Taller Forense Avanzado de Windows

OpenWebinars

¡BIENVENIDOS!

Soy Jordi Ubach Manau

Me dedico a Ciberseguridad & Forensic

https://www.linkedin.com/in/jordi-ubach-9971a1a5/

ÍNDICE

- Capturar Estructura MAC de ficheros y archivos
- Capturar contraseñas a recursos de red
- Capturar contraseñas desde navegadores
- Capturar dispositivos usb que se hayan conectado

- Captura de la configuración del firewall
- Captura de posibles Registry Spawings
- Captura de listado de aplicaciones instaladas
- Uso de volatility

Capturar Estructura MAC de ficheros y archivos

- Nos servirá para comprobar la integridad de los archivos y tener una lista ordenada por MAC

A la raíz C:

dir /t:w /a /s /o:d > arbolXfecha_Mod (t=tipo, w=fechamod,o=order,d=fechyhora)

dir /t:a /a /s /o:d > arbolXfecha_Ultacces (t=tipo, a=ultimoacceso,o=order,d=fechyhora)

dir /t:c /a /s /o:d > arbolXfecha_creac(t=tipo, c=creacion,o=order,d=fechyhora)

TOTALCDM F3 view (atributos, etc)

Capturar contraseñas a recursos de red (Netpass)

- Debemos capturar la evidencia de donde se ha conectado con usuario y contraseña. Depende mucho como está configurada la masterpassword, integración en dominio, etc.

ejecutamos Netpass como administrador

Guardamos Contrarecred.csv + hash

Verlo con el view

Capturar contraseñas desde navegadores (WebBrowserPassView)

- Depende de cómo esté configurado el navegador de forma predeterminada. Captura contraseñas de varios navegadores.
- Ejecutar Webbrowserpassview como administrador
- Nos muestra varias columnas de datos.
- Si observamos campos encriptados, posible malware chats, etc

Capturar dispositivos usb que se hayan conectado (Usbhistory)

- Auditar los usb conectados es básico, debido a que la mayoría de incidentes de fuga de información se producen a través de estos dispositivos.

- Debemos pensar que hay “mouse”, que llevan una tarjeta incorporada es reconocible por el sistema como un mouse, pero aparte crea una unidad.

- En Win 2003 xp, existía un setupapi.log, pero en posteriores versiones está en la carpeta Info.
- La información de los dispositivos en Microsoft está plagada por todo el sistema.
- Desde TOTALCMD vamos al fichero Usbhistory
- seleccionamos y abrimos ventana de DOS

Captura de la configuración del firewall

- Es importante resaltar que un atacante seguramente realizará tareas de HouseKeeping (querrá volver a entrar en el sistema) y para esto necesita comunicación mediante tcpip, para abrir el puerto se debe realizar desde el firewall de windows.
- Por eso es muy importante capturar la rama del registro que guarda la configuración. “FirewallRules” del registro.

  Equipo\HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\SharedAccess\Parameters\FirewallPolicy\FirewallRules

  Una vez localizada la rama, “botón derecho” → exportar “firewallrules.reg”

  ¡La pasamos a TOTALCMD y seleccionamos, OJO!! no doble click

  F3 view

  Siempre auditar la ultimas entradas

  Guardamos el fichero y HASH

Captura de posibles Registry Spawings

- Windows asigna a ciertos tipos de extensiones un programa por defecto, algunos creadores de virus, modifican la técnica para ejecutar ciertos componentes de troyano, ejemplo: doc-⟶ otras extensiones.
- Debemos auditar todas las extensiones posibles.

  HKEY_CLASSES_ROOT\txtfile\shell\open\command

  revisamos varias extensiones

  Creamos un fichero de texto copiar-pegar

  guardamos + hash

Captura de listado de aplicaciones instaladas

- Debemos tener copia del registro de las aplicaciones instaladas.
- Además de la ubicación nos dice un dato muy importante, La página web del desarrollador, en caso de ser malicioso, el campo name estará en blanco, pero si tendremos la dirección url.

Equipo\HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall

Guardamos el fichero + HASH

Uso de volatility

Descargar volatility

1.- volatiXXXXX imageinfo –f <Imagen.raw>(image)

2.- volatiXXXXX –profile=<suggest> pslist -f <detination memory>(procesos)

3.- volatiXXXXX –profile=<suggest> kdbgscan -f <detination memory> (kernel debug scan)

4.- volatiXXXXX –profile=<suggest> kpcrscan -f <detination memory> (kernel procesor region scan)

5.- volatiXXXXX –profile=<suggest> psscan -f <detination memory> (list services)

6.- volatiXXXXX –profile=<suggest> dlllist -f <detination memory> (listar dll)

7.- volatiXXXXX –profile=<suggest> dlllist -p 2484 -f <detination memory> (listar dll por proceso)

8.- volatiXXXXX –profile=<suggest> dlldump -D <Destination Directory> -f <destination memory> (volcado de DLL, creamos antes un directorio)

- Podemos ver las dll dentro del directorio creado:
- Incluso podemos volcar archivos DLL de procesos específicos si descubrimos que puede haberse ejecutado un proceso malicioso. Del mismo modo, podemos volcar archivos DLL de un proceso oculto utilizando su dirección de desplazamiento como se muestra a continuación.
- Aquí hay una lista de todos los procesos ocultos una vez más. Ahora hemos utilizado la dirección de desplazamiento para smss.exe, que es 0x024f1020, y volcamos las DLL en la carpeta llamada Oculto.

9.- volatiXXXXX –profile=<suggest> pstree -f <detination memory> (procesos por árbol)

10.- volatiXXXXX –profile=<suggest> consoles -f <detination memory>(consola)

Este complemento se utiliza para encontrar los diversos comandos escritos localmente o remotamente a través de puertas traseras.

11.- volatiXXXXX –profile=<suggest> hivescan -f <detination memory> (direcciones físicas de las colmenas de memoria)

12.- volatiXXXXX –profile=<suggest> hivelist -f <detination memory> (direcciones virtuales de las colmenas de memoria)

13.- volatiXXXXX –profile=<suggest> svcscan -f <detination memory> (servicios en funcionamiento)

14.- volatiXXXXX –profile=<suggest> handles -f <detination memory>(Manejadores)

- Para mostrar los identificadores abiertos en un proceso, use el comando de identificadores. Esto se aplica a archivos, claves de registro, mutexes, canalizaciones con nombre, eventos, estaciones de ventana, escritorios, subprocesos y todos los demás tipos de objetos ejecutivos asegurables.

15.- volatiXXXXX –profile=<suggest> getsids -f <detination memory> (Secure id)

- Para ver los SID (identificadores de seguridad) asociados con un proceso, sirve para identificar procesos que han escalado maliciosamente los privilegios.

16 .- volatiXXXXX –profile=<suggest> cmdscan -f <detination memory>(console escáner)

- El complemento cmdscan busca en la memoria de csrss.exe en XP / 2003 / Vista / 2008 y conhost.exe en Windows 7 los comandos que los atacantes ingresaron a través de una consola (cmd.exe).

17 .- volatiXXXXX –profile=<suggest> envars -f <detination memory>(variables de entorno de proceso)

18 .- volatiXXXXX –profile=<suggest> verinfo -f <detination memory>(Archivos PE)

- Para mostrar la información de la versión incrustada en los archivos PE, use el comando verinfo. No todos los archivos PE tienen información de versión, y muchos autores de malware falsifican que incluya datos falsos, pero este comando puede ser muy útil para identificar binarios y hacer correlaciones con otros archivos.

19 .- volatiXXXXX –profile=<suggest> memmap -f <detination memory>(mapa de memoria)

- El comando memmap le muestra exactamente qué páginas residen en la memoria, dado un proceso específico DTB (o kernel DTB si usa este complemento en el proceso inactivo o del sistema). Le muestra la dirección virtual de la página, el desplazamiento físico correspondiente de la página y el tamaño de la página. La información de mapa generada por este complemento proviene del método get_available_addresses del espacio de direcciones subyacente.

20 .- volatiXXXXX –profile=<suggest> vadwalk -f <detination memory>(procesos de los drivers)

21 .- volatiXXXXX –profile=<suggest> vadtree -f <detination memory>(procesos de los drivers en árbol)

22 .- volatiXXXXX –profile=<suggest> modules -f <detination memory>(módulos cargados)

- Esto recorre la lista doblemente vinculada de estructuras LDR_DATA_TABLE_ENTRY a las que apunta PsLoadedModuleList. Similar al comando CommandReference21 # pslist, esto se basa en encontrar la estructura KDBG.

23 .- volatiXXXXX –profile=<suggest> modscan -f <detination memory>(escaneo de módulos)

- El comando modscan encuentra estructuras LDR_DATA_TABLE_ENTRY escaneando la memoria física en busca de etiquetas de grupo. Esto puede recoger controladores descargados previamente y controladores que han sido ocultos / desvinculados por rootkits. A diferencia de CommandReference21 # modlist, el orden de los resultados no tiene relación con el orden en que se cargaron los controladores. Como puede ver a continuación, DumpIt.sys se encontró en el desplazamiento físico más bajo, pero probablemente fue uno de los últimos controladores en cargarse (ya que se utilizó para adquirir memoria)

24 .- volatiXXXXX –profile=<suggest> moddump -D <directorio_creado/> -f <detination memory>(drivers)

- Para extraer un controlador de kernel a un archivo, use el comando moddump. Proporcione el directorio de salida con -D o --dump-dir = DIR. Sin ningún parámetro adicional, todos los controladores identificados por CommandReference # 21modlist serán volcados. Si desea un controlador específico, proporcione una expresión regular del nombre del controlador con --regex = REGEX o la dirección base del módulo con --base = BASE.

25 .- volatiXXXXX –profile=<suggest> ssdt -f <detination memory>(SQL server data tools)

- Para enumerar las funciones en los SSDT nativos y de GUI, use el comando ssdt. Esto muestra el índice, el nombre de la función y el controlador propietario para cada entrada en el SSDT.

26 .- volatiXXXXX –profile=<suggest> driverscan -f <detination memory>(driver scan)

- Para encontrar DRIVER_OBJECT en la memoria física mediante el escaneo de etiquetas de grupo. Esta es otra forma de ubicar los módulos del núcleo, aunque no todos los módulos del núcleo tienen un DRIVER_OBJECT asociado. DRIVER_OBJECT es lo que contiene las 28 tablas IRP (función principal), por lo tanto, el comando CommandReference21 # driverirp se basa en la metodología utilizada por driverscan.