8636 Criptografía y Seguridad Informática

Modelado de Amenazas

Manifesto

THREAT MODELING MANIFESTO

What is threat modeling?

Threat modeling is analyzing representations of a system to highlight concerns about security and privacy characteristics.

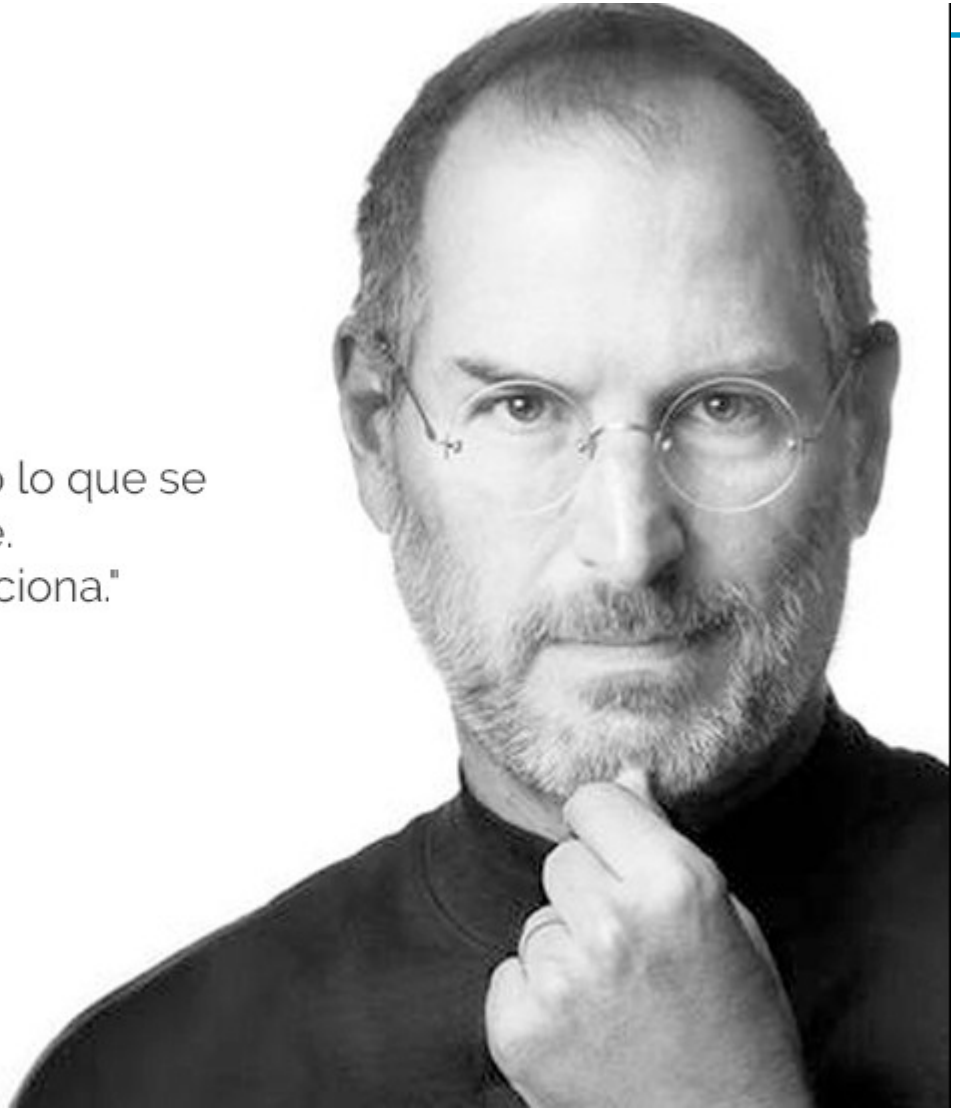
At the highest levels, when we threat model, we ask four key questions:

1. What are we working on?
2. What can go wrong?
3. What are we going to do about it?
4. Did we do a good enough job?

https://www.threatmodelingmanifesto.org/

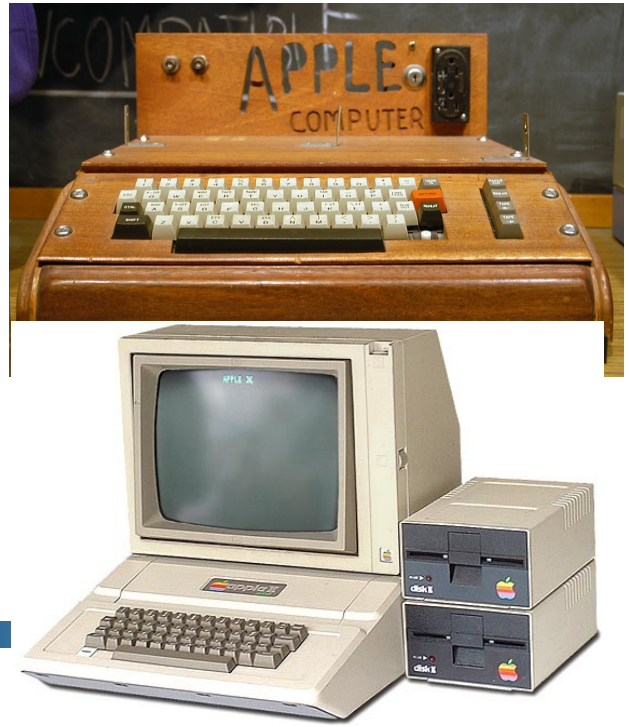
"El diseño no es solo lo que se ve o lo que se siente.
Diseño es cómo funciona."

- Steve Jobs



Steve Wozniak

aprendí a no preocuparme tanto por el resultado, sino a concentrarme en el paso en el que estaba y a intentar hacerlo de la manera más perfecta posible cuando lo estaba haciendo.

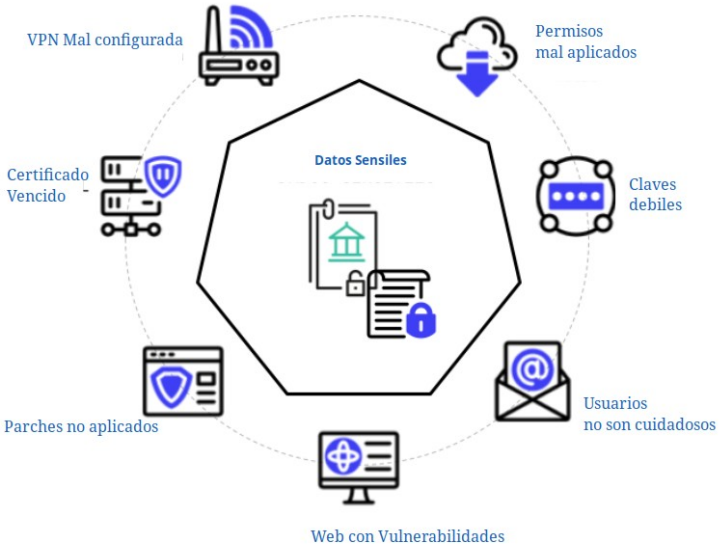


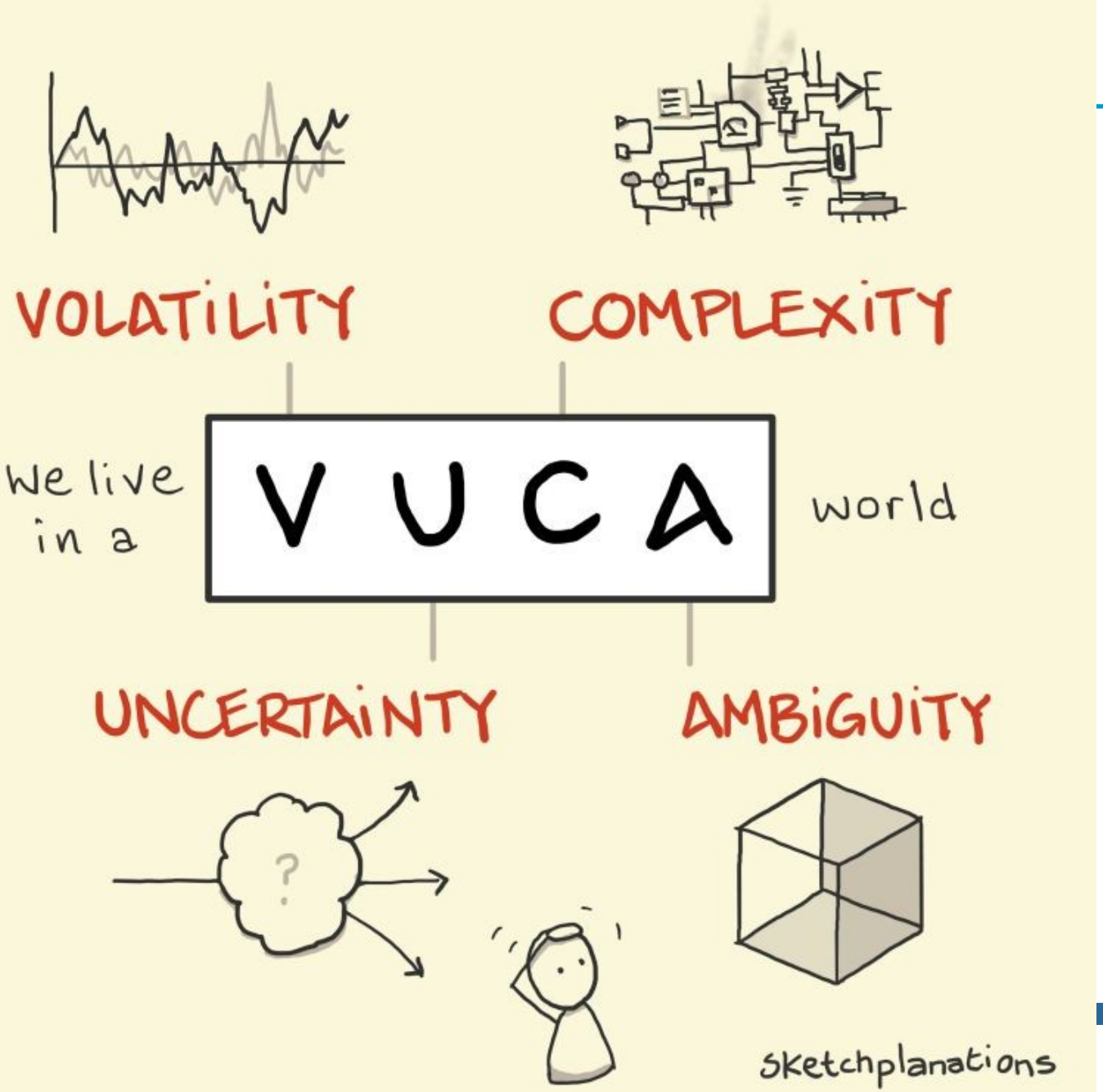
Superficie de ataque

suma de vulnerabilidades, rutas o métodos, vectores de ataque,

que se pueden usar

para obtener acceso no autorizado a la red o a datos confidenciales.




VOLATILITY
COMPLEXITY
We live in a
V U C A
world
UNCERTAINTY
AMBIGUITY
?
sketchplanations

¿Qué es el Entorno VUCA?

El Entorno en el que se desenvuelven las Organizaciones hoy en día se caracteriza por

muchos cambios, los cuáles se suceden a diario producidos por el *avance tecnológico*,

cambios de hábitos,

crisis ……etc.

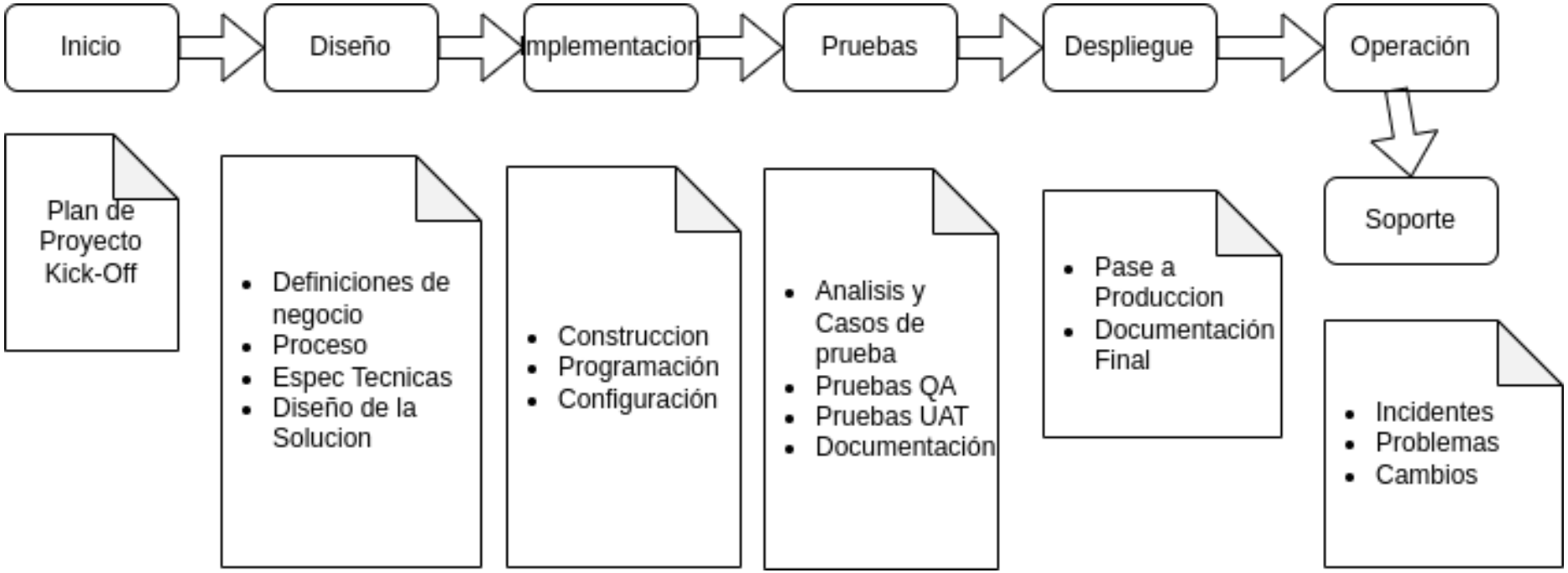
Se producen de una forma volátil, lo que genera mucha **incertidumbre** y falta de reacción, ante tantos cambios de manera **inesperada**.

Volatilidad → visión **Estratégica.** nos ayudará adaptarnos al entorno, tener visión a futuro. Minimizar sus efectos.

Incertidumbre → formación, conocimiento, mejora continua. ***Estar actualizados***

Complejidad → claridad, sencillez y "**hacerlo fácil**" en todos los procesos y tareas

Ambigüedad → Agilidad… Ser **agiles** para hacer frente a imprevistos


Inicio
Diseño
Implementacion
Pruebas
Despliegue
Operación
Soporte
Plan de Proyecto Kick-Off
Definiciones de negocio
Proceso
Espec Tecnicas
Diseño de la Solucion
Construccion
Programación
Configuración
Analisis y Casos de prueba
Pruebas QA
Pruebas UAT
Documentación
Pase a Produccion
Documentación Final
Incidentes
Problemas
Cambios

Fallos de seguridad

√ En el diseño

- ⇒ Ubicar los controles de seguridad y/o las variables de una aplicación web únicamente en el lado del cliente (navegador), lo cual puede ser manipulado fácilmente por un usuario malintencionado.
- ⇒ Omitir definir quiénes NO deben tener acceso a cierta información, lo cual puede llevar a brechas de seguridad al no restringir adecuadamente el acceso.

√ En la implementación

- ⇒ Errores al programar
- ⇒ Construir consultas SQL directamente con entradas de texto del usuario sin realizar validaciones previas, lo que puede exponer la aplicación a ataques de inyección SQL.
- ⇒ Manejar excepciones simplemente capturándolas e ignorando los errores subyacentes, lo que puede ocultar problemas críticos y afectar la estabilidad de la aplicación.

Fallos de Seguridad Comunes Diseño

En el diseño

Ubicar los controles de seguridad y/o las variables de una aplicación web únicamente en el lado del cliente (navegador), lo cual puede ser manipulado fácilmente por un usuario malintencionado.

Omitir definir quiénes NO deben tener acceso a cierta información, lo cual puede llevar a brechas de seguridad al no restringir adecuadamente el acceso.

Fallos de Seguridad Comunes Despliegue

En la implementación

Errores al programar

No validar entradas: Construir consultas SQL directamente con entradas de texto del usuario sin realizar validaciones previas, lo que puede exponer la aplicación a ataques de inyección SQL.

Incorrecto manejo excepción: Manejar excepciones simplemente capturándolas e ignorando los errores subyacentes, lo que puede ocultar problemas críticos y afectar la estabilidad de la aplicación.

Fallos de Seguridad Comunes Despliegue

En el despliegue:

Configuración insegura de servidores y servicios: Desplegar aplicaciones con configuraciones por defecto o inseguras, lo que puede dejar expuestos servicios críticos a ataques externos.

Falta de protocolos seguros: No utilizar protocolos de comunicación seguros como HTTPS, lo que expone los datos transmitidos a interceptaciones y modificaciones.

Fallos de Seguridad Comunes Operación

En la operación:

Monitoreo insuficiente: No implementar sistemas de monitoreo efectivos que permitan detectar y responder a actividades sospechosas o ataques en tiempo real.

Actualizaciones y parches desatendidos: No mantener actualizados los sistemas y aplicaciones, lo que puede dejar vulnerabilidades sin corregir y explotables por atacantes.

Fallos de Seguridad Comunes Soporte

En el soporte:

Gestión deficiente ante incidencias: No tener un proceso efectivo para manejar incidentes de seguridad, lo que puede resultar en respuestas lentas o inadecuadas ante brechas o fallos de seguridad.

Capacitación insuficiente del personal de soporte: No capacitar adecuadamente al personal de soporte en prácticas de seguridad, lo que puede llevar a errores en la gestión de configuraciones y datos sensibles.

Fallos de Seguridad Comunes Desactivar

Durante la Desactivación de Sistemas o Software

Eliminación incompleta de datos: No asegurarse de que todos los datos sensibles almacenados por el sistema sean completamente eliminados o encriptados. Esto puede dejar información confidencial expuesta a posibles filtraciones.

Mantenimiento de accesos residuales: Dejar accesos activos o cuentas de usuario sin desactivar completamente, lo que puede permitir entradas no autorizadas a sistemas aún en línea o a otros sistemas interconectados.

Desatención de copias de seguridad: No gestionar adecuadamente las copias de seguridad que contienen datos del sistema desactivado, pudiendo ser una fuente de riesgos si estas copias caen en manos equivocadas.

Fallos de Seguridad Comunes Desactivar

Durante la Desactivación de Sistemas o Software

Falta de documentación de la desactivación: No mantener un registro detallado de las acciones de desactivación, incluyendo qué, cómo y cuándo se desactivó. La falta de esta documentación puede llevar a confusiones o a la reactivación inadvertida de componentes del sistema que se suponían desactivados.

No revisar dependencias: Ignorar cómo la desactivación de un sistema podría afectar a otros sistemas interdependientes. Esto puede causar interrupciones inesperadas o fallos en otros servicios que dependen del sistema desactivado.

Seguridad perimetral olvidada: No asegurar adecuadamente el entorno físico o virtual donde residía el sistema, dejando vulnerabilidades que podrían ser explotadas aun después de que el sistema está oficialmente "fuera de servicio".

IEEE CENTER FOR SECURE DESIGN

https://ieeecs-media.computer.org/media/technical-activities/CYBSI/docs/Top-10-Flaws.pdf

1 EARN OR GIVE, BUT NEVER ASSUME, TRUST

La seguridad de sistemas de software depende de no confiar ciegamente en componentes no verificables en entornos hostiles.

- √ La cooperación de componentes esencial.
- √ Riesgo al ejecutar partes en entornos hostiles.
- √ Advertencia sobre confiar en componentes no verificables.
- √ Necesidad de precaución al offload de operaciones de seguridad.
- √ Validación rigurosa de datos no confiables al diseñar sistemas.

2 Mecanismo de Autenticación que no puedan ser eludidos o manipulados

La autenticación valida la identidad. Evitar suplantaciones, usar múltiples factores y almacenar credenciales seguras.

- √ Esencial para prevenir el acceso no autorizado.
- √ Múltiples factores de autenticación refuerzan la seguridad.
- √ Riesgo de suplantación si las credenciales son fáciles de falsificar.
- √ Debe incluir mecanismos de seguridad para evitar el acceso no autorizado.
- √ Debe tener límites de tiempo y ser segura contra suplantación.

3 Autorizar luego de autenticar

La identidad del usuario es esencial, pero no suficiente; se requiere la autorización adecuada para ciertas acciones.

- √ La autorización va más allá de la autenticación.
- √ Contexto y privilegios afectan la autorización.
- √ La revocación de autorizaciones es esencial.
- √ Operaciones sensibles pueden necesitar re-autenticación.
- √ La autorización debe ser manejada por una infraestructura común y reutilizable.

4 Separar estrictamente los datos y las instrucciones de control,

y nunca procesar instrucciones de control recibidas de fuentes no confiables.

- √ La combinación de datos y control en una entidad puede conducir a vulnerabilidades de inyección.
- √ La falta de separación estricta permite que datos no confiables controlen el flujo de ejecución.
- √ Puede resultar en vulnerabilidades de corrupción de memoria.
- √ Y puede dar lugar a vulnerabilidades de inyección.

5 Validar todos los datos

Es esencial validar explícitamente los datos en el software para evitar vulnerabilidades basadas en suposiciones implícitas sobre los datos..

√ Al diseñar sistemas de software una validación exhaustiva de datos y verificar las suposiciones es crucial.

- Bibliotecas de validación comunes
- Validación basada en el tipo de datos
- Debe ser coherente y considerar la dependencia del estado.

6 Usar bien la criptografía

La criptografía es crucial para la seguridad, pero es compleja. Evitar errores comunes.

- √ No diseñar algoritmos criptográficos propios; usar estándares y bibliotecas confiables.
- √ Entender y usar las bibliotecas correctamente.
- √ Gestionar adecuadamente las claves criptográficas.
- √ Utilizar números aleatorios criptográficamente seguros y no reutilizarlos.
- √ Prever adaptabilidad y evolución de algoritmos.

7 Identificar datos sensibles y cómo deben ser manejados.

Identificar datos sensibles y cómo protegerlos es crucial.

- √ Identificar datos sensibles y sus posibles manipulaciones o exposiciones.
- √ Regulaciones, políticas corporativas, contratos y expectativas de usuarios.
- √ Necesidades de confidencialidad, disponibilidad y integridad.
- √ Usar controles técnicos como mecanismos de acceso, cifrado y respaldos para proteger datos.
- √ La protección de datos se aplica no solo en reposo, sino también en tránsito y en entornos interconectados.

8 Tener en cuenta los usuarios

La seguridad de un sistema de software está vinculada a sus usuarios.

- √ Las características de usabilidad y experiencia del usuario son esenciales para operar de manera segura.
- √ Diseñar sistemas que faciliten la configuración y el uso seguros es un desafío importante.
- √ La falta de consideración de los usuarios puede llevar a problemas de seguridad y privacidad.
- √ Se deben considerar:
  - Las necesidades de diferentes clases de usuarios
  - Evitar la fatiga del usuario al equilibrar seguridad y usabilidad..

9 Integrar componentes externos cambia la superficie de ataque

La incorporación de componentes externos es algo común. Se debe gestionar su seguridad y efectos adecuadamente.

√ Los sistemas de software a menudo utilizan componentes externos, como bibliotecas, plataformas y aplicaciones.

- Garantizar que los componentes externos cumplan con los estándares de seguridad actuales.
- Su inclusión de componentes externos afecta el modelo de amenazas del sistema y su superficie de ataque.
- Se deben configurar los componentes externos solo para la funcionalidad requerida y validar su origen e integridad.
- Se debe verificar regularmente si necesitan actualizaciones.

10 Flexibilidad a cambios

diseñar sistemas que puedan adaptarse a cambios en el entorno, actualizaciones seguras y evolución de componentes externos, además de

√ El diseño debe ser

- adaptable a cambios en el entorno y la cantidad de usuarios.
- Actualizaciones seguras, evitando cambios masivos.
- Considerar la verificación de integridad y origen de las actualizaciones.
- Diseñar para cambios en propiedades de seguridad, como contraseñas y claves.
- Prever cambios en propiedades de componentes externos y diseñar con agilidad para adaptarse
- Garantizar la gestión adecuada de secretos y control de acceso..

IEEE CENTER FOR SECURE DESIGN

https://ieeecs-media.computer.org/media/technical-activities/CYBSI/docs/Top-10-Flaws.pdf

Seguridad desde el principio y todo el tiempo

Seguridad desde el principio y todo el tiempo

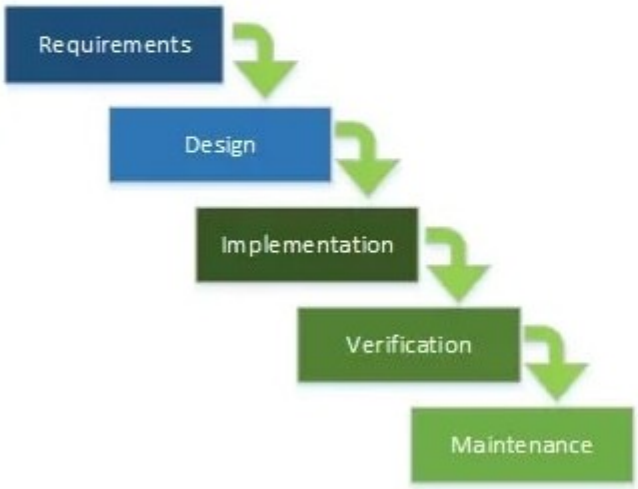
.

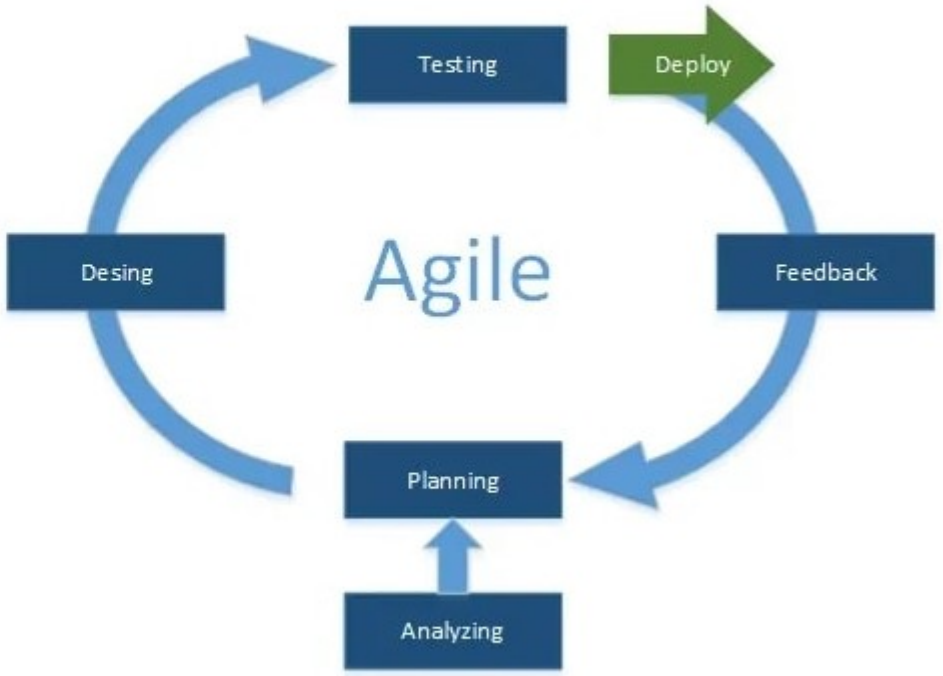
No hace falta un modelo de amenazas para:

- Definir el uso de TLS
- Verificar las entradas de una interfaz
- Definir roles desde el principio
- Ver los flujos alternativos.
- Pensar que debe suceder cuando las cosas no funcionan y hay un error.
- Estar atento a vulnerabilidades y amenazas
- validar supuestos que tenga el caso y hacer cumplir o forzar las condiciones que se impongan

https://www.youtube.com/watch?v=3Fl_7FrM_gI

Waterfall vs. Agile





Historias de usuario

Normalmente

- Definir Requisitos?
- demostrar la funcionalidad desde la perspectiva de un usuario.

→ definir personas

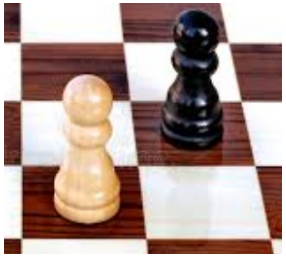
- Criterios de aceptación
- Comentarios con elementos que aportan valor
- Estimaciones de tiempo?
- Centrarse en el valor??

Dele una chance a la seguridad...

- No termine hasta que analice un modelo de amenazas del caso de uso o Historia
- Incluir criterios de aceptacion de seguridad.
- Diseñar para que los datos viajen y se guarden con confidencialidad.
- El operador no pueda negar que realizo la transacción.
- Manejar explicitamente las condiciones de error (estados de falla)
- Testeos unitarios de seguridad

https://es.smartsheet.com/user-stories-guide

Pensar como el adversario...



Historias de Abuso

- Una 'historia de abuso' es una historia de usuario desde el punto de vista de un adversario malicioso.
- Intentan ser una base para definir las actividades que deben ser bloqueadas o mitigadas activamente por el software y demostradas mediante pruebas de regresión automatizadas.

Como Cliente Autenticado,

- Veo lo que parece ser mi número de cuenta en la URL, así que lo cambio por otro número para ver qué sucede.
- Trato de entrar como otro usuario o de usar funciones a las que no tengo acceso modificando la url, cambiando parametros, etc.

https://rietta.com/blog/what-is-an-abuser-story-software/


Enter Username and password
Includes
User
User Authentication
Threatens
Brure Force Authentication
Includes
Includes
Show Generic Error Message
Mitigates
Harverst (e.g. guess) Valid User Accounts
Includes
Mitigates
Includes
Application/Server
Validate Password Minimum Length and Complexity
Mitigates
Dictionary Attack
Hacker/Malicious User
Includes
Mitigates
Lock Account After N. Failed Login Attempts

Sorry points

Story Points: medida abstracta que representa la complejidad y el esfuerzo requerido para completar una tarea.

Que pasa si se detecta una amenaza grave con impacto potencial?

- ¿que pasa si sale mal? (muy mal)
- → Sorry point… Estimacion del esfuerzo requerido para resolver dicho impacto...

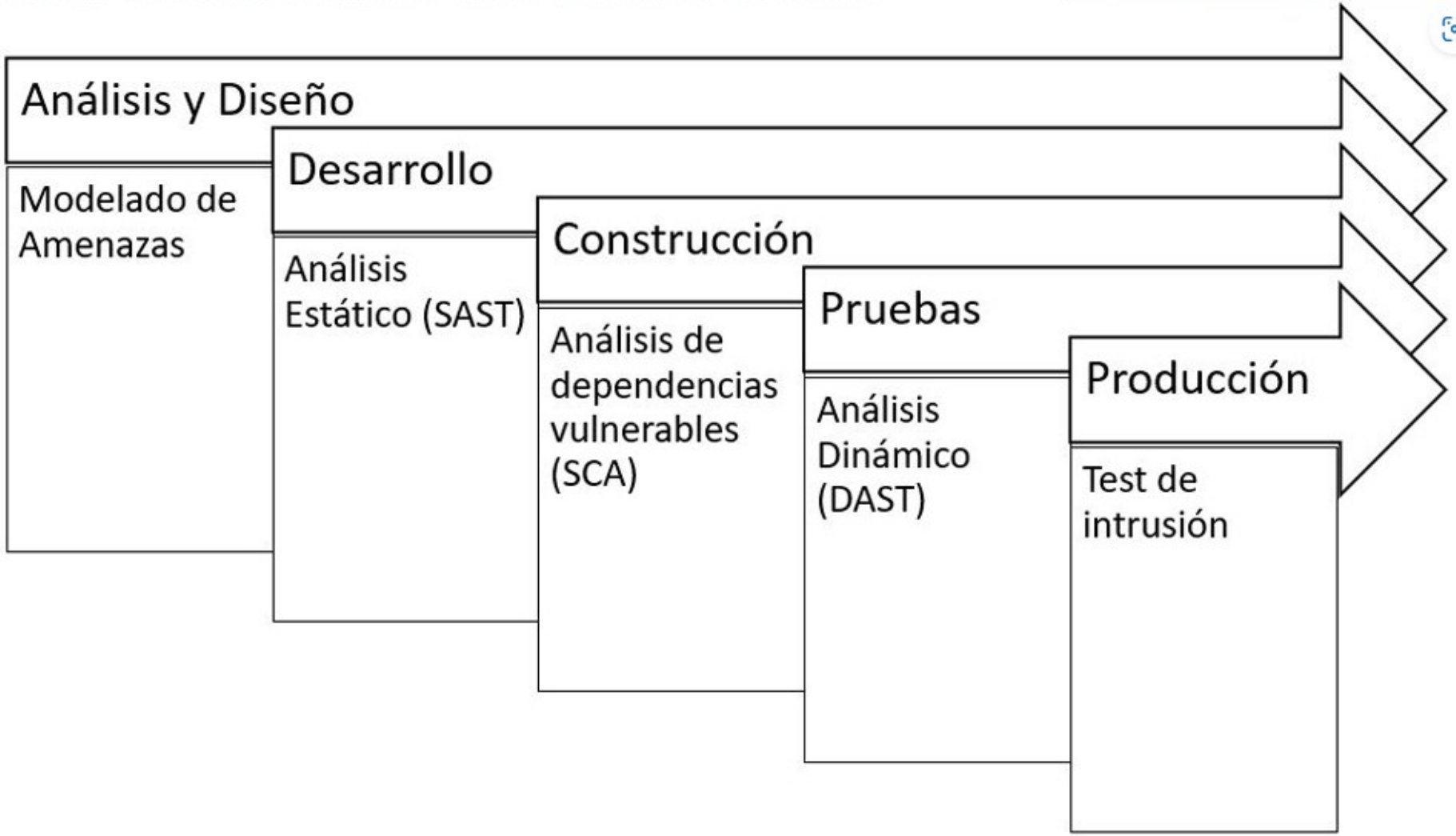
Deja de golpear al topo...

La modelización de amenazas es la clave para una defensa enfocada.

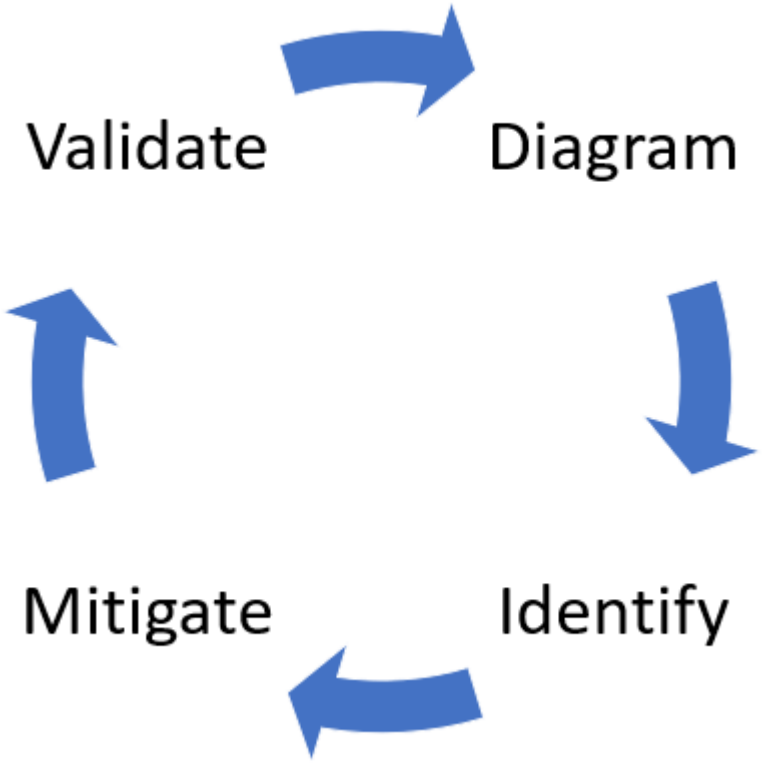
Sin la modelización de amenazas, nunca podrás dejar de jugar al 'whack-a-mole' (Golpea al topo).

Adam Shostack

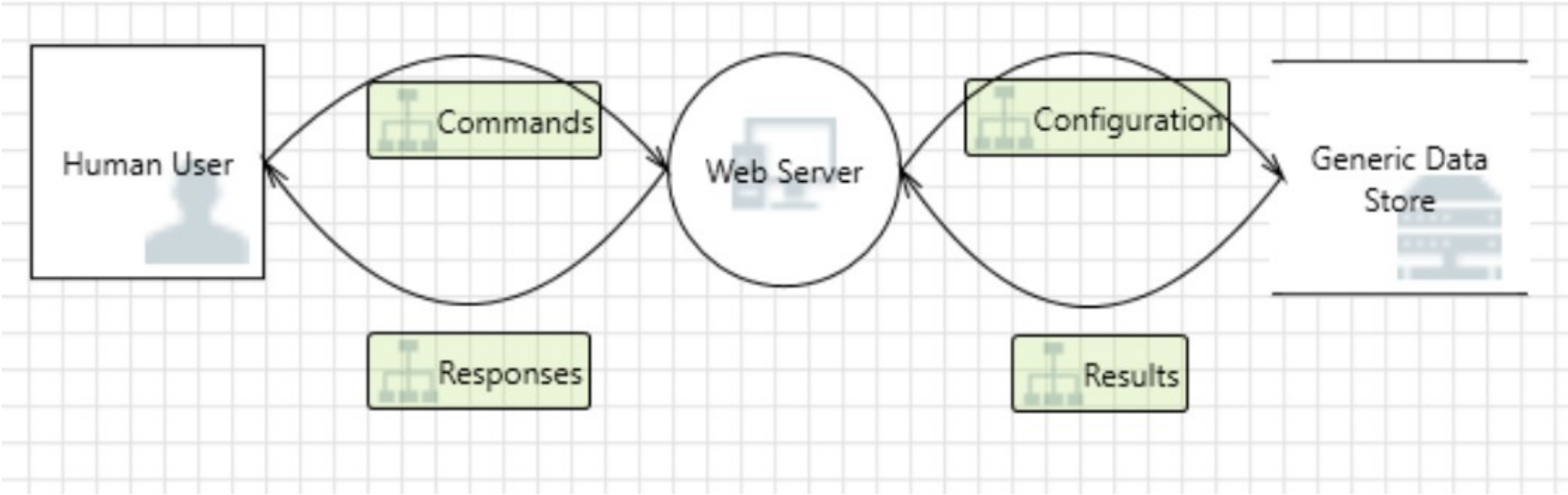
Seguridad desde el diseño

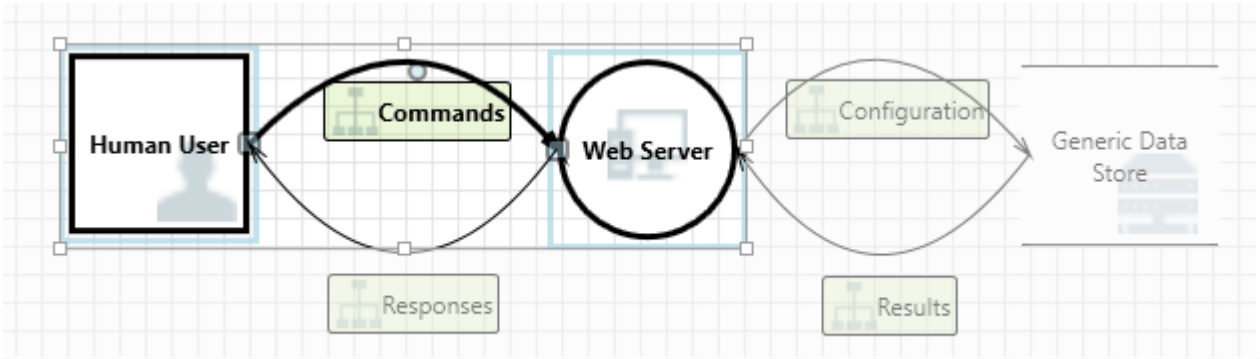


https://cuadernosdeseguridad.com/2019/01/cuida-la-seguridad-desde-el-diseno-con-el-modelado-de-amenazas/



https://learn.microsoft.com/es-es/azure/security/develop/threat-modeling-tool-getting-started


Human User
Commands
Responses
Web Server
Configuration
Results
Generic Data Store



reat Properties

: 0 Diagram: Diagram 1 Status: Not Started

Title:	Spoofing the Human User External Entity
Category:	Spoofing
escription:	Human User may be spoofed by an attacker and this may lead to unauthorized access to Web Server. Consider using a standard authentication mechanism to identify the external entity.
stification:	
nteraction:	Commands
Priority:	High

Diagram: Diagram 1

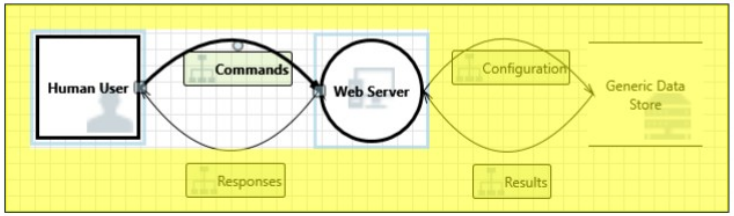
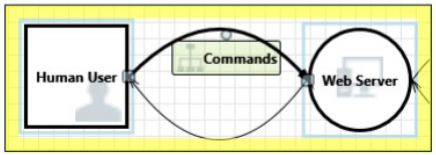


Diagram 1 Diagram Summary:

Not Started	9
Not Applicable	0
Needs Investigation	0
Mitigation Implemented	0
Total	9
Total Migrated	0

Interaction: Commands



1. Spoofing the Human User External Entity [State: Not Started] [Priority: High]

Category:	Spoofing
Description:	Human User may be spoofed by an attacker and this may lead to unauthorized access to Web Server. Consider using a standard authentication mechanism to identify the external entity.
Justification:	<no mitigation provided>
Possible Mitigation(s):	
SDL Phase:	Design

2. Cross Site Scripting [State: Not Started] [Priority: High]

Category:	Tampering
Description:	The web server 'Web Server' could be a subject to a cross-site scripting attack because it does not sanitize untrusted input.
Justification:	<no mitigation provided>
Possible Mitigation(s):	
SDL Phase:	Design