**Técnicas relacionadas con “Information Gathering”**

La recolección de información es la capacidad de obtener la mayor cantidad de datos de un potencial blanco a auditar o atacar.

La misma se puede realizar tanto de manera externa al blanco como interna, es decir que podemos utilizar un número muy grande de herramientas disponibles para recolectar la mayor cantidad de información del equipo sin haber accedido a el y por otro lado contamos con herramientas que nos brinda el propio sistema operativo.

**Herramientas disponibles**

Analizando las herramientas disponibles en Internet que permiten obtener y analizar la informacion de un servidor para detectar y corregir debilidades o vulnerabilidades, podemos diferenciar las mismas en 4 categorias principales:

* Preventivas: Aqui encontramos herramientas que buscan fallas y una vez encontradas nos permiten conocer de manera detallada su solucion (aplicar ‘X’ parche, remover ‘X’ usuario, etc)
* Reactivas: Estas herramientas se enfocan en tomar una accion al momento de detectarse un evento inusual que puede comprometer la seguridad del sistema. Podemos encontrar aqui los detectores de intrusos IDS (diferenciados como HIDS Host Intrusion Detection System, los cuales se enfocan en eventos a nivel del servidor como ser file system, integridad de archivos, programas, usuarios, procesos y NIDS Network Intrusion Detection System, enfocados exclusivamente en eventos de red). Toda la linea de solucions “IDS” se enfocan en capacidad de deteccion y alerta, mientras que existe otra linea similar denominada “IPS” Intrusion Prevention System, la cual ademas de alertar, se encarga de tomar medidas correctivas (finalizar un proceso, bloquear una conexion, etc)
* Deteccion: En esta categoria podemos encontrar herramientas especificas de deteccion de una amenaza, compromiso o indicador que haga evidente un problema ya ocurrido en un servidor pero no visible de manera normal.
* Recuperacion: Herramientas dedicadas a recuperar frente a eventos ya ocurridos y que generaron daños en el equipo afectado, imposibilitando de acceder a u nservicio, ejecutable, etc.

Si nos centramos un poco mas en detalle, podemos encontrar para cada uno de los puntos, diversas areas de cobertura. Si bien existen muchas opciones, detallamos aqui las mas significativas, ya sea por popularidad o por haber tenido la oportunidad de utilizarlas. Entedemos que podemos separar las mismas de la siguiente manera:

Analisis de vulnerabilidades

Detallamos herramientas que permiten analizar equipos y detectar vulnerabilidades, ya sea por la version del servicio expuesto o por la version del software instalado. Entre las mas destacadas mencionamos

* Nessus - <https://tenable.com> : Considerada una de las pioneras en el analisis de vulnerabilidades, la misma se creo en el año 1998 por su fundador Renaud Deraison. Su objetivo fue que la comunidad tuviese una herramienta abierta para el escaneo de vulnerabilidades, aunque luego su licencia cambio. Al dia de hoy la empresa propietaria es tenable (https://tenable.com). Es una herramienta muy completa para el analisis de vulnerabilidades, la cual integra su propio lenguaje de scripting para escribir los diferentes scripts que componen las pruebas a realizar sobre un equipo auditado. Los scripts escritos para nessus tienen la extension NASL (Nessus Attack Scripting Language) y tienen un formato bastante simple ( <https://en.wikipedia.org/wiki/Nessus_Attack_Scripting_Language> )
* Openvas - <https://openvas.org> : Es una herramienta que nacio como variante a Nessus en el año 2006, llamada inicialmente GNessus. Si bien al momento de realizar esta documentacion, la herramienta esta disponible en distribuciones de pentest y para descargar mediante paquetes y fuentes para compilar, nos parece que el proyecto luego de haber sido adquirido por la empresa GreenBone, esta cambiando su rumbo, seguramente llegando a ser en cualquier momento un producto de licencia cerrada. Ejecuta los mismos scripts de Nessu (NASL) y cuenta con muchas funcionalidades a la hora de realizar un escaneo (https://es.wikipedia.org/wiki/OpenVAS). Actualmente cuenta con una version gratuita y tambien se ofrece una version paga derivado al cierre (Gratuita con restricciones - Paga)
* Netsparker - <https://www.netsparker.com> : Es una herramienta desarrollada inicialmente en el año 2006 por Ferruh Mavituna. La misma pone foco especificamente en el analisis de aplicaciones web independientemente de la tecnologia, framework o lenguaje utilizado para su desarrollo, permitiendo escanear todos los puntos de la misma generando visualizacion muy completa del estado de seguridad de dicha aplicacion a auditar.

Auditoria - Hardening - Compliance

* <https://cisofy.com/lynis/> : Esta es una herramienta gratuita muy completa, la cual permite realizar auditoria de seguridad, hardening y chequeos de compliance. Permite ejecutar de manera simple y rapida los chequeos necesitados. Una desventaja que tiene la herramienta es la no posibilidad de agregar chequeos de manera personalizada.

Chequeo Integridad archivos/directorios e intrusion

* <https://www.nongnu.org/tiger/> : Herramienta de codigo abierto con fuerte foco en un sistema HIDS (Host Intrusion Detection System). Cuenta con varios modulos de ejecucion bastante interesantes, su ultima version es del año 2018
* <https://www.ossec.net/:> Se le puede considerar como la herramienta con mayor cantidad de funcionalidades y prestaciones para montar un servicio de deteccion y prevencion de intrusos a nivel de host. Es un proyecto de codigo abierto desarrollado inicialmente por Daniel B. Cid. Actualmente se encuentra en desarrollo y goza de muy buena salud.
* <https://aide.github.io/> : Es un proyecto de deteccion de intrusos a nivel de chequeo de modificaciones de archivos y directorios. La version actual es la 0.17.3 con fecha del 10/02/2021.

Busqueda RootKits

* <http://rkhunter.sourceforge.net/> : Proyecto iniciado en el año 2003 por M. Boelen, quien en el año 2006 entrego el mismo a u ngrupo de desarrolladores quienes se hicieron cargo desde la version 1.2.9 (actualmente la ultima estable es la 1.4.6). La herramienta se encarga de analizar el equipo sobre el cual se ejecuta a efectos de buscar señales de compromiso, como ser binarios alterados, procesos en memoria no reflejados en lista, ejecutables conocidos, etc.
* <http://www.chkrootkit.org/> : Se lo puede considerar como una version mejorada de rkhunter, desarrollado por Nelson Murilo y Klaus Steding Jessen. La version actual es la 0.55 con fecha de lanzamiento 11/06/2021 y su foco es la busqueda de alteraciones a nivel del servidor (de todo nivel, interfaces de red, procesos, binarios, etc) a efectos de diagnosticar un compromiso.

Administracion de vulnerabilidades

* <https://www.open-scap.org/tools/openscap-base/> : SCAP es un metodo de utilizacion de estandares especificos que permite automaitzar la administracion de vulnerabilidades (<https://en.wikipedia.org/wiki/Security_Content_Automation_Protocol>). La herramienta permite asistir tanto a administradores como auditores enla evaluacion, medicion y aplicacion de las lineas de base de seguridad. Para tener un detalle exacto de la definicion de SCAP, se sugiere el siguiente enlace: <https://public.cyber.mil/stigs/scap/>

Motivacion

Si bien se tienen todas las areas cubiertas, hay algo que hemos encontrado y es que para poder ampliar la capacidad de estas herramientas, se depende del nivel de “velocidad” que el proyecto avance para contar con los ultimos agregados o detecciones de nuevas amenazas, haciendo este como un aspecto negativo que aplica a todas las soluciones.

Dado que este proyecto esta pensado para administradores y es de suponer que el uso de scripts es algo de uso habitual y comun en las tareas diarias, que mejor opcion que permitirle al propio administrador escribir sus scripts y que estos sea el punto de entrada para la herramienta, permitiendo:

1. generar scripts propios que se encarguen de realizar tareas especificas a la necesidad puntual
2. no depender de la velocidad de lanzamiento de nuevas actualizaciones
3. simplificar el desarrollo en un lenguaje de scripting muy simple como es bash

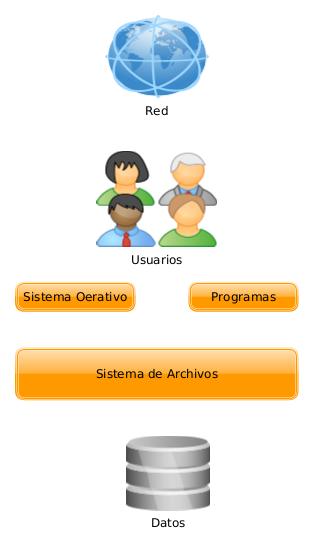
Hay que tener presente que cuando se ataca y explota un servicio determinado, este corre bajo el ID de un usuario puntual, con lo cual , cuantas más herramientas tengamos disponibles para utilizar, mayor recolección de información vamos a poder lograr.

Linux nos ofrece una cantidad de herramientas y utilidades ya instaladas por defecto, las cuales perfectamente pueden servirnos (y van a servirnos) para este proyecto.

Para poder tener claro la información que necesitamos recolectar, primero tenemos que tener claro que es lo que vamos a revisar, con lo cual vamos primero a separar en diferentes capas de acceso.

Tomando como base principal los datos a proteger (la cual sería la capa “interna”) a partir de ahí nos vamos alejando hasta la capa más “externa” que es la red.

Hecha la aclaración tenemos un diagrama que es el siguiente:



Descripción de las capas y herramientas disponibles en cada una de ella

**Red**: Puede variar de acuerdo a la red donde se encuentre el equipo. Lo importante es que aquí tenemos varias opciones a realizar a efectos de mejorar la misma.

Algunas herramientas disponibles son:

* netstat / ss: Ver conexiones activas, servicios en escucha, servicios por protocolo, estado de conexiones, procesos asociados, contextos, enrutamiento, etc
* route: Tabla de ruteo
* ip / ifconfig: información de las interfaces de red, estado, errores, dirección MAC, túneles, ruteo, etc
* arp: Tabla de direcciones MAC
* ethtool: Estado de las interfaces de red, driver asociado, estadísticas, configuraciones

**Usuarios**: Esta capa refiere a los permisos de acceso de los usuarios en el sistema e incluye autentificación, autorización y acceso.

Hay muchos controles aquí que implican verificaciones a nivel de permisos. Se mencionan algunos ejemplos:

* Usuarios con id 0
* Terminales habilitadas para realizar login
* Cuentas con o sin shell
* Grupos a los cuales pertenece un usuario
* Restricciones mediante PAM (fortalecimiento de contraseñas, forzar cambios, historial, etc)

**Sistema Operativo**: En este punto, hacemos referencia a todo lo que implica el sistema operativo auditado o comprometido. Esto incluye procesos en ejecución, servicios por defecto habilitados, aplicaciones y su configuración, manejo de dispositivos, etc.

Ejemplos de control:

* Logs de auditoría registrados y guardados en sistema centralizado
* Revisión de violaciones de acceso
* Permisos de ejecución de herramientas sensibles (o peligrosas)
* Control de accesos a comandos como root (su)
* Dispositivos y herramientas del sistema protegidos contra usuarios no autorizados (sniffers, analizadores de protocolo, herramientas de red, etc)

**Sistema de Archivos**: Es importante restringir los permisos de acceso de los usuarios a determinados lugares del sistema de archivos que no deben acceder. Ejemplos:

* Registros de sucesos
* Archivos y directorios sin propietario
* Permisos de acceso a archivos de configuración
* Minimizar los directorios del sistema operativo con permisos totales

**Datos**: Esta es la información que estamos protegiendo. Debemos tener claro que protegemos y como lo estamos haciendo, Ejemplos:

* Si es una base de datos, permisos de archivos, usuarios, contraseñas, accesos anónimos, bases de test habilitadas
* Archivos de texto, permisos mencionados anteriormente

Todos los puntos mencionados anteriormente parten de la base de tener acceso fisico al servidor a auditar, no obstante siempre hay un punto previo que implica el analisis y recoleccion de toda la informacion posible de manera pasiva, sin haber accedido aun al servidor.

Para esto, podemos dividir en diferentes “capas” de analisis como ser:

Obtencion de informacion de datos publicos:

Analisis de servicios expuestos (versiones detectadas y problemas conocidos)

Deteccion de los servicios (escaneo de puertos, consulta en bases publicas)

Analisis de vulnerabilidades (Herramientas)

Analisis de topologia (Deteccion de conectividad, gateways, red, etc)