

Trabajo de Fin de Máster

Tu Nombre

16 de agosto de 2025

Índice general

1. Alcance	3
2. Introducción	4
2.1. Reconocimiento	4
2.2. Weaponization	4
2.3. Entrega	4
2.4. Explotación	4
2.5. Instalación	5
2.6. Command and Control (C2)	5
2.7. Acciones sobre el objetivo	5
3. Malware	6
3.1. Loader	6
3.1.1. DLL Manipulation	7
3.1.2. Process Injection	7
3.2. Shellcode	7
3.3. Packer	7
4. EDR Bypass	8
4.1. Como funcionan los EDRs	8
4.1.1. Kernell callbacks	8
4.1.2. Event tracing for Windows (ETW)	8
4.1.3. Antimalware scan interface (AMSI)	8
4.1.4. System call monitoring	8
4.1.5. Monitorizacion de Memoria	9
4.2. BYOVD	9
4.3. Syscalls	9
4.3.1. Direct Syscall	9
4.3.2. Indirect Syscall	9
4.4. Call Stack Spoofing	9

5. Anexo	10
5.1. Tipos de Malware	10
5.2. Proceso	10
5.2.1. Secciones de un proceso	10
5.3. Ring level	11
5.4. Memoria Virtual	11
5.4.1. Problemas Memoria Fisica	11
5.4.2. Acceso indirecto a memoria	12
5.4.3. Paginación	12
5.4.4. Tabla de paginas	12
5.4.5. MMU	12
5.4.6. Acceso a memoria	12
5.5. Win32 API	13
5.6. SO Security	14
5.6.1. ASLR (Address Space Layout Randomization)	14
5.6.2. DEP (Data Execution Prevention)	14
5.6.3. Control Flow Guard (CFG)	14
5.6.4. Code Integrity Guard (CIG)	14
5.6.5. Protected Process Light (PPL)	14
5.6.6. Virtualization-Based Security (VBS)	14
5.6.7. Kernel-mode code signing (KMCI)	14
5.7. Genealogía de Procesos en Windows	14
5.8. Listado de EDRs	17

Capítulo 1

Alcance

Contenido: Aquí se presenta la idea del trabajo y qué se pretende conseguir.

El objetivo de este trabajo es desarrollar un escenario didáctico que permita a los investigadores de malware mejorar sus capacidades de análisis mediante la comprensión del proceso completo de creación de una malware. El resultado final del proyecto será un volcado de memoria de un sistema infectado, el cual servirá como base para ejercicios de análisis forense (blue team).

Para generar dicho recurso, el núcleo del trabajo se centrará en la concepción, diseño y desarrollo de un malware realista, su ejecución controlada en un sistema víctima, y la documentación íntegra del proceso ofensivo llevado a cabo por el red team. De este modo, el participante que reciba el volcado podrá analizar la infección desde el punto de vista del analista forense, pero también interpretar las acciones del atacante, entendiendo mejor el cómo y el por qué de cada artefacto encontrado en memoria.

En definitiva, este trabajo busca aproximar al investigador a la mentalidad del creador de malware, con el fin de reforzar sus habilidades de detección, interpretación y respuesta ante amenazas reales.

Capítulo 2

Introducción

Contenido: Esta sección dará un vistazo general a un ataque típico apoyándose en el Cyber Kill Chain de Lockheed Martin y el modelo MITRE ATTACK.

2.1. Reconocimiento

Recopilando información sobre la víctima

2.2. Weaponization

Se desarrolla un payload malicioso

2.3. Entrega

Como se entrega el payload a la víctima

Vectores típicos: phishing, vulnerabilidades sin parchear, credenciales expuestas.

El objetivo aquí es hacer llegar el malware o el loader al sistema de la víctima.

2.4. Explotación

Como conseguir ejecución del payload en la víctima

Una vez entregado, se consigue ejecutar código en el sistema.

Puede ser directo (el ransomware se ejecuta) o mediante un loader que se encarga de preparar el entorno y luego lanzar el malware.

2.5. Instalación

El malware se instala en el sistema de la víctima. El loader instala el ransomware, a menudo asegurando persistencia, o simplemente prepara el entorno para que el malware se ejecute correctamente.

2.6. Command and Control (C2)

El malware establece comunicación con el atacante para recibir instrucciones.

2.7. Acciones sobre el objetivo

El atacante realiza su objetivo final - exfiltracion - encriptacion - cripto-minero

Capítulo 3

Malware

Contenido: En esta sección se explicará todo lo relacionado con la creación de un malware

Un malware es un software diseñado para realizar acciones maliciosas en un sistema.

En este trabajo, el malware que desarrollaremos tendrá como objetivo encriptar todo el disco de la víctima.

El primer desafío consiste en conseguir ejecutar código en el sistema de la víctima. Las formas más habituales de lograrlo incluyen: phishing, explotación de vulnerabilidades sin parchear o uso de credenciales expuestas. Por ejemplo, si se dispone de credenciales que permiten acceder a una VPN, podríamos ejecutar código de manera remota.

El segundo desafío es garantizar que el malware no sea bloqueado por un EDR. Para ello, desarrollaremos un loader, un componente encargado de preparar el entorno y lanzar el malware de manera segura y silenciosa.

3.1. Loader

1. Carga en memoria (In-memory execution) 2. Inyección en procesos legítimos (Process Injection) 3. Evasión de defensas 4. Persistencia 5. Descarga o desempaquetado de payloads

3.1.1. DLL Manipulation

Phantom DLL Injection

DLL reflection

3.1.2. Process Injection

inyeccion de proceso verdadera seria desde un proceso de espacio de usuario a otro proceso de espacio de usuario objetivo benigno
por otro lado esta el process spawning y hollowing

clonando procesos (Dirty Vanity)

Para realizar process hollowing, se tiene que llamar a 5 syscalls (muy ruidoso) Pero si se llaman a 3 desde un proceso y dos desde su hijo, los edrs no detectaban esa actividad

3.2. Shellcode

3.3. Packer

Capítulo 4

EDR Bypass

Contenido: En esta sección se como los EDRs monitorizan el sistema y cómo se pueden evadir.

4.1. Como funcionan los EDRs

4.1.1. Kernell callbacks

El edr mantendra un driver en kernel land

El edr recibira callbacks segun eventos de (Procesos, Hilos, DLLs)

4.1.2. Event tracing for Windows (ETW)

Registro de eventos recopila y analizar información sobre el funcionamiento del sistema operativo y las aplicaciones.

4.1.3. Antimalware scan interface (AMSI)

AMSI es una API genérica de escaneo antimalware que permite que las aplicaciones (por ejemplo, PowerShell, WSH, VBScript, Office macros...) envíen contenido potencialmente malicioso a un proveedor antivirus registrado para su análisis antes de ser ejecutado.

4.1.4. System call monitoring

El edr pone hooks en las llamadas de win32api a syscalls pone un jmp y donde deberia saltar a la syscall salta al edr de modo que lo analiza antes de que se ejecute

4.1.5. Monitorizacion de Memoria

yara

4.2. BYOVD

4.3. Syscalls

4.3.1. Direct Syscall

Otra opcion para pasar bypassear los hooks del edr es no utilizar ntdll.dll y llamar directamente a la syscall

el problema es que hace falta el SSN y es un valor arbitrario que cambia muy frecuentemente entre builds de windows

la primera solucion de esto fue "hells gate" que cargaba ntdll.dll desde disco y de ahi recuperaba el SSN (muy ruidoso para los edr)

Versiones mas modernas: SysWhispers2

4.3.2. Indirect Syscall

Aun no se como funciona

4.4. Call Stack Spoofing

Capítulo 5

Anexo

5.1. Tipos de Malware

RAT ransomware reverse shell

5.2. Proceso

Cada proceso tiene su propio espacio de direcciones virtual, Dentro de ese espacio un proceso esta dividido en secciones,

5.2.1. Secciones de un proceso

Listado en orden de mayor direcciones de memoria a menor:

Paginas del Kernel

Esta seccion ocupa lo mismo en todos los procesos del sistema, y corresponde a todas las paginas que contienen el kernel del sistema operativo, Como todas las paginas de kernel de todos los procesos hacen referencia a los mismos frames de memoria fisica, no se desperdicia espacio cargando el kernel en cada proceso,

Para acceder a la memoria del kernel, un proceso tiene que hacer una llamada al sistema,

Stack - lectura y escritura

Almacenamiento simple LIFO Crece hacia abajo, desde el final del espacio de direcciones del proceso hasta el final de la BSS,

Cada funcion tiene su propio stack frame, que contiene las variables locales y los argumentos de la funcion,

Limitacion tipica entre 1 y 8MB, dependiendo de la arquitectura y el sistema operativo,

Utiliza el puntero de pila para reservar y liberar espacio en el stack (muy rapido)

Heap - lectura y escritura

Malloc, reserva memoria de forma dinamica Crece hacia arriba, desde el final de la BSS hasta el final del espacio de direcciones del proceso

Utiliza el puntero de heap para reservar y liberar espacio en el heap (más lento que el stack)

BSS y Data- lectura y escritura

Variables estaticas y globales del programa,

Text - lectura y ejecución

Es cargado desde el ejecutable binario desde disco

5.3. Ring level

user mode vs kernel mode

idea: pillar un programa como print hello world

desde proc mon ver todas las llamadas que hace e ir comentando cada biblioteca en que nivel esta

5.4. Memoria Virtual

5.4.1. Problemas Memoria Fisica

- Escasez de memoria
- Fragmentación
- Acceso a memoria (Seguridad)

5.4.2. Acceso indirecto a memoria

Cada proceso tiene su propio espacio de direcciones virtual, Para el proceso es como tener un espacio de memoria física contiguo reservado, Pero por detras el SO esta mapeando esas direcciones virtuales a distintas regiones de memoria física.

5.4.3. Paginación

Mapear cada byte de la memoria virtual a un byte de la memoria física seria muy ineficiente, Se crean paginas de memoria que son bloques de bytes para simplificar este proceso (normalmente de 4KB | 4096 bytes),

page - memoria virtual frame - memoria física

Cuando un proceso se carga sus paginas son cargadas en frames disponibles en memoria física

5.4.4. Tabla de paginas

Cada proceso tiene una tabla de paginas que mapea las direcciones virtuales a las físicas, La tabla de paginas es gestionada por la MMU (Memory Management Unit), Para cada pagina se mantiene, su numero de frame, permisos, etc.

5.4.5. MMU

La CPU tiene que acceder a la RAM primero para consultar la tabla de paginas y Despues para acceder a la memoria física, para cada lectura tenemos que hacer 2 accesos a memoria, Para optimizar esto, la MMU, hardware específico de la CPU, se encarga de traducir las direcciones virtuales a físicas, la MMU tiene un caché de la tabla de paginas llamado TLB (Translation Lookaside Buffer), La MMU tambien es la encargada de gestion que un proceso no acceda a memoria que no le pertenece,

5.4.6. Acceso a memoria

Cuando un proceso quiere acceder a una dirección de memoria, la CPU consulta la MMU, En la MMU la dirección se divide en dos partes, el numero de pagina y el offset dentro de la pagina,

Page number - los bits mas significativos de la dirección virtual
Page offset - los bits menos significativos de la dirección virtual

Ejemplo: En una pagina de 4KB (4096 bytes), el offset ocupa los 12 bits menos significativos, El resto de bits son los mas significativos y representan el numero de pagina,

5.5. Win32 API

Nombre de la DLL	Tareas de la DLL
User32.dll	Esta biblioteca contiene funciones para crear ventanas, manejar mensajes y procesar la entrada del usuario.
Kernel32.dll	Esta biblioteca proporciona acceso a una variedad de servicios esenciales del sistema como la gestión de memoria, operaciones de E/S y la creación de procesos e hilos.
Gdi32.dll	Esta biblioteca contiene funciones para dibujar gráficos y mostrar texto.
Comdlg32.dll	Esta biblioteca proporciona diálogos comunes como los diálogos de abrir y guardar.
Advapi32.dll	Esta biblioteca proporciona funciones para trabajar con el registro de Windows y gestionar cuentas de usuario.

Cuadro 5.1: Descripción de las principales DLLs de la Win32 API

5.6. SO Security

5.6.1. ASLR (Address Space Layout Randomization)

5.6.2. DEP (Data Execution Prevention)

5.6.3. Control Flow Guard (CFG)

5.6.4. Code Integrity Guard (CIG)

5.6.5. Protected Process Light (PPL)

5.6.6. Virtualization-Based Security (VBS)

5.6.7. Kernel-mode code signing (KMCI)

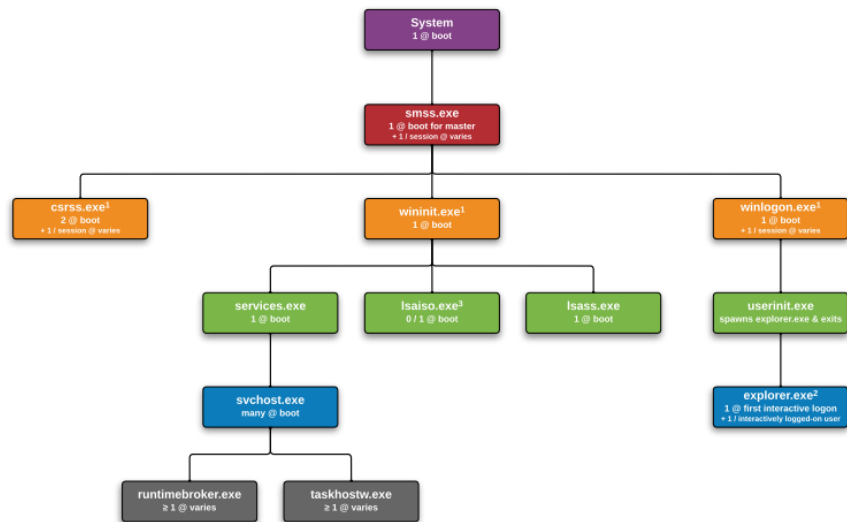
5.7. Genealogía de Procesos en Windows

La genealogía de procesos en Windows representa la jerarquía de creación de procesos desde el arranque del sistema hasta el inicio de sesión del usuario. Comprender esta estructura es esencial para el análisis forense, la respuesta a incidentes y la detección de malware, ya que permite identificar comportamientos anómalos en los procesos del sistema.

A continuación se muestra la genealogía de procesos típica en Windows:

Windows Process Genealogy

youtube.com/13cubed



Fuente: youtube.com/13cubed

Nivel 0: Proceso raíz

- **System:** Primer proceso de espacio de usuario iniciado por el kernel. Tiene un PID fijo (normalmente 4) y no tiene padre.

Nivel 1: Session Manager

- **smss.exe (Session Manager Subsystem):** Primer proceso real del espacio de usuario. Se encarga de iniciar las sesiones del sistema, lanzar procesos críticos como **csrss.exe**, **wininit.exe** y **winlogon.exe**.

Nivel 2: Procesos críticos del sistema

- **csrss.exe:** Client/Server Runtime Subsystem. Se encarga de funciones esenciales como la gestión de la consola y la creación de procesos. Existe una instancia por sesión.
- **wininit.exe:** Windows Initialization Process. Lanza procesos esenciales como **services.exe**, **lsaiso.exe** y **lsass.exe**.

- **winlogon.exe**: Gestiona la autenticación del usuario y se mantiene activo durante la sesión interactiva.

Nivel 3: Procesos del sistema

- **services.exe**: Service Control Manager. Se encarga de iniciar y gestionar los servicios del sistema, incluyendo los alojados por **svchost.exe**.
- **lsaiso.exe**: Proceso de seguridad aislado que implementa funciones de cifrado y autenticación en versiones modernas de Windows (opcional).
- **lsass.exe**: Local Security Authority Subsystem Service. Encargado de políticas de seguridad, autenticación y gestión de credenciales.

Nivel 4: Servicios alojados y utilidades del sistema

- **svchost.exe**: Proceso contenedor que aloja múltiples servicios del sistema. Existen muchas instancias según el grupo de servicios que aloje.
- **runtimebroker.exe** / **taskhostw.exe**: Procesos auxiliares para la ejecución de aplicaciones y tareas programadas.

Procesos del usuario interactivo

- **userinit.exe**: Iniciado por **winlogon.exe** tras la autenticación. Lanza el shell principal del usuario y termina.
- **explorer.exe**: Shell gráfico de Windows que representa el escritorio, la barra de tareas y el menú de inicio. Es el proceso raíz del entorno del usuario.

Importancia en Ciberseguridad

Comprender esta estructura es fundamental para:

- Identificar procesos anómalos o fuera de lugar.
- Detectar técnicas de evasión como *Process Injection* o *Parent PID Spoofing*.

- Realizar análisis forense de procesos mediante herramientas como Sysmon, EDR, Process Explorer, o Zeek.

5.8. Listado de EDRs

Sylantstrike puede ser utilizado para hacer pruebas de concepto de EDRs

- CrowdStrike Falcon
- Microsoft Defender for Endpoint