Resumen Teórica 17 : Código Malicioso

Tomás F. Melli

October 2025

Índice

1 Introducción

Malware —del inglés *malicious software*— es cualquier programa o fragmento de código diseñado con la intención de dañar, interrumpir, robar información o tomar el control de un sistema informático sin el consentimiento del usuario. Su objetivo puede ser económico, político, de espionaje, o simplemente causar interrupciones.

Qué puede hacer el malware (ejemplos de comportamientos):

- Mostrar publicidad no deseada (adware).
- Borrar o corromper archivos de configuración o del disco, dejando la computadora inoperable.
- Infectar una máquina y utilizarla para atacar a otras (botnets), haciendo que los ataques parezcan originarse desde la primera víctima.
- Recoger información sobre la persona: hábitos de uso, historial de navegación, credenciales (usuarios/contraseñas), localización, etc.
- Capturar audio y/o video del dispositivo y transmitirlo al atacante.
- Ejecutar comandos en el sistema con los privilegios de un usuario legítimo.
- Cifrar archivos y pedir un rescate económico (ransomware).
- Robar documentos y archivos sensibles: datos personales, financieros, licencias de software, etc.
- Subir y distribuir más archivos al sistema, incluidos otros componentes maliciosos, software ilegal o material inapropiado.
- Aprovechar el poder de cómputo de la máquina para minería de criptomonedas o tareas no autorizadas.

Ejemplo

A continuación se muestra un fragmento de *shell script* malicioso.

```
1 cp /bin/sh /tmp/.xyzzy
2 chmod u+s,o+x /tmp/.xyzzy
3 rm ./ls
4 ls $*
```

- La primera línea copia un intérprete de comandos a un archivo oculto en el directorio temporal. Esto crea una copia del shell con un nombre no evidente.
- La segunda línea modifica permisos para establecer el bit `setuid` en ese archivo. Conceptualmente, el bit `setuid` permite que un ejecutable se ejecute con los privilegios del propietario del archivo; en este ejemplo la idea es que el binario pueda ejecutarse con privilegios distintos a los del usuario que lo lanza.
- La tercera línea elimina o reemplaza un ejecutable legítimo (`ls`) en el directorio de trabajo, lo cual es una técnica de suplantación para engañar al usuario.
- La cuarta línea aparenta ejecutar `ls` (listado de ficheros) pero, por la sustitución previa, invocaría el binario malicioso. En términos generales, este patrón busca engañar al usuario para obtener un shell o ejecutar código con privilegios distintos, lo que supone una posible escalada de privilegios y un control no autorizado del sistema.

2 Caballo de Troya

Un **caballo de Troya** (o *troyano*) es un programa que presenta un propósito abierto y legítimo —conocido por el usuario— mientras oculta un propósito secundario y malicioso —desconocido para el usuario—. Muestra una funcionalidad útil o inofensiva para ganarse la confianza del usuario y, al mismo tiempo, ejecuta acciones no autorizadas en segundo plano.

Características típicas

- Propósito abierto: la funcionalidad visible que justifica su instalación (por ejemplo, un juego o una utilidad).
- Propósito oculto: acciones maliciosas escondidas (por ejemplo, abrir un *backdoor*, robar credenciales, espiar al usuario).
- Uso frecuente de ingeniería social para inducir la ejecución por parte del usuario.
- No siempre se replica por sí mismo (a diferencia de los gusanos), aunque existen variantes que sí lo hacen.

Un programa que aparentemente sirve para listar archivos en un directorio (propósito abierto) podría, en paralelo, contener instrucciones ocultas para crear un shell con privilegios elevados (propósito oculto). El usuario ve la funcionalidad legítima y ejecuta el programa, desconociendo el comportamiento dañino que ocurre detrás.

2.1 Ejemplo histórico: NetBus (1998)

NetBus fue una herramienta diseñada para sistemas Windows 9x/NT que alcanzó amplia difusión como ejemplo clásico de troyano de acceso remoto. Se distribuía frecuentemente disfrazada (por ejemplo, como un juego o una utilidad atractiva) para inducir a la víctima a descargarla y ejecutarla.

- Tras la ejecución en la máquina de la víctima, el programa quedaba en estado de servidor, escuchando comandos remotos desde un atacante.
- Permitía a un operador remoto realizar acciones de administración/espionaje: intercepción de pulsaciones de teclas, captura de pantallas, robo de credenciales almacenadas en aplicaciones, control de dispositivos (por ejemplo, apertura/cierre de la unidad de CD), transferencia de archivos hacia y desde la máquina infectada, entre otras.
- NetBus ilustró cómo una herramienta aparentemente útil o atractiva puede convertirse en una puerta trasera persistente para el control remoto de sistemas.

Troyanos que se replican

Aunque los troyanos tradicionales no se propagan por sí mismos (necesitan engañar al usuario para su ejecución), existen variantes llamadas *caballos de Troya autopropagables* que incluyen mecanismos para copiarse y distribuirse, reduciendo la dependencia de la interacción humana.

- Algunas primeras implementaciones experimentales de software malicioso llegaron a incorporar rutinas de copia para difundirse; en algunos casos estas rutinas se usaron con finalidades distintas (por ejemplo, autodestrucción de copias).
- En 1976, Karger y Schell propusieron la idea teórica de modificar el compilador para insertar automáticamente un troyano que se replicara en los binarios producidos, incluyendo potencialmente futuras versiones del propio compilador. Este concepto demuestra que la confianza en herramientas de construcción (compiladores, toolchains) también puede ser un vector de compromiso muy potente y difícil de detectar.

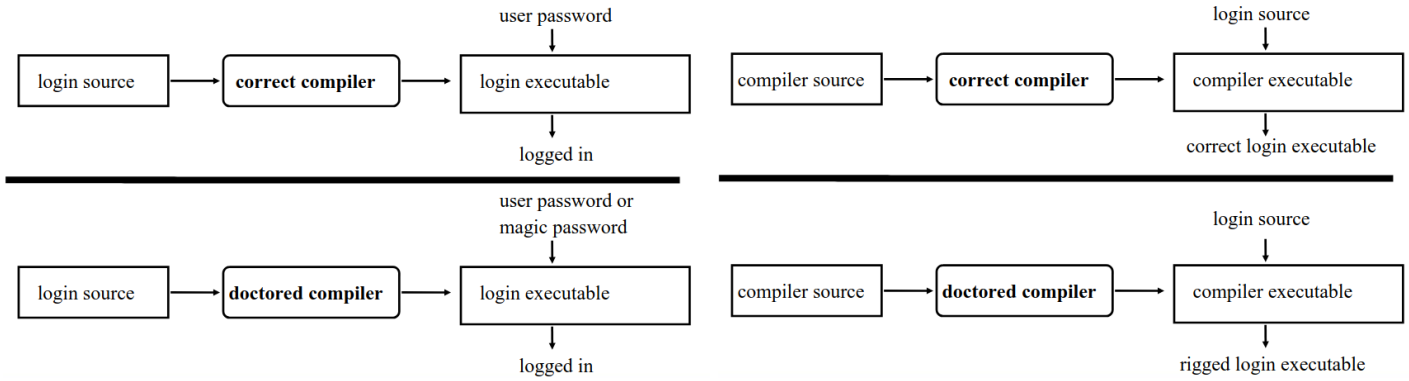
Dificultad de detección

Los troyanos autopropagables y los que se integran en toolchains son especialmente difíciles de detectar porque su actividad puede parecer parte del comportamiento normal de programas legítimos. La integración en cadenas de suministro de software o en herramientas de desarrollo puede resultar en compromisos sistémicos, afectando múltiples programas y generaciones de software.

2.2 Compilador de Thompson

El llamado *Compilador de Thompson* ilustra un vector de compromiso en la cadena de construcción: un compilador puede ser modificado para introducir código malicioso en binarios compilados (por ejemplo, el programa `login`) *sin que ese código figure en el código fuente* del programa víctima. Además, el compilador modificado puede reconocer cuando se compila una nueva versión del propio compilador e insertar automáticamente el código malicioso en la nueva versión compilada. Tras recompilar y restaurar la apariencia del código fuente, el backdoor persiste en los ejecutables producidos por el compilador infectado.

2.2.1 Descripción conceptual del ataque



Supongamos el flujo habitual de construcción:

1. Existe un **compilador correcto** y el **código fuente** del programa `login`.
2. Si se modifica el compilador para que, al compilar `login`, el ejecutable acepte además de la contraseña correcta una *clave mágica* universal, cualquier `login` compilado con ese compilador tendrá esa vulnerabilidad.
3. El atacante podrá modificar también el compilador para que, cuando se compile *el propio compilador*, inserte automáticamente ese mismo comportamiento malicioso en la versión compilada del compilador.
4. El atacante recompila el compilador usando el compilador infectado, obtiene un ejecutable de compilador que ya contiene la inserción maliciosa y, acto seguido, borra la versión del código fuente del compilador que contenía su cambio, dejando sólo el ejecutable infectado.
5. A partir de entonces, aunque el código fuente del compilador y del programa `login` parezcan correctos, cualquier compilación realizada con ese compilador infectado producirá ejecutables con el comportamiento oculto.

Tras esto, el atacante puede eliminar las modificaciones en el código fuente y dejar solamente el compilador ejecutable comprometido.

Por qué es tan peligroso

- **Subversión de la confianza:** revisar sólo el código fuente deja de ser suficiente, porque la inserción ocurre en la fase de construcción.
- **Persistencia oculta:** el backdoor puede sobrevivir a la restauración del código fuente; mientras exista el compilador infectado, todo lo compilado con él estará comprometido.
- **Difícil de detectar:** la modificación opera a nivel binario/ejecución; las auditorías de fuente no la encuentran fácilmente. Detectarla exige verificar la correspondencia entre fuente, herramientas y binarios resultantes.
- **Amplitud del impacto:** comprometer toolchains (compiladores, enlazadores, sistemas de build) puede afectar a múltiples proyectos y versiones.

2.2.2 Medidas de mitigación y buenas prácticas

Para reducir el riesgo de este tipo de compromiso en la cadena de suministro de software conviene combinar varias estrategias:

- **Reproducible builds:** garantizar que una build produce exactamente el mismo binario a partir de los mismos inputs; si varias máquinas independientes reproducen el mismo binario y coinciden los hashes, es menos probable que exista una inserción oculta.
- **Verificación de binarios frente a fuente:** comprobar que los binarios producidos pueden reproducirse a partir del código fuente público y que las herramientas usadas son las esperadas.
- **Diversidad de toolchains:** construir el mismo paquete con compiladores distintos (o en entornos separados) y comparar resultados para detectar discrepancias.
- **Firmas y procedencia:** firmar y verificar tanto herramientas como artefactos; mantener trazabilidad (provenance) de la procedencia de compiladores y binarios.

- **Auditoría de toolchains y builds:** auditar periódicamente los binarios del compilador, revisar imágenes de build y controlar quién puede modificar herramientas de construcción.
- **Segregación del entorno de build:** usar entornos de construcción aislados y reproducibles (por ejemplo contenedores inmutables o máquinas controladas), con control de acceso estricto.
- **Detección a nivel de comportamiento:** monitorizar ejecutables en tiempo de ejecución para detectar comportamientos anómalos (por ejemplo, aceptación de credenciales no válidas o conexiones inesperadas).
- **Construcción en máquinas limpias:** compilar herramientas desde fuentes verificadas en hardware y entornos iniciales conocidos "limpios".
- **Prácticas organizativas:** control de cambios para herramientas críticas, revisiones por pares, separación de funciones y gestión estricta de la cadena de suministro.

3 Virus

Un *virus* informático es un programa que se inserta a sí mismo en uno o más archivos y, en algún momento, realiza una acción determinada. Su finalidad puede ser variada: desde objetivos experimentales o demostrativos hasta daño deliberado, robo de información o persistencia en sistemas ajenos. A diferencia de otros tipos de malware, un virus requiere generalmente algún mecanismo de activación o ejecución y suele depender de la interacción humana para propagarse (por ejemplo, abrir un archivo infectado).

3.1 Fases de funcionamiento

Un virus típico presenta, de forma conceptual, dos fases principales:

3.1.1 Fase de inserción

El virus se incorpora en archivos ejecutables, sectores de arranque u otros contenedores de código. Durante esta fase el virus copia su cuerpo en uno o más objetos del sistema objetivo, de modo que cuando esos objetos sean ejecutados se desencadene la siguiente fase.

3.1.2 Fase de ejecución

Cuando el código infectado se ejecuta, el virus puede realizar una acción (que puede ir desde nula —es decir, permanecer inactivo— hasta destructiva). Las acciones incluyen, entre otras: replicación adicional, modificación de archivos, registro de teclas, despliegue de puertas traseras, o activación de una carga útil en una fecha u evento determinado.

3.2 Historia

La historia temprana de los virus informáticos mezcla experimentación académica y desarrollo informal:

- En los primeros años hubo programadores en plataformas como Apple II que escribieron ejemplares experimentales de programas que se autorreplicaban o se insertaban en otros ficheros; no siempre los denominaban "virus" y solían ser más bien ejercicios técnicos o demostrativos.
- Fred Cohen (1983–1984) es la figura clave en la formalización del concepto: siendo estudiante de posgrado, describió y analizó programas autorreplicantes y sus implicaciones. Su profesor, Len Adleman, acuñó el término *computer virus* para referirse a estos artefactos. Las primeras pruebas y demostraciones se realizaron sobre sistemas UNIX y, en ocasiones, sobre máquinas como la UNIVAC 1108 en entornos controlados.

3.3 Brain (Pakistani) virus (1986)

Brain —detectado por primera vez en 1986— suele considerarse uno de los primeros virus significativos que afectaron a PC compatibles (IBM PC y compatibles). Originado en Pakistán, Brain se propagó mediante disquetes y modificaba el sector de arranque (boot sector) de los disquetes infectados.

3.3.1 Modo de operación

- Brain alteraba el sector de arranque del disquete para incluir su código; cuando la máquina intentaba arrancar desde un disquete infectado, el código del virus se ejecutaba primero.
- Para ocultar su presencia y dificultar la detección, el virus interceptaba las llamadas de BIOS relacionadas con la lectura del sector de arranque (técnicamente, manipulaba la interrupción `INT 13h`) y, al solicitarse la lectura del sector de arranque, el virus devolvía el sector original no modificado. De este modo, utilidades simples que leían el sector podían ser engañadas y no observar la presencia del virus.
- Como efecto "visible", el virus llegaba a modificar la etiqueta del disco y a reemplazarla por la cadena "(c) Brain", lo que facilitó su identificación y su rastreo a su origen geográfico.

3.3.2 Importancia histórica

Brain es significativo por varias razones: fue ampliamente difundido en la era de los disquetes, mostró técnicas de ocultación tempranas (interceptación de rutinas de lectura del disco) y puso de manifiesto la rapidez con la que el software autorreplicante puede propagarse en soportes físicos. Además, su aparición ayudó a concienciar a la comunidad informática sobre la necesidad de medidas de protección y de buenas prácticas en el intercambio de soportes removibles.

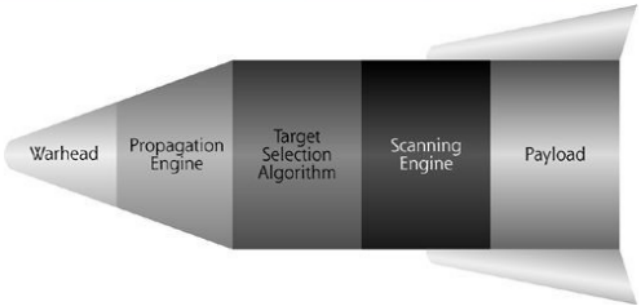
4 Gusanos

Un *gusano* es un programa que se copia a sí mismo de una computadora a otra a través de una red y que, a diferencia de un virus clásico, generalmente **no requiere interacción humana** para propagarse. Su objetivo primario es la replicación y difusión automática; además, muchos gusanos llevan una *carga útil* (*payload*) que realiza acciones adicionales en las máquinas infectadas, como instalar backdoors, participar en ataques DDoS o abrir canales de control.

Características generales

- **Autonomía de propagación:** explotan vectores de red, fallos o configuraciones inseguras para moverse entre hosts sin intervención del usuario.
- **Alta velocidad de difusión:** gracias a la automatización y al escaneo de redes, un gusano puede infectar miles de máquinas en horas o días.
- **Componentes modulares:** muchos gusanos se diseñan en módulos (exploit, motor de propagación, escáner, selector de objetivos, payload), facilitando su adaptación.
- **Impacto en disponibilidad:** incluso sin un payload destructivo, la replicación masiva puede colapsar redes y recursos, causando denegación de servicio.

4.1 Estructura típica



- **Cabeza (exploit):** código que explota una vulnerabilidad concreta en el sistema objetivo.
- **Motor de propagación:** transfiere el resto del gusano a nuevas máquinas, usando FTP, TFTP, HTTP, SMB, etc.
- **Selector de objetivos:** decide qué máquinas atacar.
- **Motor de escaneo:** busca nuevas víctimas según la selección anterior.
- **Payload:** acciones realizadas en la máquina comprometida, como instalar agentes de botnet, abrir backdoors o conectarse a servidores de control.

4.2 Ejemplos históricos

4.2.1 Internet Worm (1988)

- **Objetivos:** máquinas VAX y Sun.
- **Vectores de ataque:** vulnerabilidades en `sendmail`, utilidades de ejecución remota y fallos de configuración.
- **Comportamiento:** intentaba crackear contraseñas y usar `rsh` para propagarse.
- **Impacto operativo:** desconectar del Internet y reiniciar era necesario; se tuvieron que corregir y reinstalar programas críticos.
- **Consecuencias:** deshabilitó miles de sistemas en alrededor de 6 horas, afectando aproximadamente al 10% de los hosts de Internet de la época.
- **Análisis forense:** se desensambló el código para comprender su funcionamiento.

4.2.2 Gusano Nimda (septiembre de 2001)

- Explotaba una vulnerabilidad conocida en IIS (CVE-2000-0884).
- Se propagaba por correo electrónico usando direcciones del equipo infectado.
- Accedía a carpetas compartidas para copiarse.
- Inyectaba código malicioso en páginas web de servidores comprometidos, infectando a clientes que accedían a ellas.
- Buscaba y utilizaba "puertas traseras" dejadas por gusanos anteriores, como Code Red II y sadmind.

4.2.3 Otros gusanos

- **Santy:** explotaba vulnerabilidades en servidores web y se propagaba a través de motores de búsqueda y otras técnicas de descubrimiento masivo.
- **Conficker B (2008):** utilizaba vulnerabilidades en Windows para propagarse lateralmente, implementaba técnicas de evasión sofisticadas y mecanismos de actualización y control complejos.

4.3 Impactos operativos y económicos

- **Sobrecarga de recursos:** uso intensivo de CPU, memoria y ancho de banda.
- **Interrupción de servicios:** caída de servidores y redes.
- **Costes de remediación:** tiempo y recursos para detección, limpieza, parcheo, reinstalación y recuperación.
- **Efecto multiplicador:** gusanos combinados con puertas traseras o vectores adicionales pueden amplificar los daños.

4.4 Prevención y mitigación

- **Parcheo y gestión de vulnerabilidades:** mantener sistemas y servicios actualizados.
- **Reducción de la superficie de ataque:** deshabilitar servicios innecesarios, cerrar puertos y endurecer configuraciones.
- **Segmentación de red y controles de acceso:** VLANs, firewalls internos y filtrado entre segmentos.
- **Detección temprana:** IDS/IPS y monitoreo de patrones inusuales de tráfico.
- **Políticas de correo y filtrado:** bloquear adjuntos o enlaces sospechosos.
- **Control de acceso a recursos compartidos:** autenticación y permisos estrictos.
- **Backups y planes de recuperación:** copias de seguridad probadas y procedimientos de restauración.
- **Aislamiento y contención:** desconectar segmentos afectados y proceder a limpieza controlada.
- **Educación y concienciación:** formar usuarios y administradores sobre vectores comunes y buenas prácticas.
- **Respuesta coordinada:** colaboración con proveedores, CERTs y comunidades para bloquear vectores y compartir indicadores de compromiso.

5 Otros tipos de Malware

5.1 Backdoor

Un *backdoor* es, como su nombre lo indica, una *puerta trasera* que permite un acceso oculto a un sistema, saltándose los mecanismos de control de acceso convencionales. Los backdoors pueden ser instalados de forma intencional por administradores para mantenimiento, o de forma maliciosa por atacantes para obtener acceso persistente sin ser detectados. Permiten ejecutar comandos, modificar archivos o instalar otros programas sin que el usuario o el sistema de seguridad lo adviertan.

5.2 Adware y Spyware

- **Adware:** cualquier programa que automáticamente muestra publicidad al usuario durante su instalación o uso con el fin de generar lucro para sus autores.
- **Spyware:** programa que recopila información sobre una persona u organización sin su consentimiento. La función más común es recopilar información sobre el usuario (sitios web visitados, consultas realizadas) y distribuirla a empresas publicitarias para mostrar publicidad dirigida.

5.3 Keylogger

- Programa o componente que captura las teclas presionadas en un sistema comprometido, recolectando información sensible como claves, PINs, nombres de usuario, etc.
- Generalmente se utiliza para realizar robo de identidad o acceder a información confidencial sin autorización.

5.4 Bombas lógicas

- Un programa que realiza una acción que viola la política de seguridad cuando ocurre un evento externo específico.
- Ejemplo: un programa que borra la nómina de sueldos de una compañía cuando se elimina un registro particular (por ejemplo, el del empleado que insertó la bomba lógica). La intención es causar daño a la empresa si el creador de la bomba deja de estar en el sistema.

5.5 Rootkits

- Un *rootkit* es una herramienta o conjunto de herramientas utilizadas para ocultar procesos y archivos que permiten al intruso mantener acceso al sistema, a menudo con fines maliciosos.
- Oculta inicios de sesión (*logins*), procesos, archivos y registros (*logs*).
- Puede incluir software para interceptar datos provenientes de terminales, conexiones de red (*sniffer*) e incluso del teclado (*keylogger*).

5.6 Ransomware

- Un *ransomware* es un software malintencionado que restringe el acceso a determinadas partes o archivos del sistema infectado, generalmente usando cifrado.
- Solicita un rescate económico a cambio de eliminar la restricción y recuperar el acceso a los datos afectados.

5.7 Password Stealers

Un **password stealer** es un tipo de malware especializado en robar las contraseñas almacenadas en un sistema. Estas credenciales pueden corresponder a cuentas de correo, servicios en la nube, redes sociales, aplicaciones y otros servicios. El objetivo del malware es recopilar estas credenciales y enviarlas a los atacantes, quienes pueden utilizarlas para acceder ilegalmente a las cuentas de los usuarios, robar información personal o realizar actividades maliciosas.

Almacenamiento de claves en el navegador

Muchos password stealers buscan contraseñas almacenadas en navegadores web. Los navegadores modernos ofrecen guardar credenciales automáticamente, pero estas pueden ser vulnerables si un malware accede al perfil del usuario. El riesgo aumenta si no se utiliza un gestor de contraseñas seguro o autenticación multifactor, y si el malware logra extraer cookies o tokens de sesión, puede acceder a cuentas sin necesidad de la contraseña.

6 Técnicas de distribución

- **Correos fraudulentos:** mensajes de phishing que contienen enlaces o archivos adjuntos maliciosos.
- **Búsquedas en Internet:** el usuario puede ser dirigido a sitios web comprometidos o engañosos donde el malware se descarga automáticamente.
- **Descargas y software engañoso:** aplicaciones falsas o extensiones del navegador que ocultan código malicioso.

7 Defensas

Para protegerse de password stealers, se pueden implementar diversas medidas:

- Distinguir malware conocido mediante firmas y patrones.
- Detectar uso indebido de recursos del sistema.
- Realizar análisis dinámico en entornos aislados (*sandbox*).
- Diferenciar entre datos e instrucciones ejecutables.
- Limitar el acceso de los procesos a objetos y archivos críticos.
- Prohibir el *sharing* de recursos entre procesos inseguros.
- Detectar modificaciones no autorizadas en archivos.
- Identificar acciones fuera de las especificaciones del sistema.
- Analizar características estadísticas para reconocer comportamientos anómalos.
- Utilizar *whitelisting* para permitir solo la ejecución de programas confiables.

7.1 Soluciones Antimalware

Los sistemas antimalware combinan detección basada en firmas con análisis heurístico que identifica comportamientos sospechosos:

- Intentos de acceso al sector de booteo.
- Listado masivo de documentos en directorios.
- Modificación de programas ejecutables.
- Borrado de archivos del disco rígido.
- Auto-inserción en el inicio del sistema operativo.
- Uso de empaquetadores para ocultar código malicioso.

7.2 Soluciones EDR (Endpoint Detection and Response)

Las plataformas EDR proporcionan visibilidad y control granular sobre los endpoints para detectar y neutralizar amenazas avanzadas:

- **Recolección de telemetría profunda:** un agente monitorea continuamente procesos, conexiones de red, operaciones de archivos y llamadas a la API.
- **Detección basada en comportamiento:** mediante inteligencia artificial y *machine learning*, identifica tácticas, técnicas y procedimientos (TTPs) maliciosos siguiendo marcos como MITRE ATT&CK. Es eficaz frente a ataques sin archivos (*fileless*), *living-off-the-land* y exploits de día cero.
- **Investigación y *Threat Hunting*:** búsqueda proactiva de indicadores de ataque (IOAs) en datos históricos y en tiempo real.
- **Respuesta rápida y remota:** aislamiento de endpoints, terminación de procesos, eliminación de artefactos y análisis forense remoto para remediación completa.

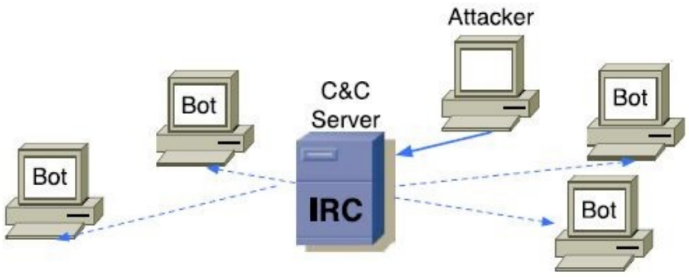
7.3 XDR (Extended Detection and Response)

Mientras que el EDR se enfoca en endpoints como PCs y servidores, el XDR amplía la protección integrando y correlacionando datos de múltiples capas de seguridad: red, nube y correo electrónico. Esto permite tener una visión unificada del ataque y responder de manera más efectiva frente a amenazas complejas.

Punto clave

La protección contra password stealers y otras amenazas de seguridad es un problema complejo. La gran pregunta que enfrentan los sistemas de seguridad es: ¿cómo detectar que lo que el usuario solicita o ejecuta no es realmente lo que él quería hacer? La combinación de defensas proactivas, monitoreo de comportamiento y controles de acceso robustos es esencial para reducir riesgos y proteger la información sensible.

8 Bots y Botnets



Luego de un ataque exitoso contra un equipo, un **bot** (también conocido como zombie o drone) puede ser instalado en el sistema. Este programa habilita un mecanismo de control remoto, permitiendo al atacante ejecutar comandos arbitrarios y tomar control total del sistema víctima.

Una **botnet** es una red de equipos comprometidos que puede ser controlada de forma remota por un atacante. Los usos más comunes de las botnets incluyen:

- Ataques de denegación de servicio distribuido (DDoS)
- Envío masivo de spam
- Sniffing de tráfico de red
- Robo de información sensible
- Registro de teclas (*keylogging*)
- Difusión de nuevo malware
- Instalación de spyware
- Abuso de servicios como Google Adsense
- Manipulación de encuestas online
- Robo de identidad

8.1 Mirai Botnet

Mirai es una botnet que se propaga principalmente a través de dispositivos IoT (Internet of Things), como cámaras, routers y grabadoras de video. Aprovecha credenciales por defecto o débiles para infectar dispositivos, convirtiéndolos en "bots" controlados de manera remota por los atacantes. La botnet se ha usado para ejecutar ataques DDoS de gran magnitud.

8.2 Ransomware WannaCry

WannaCry es un ransomware que se difundió globalmente en mayo de 2017, explotando la vulnerabilidad SMBv1 de Windows conocida como EternalBlue. Una vez infectado un equipo, cifra los archivos del usuario y exige un rescate en Bitcoin para restaurar el acceso. Su rápida propagación se debió a su capacidad de auto-replicarse en redes locales.

8.3 Miners

Los *miners* o criptominers son programas maliciosos que utilizan la potencia de cómputo de los dispositivos infectados para minar criptomonedas sin el consentimiento del propietario. Esto puede ralentizar gravemente los sistemas afectados y aumentar el consumo eléctrico de manera significativa.

8.4 Arc Browser – mayo 2024

Se reportaron vulnerabilidades y exploits en versiones específicas del navegador Arc en mayo de 2024, que podían ser aprovechados por malware para comprometer sesiones de usuario, instalar extensiones maliciosas o robar información de navegación.

8.5 AVGater

AVGater es un malware que combina funcionalidades de troyano y backdoor. Su objetivo principal es robar información sensible y proporcionar acceso remoto al atacante. Se propaga a través de descargas engañosas y archivos adjuntos en correos electrónicos fraudulentos.

8.6 Packers y Joiners

- **Packer:** Compresor de ejecutables que permite la descompresión en tiempo real cuando el programa se ejecuta. Si el packer no es reconocido por un antivirus, puede engañarlo y evitar la detección. Ejemplo: UPX.
- **Joiner:** Programa que permite unir dos o más archivos generando un único .EXE. Son comúnmente usados para insertar un troyano en un archivo aparentemente inofensivo, como una tarjeta de saludo o un juego, que luego se comparte por correo o redes P2P.

8.7 Emotet

Emotet es un malware que se distribuye principalmente por correos electrónicos de phishing. Originalmente un *banking trojan*, evolucionó para incluir funcionalidades de *loader*, descargando otros malware en sistemas comprometidos. Más información oficial: https://www.us-cert.gov/ncas/alerts/TA18-201A.

9 Servicios de análisis

9.1 VirusTotal

VirusTotal es un servicio gratuito en línea que analiza archivos y URLs sospechosas para detectar malware y otras amenazas. Utiliza más de 70 escáneres de antivirus y servicios de listas negras de dominios/URL, incluyendo análisis de comportamiento en sandbox.

9.2 Hybrid Analysis

Hybrid Analysis es un servicio gratuito especializado en el análisis profundo de archivos y URLs sospechosas. Su principal característica es el uso de una tecnología de *sandbox*, un entorno virtual aislado donde el malware se ejecuta de forma segura para observar su comportamiento sin riesgo.

Capacidades clave

- **Análisis Dinámico (Sandbox):** Observa en tiempo real qué hace un archivo: procesos creados, sitios a los que se conecta, modificaciones al registro y archivos alterados.
- **Análisis Estático:** Examina el código del archivo sin ejecutarlo para extraer metadatos, cadenas de texto e indicadores de compromiso (IOCs) potenciales.
- **Informes de Comportamiento:** Genera reportes detallados con capturas de pantalla, listado de acciones maliciosas y mapeo con tácticas y técnicas de la matriz MITRE ATT&CK®.
- **Extracción de Indicadores:** Identifica y extrae automáticamente indicadores de compromiso como hashes de archivos, direcciones IP, dominios y mutaciones del registro, cruciales para la respuesta a incidentes y la caza de amenazas.