Criptografía y Seguridad

Seguridad en aplicaciones: Principios de diseño

Principios de diseño

- Bases que promueven un diseño que tenga como resultado un sistema robusto y seguro
- Son principios guía de alto nivel*
- Basados en dos ideas:
  - Simplicidad
    - Menos cosas pueden salir mal
    - Menos inconsistencias y zonas no definidas
    - Fácil de entender y verificar
  - Restricción
    - Minimizar el acceso
    - Minimizar la comunicación

(*) *"The protection of information in computer systems"* (Salzer & Schroeder)

1. Menor privilegio

- Un sujeto debe recibir solo los privilegios necesarios para completar su tarea
- Asignar privilegios por función, no por identidad
  - Ejemplo: El CEO de una compañía, no tiene por que tener acceso a todos los archivos confidenciales
- Si una tarea requiere derechos adicionales, asignarlos y desecharlos luego de su uso
- Muchas veces el S.O. o sistema no posee el nivel de granularidad deseado

Ejemplo

- Web server
  - Debería poder leer las carpetas de archivos web
  - Debería poder leer sus archivos de configuración
  - Debería poder escribir en sus carpetas de logs (solo en modo append)
  - NO debería poder leer otros archivos
  - NO debería poder escribir en otros lugares (*)
  - NO debería poder sobreescribir logs

¿Alguien vio un web server configurado de esta manera?

(*) Salvo lugares de upload o webdav

2. Valores iniciales seguros

- El acceso a cualquier objeto debe ser denegado por defecto
- Si una acción falla por seguridad, el sistema debe volver al estado de seguridad inicial
- Ejemplo: sshd intenta abrir el puerto 22 y falla.
  - sshd NO debe abrir otro puerto
  - sshd NO debe elevar sus privilegios para reintentar

Si lo hiciese, es probable que se pueda atacar el sistema.

Ejemplo

- Las instalaciones antiguas de Oracle definen usuarios administrativos iniciales
  - Históricamente, el instalador crea los usuarios con claves predefinidas
  - Algunos usuarios no son tan evidentes
  - Es responsabilidad del administrador modificarlos
- En la actualidad
  - Los instaladores solicitan la clave inicial

No se admite que el sistema inicie usando las password default

3. Economía de mecanismos

- Los mecanismos de seguridad deben ser simples
- Menos cosas pueden salir mal
- Se simplifica la verificación (formal e informal)
- Si algo sale mal es más fácil de corregir
- Las relaciones de confianza son más visibles
  - Confianza en entradas y salidas

Ejemplo

- Protocolo finger

  Es un protocolo antiguo que no se usa mas

  - Un host puede pedir información de un usuario registrado en otro host
  - Muchas implementaciones asumen que la respuesta del servidor esta bien formada
    - Este hecho está muchas veces oculto dentro de las complejidades propias del protocolo
  - Un servidor puede generar un mensaje de respuesta infinito
    - Se llenan logs, disco, Denial of service
    - Se cae el servicio, Denial of service
    - Un buffer overflow, ejecución remota de código

4. Mediación completa

- TODOS los accesos a objetos deben ser verificados

  y esa verificacion se tiene que hacer siempre en un unico punto

- Incluso si el objeto es accedido varias veces
- Puede ir en contra de la eficiencia
  - No permite utilizar caches
- Complejo de implementar
  - ¿Que ocurre si mientras se esta accediendo a un objeto (ej: un archivo), nos quitan el permiso?

La idea de mediacion completa es siempre hacer la verificacion en el unico punto. Los problemas vienen cuando ese unico punto se transforma en un SPOF y falla.
Entonces choca la idea de tener redundancia con la de tener un unico punto de verificacion.

5. Diseño abierto

- La seguridad no debe depender del secreto del diseño o la implementación
- No significa que deba publicarse el código fuente
- Un atacante puede conseguir el algoritmo
  - Desensamblando el ejecutable
  - Sobornando/coercionando a un desarrollador
  - Buscando en desechos
- Esto no aplica a las claves, sino a los algoritmos
- La violación del principio se denomina “seguridad por oscuridad”

Lo unico que debe estar oculto es la clave. Nosotros tenemos que diseñar el sistema sin asumir que nuestro codigo siempre va a ser secreto.

6. Separación de privilegios

- Un sistema no debe otorgar permisos basado en una sola condición
- Se refiere a la asignación de permisos
- Busca evitar que un sujeto pueda obtener privilegios y usarlos
  - Separación de tareas
  - Defensa en profundidad

Por ejemplo, el que administra una base de datos no debe ser el desarrollador

Ejemplo

- Para aprobar transacciones electrónicas por sobre cierto monto, los bancos requieren dos firmas
- Algunos sistemas no permiten modificar los permisos de otro administrador

7. Mecanismos exclusivos

- Los mecanismos de seguridad no deben compartirse

  No podemos hacer que el mecanismo de seguridad comparta codigo con otras cosas como logica de empresa

  - Puede fluir información entre variables compartidas
  - Canales ocultos
- Promueve aislación
  - Maquinas virtuales
  - Sandboxing

  Si yo tengo un proceso ejecutandose en una maquina, no quiero que ese proceso pueda ver al resto de los procesos ejecutandose en ese dispositivo.
  Por ejemplo, en Android o iOS las aplicaciones corren en un sandbox y no comparten ni siquiera /tmp con las otras aplicaciones.

8. Aceptación psicológica

- Los mecanismos de seguridad no deben dificultar el acceso al recurso
- Requiere ocultar el mecanismo
- En general es muy dificil o imposible
  - Simplificar instalación y configuración
  - Evitar necesidad de conocimientos técnicos

Por ejemplo, no tiene sentido que nos pidan una password de 40 digitos.
Hay que ocultar la complejidad del mecanismo al usuario siempre que se pueda.

Ejemplo

- Para proteger computadoras hoy día se utilizan firewalls personales
  - La mayoría requiere que el usuario identifique redes como internas o externas
  - El usuario debe poder determinar cuando una aplicación puede actuar como servidor o acceder a la red
- Windows Vista UAC
  - Cada operación privilegiada muestra una ventana solicitando acceso
  - El usuario debe decidir si la operación legítimamente necesita el acceso o debe ser denegado

Lectura Recomendada

TODOS los sistemas tienen que hacer un balance entre 3 cosas:
- Seguridad
- Features
- Performance

Capítulo 12-13
Computer Security Art and Science

Matt Bishop

Es importante la idea de seguridad por capas.
Yo voy haciendo una capa detras de otra. Si un atacante supera la primera capa, entonces tiene otra capa y otra.