virtuales, experiencia en redes, etc.)
- Monitoreo del entorno :
  - ejecutar y mantener herramientas similares a las utilizadas en el entorno de producción con herramientas de prueba de rendimiento capaces de vincularse con esos resultados de monitoreo.

Perfiles

- Un **perfil operativo** describe en términos cuantitativos cómo un tipo particular de usuario utiliza el sistema. El perfil describe qué transacciones realiza un usuario y con qué frecuencia. Un perfil operativo describe el uso realista respondiendo a la pregunta: "Cuando el sistema está en esta condición, ¿qué posibilidades hay de que el usuario lleve a cabo esta acción?" Este perfil proporciona un promedio estadístico de cómo 'el usuario' maneja el sistema y, si se distinguen tipos de usuarios con comportamientos significativamente diferente, es aconsejable crear un perfil operativo separado para cada tipo. La prueba tiene como objetivo examinar si: "El sistema sigue funcionando correctamente cuando las transacciones se han realizado con frecuencia y durante un largo tiempo".
- Los **perfiles de carga** muestran el grado en que los recursos del sistema (CPU, memoria, capacidad de red) están cargados en realidad. La carga suele mostrarse en términos de la cantidad de usuarios o transacciones en un período determinado. Por lo general, la carga de un sistema no es uniforme de forma continua, sino que varía a lo largo del tiempo: hay picos y valles en un tramo de 24 horas. A menudo, los fines de semana mostrarán una carga diferente a la de los días laborables. Y durante los períodos de vacaciones y días festivos, la carga de un sistema puede volver a verse diferente.

Diseño de pruebas

- Un **modelo de carga** es una descripción de los distintos niveles de carga esperados en un sistema de TI, que es la base para especificar y ejecutar pruebas de rendimiento.
- El modelo de carga consta de las siguientes partes:
  - Requisitos de desempeño: Los requisitos están inventariados y, si es necesario, detallados. Un requisito en cuanto al número de usuarios simultáneos, por ejemplo, puede ser inexacto debido al ritmo de trabajo variable, también porque no se sabe con qué frecuencia un usuario realiza una transacción. Por lo tanto, se requiere información adicional, preferiblemente el número de transacciones/accesos por unidad de tiempo.
  - Objeto de prueba: En base a los requisitos y la estimación de riesgos, se decide sobre el objeto de prueba (p. ej., la aplicación (cadena) completa o tal vez solo una base de datos individual o un servidor de aplicaciones). Esto tiene un impacto directo en los entornos de prueba requeridos.
  - Carga a generar: El objetivo final de un modelo de carga es generar una carga representativa en el sistema. Esta carga consta de un número específico de usuarios, cada uno de los cuales realiza determinadas tareas/transacciones (lo mas realista posible, usar un record & replay). Posteriormente se determina la frecuencia con la que se están realizando las transacciones y el número de usuarios requeridos para un período de tiempo específico.

Diseño de pruebas

- El **modelo de iteración** describe a nivel técnico cómo se cargará el objeto de prueba y cómo se medirá el rendimiento. Un modelo de iteración describe la forma (técnica) en la que se distribuye la carga, de modo que se pueda lograr rápidamente una combinación dinámica de acciones, que luego se puede mantener durante un cierto período de tiempo. El modelo de iteración depende del tipo de prueba de rendimiento que se esté diseñando.
- El **plan de métricas de desempeño** indica qué partes del entorno de prueba se monitorean como parte de la ejecución de la prueba. Se indica por parte qué métricas se recopilan, en qué intervalos y cómo se protegerá y presentará la información una vez finalizada. Los perfiles de carga muestran el grado en que los recursos del sistema (CPU, memoria, red) están cargados en realidad.
- El análisis deberá tener en cuenta el posible impacto del seguimiento del rendimiento y deberán restablecerse los ajustes para la continuación de las pruebas periódicas.

Inicio temprano

- Con la integración de las pruebas de rendimiento en el ciclo de vida de entrega de TI, es posible realizar una serie de mejoras que permitirán un nivel más exhaustivo de pruebas de rendimiento:
  - Haga que el rendimiento sea una parte importante de la definición de hecho. Esto no sólo requiere una atención especial al diseño y desarrollo para el rendimiento; también requiere una infraestructura de pruebas y una configuración de herramientas que proporcionen los beneficios de todas las variedades relevantes de pruebas de rendimiento durante los sprints.
  - Programe pruebas de rendimiento de un extremo a otro de manera oportuna con respecto a la versión de los sistemas de TI para permitir las últimas correcciones.
  - Facilite el ciclo de retroalimentación desde las pruebas de rendimiento de un extremo a otro y el monitoreo del rendimiento de la producción hasta cualquier parte del flujo de información de DevOps.
  - Realice pruebas de aceptación de rendimiento dedicadas (unidad/componente) integradas en un proceso de CI/CD.

https://jmeter.apache.org/

JMETER

Introducción

- Apache JMeter es un software de código abierto, una aplicación Java 100% pura, diseñada para cargar el comportamiento funcional de las pruebas y medir el rendimiento.
- Originalmente fue diseñado para probar aplicaciones web, pero desde entonces se ha expandido a otras funciones de prueba.
- Apache JMeter puede usarse para probar el rendimiento tanto en recursos estáticos como dinámicos: aplicaciones web dinámicas.
- Se puede usar para simular una carga pesada en un servidor, grupo de servidores, red u objeto para probar su resistencia o para analizar el rendimiento general bajo diferentes tipos de carga.
- JMeter no es apropiado para pruebas E2E dado que no es un navegador, funciona a nivel de protocolo. En lo que respecta a servicios web y servicios remotos, JMeter se parece a un navegador (o más bien, a múltiples navegadores); sin embargo, JMeter no realiza todas las acciones admitidas por los navegadores. En particular, JMeter no ejecuta el Javascript que se encuentra en las páginas HTML. Tampoco representa las páginas HTML como lo hace un navegador (es posible ver la respuesta como HTML, etc., pero los tiempos no se incluyen en ninguna muestra, y solo una muestra en una secuencia se muestra a la vez).

Características

- IDE de prueba con todas las funciones que permite la grabación rápida del plan de prueba (desde navegadores o aplicaciones nativas), construcción y depuración .
- Modo en línea de comandos para cargar la prueba desde cualquier SO compatible con Java (Linux, Windows, Mac OSX, ...)
- Un informe HTML completo y listo para presentar.
- Correlación sencilla mediante la capacidad de extraer datos de los formatos de respuesta más populares: HTML, JSON, XML o cualquier formato de texto
- Portabilidad completa y 100% Java puro.
- El marco completo de subprocesos múltiples permite el muestreo simultáneo mediante varios subprocesos y el muestreo simultáneo de diferentes funciones mediante grupos de subprocesos separados.
- Caché y análisis/reproducción fuera de línea de los resultados de las pruebas.

Características

- Núcleo altamente extensible:
  - Los muestreadores enchufables permiten capacidades de prueba ilimitadas.
  - Muestreadores de secuencias de comandos (lenguajes compatible con JSR223 como Groovy y BeanShell)
  - Se pueden elegir varias estadísticas de carga con temporizadores conectables.
  - Los complementos de análisis y visualización de datos permiten una gran extensibilidad y personalización.
  - Las funciones se pueden utilizar para proporcionar una entrada dinámica a una prueba o para proporcionar manipulación de datos.
  - Integración Continua simplificada a través de bibliotecas de código abierto de terceras partes para Maven, Gradle y Jenkins.

Tipos de aplicaciones/servidores/protocolo

- Web: HTTP, HTTPS (Java, NodeJS, PHP, ASP.NET, ...)
- Servicios Web de SOAP / REST
- FTP
- Base de datos a través de JDBC
- LDAP
- Middleware orientado a mensajes (MOM) a través de JMS
- Correo: SMTP (S), POP3 (S) e IMAP (S)
- Comandos nativos o scripts de shell
- TCP
- Objetos Java

Debilidades

- Aunque permite también el diseño y automatización de pruebas funcionales, existen herramientas más adecuadas para ello.
- JMeter NO se comporta como un navegador. Esto tiene varias implicaciones importantes: por defecto no guarda ni envía cookies, no interpreta código JavaScript, … Cualquiera de estas funcionalidades debe ser implementada específicamente en el testplan
- Con JMeter el tester trabaja a nivel de protocolos: el desarrollador de un testplan debe descender a este nivel, por lo que normalmente el tester tiene que apoyarse en herramientas adicionales durante el desarrollo de un testplan, como Firebug, HttpFox, SoapUI, Badboy, ...
- Los tipos de aplicaciones que se pueden testear con JMeter dependen de los protocolos que implementen las interfaces de acceso a la aplicación. Con JMeter se pueden testear los siguientes tipos de interfaces: HTTP, HTTPS, SOAP (sobre HTTP), XML-RPC (sobre HTTP), FTP, LDAP, POP3, IMAP, SMTP, JMS, JDBC y TCP. El testeo de otro tipo de interfaces no es inmediato. En concreto, con JMeter no es posible (con un esfuerzo razonable), implementar un testplan para una interfaz RMI o WebDav

Monitor de rendimiento de Windows

- El Monitor de rendimiento es la utilidad gráfica del sistema operativo que permite supervisar al sistema operativo Windows, principalmente para buscar cuellos de botella.
- Se basa en contadores de rendimiento asociados a los diferentes objetos del sistema, como procesadores, memoria, caché, subprocesos y procesos.
- Cada uno de estos objetos tiene asociado un conjunto de contadores que miden el uso de los dispositivos, la longitud de las colas, las demoras y otros indicadores del rendimiento y la congestión interna.

Contadores

- Disco:
  - PhysicalDisk: Avg. Disk Queue Length
  - PhysicalDisk: Avg. Disk sec/Read, PhysicalDisk: Avg. Disk sec/Write
  - Processor:% Privileged Time
- Procesador
  - Processor: % Processor Time
  - System: Processor Queue Length
- Memoria:
  - Memory: Page Faults/sec
  - Memory: Available Bytes
  - Memory: Pages/sec
- Red:
  - Network Interface: Bytes Total/sec
  - Network Interface: Current Bandwidth

Instalación

- Descargar JMeter
  - https://jmeter.apache.org/
- Descomprimir
- Descargar JMeter Plugins
  - https://jmeter-plugins.org/install
- Copiar plugins-manager.jar en jmeter/lib/ext
- Configurar, si es necesario, jmeter/bin/jmeter.bat
- Abrir GUI: jmeter/bin/jmeterw.cmd

Modos

- Modo GUI
- Modo sin GUI (modo de línea de comandos)
  - Para las pruebas de carga, se debe ejecutar JMeter en modo sin GUI para obtener los resultados óptimos.

    jmeter -n -t my_test.jmx -l log.jtl
- Modo de servidor
  - Para las pruebas distribuidas, se ejecuta JMeter en modo servidor en los nodos remotos y se controla los servidores desde el GUI. También puede utilizar el modo no GUI para ejecutar pruebas remotas. Para iniciar cada servidor en cada host:

    jmeter-server

Elementos de un plan de prueba

- Grupo de hilos
- Controladores
- Receptores
- Temporizadores
- Aserciones
- Elementos de configuración
- Elementos del preprocesador
- Elementos de postprocesador
- Propiedades y variables
- Funciones

Grupo de hilos

- Los elementos del grupo de hilos son los puntos de inicio de cualquier plan de prueba. Todos los controladores y muestreadores deben estar bajo un grupo de subprocesos. El grupo de hilos controla el número de hilos que JMeter usará para ejecutar la prueba. Los controles para un grupo de hilos le permiten:
  - Establecer el número de hilos
  - Establecer el período de aceleración
  - Establecer el número de veces para ejecutar la prueba.
- Cada hilo ejecutará el plan de prueba en su totalidad y de forma completamente independiente de otros hilos de prueba. Se utilizan varios subprocesos para simular conexiones simultáneas a su aplicación de servidor.
- El período de aceleración le dice a JMeter cuánto tiempo se tarda en "aumentar" la cantidad total de hilos elegidos.
- La expansión debe ser lo suficientemente larga para evitar una carga de trabajo demasiado grande al inicio de una prueba, y lo suficientemente corta para que los últimos hilos comiencen a ejecutarse antes de que terminen los primeros (a menos que uno quiera que eso ocurra).
- Así mismo permite controlar la acción a tomar después de un error del muestrador.

Controladores

- Los controladores conducen el procesamiento de una prueba. JMeter tiene dos tipos de controladores: muestreadores y controladores lógicos.
- Los muestreadores le dicen a JMeter que envíe solicitudes a un servidor. Por ejemplo, se agrega un muestreador de Petición HTTP si se desea que JMeter envíe una solicitud HTTP. También se puede personalizar una solicitud agregando uno o más elementos de configuración a un muestreador.
- Los controladores lógicos permiten personalizar la lógica que JMeter utiliza para decidir cuándo enviar solicitudes. Por ejemplo, puede agregar un controlador de lógica de intercalación para alternar entre dos muestreadores de solicitud HTTP.
- El Fragmento de prueba es un tipo especial de controlador que existe en el árbol del Plan de prueba al mismo nivel que el Grupo de subprocesos. Se distingue de un Grupo de subprocesos en que no se ejecuta a menos que sea referenciado por un Controlador de Módulo o de Include. Este elemento es únicamente para reutilizar el código dentro de los planes de prueba.

Muestreadores

- Los muestreadores le dicen a JMeter que envíe solicitudes a un servidor y espere una respuesta. Se procesan en el orden en que aparecen en el árbol. Los controladores se pueden utilizar para modificar el número de repeticiones de una muestra.
- Los muestreadores JMeter incluyen:
  - Solicitud HTTP (también se puede utilizar para el servicio web SOAP o REST)
  - Solicitud de FTP
  - Solicitud de correo
  - Solicitud de TCP
  - Solicitud de objeto Java
  - Solicitud JDBC
  - Solicitud JMS
  - Solicitud de prueba de JUnit
  - Solicitud LDAP
  - Solicitud de proceso OS

Controladores lógicos

- Los controladores lógicos le permiten personalizar la lógica que JMeter utiliza para decidir cuándo enviar solicitudes. Los controladores lógicos pueden cambiar el orden de las solicitudes provenientes de sus elementos secundarios. Pueden modificar las solicitudes ellos mismos, hacer que JMeter repita las solicitudes, etc.
- Mediante los controladores lógicos se pueden establecer transacciones, orden, frecuencia y el flujo de control con condicionales, bucles, …

Logic Controllers

- Simple Controller
- Loop Controller
- Once Only Controller
- Interleave Controller
- Random Controller
- Random Order Controller
- Throughput Controller
- Runtime Controller
- If Controller
- While Controller
- Switch Controller
- ForEach Controller
- Module Controller
- Include Controller
- Transaction Controller
- Recording Controller
- Critical Section Controller

Receptores

- Los receptores proporcionan acceso a la información que JMeter recopila sobre los casos de prueba mientras JMeter se ejecuta. La mayoría de los receptores realizan varias funciones además de "escuchar" los resultados de la prueba. También proporcionan medios para ver, guardar y leer los resultados de las pruebas guardadas.
- Se pueden agregar receptores en cualquier parte de la prueba, incluso directamente en el plan de prueba. Recopilarán datos solo de elementos en el que están asociado o por debajo de su nivel.
- Todos los receptores guardan los mismos datos, la única diferencia está en la forma en que se presentan los datos en pantalla.
- Los receptores pueden dirigir los datos a un archivo para su uso posterior, proporcionan un campo para indicar el archivo para almacenar datos y permiten elegir qué campos guardar así como el formato: CSV o XML.
- Los receptores pueden presentar los datos en formato grafico, tabular, árbol, especializado y para depurar. Algunos proporcionan información de resumen o agregación.

Listeners

- Sample Result Save Configuration
- Graph Results
- Assertion Results
- View Results Tree
- Aggregate Report
- View Results in Table
- Simple Data Writer
- Aggregate Graph
- Response Time Graph
- Mailer Visualizer
- BeanShell Listener
- Summary Report
- Save Responses to a file
- JSR223 Listener
- Generate Summary Results
- Comparison Assertion Visualizer
- Backend Listener

Temporizadores

- De forma predeterminada, un hilo de JMeter ejecuta los muestreadores en secuencia sin pausa. Es recomendable especificar un retraso agregando uno de los temporizadores disponibles al grupo de subprocesos, si no, JMeter podría saturar al servidor al realizar demasiadas solicitudes en un período de tiempo muy corto.
- Un temporizador hará que JMeter demore un cierto tiempo antes de cada muestra que esté dentro de su alcance.
- Si se agrega más de un temporizador a un grupo de subprocesos, JMeter toma la suma de los temporizadores y se detiene durante ese tiempo antes de ejecutar los muestreadores a los que se aplican los temporizadores. Los temporizadores se pueden agregar como hijos de muestreadores o controladores para restringir a los que se aplican.
- Para proporcionar una pausa en un solo lugar en un plan de prueba, se puede usar el Muestreador de acciones de prueba.

Timers

- Constant Timer
- Gaussian Random Timer
- Uniform Random Timer
- Constant Throughput Timer
- Precise Throughput Timer
- Synchronizing Timer
- BeanShell Timer
- JSR223 Timer
- Poisson Random Timer

Aserciones

- Las aserciones permiten afirmar hechos sobre las respuestas recibidas del servidor que se está probando, en caso de no cumplirse JMeter lo marcará como una solicitud fallida. Permiten comprobar que la aplicación está devolviendo los resultados esperados.
- Se pueden establecer aserciones sobre el contenido, tamaño y duración de las respuestas. Hay disponibles aserciones especificas para determinados muestreadores.
- Las aserciones se aplican a todos los muestreadores que se encuentran en su alcance. Para restringir una aserción a un solo muestreador se le debe agregar como elemento secundario.
- Para ver los resultados de la aserción, se agrega un receptor de resultado de aserción. Las aserciones fallidas también se mostrarán en la vista de árbol y en los receptores de la tabla, y contarán para el % de error en los informes de agregados y resumen.

Assertions

- Response Assertion
- Duration Assertion
- Size Assertion
- XML Assertion
- BeanShell Assertion
- MD5Hex Assertion
- HTML Assertion
- XPath Assertion
- XPath2 Assertion
- XML Schema Assertion
- JSR223 Assertion
- Compare Assertion
- SMIME Assertion
- JSON Assertion
- JSON JMESPath Assertion

Elementos de configuración

- Los elementos de configuración se pueden usar para configurar valores predeterminados y variables para su uso posterior por parte de los muestreadores. Hay que tener en cuenta que estos elementos se procesan al inicio del alcance en el que se encuentran, es decir, antes de cualquier muestreador en el mismo alcance.
- Están disponibles configuradores para:
  - Valores predeterminados para peticiones HTTP, FTP, Java, LDAP
  - Configuración de conexión JDBC, del almacén de claves, de muestra de TCP, de inicio de sesión
  - Variables definidas por el usuario, aleatorias, contadores, conjuntos de datos CSV
  - Administrador de caché HTTP y DNS
  - Administrador de autorización, cookies y encabezados HTTP

Configuration Elements

- CSV Data Set Config
- FTP Request Defaults
- DNS Cache Manager
- HTTP Authorization Manager
- HTTP Cache Manager
- HTTP Cookie Manager
- HTTP Request Defaults
- HTTP Header Manager
- Java Request Defaults
- JDBC Connection Configuration
- Keystore Configuration
- Login Config Element
- LDAP Request Defaults
- LDAP Extended Request Defaults
- TCP Sampler Config
- User Defined Variables
- Random Variable
- Counter
- Simple Config Element
- MongoDB Source Config (DEPRECATED)
- Bolt Connection Configuration

Elementos del preprocesador

- Un preprocesador ejecuta alguna acción antes de que se realice una solicitud de muestra.
- Si se adjunta un preprocesador a un elemento muestreador, entonces se ejecutará justo antes de que se ejecute ese elemento.
- Un preprocesador se utiliza principalmente para modificar la configuración de una solicitud de muestra justo antes de que se ejecute, o para actualizar las variables que no se extraen del texto de respuesta.

Elementos del postprocesador

- Un postprocesador ejecuta alguna acción después de que se haya realizado una solicitud de muestra.
- Si un postprocesador está conectado a un elemento Sampler, se ejecutará justo después de que se ejecute ese elemento.
- Un postprocesador se utiliza para procesar los datos de respuesta, a menudo para extraer valores de él.

Processors

Pre Processors

- HTML Link Parser
- HTTP URL Re-writing Modifier
- User Parameters
- BeanShell PreProcessor
- JSR223 PreProcessor
- JDBC PreProcessor
- RegEx User Parameters
- Sample Timeout

Post Processors

- Regular Expression Extractor
- CSS Selector Extractor (was: CSS/JQuery Extractor)
- XPath2 Extractor
- XPath Extractor
- JSON JMESPath Extractor
- Result Status Action Handler
- BeanShell PostProcessor
- JSR223 PostProcessor
- JDBC PostProcessor
- JSON Extractor
- Boundary Extractor

Propiedades y variables

- Las propiedades son globales para jmeter, y se usan principalmente para definir algunos de los valores predeterminados que usa JMeter. Por ejemplo, la propiedad remote_hosts define los servidores que JMeter intentará ejecutar de forma remota. Se puede hacer referencia a las propiedades en los planes de prueba pero no se puede usar para valores específicos de hilos. Las propiedades de JMeter se definen en los ficheros jmeter.properties y user.properties o en la línea de comandos.
- Las variables de JMeter son locales para cada hilo. Los valores pueden ser los mismos para cada hilo, o pueden ser diferentes. Si una variable es actualizada por un hilo, solo se cambia la copia de la variable en el hilo. Por ejemplo, el Postprocesador de Expresión Regular Extractor establecerá sus variables de acuerdo con la muestra que su hilo haya leído, y estas pueden usarse más adelante por el mismo hilo.
- Los valores definidos por el plan de prueba y el configurador de variables definidas por el usuario (UDV) están disponibles para todo el plan de prueba desde el inicio. Si la misma variable está definida por múltiples UDV, solo tendrá efecto la última.

Propiedades y variables

- Una vez que un hilo ha comenzado, el conjunto inicial de variables se copia a cada hilo. Otros elementos, como el Preprocesador de parámetros de usuario o el Postprocesador de extracción de expresiones regulares, se pueden usar para redefinir las mismas variables (o crear nuevas). Estas redefiniciones solo se aplican al hilo actual.
- Las variables no tienen por que variar, se pueden definir una vez y, si se dejan solas, no cambiarán de valor. Por lo tanto, se pueden usar como abreviatura para expresiones que aparecen con frecuencia en un plan de prueba o para los elementos que son constantes durante una ejecución pero que pueden variar entre ejecuciones. Por ejemplo, el nombre de un host.
- Tanto las variables como las propiedades distinguen entre mayúsculas y minúsculas.

Funciones

- Las funciones de JMeter son valores calculados que pueden rellenar los campos de cualquier muestreador u otro elemento en un árbol de prueba.
- Hay dos tipos de funciones: valores estáticos definidos por el usuario (o variables) y funciones incorporadas.
- Los valores estáticos definidos por el usuario permiten al usuario definir las variables que se reemplazarán con su valor estático cuando se compila un árbol de prueba y se envía para que se ejecute. Este reemplazo ocurre una vez al comienzo de la prueba.
- Con las funciones integradas, se pueden calcular nuevos valores en tiempo de ejecución dependiendo de los datos de respuesta anteriores, en qué hilo se encuentra la función, la hora y muchas otras fuentes. Se generan nuevos valores para cada solicitud a lo largo del curso de la prueba.

Funciones

- Las funciones y variables se pueden escribir en cualquier campo de cualquier componente de prueba.
- Para invocar las funciones:
  ${__functionName (var1, var2, var3)}
- Si un parámetro de función contiene una coma, entonces asegúrese de escapar con " \ ", de lo contrario JMeter lo tratará como un delimitador de parámetros. Las funciones que no requieren parámetros pueden omitir los paréntesis.
- Las variables se referencian de la siguiente manera:
  ${VARIABLE}
- Las propiedades deben ser referenciadas utilizando la función __P o __property:
  ${__property(user.dir)}
- Si se hace referencia a una función o variable no definida, JMeter NO informa/registra un error: la referencia se devuelve sin cambios.

Orden de ejecución

1. Elementos de configuración
2. Preprocesadores
3. Temporizadores
4. Controladores
5. Postprocesadores (si SampleResult no es nulo)
6. Aserciones (si SampleResult no es nulo)
7. Receptores (si SampleResult no es nulo)

   – Los temporizadores, las aserciones, los preprocesadores y los postprocesadores solo se procesan si hay un muestreador al que aplicarlo.
   – Los controladores lógicos y los muestreadores se procesan en el orden en que aparecen en el árbol.
   – Otros elementos de prueba se procesan de acuerdo con el alcance en el que se encuentran y el tipo de elemento de prueba.

Reglas de alcance

- El árbol de la prueba JMeter contiene elementos que son jerárquicos y están ordenados. Algunos elementos en los árboles de prueba son estrictamente jerárquicos (receptores, elementos de configuración, postprocesadores, preprocesadores, aserciones, temporizadores) y algunos están ordenados por principio (controladores, muestreadores).
- Cuando se crea el plan de prueba, se creará una lista ordenada de solicitudes de muestra (a través de muestreadores) que representa un conjunto de pasos a ejecutar. Estas solicitudes a menudo se organizan dentro de los controladores que también se ordenan.
- Otros elementos son jerárquicos. Una aserción, por ejemplo, es jerárquica en el árbol de prueba. Si su padre es una solicitud, entonces se aplica a esa solicitud. Si su padre es un controlador, afecta a todas las solicitudes que contiene.

Proceso de Prueba

- Preguntas iniciales¶
  - ¿Cuál es el número promedio anticipado de usuarios (carga normal)?
  - ¿Cuál es el número máximo de usuarios previsto?
  - ¿Cuándo es un buen momento para cargar y probar la aplicación (es decir, fuera de horario o fines de semana), teniendo en cuenta que esto puede muy bien bloquear uno o más de los servidores?
  - ¿La aplicación tiene estado? Si es así, ¿cómo la administra la aplicación (cookies, reescritura de URL o algún otro método)?
  - ¿Cuál es el objetivo que se pretende lograr con la prueba?
  - ¿Quién sabe cómo funciona la aplicación?
- La secuencia normal es:
  - Prueba (volumen bajo, ¿podemos comparar nuestra aplicación?)
  - Rendimiento (el número promedio de usuarios)
  - Carga (el número máximo de usuarios)
  - Estrés (¿cuál es el límite?)
- El proceso de prueba puede pasar de la prueba de caja negra a la prueba de caja blanca (la diferencia es que la primera no requiere conocimiento de la aplicación [se trata como una "caja negra"] mientras que la segunda requiere algún conocimiento de la aplicación). No es raro que se descubran problemas con la aplicación durante este proceso, así que prepárate para defender tu trabajo.

Prueba de carga

- En el GUI:
  - Crear y Validar Plan de Pruebas
  - Quitar/deshabilitar Receptores
  - Cerrar
- Ejecutar el plan en modo comando y exportar resultados:
  - jmeter –n *-f* -t demo.jmx -l demo.jtl -e -o demoinf
    - Sumario cada 30 segundos, totalizados
- Analizar resultados en el GUI:
  - Crear plan con receptores.
  - Cargar los ficheros de resultados
  - Ver informe HTML de resultados

Pruebas distribuidas

- En los sistemas esclavos, ejecutar jmeter/bin/ejecute jmeter-server.bat (jmeter-server en unix).
- En el sistema maestro, modificar el fichero de propiedades jmeter/bin/jmeter.properties:
  remote_hosts=192.168.0.10,192.168.0.11,192.168.0.12
- En el sistema maestro, iniciar JMeter y abrir el plan de prueba que se desea utilizar.
- Para ejecutar la prueba:
  - Iniciar un solo cliente: Run > Remote Start > 192.168.0.10
  - Iniciar todos los clientes: Run > Remote Start All
- Para detener los sistemas esclavos, Run > Remote Stop > 192.168.0.10 ó Run > Remote Stop All

Informes y Análisis

- Apache JMeter Dashboard
  - Generación después de la prueba de carga:

    jmeter -n -t <archivo de prueba JMX> -l <archivo de registro de prueba> -e -o <Ruta a la carpeta de salida>
  - Generación a partir de un archivo de registro CSV de muestra existente

    jmeter -g <archivo de registro> -o <Ruta a la carpeta de salida>
- Respondedores

Respondedores de informe

- El receptor de resultados gráficos genera un gráfico simple que representa todos los tiempos de muestra. En la parte inferior del gráfico, la muestra actual (negro), el promedio actual de todas las muestras (azul), la desviación estándar actual (rojo) y la tasa de rendimiento actual (verde) se muestran en milisegundos.
- El informe agregado crea una fila de tabla para cada solicitud con un nombre diferente en la prueba. Para cada solicitud, totaliza la información de respuesta y proporciona el recuento de solicitudes, mínimo, máximo, promedio, tasa de error, rendimiento aproximado (solicitud / segundo) y rendimiento de Kilobytes por segundo. Una vez que se realiza la prueba, el rendimiento es el real a lo largo de toda la prueba.
- El gráfico agregado es similar al informe agregado. La principal diferencia es que el gráfico agregado proporciona una manera fácil de generar gráficos de barras exportables a PNG.
- El informe de resumen crea una fila de tabla para cada solicitud con un nombre diferente en su prueba. Similar al Informe agregado, pero usando menos memoria.
- El gráfico de tiempo de respuesta dibuja un gráfico de líneas que muestra la evolución del tiempo de respuesta durante la prueba, para cada solicitud etiquetada. Si existen muchas muestras para la misma marca de tiempo, se muestra el valor medio.

PRUEBAS DE SEGURIDAD

Introducción

- Seguridad es la capacidad de los sistemas de información o de las redes para resistir, con un determinado nivel de confianza, los accidentes o las acciones ilícitas o malintencionadas que comprometan la disponibilidad, autenticidad, integridad y confidencialidad de los datos almacenados o transmitidos y de los servicios que dichas redes y sistemas ofrecen o hacen accesibles.
- El objetivo a proteger es la misión de la Organización, teniendo en cuenta las dimensiones de la seguridad:
  - **Disponibilidad**: o disposición de los servicios a ser usados cuando sea necesario. La carencia de disponibilidad supone una interrupción del servicio y afecta directamente a la productividad de las organizaciones.
  - **Integridad**: o mantenimiento de las características de completitud y corrección de los datos. Contra la integridad, la información puede aparecer manipulada, corrupta o incompleta y afecta directamente al correcto desempeño de las funciones de una Organización.
  - **Confidencialidad**: o que la información llegue solamente a las personas autorizadas. Contra la confidencialidad o secreto pueden darse fugas y filtraciones de información, así como accesos no autoriza- dos. La confidencialidad es una propiedad de difícil recuperación, pudiendo minar la confianza de los demás en la organización que no es diligente en el mantenimiento del secreto, y pudiendo suponer el incumplimiento de leyes y compromisos contractuales relativos a la custodia de los datos.
- Se pueden añadir otras derivadas de la percepción de los usuarios de los sistemas de información:
  - **Autenticidad**: Propiedad o característica consistente en que una entidad es quien dice ser o bien que garantiza la fuente de la que proceden los datos. Contra la autenticidad de la información podemos tener manipulación del origen o el contenido de los datos. Contra la autenticidad de los usuarios de los servicios de acceso, podemos tener suplantación de identidad.
  - **Trazabilidad**: Aseguramiento de que en todo momento se podrá determinar quién hizo qué y en qué momento. La trazabilidad es esencial para analizar los incidentes, perseguir a los atacantes y aprender de la experiencia. La trazabilidad se materializa en la integridad de los registros de actividad.

MAGERIT v3

- MAGERIT versión 3, siglas de Metodología de Análisis y Gestión de Riesgos de los Sistemas de Información, es la metodología de análisis y gestión de riesgos elaborada en su día por el antiguo Consejo Superior de Administración Electrónica y actualmente mantenida por la Secretaría General de Administración Digital (Ministerio de Asuntos Económicos y Transformación Digital) con la colaboración del Centro Criptológico Nacional (CCN).
- MAGERIT es una metodología de carácter público que puede ser utilizada libremente y no requiere autorización previa. Interesa principalmente a las entidades en el ámbito de aplicación del Esquema Nacional de Seguridad (ENS) para satisfacer el principio de la gestión de la seguridad basada en riesgos, así como el requisito de análisis y gestión de riesgos, considerando la dependencia de las tecnologías de la información para cumplir misiones, prestar servicios y alcanzar los objetivos de la organización.
- Siguiendo la terminología de la normativa ISO 31000, MAGERIT responde a lo que se denomina Proceso de Gestión de los Riesgos. En otras palabras, MAGERIT implementa el Proceso de Gestión de Riesgos dentro de un marco de trabajo para que los órganos de gobierno tomen decisiones teniendo en cuenta los riesgos derivados del uso de tecnologías de la información.

MAGERIT v3

- La gestión de los riesgos es una piedra angular en las guías de buen gobierno [ISO 38500], público o privado, donde se considera un principio fundamental que las decisiones de gobierno se fundamenten en el conocimiento de los riesgos que implican.
- MAGERIT persigue los siguientes Objetivos Directos:
  - Concienciar a los responsables de las organizaciones de información de la existencia de riesgos y de la necesidad de gestionarlos
  - Ofrecer un método sistemático para analizar los riesgos derivados del uso de tecnologías de la información y comunicaciones (TIC)
  - Ayudar a descubrir y planificar el tratamiento oportuno para mantener los riesgos bajo control Indirectos
  - Preparar a la Organización para procesos de evaluación, auditoría, certificación o acreditación, según corresponda en cada caso
- MAGERIT versión 3 se estructura en tres libros: "Método", "Catálogo de Elementos" y "Guía de Técnicas".
- PILAR es una herramienta que implementa la metodología MAGERIT de análisis y gestión de riesgos, desarrollada por el CCN y de amplia utilización en la administración pública.

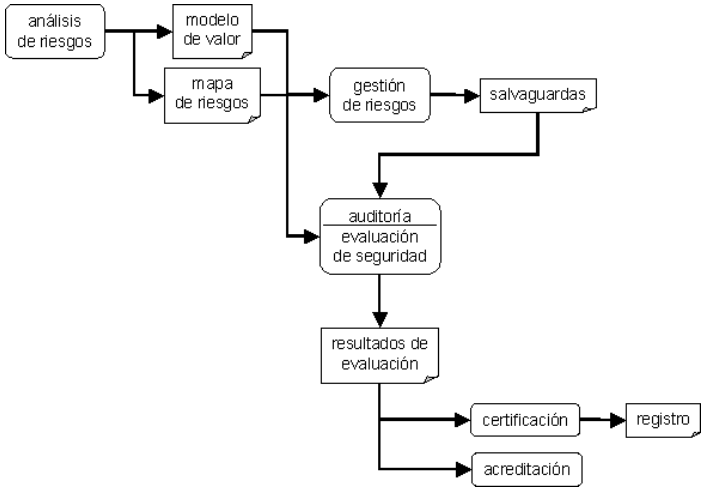
MAGERIT v3

- El **Método de Análisis de Riesgos (MAR)** constituye el núcleo del enfoque de Magerit v3. Proporciona una estructura sistemática y bien definida para evaluar los riesgos asociados con los activos de información, las amenazas que los acechan y las vulnerabilidades presentes en los sistemas. El MAR se basa en una serie de pasos bien definidos, que incluyen la identificación y valoración de activos, la identificación y análisis de amenazas, la evaluación de vulnerabilidades y la estimación de riesgos. Este método permite a los profesionales de la seguridad tomar decisiones informadas y priorizar las medidas de protección necesarias para mitigar los riesgos identificados.
- Los **Proyectos de Análisis de Riesgos (PAR)** son implementaciones específicas del MAR en entornos concretos. Estos proyectos se llevan a cabo en organizaciones para evaluar los riesgos asociados con sus sistemas de información y establecer medidas de seguridad adecuadas. Los PAR se inician con la definición de los objetivos y alcance del proyecto, seguidos de la identificación y valoración de activos críticos, amenazas relevantes y vulnerabilidades existentes. A través de un enfoque sistemático, los PAR permiten identificar los riesgos más significativos y diseñar estrategias de mitigación adecuadas. Los resultados obtenidos de los PAR proporcionan una base sólida para la toma de decisiones en materia de seguridad.
- El **Plan de Seguridad (PS)** es el documento final que se deriva del proceso de análisis de riesgos y define las medidas de seguridad necesarias para proteger los activos de información de una organización. El PS se basa en los resultados obtenidos del análisis de riesgos y proporciona una visión clara de las acciones y controles de seguridad que deben implementarse. Incluye políticas, procedimientos, directrices y normas específicas que guían la gestión de la seguridad de la información en la organización. El PS se actualiza periódicamente para reflejar los cambios en el entorno tecnológico y las nuevas amenazas emergentes.

Riesgo

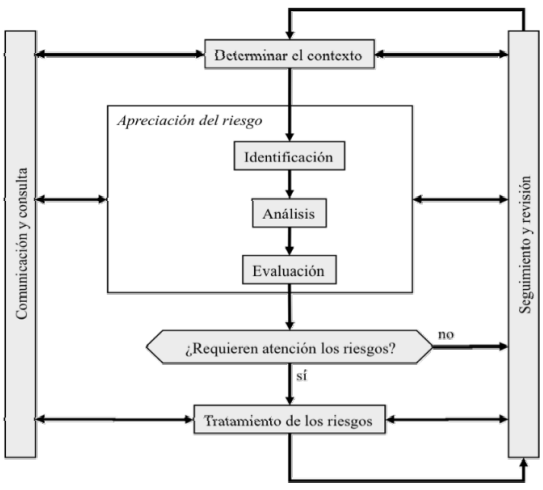
- Todas estas características pueden ser requeridas o no dependiendo de cada caso. Cuando se requieren, no es evidente que se disfruten sin más. Lo habitual que haya que poner medios y esfuerzo para conseguirlas. A racionalizar este esfuerzo se dedican las metodologías de análisis y gestión de riesgos que comienzan con una definición:
  - **Riesgo**: estimación del grado de exposición a que una amenaza se materialice sobre uno o más activos causando daños o perjuicios a la Organización.
- El riesgo indica lo que le podría pasar a los activos si no se protegieran adecuadamente. Es importante saber qué características son de interés en cada activo, así como saber en qué medida estas características están en peligro y sabiendo lo que podría pasar, hay que tomar decisiones:
  - **Análisis de riesgos**: proceso sistemático para estimar la magnitud de los riesgos a que está expuesta una Organización.
  - **Tratamiento de los riesgos**: proceso destinado a modificar el riesgo.
- El análisis de riesgos es una piedra angular de los procesos de evaluación, certificación, auditoría y acreditación que formalizan la confianza que merece un sistema de información.

Contexto



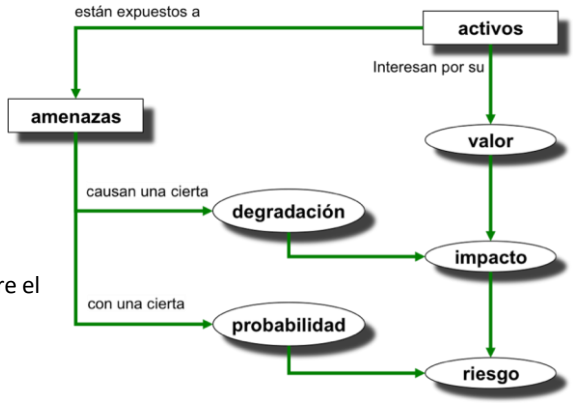
Análisis de riesgos

- El análisis de riesgos considera los siguientes elementos:
  - activos, que son los elementos del sistema de información (o estrechamente relacionados con este) que soportan la misión de la Organización
  - amenazas, que son cosas que les pueden pasar a los activos causando un perjuicio a la Organización
  - salvaguardas (o contra medidas), que son medidas de protección desplegadas para que aquellas amenazas no causen [tanto] daño.
- Con estos elementos se puede estimar:
  - el impacto: lo que podría pasar
  - el riesgo: lo que probablemente pase
- El análisis de riesgos permite analizar estos elementos de forma metódica para llegar a conclusiones con fundamento y proceder a la fase de tratamiento.



Método de análisis de riesgos

- El análisis de riesgos es una aproximación metódica para determinar el riesgo siguiendo unos pasos pautados:
  1. determinar los activos relevantes para la Organización, su interrelación y su valor, en el sentido de qué perjuicio (coste) supondría su degradación
  2. determinar a qué amenazas están expuestos aquellos activos
  3. determinar qué salvaguardas hay dispuestas y cuán eficaces son frente al riesgo
  4. estimar el impacto, definido como el daño sobre el activo derivado de la materialización de la amenaza
  5. estimar el riesgo, definido como el impacto ponderado con la tasa de ocurrencia (o expectativa de materialización) de la amenaza



Activos

- Un activo es un componente o funcionalidad de un sistema de información susceptible de ser atacado deliberada o accidentalmente con consecuencias para la organización.
- En un sistema de información, las dos cosas esenciales son la **información** que maneja y los **servicios** que presta. Estos son los activos esenciales y marcan los requisitos de seguridad para todos los demás activos del sistema que dependen de ellos, como son:
  - **Datos** que materializan la información.
  - **Servicios auxiliares** que se necesitan para poder organizar el sistema.
  - Las **aplicaciones** informáticas (software) que permiten manejar los datos.
  - Los **equipos** informáticos (hardware) y que permiten hospedar datos, aplicaciones y servicios.
  - Los **soportes** de información que son dispositivos de almacenamiento de datos.
  - El **equipamiento auxiliar** que complementa el material informático.
  - Las redes de **comunicaciones** que permiten intercambiar datos.
  - Las **instalaciones** que acogen equipos informáticos y de comunicaciones.
  - Las **personas** que explotan u operan todos los elementos anteriormente citados.
- El primer paso es identificar, dimensionar y valorar los activos.

Dimensiones de los activos

- De un activo puede interesar calibrar diferentes dimensiones [UNE 71504:2008]:
  - **Confidencialidad** [C]: Propiedad o característica consistente en que la información ni se pone a disposición, ni se revela a individuos, entidades o procesos no autorizados. ¿qué daño causaría que lo conociera quien no debe? Esta valoración es típica de datos.
  - **Integridad** [I]: Propiedad o característica consistente en que el activo de información no ha sido alterado de manera no autorizada. ¿qué perjuicio causaría que estuviera dañado o corrupto? Esta valoración es típica de los datos, que pueden estar manipulados, ser total o parcialmente falsos o, incluso, faltar datos.
  - **Disponibilidad** [D]: Propiedad o característica de los activos consistente en que las entidades o procesos autorizados tienen acceso a los mismos cuando lo requieren. ¿qué perjuicio causaría no tenerlo o no poder utilizarlo? Esta valoración es típica de los servicios.
- En los activos esenciales frecuentemente es útil valorar:
  - **Autenticidad** [A]: Propiedad o característica consistente en que una entidad es quien dice ser o bien que garantiza la fuente de la que proceden los datos. ¿qué perjuicio causaría no saber exactamente quien hace o ha hecho cada cosa? Esta valoración es típica de servicios (autenticidad del usuario) y de los datos (autenticidad de quien accede a los datos para escribir o, simplemente, consultar)
  - **Trazabilidad** [T] del uso del servicio: Propiedad o característica consistente en que las actuaciones de una entidad pueden ser imputadas exclusivamente a dicha entidad. ¿qué daño causaría no saber a quién se le presta tal servicio? O sea, ¿quién hace qué y cuándo?

Valoración de los activos

- Una vez determinadas qué dimensiones (de seguridad) interesan de un activo hay que proceder a valorarlo. La valoración es la determinación del coste que supondría recuperarse de una incidencia que destrozara el activo.
- Hay muchos factores a considerar:
  - coste de reposición: adquisición e instalación
  - coste de mano de obra (especializada) invertida en recuperar (el valor) del activo
  - lucro cesante: pérdida de ingresos
  - capacidad de operar: confianza de los usuarios y proveedores que se traduce en una pérdida de actividad o en peores condiciones económicas
  - sanciones por incumplimiento de la ley u obligaciones contractuales
  - daños a otros activos (propios o ajenos), a personas, medioambientales, …
- La valoración puede ser cuantitativa (con una cantidad numérica) o cualitativa (en alguna escala de niveles). Los criterios más importantes a respetar son:
  - la homogeneidad: es importante poder comparar valores aunque sean de diferentes dimensiones a fin de poder combinar valores propios y valores acumulados, así como poder determinar si es más grave el daño en una dimensión o en otra
  - la relatividad: es importante poder relativizar el valor de un activo en comparación con otros activos

Amenazas

- El siguiente paso consiste en determinar las amenazas que pueden afectar a cada activo. Y, de todo lo que puede ocurrir, interesa lo que puede pasarle a nuestros activos y causar un daño.
- Una amenazas es una causa potencial de un incidente que puede causar daños a un sistema de información o a una organización.
- Para la Identificación de las amenazas, las causas típicas y dimensiones afectadas son:
  - De origen natural [D]: Hay accidentes naturales (terremotos, inundaciones, ...). Ante esos avatares el sistema de información es víctima pasiva, pero de todas formas tendremos en cuenta lo que puede suceder.
  - Del entorno (de origen industrial) [D]: Hay desastres industriales (contaminación, fallos eléctricos, ...) ante los cuales el sistema de información es víctima pasiva pero no hay que permanecer indefensos.
  - Defectos de las aplicaciones [C, I, D, A, T]: Hay problemas que nacen directamente en el equipamiento propio por defectos en su diseño o en su implementación, con consecuencias potencialmente negativas sobre el sistema. Frecuentemente se denominan vulnerabilidades técnicas o, simplemente, 'vulnerabilidades'.
  - Causadas por las personas de forma accidental [C, I, D]: Las personas con acceso al sistema de información pueden ser causa de problemas no intencionados, típicamente por error o por omisión.
  - Causadas por las personas de forma deliberada [C, I, D, A, T]: Las personas con acceso al sistema de información pueden ser causa de problemas intencionados: ataques deliberados; bien con ánimo de beneficiarse indebidamente, bien con ánimo de causar daños y perjuicios a los legítimos propietarios.

Clasificación de las amenazas

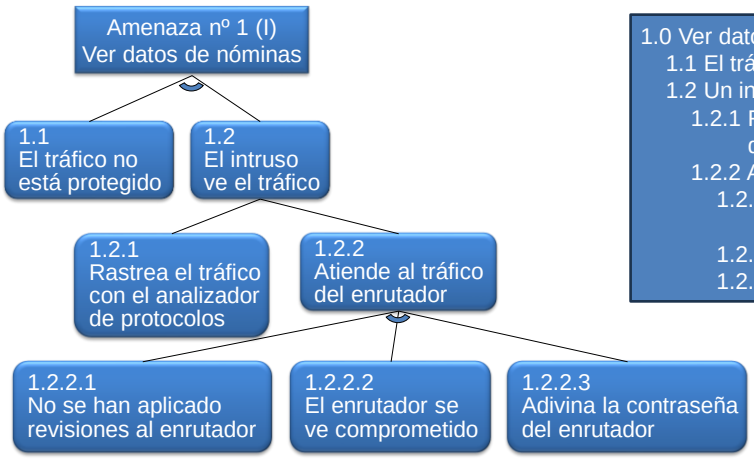
Matriz activos-amenazas

	Origen natural	Entorno	Defectos	Accidentales	Deliberadas
Datos					
Servicios					
Aplicaciones					
Equipos					
…					

Matriz dimensiones-amenazas

	Origen natural	Entorno	Defectos	Accidentales	Deliberadas
Confidencialidad					
Integridad					
Disponibilidad					
Autenticidad					
Trazabilidad					

Identificar las amenazas mediante árboles de ataques



1.0 Ver datos de nóminas (I)
- 1.1 El tráfico no está protegido (Y)
- 1.2 Un intruso ve el tráfico
  - 1.2.1 Rastrea al tráfico con el analizador de protocolos
  - 1.2.2 Atiende al tráfico del enrutador
    - 1.2.2.1 No se han aplicado revisiones al enrutador (Y)
    - 1.2.2.2 El enrutador se ve comprometido
    - 1.2.2.3 Adivina la contraseña del enrutador

Valoración de las amenazas

- No todas las amenazas afectan a todos los activos, sino que hay una cierta relación entre el tipo de activo y lo que le podría ocurrir. Cuando un activo es víctima de una amenaza, no se ve afectado en todas sus dimensiones, ni en la misma cuantía.
- Una vez determinado que una amenaza puede perjudicar a un activo, hay que valorar su influencia en el valor del activo, en dos sentidos:
  - **degradación**: cuán perjudicado resultaría el [valor del] activo
  - **probabilidad**: cuán probable o improbable es que se materialice la amenaza
- Se denomina **impacto** a la medida del daño sobre el activo derivado de la materialización de una amenaza. Conociendo el valor de los activos (en varias dimensiones) y la degradación que causan las amenazas, se puede calcular el impacto que estas tendrían sobre el sistema. El impacto se calcula para cada activo, por cada amenaza y en cada dimensión.
  - El impacto acumulado es el calculado sobre un activo teniendo en cuenta su valor acumulado (el propio mas el acumulado de los activos que dependen de él) y las amenazas a que está expuesto. Permite determinar las salvaguardas de que hay que dotar a los medios de trabajo: protección de los equipos, copias de respaldo, etc.
  - El impacto repercutido es el calculado sobre un activo teniendo en cuenta su valor propio y las amenazas a que están expuestos los activos de los que depende. Permite determinar las consecuencias de las incidencias técnicas sobre la misión del sistema de información.

Riesgo potencial

- Se denomina **riesgo** a la medida del daño probable sobre un sistema. El riesgo crece con el impacto y con la probabilidad de ocurrencia, pudiendo distinguirse una serie de zonas a tener en cuenta en el tratamiento del riesgo.
- Conociendo el impacto de las amenazas sobre los activos, se puede calcular el riesgo sin más que tener en cuenta la probabilidad de ocurrencia. El riesgo se calcula para cada activo, por cada amenaza y en cada dimensión.
  - El riesgo acumulado se calcula sobre un activo teniendo en cuenta el impacto acumulado sobre un activo debido a una amenaza y la probabilidad de la amenaza. Permite determinar las salvaguardas de que hay que dotar a los medios de trabajo.
  - El riesgo repercutido se calcula sobre un activo teniendo en cuenta el impacto repercutido sobre un activo debido a una amenaza y la probabilidad de la amenaza. Permite determinar las consecuencias de las incidencias técnicas sobre la misión del sistema de información que ayuda a tomar una de las decisiones críticas de un análisis de riesgos: aceptar un cierto nivel de riesgo.

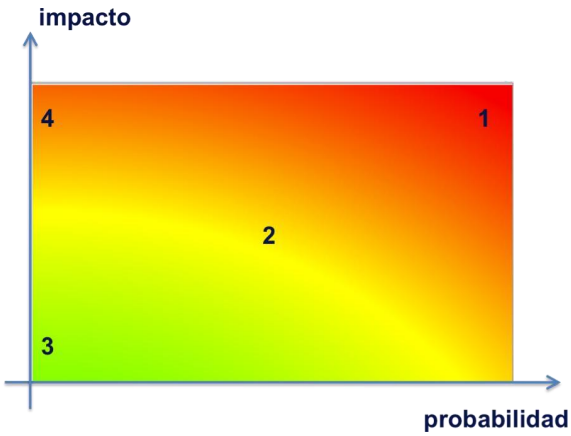
Zonas de riesgo

zona 1: riesgos muy probables y de muy alto impacto

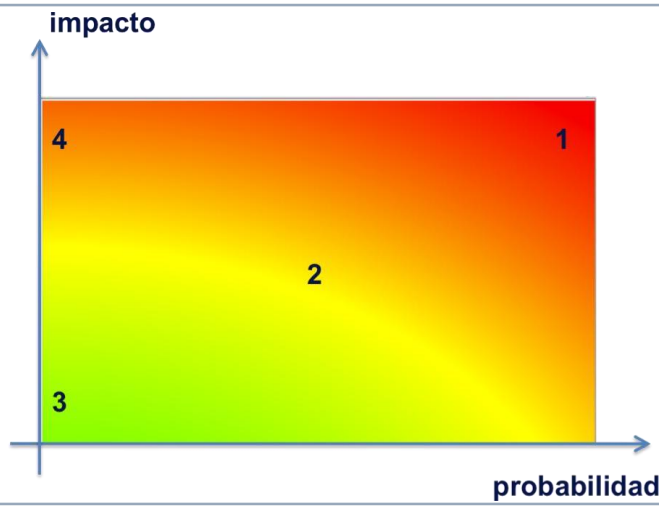
zona 2: cubre un amplio rango desde situaciones improbables y de impacto medio, hasta situaciones muy probables pero de impacto bajo o muy bajo (franja amarilla)

zona 3: riesgos improbables y de bajo impacto

zona 4: riesgos improbables pero de muy alto impacto



Zonas de riesgo



Salvaguardas

- Las salvaguardas o contra medidas son aquellos procedimientos o mecanismos tecnológicos que reducen el riesgo o impacto. Hay amenazas que se conjuran simplemente organizándose adecuadamente, otras requieren elementos técnicos (programas o equipos), otras seguridad física y, por último, está la política de personal.
- Ante el amplio abanico de posibles salvaguardas a considerar, es necesario hacer una criba inicial para quedarnos con aquellas que son relevantes para lo que hay que proteger y teniendo en cuenta:
  - tipo de activos a proteger, pues cada tipo se protege de una forma específica
  - dimensión o dimensiones de seguridad que requieren protección
  - amenazas de las que necesitamos protegernos
  - si existen salvaguardas alternativas
- Además, es prudente establecer un principio de proporcionalidad y tener en cuenta:
  - el mayor o menor valor propio o acumulado sobre un activo, centrándonos en lo más valioso y obviando lo irrelevante
  - la mayor o menor probabilidad de que una amenaza ocurra, centrándonos en los riesgos más importantes (zonas rojas de riesgo)
  - la cobertura del riesgo que proporcionan salvaguardas alternativas
- Esto lleva a dos tipos de declaraciones para excluir una cierta salvaguarda de las que conviene analizar:
  - no aplicable: porque técnicamente no es adecuada al tipo de activos a proteger, no protege la dimensión necesaria o no protege frente a la amenaza en consideración
  - no se justifica: porque, aunque aplicable, es desproporcionada al riesgo que tenemos que proteger
- Como resultado de estas consideraciones dispondremos de una “declaración de aplicabilidad” o relación de salvaguardas que deben ser analizadas como componentes de nuestro sistema de protección.

Tipos de salvaguardas

- Las salvaguardas entran en el cálculo del riesgo de dos formas:
  - Reduciendo la probabilidad de las amenazas:
    - Se llaman salvaguardas preventivas. Las ideales llegan a impedir completamente que la amenaza se materialice.
  - Limitando el daño causado:
    - Hay salvaguardas que directamente limitan la posible degradación, mientras que otras permiten detectar inmediatamente el ataque para frenar que la degradación avance. Incluso algunas salvaguardas se limitan a permitir la pronta recuperación del sistema cuando la amenaza lo destruye. En cualquiera de las versiones, la amenaza se materializa pero las consecuencias se limitan.

Efecto	Tipo
preventivas	[PR] preventivas [DR] disuasorias [EL] eliminatorias
acotan la degradación	[IM] minimizadoras [CR] correctivas [RC] recuperativas
consolidan el efecto de las demás	[MN] monitorización [DC] detección [AW] concienciación [AD] administrativas

Tipos de protección

[PR] **prevención**: cuando reduce las oportunidades de que un incidente ocurra. Si falla, no reducen los daños.

[DR] **disuasión**: cuando tiene un efecto tal sobre los atacantes que estos no se atreven o se lo piensan dos veces antes de atacar. Son salvaguardas que actúan antes del incidente, reduciendo las probabilidades de que ocurra; pero que no tienen influencia sobre los daños causados caso de que el atacante realmente se atreva.

[EL] **eliminación**: cuando impide que éste tenga lugar. Actúan antes de que el incidente se haya producido. No reducen los daños caso de que la salvaguarda no sea perfecta y el incidente llegue a ocurrir.

[IM] **minimización** del impacto / limitación del impacto: cuando acota las consecuencias de un incidente.

[CR] **corrección**: cuando, habiéndose producido un daño, lo repara. Actúan después de que el incidente se haya producido y por tanto reducen los daños.

[RC] **recuperación**: cuando permite regresar al estado anterior al incidente. No reducen las probabilidades del incidente, pero acotan los daños a un periodo de tiempo.

[MN] **monitorización**: cuando supervisan lo que está ocurriendo o lo que ha ocurrido. Si se detectan cosas en tiempo real, podemos reaccionar atajando el incidente para limitar el impacto; si se detectan cosas a posteriori, podemos aprender del incidente y mejorar el sistema de salvaguardas de cara al futuro.

[DC] **detección**: cuando informa de que el ataque está ocurriendo. Aunque no impide el ataque, sí permite que entren en operación otras medidas que atajen la progresión del ataque, minimizando daños.

[AW] **concienciación**: Son las actividades de formación de las personas anexas al sistema que pueden tener una influencia sobre él. La formación reduce los errores de los usuarios, lo cual tiene un efecto preventivo. También mejora las salvaguardas de todo tipo pues los que las operan lo hacen con eficacia y rapidez, potenciando su efecto o, al menos, no menoscabándolo por una mala operación.

[AD] **administración**: Se refiere a las salvaguardas relacionadas con los componentes de seguridad del sistema. Una buena administración evita el desconocimiento de lo que hay y por tanto impide que hayan puertas desconocidas por las que pudiera tener éxito un ataque. En general pueden considerarse medidas de tipo preventivo.

Eficacia de la protección

- Las salvaguardas se caracterizan por su eficacia frente al riesgo que pretenden conjurar. La salvaguarda ideal es 100% eficaz. La eficacia que combina dos factores:
  - desde el punto de vista técnico:
    - es técnicamente idónea para enfrentarse al riesgo que protege
    - se emplea siempre
  - desde el punto de vista de operación de la salvaguarda
    - está perfectamente desplegada, configurada y mantenida
    - existen procedimientos claros de uso normal y en caso de incidencias
    - los usuarios están formados y concienciados
    - existen controles que avisan de posibles fallos
- Entre una eficacia del 0% para aquellas que faltan y el 100% para aquellas que son idóneas y que están perfectamente implantadas, se estimará un grado de eficacia real en cada caso concreto. Se puede emplear una escala de madurez que recoja en forma de factor corrector la confianza que merece el proceso de gestión de la salvaguarda (del 0% al 100%):
  - L0: inexistente, L1: inicial / ad hoc, L2: reproducible, pero intuitivo, L3: proceso definido, L4: gestionado y medible, L5: optimizado

Establecer salvaguardas y valores residuales

- Se denomina vulnerabilidad a toda debilidad que puede ser aprovechada por una amenaza, o más detalladamente a las debilidades de los activos o de sus medidas de protección que facilitan el éxito de una amenaza potencial.
- Son vulnerabilidades todas las ausencias o ineficacias de las salvaguardas pertinentes para preservar el valor propio o acumulado de un activo. A veces se emplea el término “insuficiencia” para resaltar el hecho de que la eficacia medida de la salvaguarda es insuficiente para preservar el valor del activo expuesto a una amenaza.
- El tercer paso consiste establecer las salvaguardas que eliminen o reduzcan la probabilidad de las amenazas y, que en caso se producirse, limiten el daño causado a los activos. La elección de las salvaguardas vienen determinadas por los impactos y riesgos a que estarían expuestos los activos si no se protegieran en absoluto, teniendo en cuenta su aplicabilidad y el principio de proporcionalidad. El resultado de esta actividad se concreta en varios informes:
  - declaración de aplicabilidad
  - evaluación de salvaguardas
  - insuficiencias (o vulnerabilidades del sistema de protección)
- Los valores residuales son el impacto y riesgo que aún soporta el sistema tras el despliegue de las salvaguardas.
- El cuarto paso consiste en calcular el impacto residual, desde un valor potencial a un valor residual, teniendo en cuenta el efecto de las salvaguardas en el nuevo nivel de degradación.
- El quinto paso calcula el riesgo residual, desde un valor potencial a un valor residual, teniendo en cuenta el efecto de las salvaguardas en la probabilidad, probabilidad residual, y el impacto residual.

Método de Análisis de Riesgos (MAR)

- MAR.1 – Caracterización de los activos
  - MAR.11 – Identificación de los activos
  - MAR.12 – Dependencias entre activos
  - MAR.13 – Valoración de los activos
- MAR.2 – Caracterización de las amenazas
  - MAR.21 – Identificación de las amenazas
  - MAR.22 – Valoración de las amenazas
- MAR.3 – Caracterización de las salvaguardas
  - MAR.31 – Identificación de las salvaguardas pertinentes
  - MAR.32 – Valoración de las salvaguardas
- MAR.4 – Estimación del estado de riesgo
  - MAR.41 – Estimación del impacto
  - MAR.42 – Estimación del riesgo

Proceso de gestión de riesgos

- A la vista de los impactos y riesgos a que está expuesto el sistema, hay que tomar una serie de decisiones condicionadas por diversos factores:
  - la gravedad del impacto y/o del riesgo
  - las obligaciones a las que por ley, reglamentos sectoriales y contratos a que esté sometida la Organización
- Dentro del margen de maniobra que permita este marco, pueden aparecer consideraciones adicionales para aceptar ciertos impactos de naturaleza intangible tales como:
  - imagen pública de cara a la Sociedad (aspectos reputacionales)
  - política interna: relaciones con los propios empleados, tales como capacidad de contratar al personal idóneo, capacidad de retener a los mejores, capacidad de soportar rotaciones de personas, capacidad de ofrecer una carrera profesional atractiva, etc.
  - relaciones con los proveedores, tales como capacidad de llegar a acuerdos ventajosos a corto, medio o largo plazo, capacidad de obtener trato prioritario, etc.
  - relaciones con los clientes o usuarios, tales como capacidad de retención, capacidad de incrementar la oferta, capacidad de diferenciarse frente a la competencia, ...
  - relaciones con otras organizaciones, tales como capacidad de alcanzar acuerdos estratégicos, alianzas, etc.
  - nuevas oportunidades de negocio, tales como formas de recuperar la inversión en seguridad
  - acceso a sellos o calificaciones reconocidas de seguridad

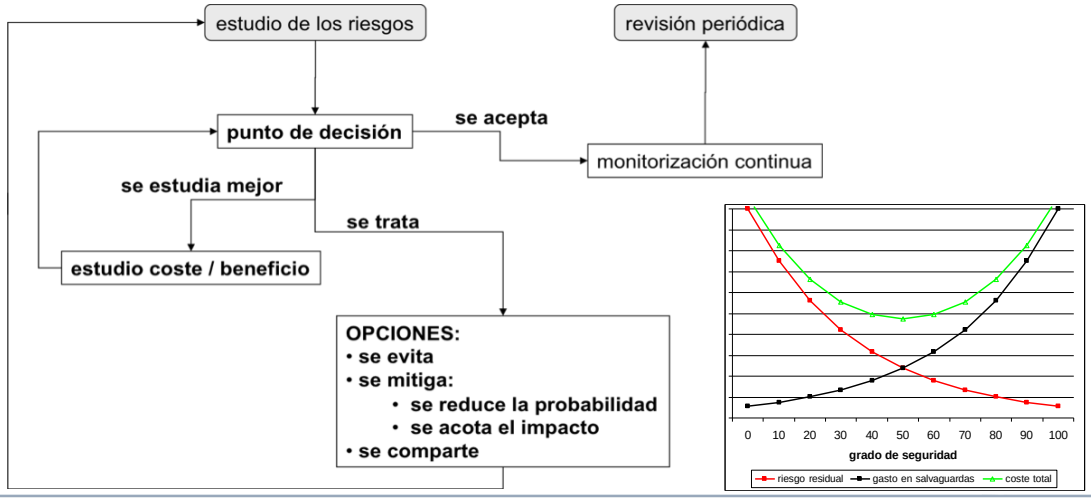
Proceso de gestión de riesgos

- Todas las consideraciones anteriores desembocan en una calificación de cada riesgo significativo, determinándose si ...
  - es crítico en el sentido de que requiere atención urgente
  - es grave en el sentido de que requiere atención
  - es apreciable en el sentido de que pueda ser objeto de estudio para su tratamiento
  - es asumible en el sentido de que no se van a tomar acciones para atajarlo
- La última opción, asumir o aceptar el riesgo, siempre es arriesgada y hay que tomarla con prudencia y justificación. Las razones que pueden llevar a esta aceptación son:
  - cuando el impacto residual es asumible
  - cuando el riesgo residual es asumible
  - cuando el coste de las salvaguardas oportunas es desproporcionado en comparación al impacto y riesgo residuales
- La calificación de los riesgos tendrá consecuencias en las tareas subsiguientes, siendo un factor básico para establecer la prioridad relativa de las diferentes actuaciones.

Proceso de gestión de riesgos

- El análisis de riesgos determina impactos y riesgos. Los impactos recogen daños absolutos, independientemente de que sea más o menos probable que se dé la circunstancia. En cambio, el riesgo pondera la probabilidad de que ocurra. El impacto refleja el daño posible (lo peor que puede ocurrir), mientras que el riesgo refleja el daño probable (lo que probablemente ocurra).
- El resultado del análisis es sólo un análisis. A partir de el disponemos de información para tomar decisiones conociendo lo que queremos proteger (activos valorados), de qué lo queremos proteger (amenazas valoradas) y qué hemos hecho por protegerlo (salvaguardas valoradas). Todo ello sintetizado en los valores de impacto y riesgo.
- A partir de aquí, las decisiones son de los órganos de gobierno de la Organización que actuarán en 2 pasos:
  1. Evaluación
  2. Tratamiento

Proceso de gestión de riesgos



Opciones de tratamiento

- La **eliminación** de la fuente de riesgo es una opción frente a un riesgo que no es aceptable. Esta opción puede tomar diferentes formas:
  - Eliminar cierto tipo de activos, emplean otros en su lugar. Por ejemplo: cambiar de sistema operativo, de fabricante de equipos, …
  - Reordenar la arquitectura del sistema de forma que alteremos el valor acumulado en ciertos activos expuestos a grandes amenazas. Por ejemplo: segregar redes, desdoblar equipos para atender a necesidades concretas, alejando lo más valioso de lo más expuesto, …
- La **mitigación** del riesgo se refiere a una de dos opciones:
  - reducir la degradación causada por una amenaza (a veces se usa la expresión 'acotar el impacto')
  - reducir la probabilidad de que una amenaza de materializa
- Se comparte el riesgo mediante la transferencia del riesgo, que puede ser parcial o total. Hay dos formas básicas de **compartir** riesgo:
  - Riesgo cualitativo: se comparte por medio de la externalización de componentes del sistema, de forma que se reparten responsabilidades técnicas, para el que opera el componente técnico, y legales, según el acuerdo que se establezca de prestación del servicio.
  - Riesgo cuantitativo: se comparte por medio de la contratación de seguros, de forma que a cambio de una prima, el tomador reduce el impacto de las posibles amenazas y el asegurador corre con las consecuencias. Hay multitud de tipos y cláusulas de seguros para concretar el grado de responsabilidad de cada una de las partes.
- Cuando se **acepta** un riesgo, la Organización hará bien en reservar fondos para el caso de que el riesgo se concrete y haya que responder de sus consecuencias. A veces de habla de 'fondos de contingencia' y también puede ser parte de los contratos de aseguramiento.

Marco de seguridad SD^3

Seguro por diseño	• Arquitectura y código seguros • Análisis de amenazas • Reducción de vulnerabilidades
Seguro por defecto	• Se reduce el área de ataques • Características no utilizadas desactivadas de forma predeterminada • Uso de privilegios mínimos
Seguro por despliegue	• Protección: detección, defensa, recuperación, administración • Proceso: guías de procedimientos, guías de arquitecturas • Personas: entrenamiento

DevSecOps

- DevSecOps significa desarrollo, seguridad y operaciones. Se trata de un enfoque que aborda la cultura, la automatización y el diseño de plataformas, e integra la seguridad como una responsabilidad compartida durante todo el ciclo de vida.
- DevOps no solo concierne a los equipos de desarrollo y operaciones. Si desea aprovechar al máximo la agilidad y la capacidad de respuesta de los enfoques de DevOps, la seguridad de la TI también debe desempeñar un papel integrado en el ciclo de vida completo de sus aplicaciones.
- Antes, el papel de la seguridad estaba aislado y a cargo de un equipo específico en la etapa final del desarrollo. Cuando los ciclos de desarrollo duraban meses o incluso años, no pasaba nada. Pero eso quedó en el pasado. Una metodología efectiva de DevOps garantiza ciclos de desarrollo rápidos y frecuentes (a veces de semanas o días), pero las prácticas de seguridad obsoletas pueden revertir incluso las iniciativas de DevOps más eficientes.

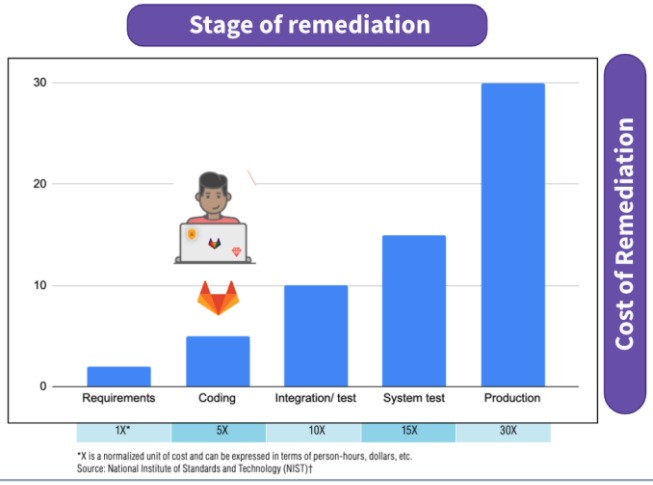
Desafíos

- **Los equipos son reacios a integrarse**: La esencia de DevSecOps es la integración de equipos para que puedan trabajar juntos en lugar de forma independiente. Sin embargo, no todo el mundo está preparado para hacer el cambio porque ya está acostumbrado a los procesos de desarrollo actuales.
- **Guerra de herramientas**: Dado que los equipos han estado trabajando por separado, han estado utilizando diferentes métricas y herramientas. En consecuencia, les resulta difícil ponerse de acuerdo sobre dónde tiene sentido integrar las herramientas y dónde no. No es fácil reunir herramientas de varios departamentos e integrarlas en una plataforma. Por lo tanto, el desafío es seleccionar las herramientas adecuadas e integrarlas adecuadamente para construir, implementar y probar el software de manera continua.
- **Implementar seguridad en IC/DC**: Generalmente, se ha pensado en la seguridad como algo que llega al final del ciclo de desarrollo. Sin embargo, con DevSecOps, la seguridad es parte de la integración y el desarrollo continuos (IC/DC). Para que DevSecOps tenga éxito, los equipos no pueden esperar que los procesos y herramientas de DevOps se adapten a los viejos métodos de seguridad. Al integrar los controles de seguridad en DevOps, las organizaciones están adoptando el nuevo modelo DevSecOps para aprovechar todo el potencial de IC/DC. Cuando las empresas implementan tecnologías de control de acceso o seguridad desde el principio, se aseguran de que esos controles estén en línea con un flujo de IC/DC.

Beneficios

- Desarrollar software seguro desde el diseño.
- Identificación temprana de vulnerabilidades en el código y ataques.
- Aplicar la seguridad de forma más rápida y ágil.
- Responder a los cambios y requisitos rápidamente.
- Mejorar la colaboración y comunicación entre equipos.
- Generar una mayor conciencia sobre seguridad entre todos los miembros.
- Enfocarse en generar el mayor valor de negocio.

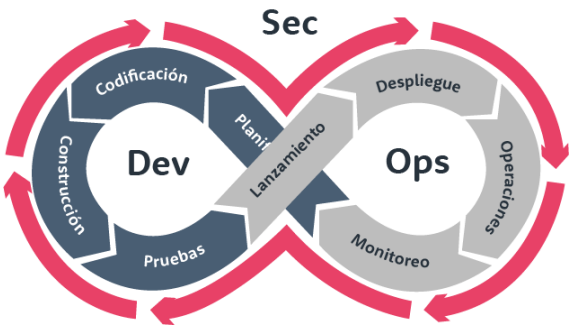
Detección temprana



Implementación

- DevSecOps supone un cambio en la filosofía de trabajo, donde la seguridad se convierte en una responsabilidad compartida (extendida a los equipos de desarrollo y operaciones) e integrada a lo largo de todo el proceso de desarrollo y despliegue. Para poner en práctica DevSecOps, la seguridad debe convertirse en una condición más en el proceso de diseño, desarrollo y entrega de software.
- DevSecOps consta de dos partes diferenciadas pero que forman parte del mismo proceso.
  - Seguridad como código (SaC): cuando se incorporan la seguridad al desarrollo y las operaciones de herramientas informáticas o aplicaciones.
  - Infraestructura como código (IaC): hace referencia al conjunto de herramientas de desarrollo de operaciones (DevOps) utilizadas para configurar y actualizar los componentes de la infraestructura para garantizar un entorno controlado. Trata las infraestructuras como un software.

Ciclo de vida



- Asesorar a los desarrolladores sobre las vulnerabilidades de seguridad para ayudar a configurar las herramientas y las opciones de diseño.
- Trabajar con el equipo de operaciones para llevar a cabo un mantenimiento de seguridad regular y actualizaciones de la infraestructura.
- Implementar pruebas de seguridad automatizadas para la arquitectura, diseño y despliegue en cada proyecto nuevo.
- Usar técnicas de monitorización continua para asegurar que los sistemas de seguridad están trabajando según lo previsto.

Buenas practicas

- **Implemente la automatización para proteger el entorno de IC/DC**: Uno de los aspectos clave del entorno IC/DC es la velocidad. Y eso significa que la automatización es necesaria para integrar la seguridad en este entorno, al igual que incorporar los controles y pruebas de seguridad esenciales a lo largo del ciclo de vida del desarrollo. También es importante implementar pruebas de seguridad automatizadas en las canalizaciones de IC/DC para permitir el análisis de vulnerabilidades en tiempo real.
- **Aborde los problemas de seguridad de la tecnología de código abierto**: Está aumentando el uso de herramientas de código abierto para el desarrollo de aplicaciones. Por lo tanto, las organizaciones deben abordar las preocupaciones de seguridad relacionadas con el uso de dichas tecnologías. Sin embargo, dado que los desarrolladores están demasiado ocupados para revisar el código fuente abierto, es importante implementar procesos automatizados para administrar el código fuente abierto, así como otras herramientas y tecnologías de terceros. Por ejemplo, utilidades como Open Web Application Security Project (OWASP) pueden verificar que no haya vulnerabilidades en el código que dependa de componentes de código abierto.
- **Integre el sistema de seguridad de la aplicación con el sistema de gestión de tareas**: Esto creará automáticamente una lista de tareas con errores que el equipo de seguridad de la información puede ejecutar. Además, proporcionará detalles procesables, incluida la naturaleza del defecto, su gravedad y la mitigación necesaria. Como tal, el equipo de seguridad puede solucionar problemas antes de que terminen en los entornos de desarrollo y producción.

Seguridad del entorno y de los datos

- **Estandarice y automatice el entorno**: los servicios deben tener la menor cantidad de privilegios posible para reducir las conexiones y los accesos no autorizados.
- **Centralice las funciones de control de acceso y de identidad de los usuarios**: el control de acceso estricto y los mecanismos de autenticación centralizados son fundamentales para proteger los microservicios, ya que la autenticación se inicia en varios puntos.
- **Aísle de la red y entre sí aquellos contenedores que ejecutan microservicios**: abarca tanto los datos en tránsito como en reposo, ya que ambos pueden ser objetivos valiosos para los atacantes.
- **Cifre los datos entre las aplicaciones y los servicios**: una plataforma de organización de contenedores con funciones de seguridad integradas disminuye las posibilidades de accesos no autorizados.
- **Incorpore puertas de enlace de API seguras**: las API seguras aumentan el control de los enrutamientos y las autorizaciones. Al disminuir la cantidad de API expuestas, las empresas pueden reducir las superficies de ataque.

Seguridad del proceso de CI/CD

- **Integre análisis de seguridad para los contenedores**: estos deberían ser parte del proceso para agregar contenedores al registro.
- **Automatice las pruebas de seguridad en el proceso de integración continua**: esto incluye ejecutar herramientas de análisis de seguridad estática como parte de las compilaciones, así como examinar las imágenes de contenedores prediseñadas para detectar puntos vulnerables de seguridad conocidos a medida que se incorporan al canal de diseño.
- **Incorpore pruebas automatizadas para las funciones de seguridad al proceso de pruebas de aceptación**: automatice las pruebas de validación de los datos de entrada, así como las funciones de verificación, autenticación y autorización.
- **Automatice las actualizaciones de seguridad, como la aplicación de parches para los puntos vulnerables conocidos**: lleve a cabo este proceso mediante el canal de DevOps. Esto elimina la necesidad de que los administradores inicien sesión en los sistemas de producción mientras crean un registro de cambios documentado y rastreable.
- **Automatice las funciones de gestión de la configuración de los sistemas y los servicios**: de esta manera, podrá cumplir con las políticas de seguridad y eliminar los errores manuales. También se recomienda automatizar los procesos de auditoría y corrección.

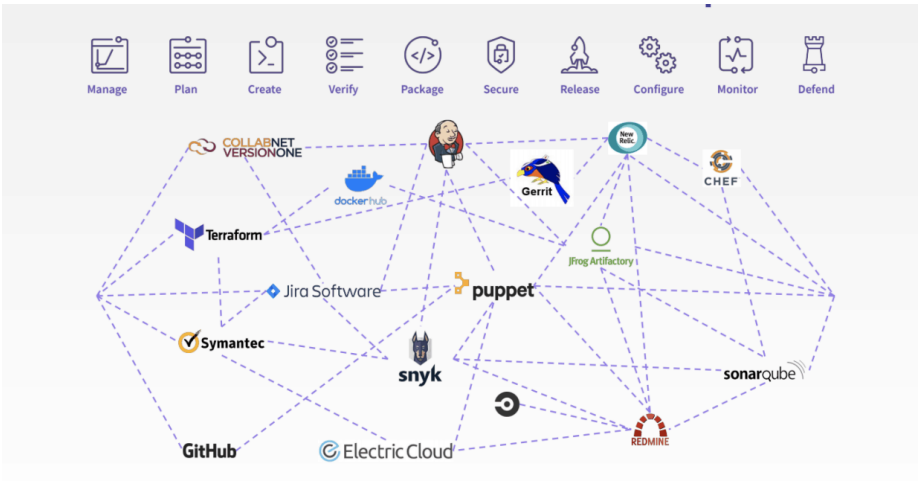
Integración en el ciclo de vida

- Pruebas estáticas de seguridad del software
  - Escanea el código fuente de la aplicación y los archivos binarios para detectar posibles vulnerabilidades.
- Pruebas dinámicas de seguridad del software
  - Escanea la aplicación en tiempo de ejecución para detectar vulnerabilidades como:
    - SQL injection
    - DoS attack
    - XSS
  - Es necesario que la aplicación se este ejecutando en un entorno de laboratorio, donde se realizan las pruebas de afuera hacia adentro sin tener acceso al código fuente. También hacen parte de los tipos de pruebas conocidas como de caja negra.
- Escaneo de dependencias
  - Analiza las dependencias externas del proyecto como npm, ruby gem, composer, entre otras para detectar vulnerabilidades en cada commit de código. Por ejemplo, verificar que no estamos utilizando librerías de código obsoletas o deprecadas.
- Cumplimiento de licencias
  - Busca automáticamente en las dependencias del proyecto las licencias aprobadas o en lista negra de su organización para evitar utilizar software pirata.

¿Es DevSecOps adecuado para tu equipo?

- Los beneficios de DevSecOps son claros: velocidad, eficiencia y colaboración. Pero, ¿cómo saber si es adecuado para tu equipo? Si tu organización se enfrenta alguno de los siguientes desafíos, un enfoque de DevSecOps podría ser una buena medida:
  - Los equipos de desarrollo, seguridad y operaciones están aislados. Si el desarrollo y las operaciones están aislados de los problemas de seguridad, no pueden crear software seguro. Y si los equipos de seguridad no forman parte del proceso de desarrollo, no pueden identificar los riesgos de manera proactiva. DevSecOps reúne a los equipos para mejorar los flujos de trabajo y compartir ideas. Las organizaciones podrían incluso ver una mejora en la moral y la retención de los empleados.
  - Los largos ciclos de desarrollo dificultan satisfacer las demandas de los clientes o de las partes interesadas. Una de las razones de la dificultad podría ser la seguridad. DevSecOps implementa la seguridad en cada paso del ciclo de vida del desarrollo, lo que significa que una seguridad sólida no requiere que todo el proceso se detenga.
  - Se está migrando a la nube (o considerando hacerlo). Mudarse a la nube a menudo significa incorporar nuevos procesos, herramientas y sistemas de desarrollo. Es el momento perfecto para hacer que los procesos sean más rápidos y seguros, y DevSecOps podría hacerlo mucho más fácil.

Herramientas



Pruebas de seguridad

- Para mantener seguro un sistema de TI, se toman una amplia variedad de medidas (salvaguardas). Algunas son medidas de seguridad física, como instalar cerraduras, otras son medidas de TI, como instalar firewalls en una red y otras son medidas de procedimiento, como asignar contraseñas a las personas que necesitan tener acceso.
- Una vez implementado el sistema, las pruebas de seguridad deben demostrar que el sistema efectivamente cumple con las pautas, reglas y regulaciones de seguridad.
- Las pruebas de seguridad son una variedad de pruebas importante que tiene como objetivo garantizar la confidencialidad, integridad y disponibilidad (a menudo denominada CIA) de todas las formas de información y datos.
- Las pruebas de seguridad tienen tres áreas separadas que funcionan juntas y se conocen comúnmente como AAA: control de acceso, autenticación y auditoría.
- Debido a la naturaleza especializada de las pruebas de seguridad, su criticidad y la rápida evolución de los modelos de amenazas, la mayoría de los equipos solo pueden realizar una parte de las pruebas de seguridad y solicitarán la ayuda de una persona o equipo especializado para realizar más pruebas de seguridad.

Diseño de pruebas

- Como cualquier prueba, las pruebas de seguridad deben comenzar temprano con pruebas estáticas, como revisiones de la arquitectura del sistema, centradas en el diseño seguro de la tecnología de la información. Una técnica de prueba estática que se utiliza a menudo para esto es el modelado de amenazas (STRIDE) que utiliza una variedad de categorías de ataques para investigar si la arquitectura es segura para esa categoría específica.
- En las pruebas funcionales, los casos de uso son una técnica de diseño común. En las pruebas de seguridad, las técnicas son casos de abuso y casos de uso indebido. La creación de estos casos de abuso y uso indebido comienza agregando la palabra "no" a los casos de uso funcionales.
- En las pruebas de seguridad existe una distinción entre fallos y defectos. Donde un fallo significa que hay algo mal en el sistema y un defecto significa que hay algo mal en un proceso o procedimiento de seguridad. En tal caso, puede pasar que el sistema soporte perfectamente el procedimiento (por lo que no hay fallo) pero el procedimiento en sí no es seguro. Generalmente, los defectos tienen un impacto mucho mayor y también suelen ser más difíciles de encontrar y corregir que los fallo, las pruebas de seguridad deberían poner más énfasis en la detección de defectos.
- En las pruebas de seguridad, un defecto es una debilidad en un proceso o sistema que lo hace vulnerable a amenazas a la seguridad, comúnmente conocido como vulnerabilidad. Una vulnerabilidad se puede convertir en un exploit, un fragmento de datos o una secuencia de comandos que permite aprovechar (explotar) maliciosamente la vulnerabilidad.

Identificar las amenazas con STRIDE

Tipos de amenazas	Ejemplos
Suplantación	• Falsificar mensajes de correo electrónico • Reproducir paquetes de autenticación
Al**T**eración	• Alterar datos durante la transmisión • Cambiar datos en archivos
Repudio	• Eliminar un archivo esencial y denegar este hecho • Adquirir un producto y negar que se ha adquirido
D**I**vulgación de información	• Exponer información en mensajes de error • Exponer código en sitios Web
Denegación de servicio	• Inundar una red con paquetes de sincronización • Inundar una red con paquetes ICMP falsificados
Elevación de privilegios	• Aprovechar la saturación de un búfer para obtener privilegios en el sistema • Obtener privilegios de administrador de forma ilegítima

Enfoques de las pruebas de seguridad

- El enfoque de las pruebas de seguridad generalmente se centra en: infraestructura, organización y aplicaciones.
- Al probar la seguridad de la infraestructura y las aplicaciones, distinguimos las pruebas de seguridad y el escaneo de seguridad. El escaneo de seguridad es un tipo de prueba estática donde se utilizan herramientas para evaluar la seguridad del sistema de TI. Las pruebas de seguridad son pruebas dinámicas que normalmente las realizan personas manualmente (que tendrán herramientas de soporte). Uno de los tipos de pruebas de seguridad es la prueba de penetración, un ataque simulado autorizado al sistema de TI que se realiza para revelar vulnerabilidades que, normalmente, la realiza una persona o un equipo especializado.
- Para probar una organización, los evaluadores suelen aplicar "ingeniería social" en la que intentan obtener información engañando a los empleados.
- Para probar exhaustivamente la seguridad de una organización, incluidas su gente, su infraestructura y sus aplicaciones, un enfoque relativamente nuevo es el hacking ético o "equipo rojo". El equipo rojo es el equipo atacante y el equipo azul es el equipo defensor. Los hackers de sombrero blanco, también llamados hackers éticos, se caracterizan por trabajar buscando posibles vulnerabilidades, fallas o riesgos dentro de un sistema informático para notificar a sus clientes y mejorar la calidad de su ciberseguridad.

Pruebas estáticas de seguridad de las aplicaciones (SAST)

- Las pruebas estáticas de seguridad de las aplicaciones (SAST) se apoyan en herramientas que escanean el código fuente, binario o de bytes de una aplicación mediante pruebas de caja blanca para identificar la causa raíz de las vulnerabilidades y ayuda a remediar las fallas de seguridad subyacentes. Las soluciones SAST analizan una aplicación desde “adentro hacia afuera” y no utilizan un sistema en ejecución para realizar un análisis.
- Las SAST reducen los riesgos de seguridad en las aplicaciones al proporcionar comentarios inmediatos a los desarrolladores sobre los problemas introducidos en el código durante el desarrollo. Ayuda a educar a los desarrolladores sobre la seguridad mientras trabajan, brindándoles acceso en tiempo real a recomendaciones y navegación por líneas de código, lo que permite un descubrimiento de vulnerabilidades más rápido y una auditoría colaborativa. Esto permite a los desarrolladores crear más código que sea menos vulnerable a verse comprometido, lo que conduce a una aplicación más segura. Sin embargo, las herramientas SAST no son capaces de identificar vulnerabilidades fuera del código.
- Las ventajas de las SAST son:
  - Escanea el código fuente para encontrar debilidades que conduzcan a vulnerabilidades.
  - Proporciona informes en tiempo real
  - Cubrir los lenguajes de programación que utilizan los desarrolladores
- Los inconvenientes de las SAST son:
  - No es capaz de identificar vulnerabilidades en entornos dinámicos.
  - Alto riesgo de informar falsos positivos
  - Dado que el informe es estático, queda obsoleto rápidamente.

Pruebas dinámicas de seguridad de aplicaciones (DAST)

- Las pruebas dinámicas de seguridad de aplicaciones (DAST) es el proceso de analizar una aplicación web a través del frontend para encontrar vulnerabilidades mediante ataques simulados. Este tipo de enfoque evalúa la aplicación desde "fuera hacia dentro" atacando una aplicación como lo haría un usuario malintencionado.
- Después de que un escáner de DAST realice estos ataques, busca resultados que no formen parte del conjunto de resultados esperado e identifica vulnerabilidades de seguridad.
- Las ventajas de las DAST son:
  - Independiente de la aplicación
  - Detecta inmediatamente vulnerabilidades que podrían ser explotadas
  - No requiere acceso al código fuente
- Los inconvenientes de las DAST son:
  - No encuentra la ubicación exacta de una vulnerabilidad en el código
  - Se necesitan conocimientos de seguridad para interpretar los informes
  - Las pruebas pueden llevar mucho tiempo

Análisis de Composición de Software (SCA)

- La cadena es tan fuerte como el eslabón más débil. Las vulnerabilidades de código están aumentando en frecuencia e impacto. Dado que el software se compone cada vez más de piezas de diferentes proveedores, muchos de ellos Open Source, a las que a menudo se hace referencia como la cadena de suministro de software, puede resultar difícil encontrarlas y repararlas rápidamente. Hay que realizar un seguimiento del uso de bibliotecas y marcos, aplicaciones, contenedores, sistemas operativos, firmware, hardware y servicios.
- Las herramientas SCA (Software Composition Analysis) se enfocan en escanear el código fuente y/o el paquete de distribución de la aplicación para identificar las bibliotecas y componentes de terceros que se utilizan. Luego, las herramientas SCA analizan estas bibliotecas y componentes para identificar vulnerabilidades conocidas de seguridad, así como para evaluar la calidad del software y cumplimiento de políticas de seguridad. Algunas de las herramientas SCA más populares incluyen Black Duck, Snyk, Sonatype, WhiteSource, entre otras.
- Es importante tener en cuenta que, aunque las herramientas SCA son muy útiles para identificar vulnerabilidades en bibliotecas y componentes de terceros, no son capaces de identificar todas las vulnerabilidades de seguridad en una aplicación. Por lo tanto, es recomendable complementar las pruebas de seguridad con herramientas DAST y SAST y pruebas manuales de seguridad.
- El escaneo debe realizarse periódicamente para detectar las nuevas vulnerabilidades aparecidas.