SEG 118 – Desarrollo Seguro basado en OWASP

Ing. Carlos Carreño

Nov, 2021

MODULO 1: INTRODUCCIÓN A LA SEGURIDAD EN EL DESARROLLO DE SOFTWARE

- Casos reales de vulnerabilidades y su impacto.
- Problemática de las aplicaciones inseguras.
- Derribando mitos.
- Participación de Seguridad Informática en el desarrollo del software

Casos reales de vulnerabilidades y su impacto

- WikiLeaks, noviembre del 2010
- Sony PlayStation Network , Abril 2011
- Dropbox, Agosto 2012
- Target, Diciembre 2013
- eBay, Mayo 2014
- Elecciones USA, Diciembre 2015
- Friend Finder, Noviembre 2016
- Uber , Noviembre 2017
- Cambridge Analytica, Marzo 2018
- Facebook, Marzo 2019

Casos reales de vulnerabilidades y su impacto 2020

- SolarWinds, FireEyey otros
- Laboratorios de Vacunas contra el COVID 2019
- Ransomware
- Phishing
- Twitter
- Filtración del Código Fuente de Windows XP

2021 Que otros ataque podrían ocurrir

Problemática de las aplicaciones inseguras

- 90% de la aplicaciones web son inseguras [4]
  - Cross-site 80%
  - Inyección SQL 62%
  - Falsificación de parámetros 60%
- 75% de los ataques son a través de las aplicaciones web en internet

Derribando mitos

- El uso de las defensas habituales como firewalls, detectores de intrusos, etc. en la mayoría de los casos se muestran ineficientes.
- Los atacantes podrán acceder a datos de usuarios, de la empresa, detener sitios web, modificar la información del sitio web, e incluso llegar a ejecutar comandos en el servidor sin ser detectado en ningún momento. [4]

Se pueden detener los ataques?

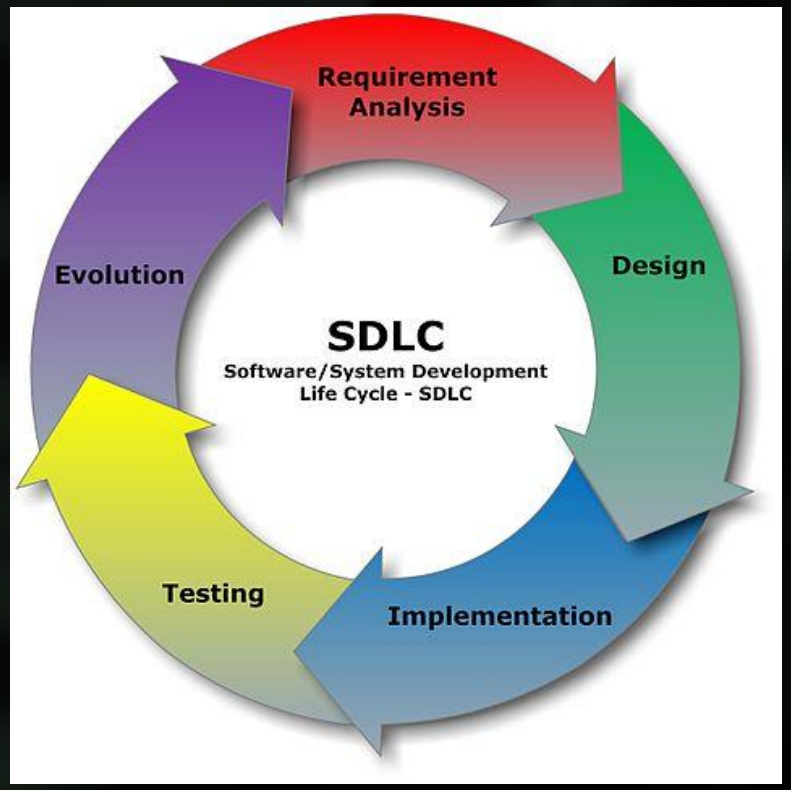
- https://www.youtube.com/watch?v=hSQf3hUx9J0

Tipos de Ataques

- https://www.youtube.com/watch?v=Dk-ZqQ-bfy4

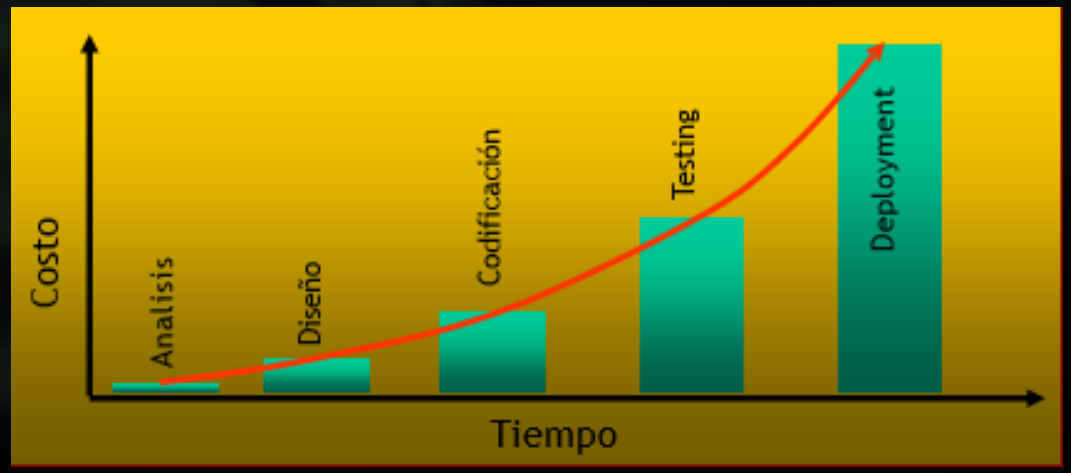
Participación de Seguridad Informática en el desarrollo del software

- Que es el SDLC?



Introducción a la seguridad en el SDLC

- Actualmente la mayoría de los procesos de desarrollo no incluyen seguridad, o bien la incluyen al final (etapa de testing)
- El costo de solucionar las vulnerabilidades es mayor cuanto más tarde se detectan las mismas (igual que los bugs)



La Seguridad y el SDLC

- La seguridad de la aplicación es responsabilidad del programador.
- Nadie sabe cómo funciona, por ende, no la van a atacar.
- Si no se encontraron vulnerabilidades hasta ahora…
- A nadie le interesaría atacar nuestra aplicación.
- La aplicación es segura porque corre detrás de un firewall.
- La aplicación es segura porque usa encripción.
- Si no corre como Administrator / root, no funciona.
- Si, ese feature (que es inseguro) viene habilitado por default, pero el administrador lo puede deshabilitar.
- No hay manera de explotarla.
- No hay tiempo para incluir seguridad.

Tendencia Actual

- Participación de Seguridad Informática desde el comienzo de todos los proyectos de desarrollo.
- Incorporar seguridad a lo largo de todas las etapas del ciclo de vida del desarrollo de software (SDLC).
  - *Análisis*
  - *Diseño*
  - *Codificación*
  - *Testing*
  - *Deployment*

Seguridad en el análisis de requerimientos

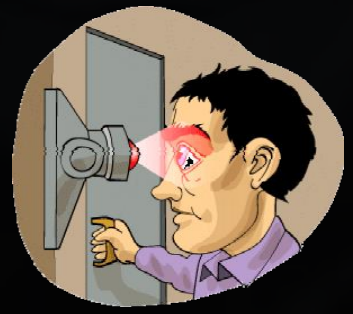
• Durante el análisis de requerimientos, se pueden identificar diversas características que derivarán en los requerimientos de seguridad del software. Por ejemplo:

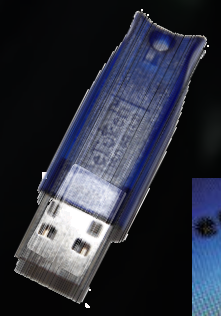
- Arquitectura de la aplicación
  - *¿Cliente/servidor o Desktop?*
- Plataforma donde correrá la aplicación
  - *PC /Palm /Teléfono celular*
- Tipos de datos que se almacenan o transfieren
  - *Confidenciales / públicos*
- Requerimiento de compliance con normativas y marcos regulatorios
  - *SOX, PCI-DSS, "A" 4609*

Security Standards Council

Seguridad en el análisis de requerimientos

- Tipos de registro que el sistema debe generar
  - *Acceso a recursos, uso de privilegios, etc.*
- Perfiles de usuario necesarios para la aplicación
  - *Administrador, revisor, editor, usuario básico, etc.*
- Tipos de acceso a los datos por parte de cada perfil
  - *Lectura, escritura, modificación, agregado, borrado, etc.*
- Acciones sobre el sistema que puede hacer cada perfil
  - *Cambiar la configuración del sistema*
  - *Arrancar o detener servicios*
- Modos de autenticación
  - *Passwords, Tokens, Biométricos*
  - *1 factor, 2 factores, etc*





Seguridad en el Diseño

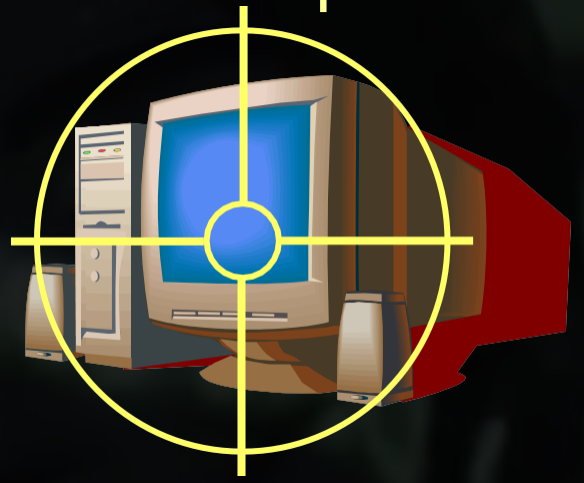
- Muchas de las vulnerabilidades encontradas en aplicaciones tuvieron su causa en errores de diseño. Ej:

- Aplicación Home banking (WEB)
  - Incluía el número de cuenta en el request de transferencias
  - No validaba que la cuenta origen perteneciera al usuario logueado
  - Vulnerabilidad: Transferencias desde cuentas ajenas

- Aplicación de Adm y Finanzas (Consola Unix)
  - Tomaba el usuario de una variable de entorno
  - Permitía que el usuario "escapara" a un shell
  - Vulnerabilidad: Se puede establecer un usuario arbitrario

Seguridad en el Diseño

- Buenas prácticas para el diseño de una aplicación segura
  - Reducción de Superficie de ataque
  - Criterio del menor privilegio
  - Fallar de manera segura
  - Criterio de defensa en profundidad
  - Diseño seguro de mensajes de error
  - Diseño seguro de autenticación
  - Separación de privilegios
  - Interacción “amigable” con Firewalls e IDS's.
  - Administración segura información Sensible
  - Diseño de Auditoría y Logging
  - Análisis de riesgo



Seguridad en el Diseño

- Análisis de riesgo – Threat Modeling

- Técnica formal, estructurada y repetible que permite determinar y ponderar los riesgos y amenazas a los que estará expuesta nuestra aplicación.

- Consta de las siguientes etapas:

1) Conformar un grupo de análisis de riesgos
2) Descomponer la aplicación e identificar componentes clave.
3) Determinar las amenazas a cada componente de la aplicación.
4) Asignar un valor a cada amenaza.
5) Decidir cómo responder a las amenazas.
6) Identificar las técnicas y tecnologías necesarias para mitigar los riesgos identificados.

Seguridad en el Diseño

- Método STRIDE

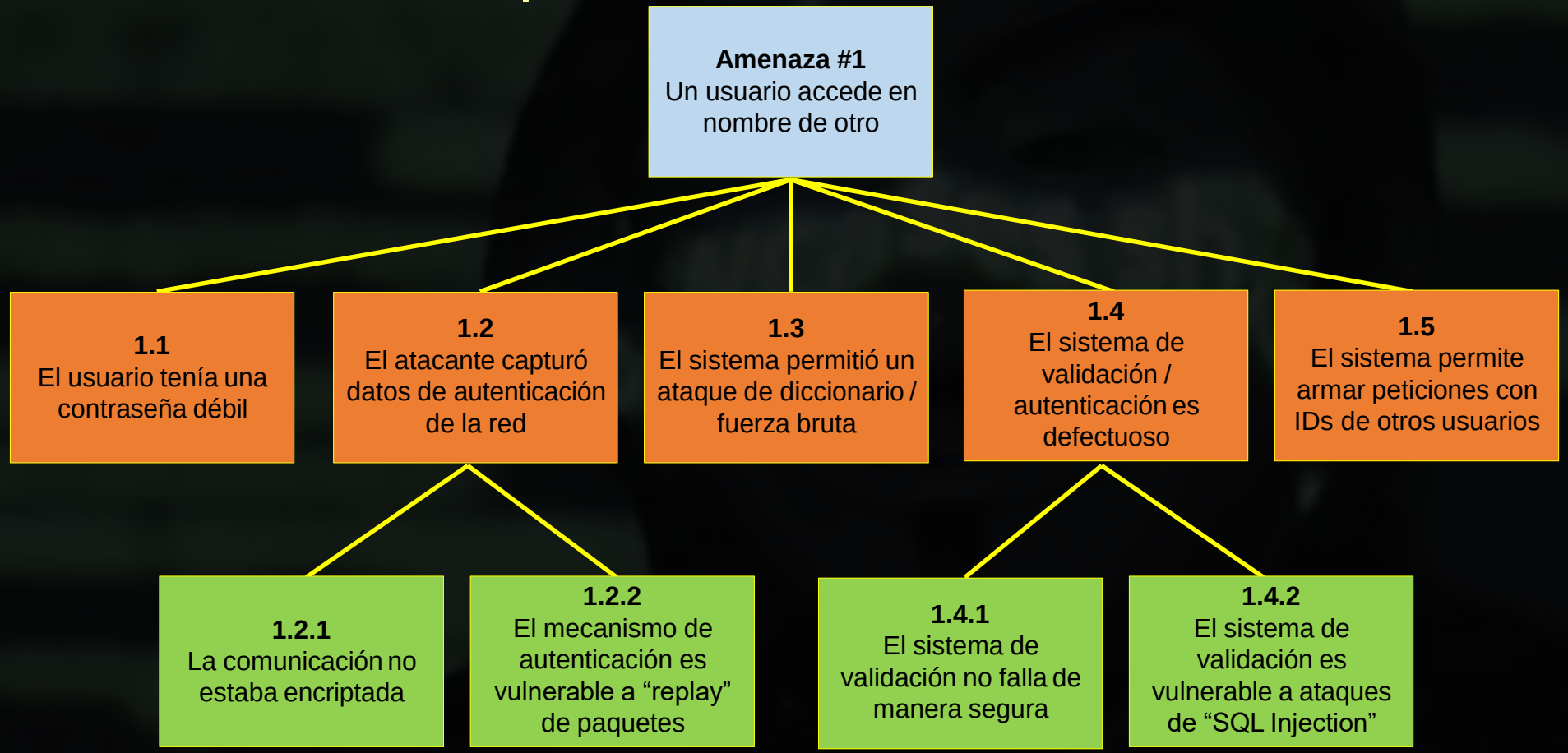
- Ayuda a identificar amenazas en los componentes de un sistema
- Su nombre es un acrónimo de:

•Spoofing Identity: Suplantar la identidad de otro usuario o servicio. Tampering with Data: Modificar maliciosamente datos almacenados. Repudiation: Imposibilidad de identificar el autor de una acción.

- Information Disclosure: Divulgar información a usuarios no autorizados. Denial of Service: Provocar que un servicio deje de funcionar.
- Elevation of privilege: Conseguir privilegios mayores a los asignados

Seguridad en el Diseño

- Árboles de ataque



Seguridad en el Diseño

- Método DREAD
  - Ayuda a ponderar las amenazas identificadas.
  - Es un acrónimo de los siguientes 5 ítems:

- Damage Potencial: ¿Cuán importante es el daño de esta amenaza?
- Reproducibility: ¿Cuán reproducible es la vulnerabilidad?
- Exploitability: ¿Cuán fácil es de explotar?
- Affected Users: ¿Cuáles y cuántos usuarios se verían afectados?
- Discoverability: ¿Cuán fácil de descubrir es la vulnerabilidad?

Seguridad en la codificación

- La falta de controles adecuados en la codificación, muchas veces deriva en vulnerabilidades que pueden comprometer a la aplicación o a los datos de la misma
- Los tipos de vulnerabilidades más habituales son:

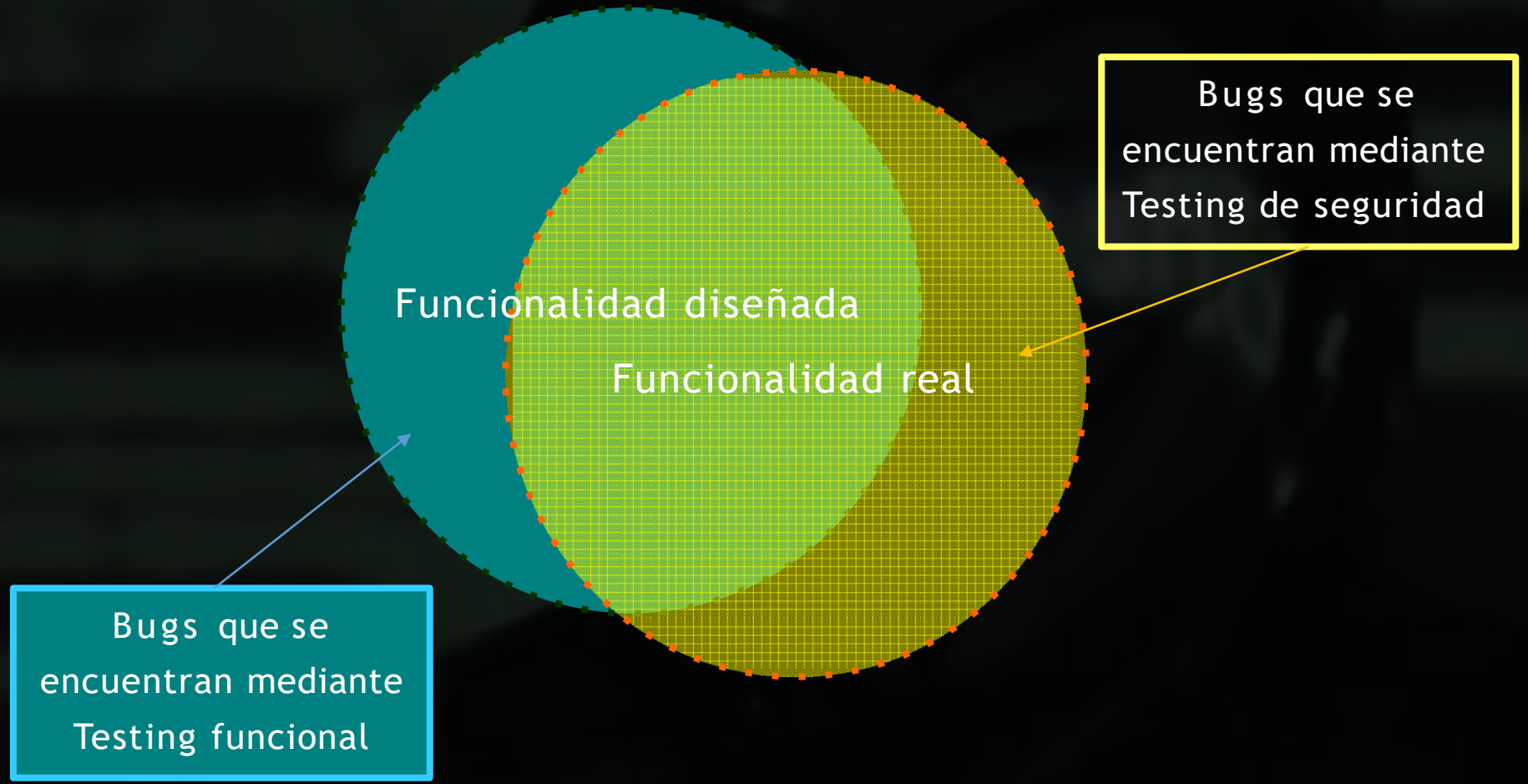
- *Stack buffer overflows*
- *Heap buffer overflows*
- *SQL Injections*
- *Cross Site Scripting (XSS)*
- *Directory Traversal*
- *Authentication Bypass*
- *Information Disclosure*
- *Escalamiento de privilegios*
- *Manejo inseguro de sesiones*
- *Denegación de servicio*

Buenas prácticas para una codificación segura

- Validar siempre los datos de entrada antes de procesarlos
- Nunca confiar en que los datos recibidos sean correctos.
- Realizar validación de datos en todas las capas
- Usar siempre criterio de “White List” en las validaciones
- Controlar tamaño y tipo de datos
- “Sanitizar” los valores de entrada y salida
- Eliminar o “escapear” caracteres especiales
- Transformar los datos de entrada a un “encoding” establecido
- Reemplazar sentencias SQL dinámicas por Stored Procedures
- Evitar generar código con valores ingresados por el usuario
- No mezclar datos con código
- Capturar errores de capas inferiores y no mostrarlos al usuario

Testing de seguridad

- Testing funcional Vs. Testing de seguridad



Testing de seguridad

- Técnicas de testing de seguridad
  - Testing Funcional (clásico) aplicado a las funcionalidades de seguridad de una aplicación. Ej:

- *Req. de autenticación*
- *Req. de complejidad de contraseñas*
- *Bloqueo automático de cuentas*
- *Funcionalidad de captchas*
- *Restricciones de acceso según diseño*
- *Mecanismos de registro y logging*
- *Mensajes de error especificados*

Testing de seguridad basado en Riesgo

- Técnica que se desprende del Threat Modelling
- Se identifican todas las interfaces de la aplicación
- Se generan casos de test sobre cada interfaz basado en STRIDE

Testing de seguridad

Técnicas de testing de seguridad

Testing con un cliente /server falso

Consiste en diseñar un cliente o server "Ad Hoc" que permita:

- *Enviar peticiones /respuestas incorrectas o inválidas.*
- *Enviar peticiones/ respuestas fuera de orden.*
- *Insertar delays arbitrarios.*
- *Comportarse de manera diferente al cliente o servidor real.*

Test de stress

Consiste en llevar la carga o funcionalidad de la aplicación al límite

- *Generar una carga alta de peticiones/transacciones a la aplicación.*
- *Mantener esta carga durante tiempos prolongados.*
- *Simular tráfico en ráfagas.*

Testing de seguridad

Técnicas de testing de seguridad

Test de mutación de datos

Se testea la aplicación ingresando en sus interfaces datos "mutados"

- *Diferente signo*
- *Diferente tipo*
- *Diferente longitud*
- *Fuera de rango*
- *Caracteres especiales*
- *Código (ej: javascripts)*
- *Valores nulos*
- *Valores aleatorios*

Revisión de código

Permite encontrar vulnerabilidades que son muy difíciles de

con otros métodos de testing de seguridad (ej: BackDoors)

- *Enfoque tradicional: Grupo de revisión*
- *Enfoque rápido: Revision por pares*

Seguridad en la Implementación

Seguridad en la implementación (deployment)

Si no se implementa la aplicación de forma segura, se pueden echar por tierra los esfuerzos de las etapas anteriores.

- Hardening de software de base
  - *Servicios innecesarios*
  - *Usuarios y contraseñas default*
  - *Configuración de intérpretes*
- Proceso de implementación
  - *Separación de ambientes*
- Administración de implementación y mantenimiento
  - *Releases y Patches*
  - *Firma de código*

Recomendaciones

- Integrando seguridad a lo largo de todas las etapas del SLDC se ahorra tiempo y dinero.
- La tendencia actual es que SI participe desde el principio de los proyectos de desarrollo.
- La mayoría de las vulnerabilidades no se deben a errores de codificación, sino a defectos de diseño.
- Existen buenas prácticas y técnicas específicas para insertar seguridad en cada etapa del SDLC.

Laboratorio

- Preparación del entorno de practicas de tipos de ataques segun OWASP.
  - Oracle VirtualBox
  - Centos 7
  - Java 15

Referencias

[1] https://www.computing.es/seguridad/noticias/1116703002501/10-ciberataques-mas-grandes-de-decada.1.html

[2] https://www.muycomputer.com/2020/12/30/ciberseguridad-en-2020/

[3] https://sofecom.com/ataques-ciberneticos-empresas/

[4] https://unaaldia.hispasec.com/2004/02/un-90-de-las-aplicaciones-web-son-inseguras.html