

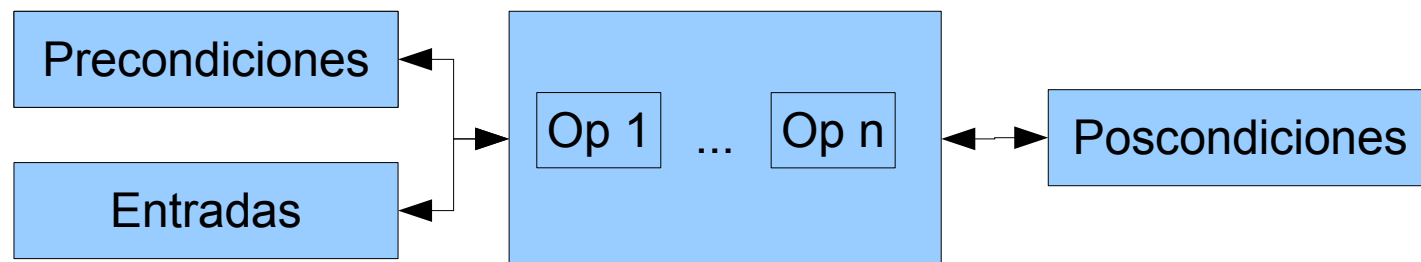


Criptografía y Seguridad

Probando la
seguridad:
Penetration testing

Verificación formal

- Verificación matemática de que un sistema cumple con ciertas restricciones
- Precondiciones → hipótesis sobre el estado del sistema
- Poscondiciones → resultado de aplicar operaciones del sistema a un input dado



- Requerimiento: las poscondiciones cumplen las restricciones

Prueba de penetración

- *Prueba* para verificar que un sistema cumple con ciertas restricciones
- Precondiciones → hipótesis sobre el estado del sistema y la existencia de una vulnerabilidad
- Poscondiciones → Sistema comprometido
- Ejecución:
 - Aplicar pruebas para intentar mover al sistema del estado inicial al estado comprometido

Similitudes y diferencias

- Prueba de penetración y verificación formal
 - Prueba la existencia de vulnerabilidades
- Pero la prueba de penetración
 - NO prueba la ausencia de las mismas
- Mientras que una verificación formal
 - Prueba la ausencia de vulnerabilidades
 - Aunque para ello debe incluir TODOS los factores externos
 - En la práctica se prueba ausencia de vulnerabilidades en determinado algoritmo, programa o ambiente, y no en un sistema
 - Se ignoran: Instalación, Uso

Objetivos

- Probar la eficacia de los controles de seguridad de un sistema
 - Intentando violar la política de seguridad
 - Requiere ejecutar técnicas similares a las de un atacante
 - También llamado 'ethical hacking'
 - Análogo a pruebas manuales de un sistema
 - No reemplaza un buen diseño e implementación
 - Prueba al sistema como un todo
 - No solo aspectos técnicos

Metodología (informal)

- Determinar y cuantificar objetivos
 - Ejemplo: obtener información de clientes
- Encontrar cierta cantidad de vulnerabilidades o buscarlas durante un periodo de tiempo
 - El estudio se conduce desde el punto de vista de un atacante
- Estudiar y categorizar los hallazgos
 - El éxito de una prueba de penetración proviene del análisis de los hallazgos
 - Permite detectar problemas recurrentes
 - Permite enfocarse en mejorar sistemáticamente familias de problemas

Metodología de Hipótesis de Falla

1.Recolección de información

Para conocer el sistema y su funcionamiento

2.Hipótesis

Asumir existencia de vulnerabilidades concretas

3.Prueba

Probar las vulnerabilidades

4.Generalización

Generalizar patrones y hallar otras vulnerabilidades del mismo tipo

5.(opcional) Eliminación

Determinar pasos necesarios para eliminarlas

Recolección de información

- Componer un modelo del sistema y sus partes
 - Buscar discrepancias
 - Revisar interfases
- Es necesario conocer bien el sistema
 - Utilizar documentación de diseño, manuales (si están disponibles)
 - Especialmente buscar secciones poco especificadas o ambiguas
 - Ver manejo de privilegios y tipos de cuentas
- También el ambiente
 - Conseguir nombres de usuarios / servidores
 - Estructura de red del sistema

Punto de partida

- Identifica la información inicial con la que cuenta el equipo
- Niveles incrementales
 - Atacante externo sin conocimiento del sistema
 - Atacante externo con conocimiento del sistema
 - Atacante con acceso al sistema
- Dependiendo del tipo de prueba, algunos niveles son irrelevantes
 - Ejemplo: Durante el diseño, se ignora el primer nivel
 - Ejemplo: En sistemas con registro abierto, se ignora el segundo nivel

Hipótesis

- Buscar posibles vulnerabilidades
- Examinando políticas y procedimientos
 - Buscar inconsistencias
 - Buscar inconsistencias entre políticas y mecanismos
 - Los procedimientos pueden no cumplirse
- Examinando implementaciones
 - Utilizar modelos de vulnerabilidades
 - Revisar vulnerabilidades comunes/conocidas (vulnerability scanning, e.g. portmapper)
 - Usar manuales (exceder limites, omitir pasos de las secuencias, etc)

Hipótesis

- Seguridad en APIs
 - Política de desarrollo de Microservicios
 - Software Supply Chain (API) Management
- RASP: Runtime Application Self Protection
 - Mal implementados
- Verificar Federadores
 - Vulnerabilidades conocidas sobre protocolos.
- Software EOL
 - Software fuera de soporte donde no existen actualizaciones de seguridad o que no se encuentren actualizados.

Hipótesis

- Examinando mecanismos
 - Pueden estar mal implementados
 - El ambiente donde corren puede introducir errores
 - Pueden no ser seguros
- Comparando con otros sistemas
 - Sistemas parecidos tienen problemas parecidos
- Resultado:
 - Una lista de posibles vulnerabilidades

Prueba

- Priorizar la lista de posibles vulnerabilidades
 - Depende del objetivo de la prueba
 - Por lo general: se prioriza por nivel de acceso requerido
- Definir como probar la existencia de la vulnerabilidad
 - La mejor manera: resultado de analizar documentación o comportamiento
 - Otra manera: intentar explotarla
 - Ultimo recurso
 - Menos eficiente
 - Excepción: Pivoting ! (explicado luego)

Prueba

- Diseñar el test para que sea lo menos intrusivo posible
 - Algunas vulnerabilidades pueden denegar el servicio!
- Proceso normal:
 - Resguardar los datos de todo el sistema
 - Documentar requerimientos para detectar la vulnerabilidad
 - Intentar detectar la vulnerabilidad
- La prueba **DEBE SER REPETIBLE**

Pivoting

- Es probable que la existencia de una vulnerabilidad provea más información
 - Ejemplo: Acceso al S.O.
 - Se debe volver a la fase de recolección de información
- Pivoting es el uso de un punto de acceso comprometido para continuar un ataque
 - Ejemplo: Se compromete el firewall y desde allí se continúa atacando la red interna

Generalización

- A medida que las pruebas resultan exitosas, emergen patrones
- Algunas veces dos vulnerabilidades combinadas constituyen un problema grave
 - Ejemplo:
 - Cuenta invitado habilitada permite conexión remota
 - Buffer overflow local permite derechos de administrador
- Constituye la parte más importante de la prueba

Eliminación

- Por lo general solo se incluyen recomendaciones
 - Quien ejecuta la prueba no suele ser el diseñador/desarrollador
- Es importante que quede claro el contexto, detalles y mecanismo de explotación
 - Para poder corregir el sistema
 - Para poder impedir / monitorear mientras tanto
 - Para verificar si fue explotado

Ejemplo: Michigan Terminal System

- Sistema operativo que corre en mainframes IBM 360/370
- Objetivo de la prueba: obtener acceso a las estructuras de control
- Nivel inicial: Cuenta autorizada (nivel 3)

Esta prueba de penetración
contaba con la aprobación
de los administradores

Paso 1: Recolección de información

- Aprender detalles del mecanismo de control del sistema
 - Al correr un programa, la memoria se divide en segmentos
 - 0-4: supervisor, programas de sistema y estado
 - Protegidos por mecanismos de hardware
 - 5: Area de trabajo del sistema, información del proceso
 - Incluido el nivel de privilegio
 - El proceso no puede alterar esta información
 - 6+: Información del proceso
 - El proceso puede modificar libremente estas zonas

Paso 1: Recolección de información

- El segmento 5
 - Protegido por el sistema de protección de memoria virtual
 - Modo sistema (kernel): el proceso puede acceder, modificar datos y ejecutar funciones privilegiadas
 - Modo usuario: El segmento 5 no se mapea al espacio de direcciones del usuario
- Un proceso corre en modo usuario
 - De modo usuario se realizan system calls
 - Las system calls corren en modo sistema

Paso 1: Recolección de información

- Ejecución de un system call
 - El código del system call revisa los parámetros
 - Especialmente, revisa que los punteros pertenezcan a zonas accesibles por el proceso
 - Los parámetros se construyen como una lista de direcciones en el segmento de usuario
 - La lista de parámetros se pasa como dirección en un registro

Paso 2: Hipótesis

- ¿Que ocurriría si la dirección de un parámetro apunta a la lista misma de parámetros?
 - Sin controles extra, sería posible que algún system call escriba la dirección del parámetro después de haber pasado la validación
 - Si se puede controlar que valor escribir, técnicamente sería posible leer o escribir cualquier zona del segmento 5

Paso 3: Prueba

- Buscar un system call que
 - Use la convención de llamada estudiada
 - Tenga al menos dos parámetros y altere 1
 - Pueda hacer que se altere el parámetro con cualquier valor
- Entre los candidatos se eligió : line input
 - Lee una línea de texto y retorna número de línea y longitud de línea
- Setup
 - Hacer que la dirección donde se almacenará el número de línea sea la de la longitud de línea

Paso 3: Prueba

- Ejecución:
 - El system call valida los parametros
 - Todos pertenecen al segmento del proceso
 - Ejecuta método read input line
 - La longitud de la línea ingresada es el valor a escribir
 - El número de línea se guarda según la lista de parámetros
 - Hacer que sea una dirección del segmento 5
 - Esto reescribe la dirección del otro parámetro
 - La longitud se guarda según la lista de parámetros
 - Esto escribe el valor en el segmento 5

Paso 4: Generalización

- No se puede escribir en los segmentos 0-4
 - Protegidos por hardware
- Pero en el segmento 5, el nivel de privilegio determina si pueden hacerse llamadas de nivel supervisor (en lugar de system calls)
 - Y una función de nivel supervisor apaga la protección por hardware
- Conclusión:
 - Esta vulnerabilidad permite que un atacante modifique cualquier dirección de cualquier segmento, controlando completamente la computadora

Ejemplo: Ataque externo

- Objetivo:
 - Determinar si las medidas de seguridad de una empresa eran efectivas para evitar que un atacante ingrese a un sistema
- El test se enfoca en políticas y procedimientos
 - Tanto técnicos como no técnicos

Paso 1: Recolección de información

- Búsqueda en internet
 - Nombres de empleados y directores
 - Teléfono de la sucursal local. De la sucursal local consiguieron el reporte anual (público – la empresa cotizaba en bolsa)
- Reconstrucción de estructura
 - Partes del organigrama
 - Nombres de proyectos y personas que trabajan en cada uno

Paso 1: Recolección de información

- Listado de empleados
 - El tester se hizo pasar por un nuevo empleado
 - Aprendió que para enviar cualquier documento necesitaría: número de empleado y centro de costos
 - El tester llamó a la secretaria del director sobre el que había recabado mas información
 - Primero haciéndose pasar por un empleado, consiguió el número de empleado del director
 - Luego, haciéndose pasar por un auditor, el centro de costos
 - Con éstos datos se solicito enviar el listado a una consultora externa

Paso 2: Hipótesis

- Los empleados nuevos no conocen todos los procesos y controles
 - Es posible que un nuevo empleado brinde información sensible

Paso 3: Prueba

- El tester llamó a RRHH impersonando a la secretaria de un director
 - Se quejo que no le habían informado de los nuevos ingresos y los pidió
 - Obtuvo los nombres de los nuevos empleados
- El tester llamó a los nuevos empleados
 - Se hizo pasar por operador del centro de computos
 - Les brindo un entrenamiento sobre seguridad por teléfono
 - Durante el entrenamiento obtuvo: tipos de sistemas usados, número de usuarios, logins y claves

Para discutir

- ¿Cuán válidos son los tests de penetración?
 - No sustituyen una buena especificación, diseño, implementación y pruebas
 - Es una técnica importante para probar un sistema luego de ser instalado
 - Idealmente no es necesario. En la práctica, si
 - Encuentra problemas introducidos por la interacción del sistema con los usuarios y el ambiente
 - Este tipo de problemas suelen quedar fuera del análisis y pruebas normales

Para discutir

- ¿Que determina la calidad de una prueba de penetración?
 - La metodología de hipótesis de fallas depende de la capacidad de los testers para formular hipótesis
 - Particularmente, no provee una forma sistemática de revisar un sistema
 - Los resultados de un test sirven solo marginalmente para otros. Cada test es un mundo aparte
 - Hay herramientas que automatizan algunos aspectos de una prueba, pero no todos

Lectura recomendada

Capítulo 23: 1-2
Computer Security Art and Science

Matt Bishop

OSSTMM - Open Source Security Testing
Methodology Manual
<http://www.isecom.org/osstmm/>