Universitat Oberta de Catalunya
uoc.edu

Hacking the running Linux kernel

Leoncio Armas Castro
Máster Universitario en Seguridad de las Tecnologías de la Información y de las Comunicaciones
Seguridad empresarial

Víctor Méndez Muñoz
Víctor García Font

02/06/2020

FICHA DEL TRABAJO FINAL

Título del trabajo:	*Hacking the running Linux kernel*
Nombre del autor:	*Leoncio Armas Castro*
Nombre del consultor/a:	*Víctor Méndez Muñoz*
Nombre del PRA:	*Víctor García Font*
Fecha de entrega (mm/aaaa):	06/2020
Titulación:	*Máster Universitario en Seguridad de las Tecnologías de la Información y de las Comunicaciones*
Área del Trabajo Final:	*Seguridad empresarial*
Idioma del trabajo:	*Castellano*
Palabras clave	*Memoria, Linux, Kernel*

Resumen del Trabajo (máximo 250 palabras): *Con la finalidad, contexto de aplicación, metodología, resultados i conclusiones del trabajo.*

Linux es el kernel más utilizado en los sistemas empotrados, dispositivos cuya disponibilidad pública convierte al kernel en el punto crítico de muchos de los sitemas de seguridad integrada (validación de firmware, passwords, semillas de cifrado, raices hardware, etc). Diferentes setup de placa son susceptibles ante ataques de lectura de memoria RAM en caliente, ya sea en bus SCI (ataques JTAG tipo BusPirate) o buses de RAM conectados a la CPU (ataques con electrónica Logic Analizer Module).

En el presente trabajo, se analizarán las diferentes herramientas y técnicas utilizados en el análisis de volcados de memoria. Estas herramientas son muy utilizadas para saber que se estaban ejecutando en el dispositivo al momento de la captura de información.

Posteriormente, para comparar la utilidad de las herramientas revisadas y haciendo uso de una muestra de volcado de memoria, se obtendrá información sobre el dispositivo de donde se obtuvo la muestra.

Abstract (in English, 250 words or less):

Linux is the most used kernel in embedded systems, devices whose public availability makes the kernel the critical point of many of the integrated security systems (firmware validation, passwords, encryption seeds, hardware roots, etc.). Different board setups are susceptible to hot RAM read attacks, either on SCI bus (JTAG BusPirate type attacks) or RAM buses connected to the CPU (attacks with electronic Logic Analyzer Module).

In this paper, the different tools and techniques used in the analysis of memory dumps will be analyzed. These tools are widely used to know that they were running on the device at the time of information capture.

Subsequently, to evaluate the different tools using a memory dump sample, information will be obtained on the device from which the sample was obtained.

Índice

Lista de figuras

Lista de tablas

1. Introducción

1.1 Contexto y justificación del Trabajo

Actualmente, con la expansión de las redes de comunicaciones y el uso de Internet, aparecen diversos dispositivos autónomos (IoT) que se conectan a las redes para consumir/compartir información.

En los últimos años, se observa un incremento en el uso de los dispositivos IoT (Internet of Things), los cuales son objetos dotados con cierto software y conexión a internet con cierta autonomía para el procesamiento de información y que pueden ser controlados o monitoreados remotamente, las aplicaciones para estos dispositivos pueden ser de consumo (como la domótica para el control del hogar, los wearables para llevarlos puesto o incorporado en la ropa), empresarial (permite la automatización industrial, uso de sensores, actuadores) y de infraestructura (seguimiento y control a vías férreas, puentes, etc).

Los dispositivos IoT presentan deficiencias en el manejo de las credenciales (contraseñas, certificados, tokens, etc) y en el cifrado de los datos, estas vulnerabilidades generan problemas para la seguridad del dispositivo y las comunicaciones, que fácilmente pueden ser aprovechados por terceros para el robo de información o el monitoreo sin consentimiento.

El presente trabajo de fin de máster tiene por finalidad la revisión de las herramientas y técnicas que existen para el análisis de volcados de memoria muy utilizados en el análisis forense, análisis de debilidades en seguridad, detección de intrusos, protección de habilidades, protección de datos sensibles (dispositivos médicos), en general para realizar un análisis en la seguridad del dispositivo IoT.

Una vez revisadas las herramientas y técnicas se procederá a aplicarlas en una muestra de volcado de memoria para comprobar su uso y eficacia en la obtención de datos relevantes sobre el dispositivo del cual se obtuvo la muestra y analizar si existe algún indicio de actividades sospechosas.

1.2 Objetivos del Trabajo

El objetivo del presente trabajo es el estudio de las diversas herramientas y técnicas que existen para realizar el análisis de volcados de memoria, para alcanzar el objetivo se revisarán y analizarán las literaturas académicas que existen a fin de poder documentar las características, las opciones y los modos de uso de las herramientas.

Previamente se explicarán algunos conceptos sobre los dispositivos IoT, la memoria RAM y algunos comandos básicos para la gestión del sistema operativo Linux.

Posteriormente, se aplicarán las herramientas estudiadas sobre una muestra de volcado de memoria Linux, demostrando los modos de uso de las diversas opciones que presentan dichas herramientas.

A modo de resumen, los objetivos del TFM serán:

Objetivos de estudio

- Concepto y descripción de los dispositivos IoT.
- Concepto y descripción sobre las memorias RAM.
- Descripción de comandos para gestión del sistema operativo Linux.
- Revisión de las herramientas y técnicas para el análisis de volcados de memoria.

Objetivos de análisis

- Poner a prueba las herramientas estudiadas para realizar el análisis de una muestra de volcado de memoria del sistema Linux.
- Documentar la información obtenida de las pruebas realizadas.

1.3 Enfoque y método seguido

El enfoque del proyecto consiste en la revisión de la literatura para el estudio de las herramientas y técnicas que existen para realizar el análisis de volcados de memoria, posteriormente se aplicarán las herramientas estudiadas a una muestra de volcado de memoria para luego obtener conclusiones de los análisis comparativos a los resultados obtenidos.

El proyecto se divide en dos partes:

- La primera parte consiste en realizar una búsqueda y posterior estudio de las herramientas y técnicas que existen para poder realizar el análisis de volcado de memoria.

- La segunda parte consiste en aplicar las herramientas y técnicas estudiadas en la primera parte a una muestra de volcado de memoria del sistema Linux, para seguidamente mostrar los resultados obtenidos.

Para conseguir los objetivos, el trabajo se divide en cuatro entregas: la primera es la elaboración del plan de trabajo, la segunda consta en la investigación y estudio teórico de las herramientas, la tercera consta de los resultados del análisis de volcado de memoria realizado con las herramientas estudiadas y la cuarta consta en la entrega de la memoria final del trabajo realizado.

1.4 Planificación del Trabajo

A continuación, se presenta la planificación temporal del Trabajo.

Item	Actividad	Inicio	Fin	Duración
0	Trabajo Fin de Máster (TFM)	19/02/2020	09/06/2020	112 días
1	Definición y planificación del TFM	19/02/2020	03/03/2020	14 días
1.1	Definición del trabajo a realizar	19/02/2020	20/02/2020	02 días
1.2	Contexto y justificación	21/02/2020	22/02/2020	02 días
1.3	Enfoque y método	23/02/2020	25/02/2020	03 días
1.4	Elaboración de la planificación	26/02/2020	27/02/2020	02 días
1.5	Estado del arte	28/02/2020	29/02/2020	02 días
1.6	Resumen de los capítulos	01/03/2020	02/03/2020	02 días
1.7	Resumen del trabajo	03/03/2020	03/03/2020	01 día
1.8	**Entrega 1: Plan de trabajo**	**03/03/2020**	**03/03/2020**	**Hito**
2	Investigación y estudio	04/03/2020	31/03/2020	28 días
2.1	Dispositivos IoT	04/03/2020	07/03/2020	04 días
2.1.1	Sensores	04/03/2020	05/03/2020	02 días
2.1.2	Actuadores	06/03/2020	07/03/2020	02 días
2.2	Memorias RAM	08/03/2020	11/03/2020	04 días
2.2.1	Jerarquía de memoria	08/03/2020	09/03/2020	02 días
2.2.2	Tipos de memoria	10/03/2020	11/03/2020	02 días
2.3	Comandos Linux	12/03/2020	13/03/2020	02 días
2.4	Revisión de herramientas	14/03/2020	31/03/2020	18 días
2.4.1	Volatility	14/03/2020	16/03/2020	03 días
2.4.2	Volatilitux	17/03/2020	18/03/2020	02 días
2.4.3	Red Hat Crash Utility	19/03/2020	20/03/2020	02 días
2.4.4	Rekall	21/03/2020	21/03/2020	01 días
2.4.5	Draugr	22/03/2020	22/03/2020	01 días
2.4.6	DAMM	23/03/2020	23/03/2020	01 días
2.4.7	Invtero.net	24/03/2020	24/03/2020	01 días
2.4.8	Foriana	25/03/2020	25/03/2020	01 días
2.4.9	FATKit	26/03/2020	27/03/2020	02 días
2.4.10	Memoryze	28/03/2020	29/03/2020	02 días
2.4.11	BlackLight	30/03/2020	31/03/2020	02 días
2.5	**Entrega 2**	**31/03/2020**	**31/032020**	**Hito**
3	Análisis de las herramientas	01/04/2020	28/04/2020	28 días
3.1	Comparación de las herramientas	01/04/2020	02/04/2020	02 días
3.2	Caso de uso de herramientas	03/04/2020	28/04/2020	26 días
3.3	**Entrega 3**	**28/04/2020**	**28/04/2020**	**Hito**
4	Presentación y defensa del TFM	29/04/2020	09/06/2020	42 días
4.1	Conclusiones del TFM	29/04/2020	18/05/2020	20 días
4.2	Revisión de la memoria	19/05/2020	02/06/2020	15 días
4.3	**Entrega 4: Memoria final**	**02/06/2020**	**02/06/2020**	**Hito**
4.4	Elaboración del vídeo del TFM	03/06/2020	09/06/2020	07 días
4.5	**Entrega 5: Presentación en vídeo**	**09/06/2020**	**09/06/2020**	**Hito**
4.6	**Defensa del TFM**	**15/09/2020**	**19/06/2020**	**Hito**

Tabla 1: Planificación del trabajo

Ítem	Actividad	Inicio	Final
0	Trabajo Fin de Master (TFM)	19/02/2020	9/06/2020
1	Definición y planificación del TFM	19/02/2020	3/03/2020
1.1	Definición del trabajo a realizar	19/02/2020	20/02/2020
1.2	Contexto y justificación	21/02/2020	22/02/2020
1.3	Enfoque y método	23/02/2020	25/02/2020
1.4	Elaboración de la planificación	26/02/2020	27/02/2020
1.5	Estado del arte	28/02/2020	29/02/2020
1.6	Resumen de los capítulos	1/03/2020	2/03/2020
1.7	Resumen del trabajo	3/03/2020	3/03/2020
1.8	**Entrega 1: Plan de trabajo**	**3/03/2020**	**3/03/2020**
2	Investigación y estudio	4/03/2020	31/03/2020
2.1	Dispositivos IoT	4/03/2020	7/03/2020
2.2	Memorias RAM	8/03/2020	11/03/2020
2.3	Comandos Linux	12/03/2020	13/03/2020
2.4	Revisión de herramientas	14/03/2020	31/03/2020
2.5	**Entrega 2**	**31/03/2020**	**31/03/2020**
3	Análisis de las herramientas	1/04/2020	28/04/2020
3.1	Comparación de las herramientas	1/04/2020	2/04/2020
3.2	Caso de uso de herramientas	3/04/2020	28/04/2020
3.3	**Entrega 3**	**28/04/2020**	**28/04/2020**
4	Presentación y defensa del TFM	29/04/2020	9/06/2020
4.1	Conclusiones del TFM	29/04/2020	18/05/2020
4.2	Revisión de la memoria	19/05/2020	2/06/2020
4.3	**Entrega 4: Memoria final**	**2/06/2020**	**2/06/2020**
4.4	Elaboración del vídeo del TFM	3/06/2020	9/06/2020
4.5	**Entrega 5: Presentación en vídeo**	**9/06/2020**	**9/06/2020**
4.6	**Defensa del TFM**	**15/06/2020**	**19/06/2020**

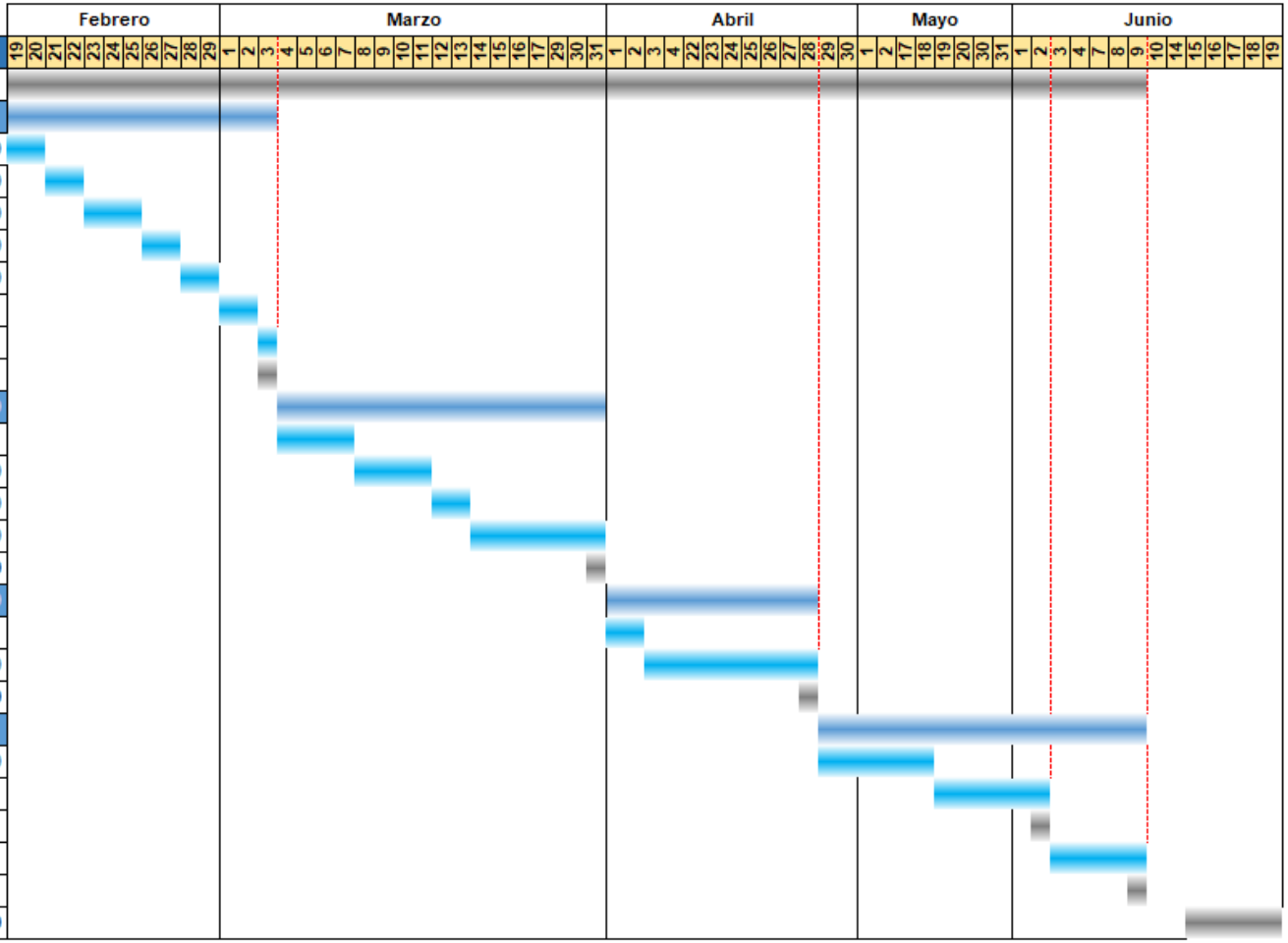


Ilustración 1: Diagrama de Gantt

1.5 Estado del arte

Los atacantes para ganar acceso remoto a los sistemas usan rootkit que son un conjunto de programas maliciosos con capacidad de ocultar procesos, archivos y actividades frente a los antivirus y programas de seguridad.

El análisis de volcado de memoria de los equipos de cómputo y dispositivos IoT es una técnica muy usada en la investigación forense para buscar pistas sobre algún ataque o actividad sospechosa en el sistema, puesto que la memoria RAM almacena información administrativa del sistema operativo, procesos que están ejecutándose, conexiones del equipo, información confidencial de los usuarios, correos, credenciales, etc. todo de manera temporal hasta que el sistema se reinicie o apague.

En [1] se usan las herramientas Volatility y PTFinder (ahora no disponible) en el análisis de la imagen de memoria del sistema operativo Windows XP, pero no se muestran la sintaxis de los comandos empleados para la recuperación de datos también se observa que los resultados son similares a las respuestas obtenidas de las consultas en vivo a la memoria RAM.

En [2] las herramientas Volatility, Rekall y Mimikatz (no disponible) se presentan como framework forense para el análisis de memoria entregando una breve descripción de cada uno de ellos.

El estudio sobre el análisis forense de la memoria física [3], toma en cuenta dos aspectos:

- Adquisición de memoria física, donde se pueden emplear:
  - Herramientas basadas en hardware, usa dispositivo físico que abre un puerto dedicado para copiar el contenido de la memoria física:
    * Tarjeta PCI Tribble que debe ser previamente instalado.
    * Dispositivo FireWire (IEEE 1394) usa DMA y accede a memoria.
  - Herramientas basadas en software, son programas que permiten obtener una imagen de la memoria física:
    * Data Dumper (DD) usado en los sistemas tipo Unix.
    * DumpIt herramienta portable.
- Análisis de la memoria física, una vez obtenido la imagen de la memoria ya se puede extraer información:
  - Volatility es la herramienta empleada para obtener información de la imagen, listado de procesos, información DLL, llaves de registro, etc.

En [4] se describe una extensión del uso de la herramienta Volatility para la adquisición de datos en dispositivos móviles, mediante el reinicio o arranque en frío del dispositivo método que consiste en explotar el efecto de remanencia de la memoria dinámica (DRAM) donde los datos se desvanecen lentamente y pueden ser capturados si se inicia inmediatamente el dispositivo.

1.6 Breve sumario de productos obtenidos

El Trabajo Final de Máster se divide en los siguientes entregables parciales, que en conjunto forman el resultado final del proyecto:

- Entrega 1: Esta entrega corresponde al plan de trabajo, donde se define el ámbito del trabajo, los objetivos que se esperan alcanzar, la metodología a seguir, la planificación con la previsión temporal de las actividades a realizar para alcanzar los objetivos.

- Entrega 2: Trata de una entrega parcial que contiene conceptos teóricos, incluye descripción sobre los dispositivos IoT, una descripción sobre las memorias, conceptos de comandos Linux y la descripción del estudio de las herramientas y técnicas existentes para el análisis de volcado de memoria.

- Entrega 3: Trata de una entrega parcial que incluye como caso práctico el uso de herramientas para el análisis de un volcado de memoria del sistema Linux documentando los resultados obtenidos.

- Entrega 4: En esta entrega se reúnen las entregas anteriores y se completa con las conclusiones, el glosario, la bibliografía y el anexo, se revisan y mejoran los aspectos formales de todo el documento y se genera la memoria del Trabajo Final de Máster.

- Entrega 5: Esta entrega corresponde a realizar un video con una presentación resumen del trabajo realizado con voz en off.

1.7 Breve descripción de los otros capítulos de la memoria

El Trabajo Final de Máster consta de dos grandes partes. La primera parte de investigación y estudio consta en la revisión de los conceptos teóricos sobre los dispositivos y programas involucrados en este trabajo.

Existe un capítulo sobre los dispositivos IoT donde se define en qué consisten, sus características, se revisa las tecnologías que utiliza, las aplicaciones y su manejo de la seguridad.

Existe un capítulo que trata sobre las memorias RAM, donde encontramos su definición, jerarquía de memorias, tipos de memorias.

Existe un capítulo sobre comandos Linux, donde se describen de manera básica los comandos utilizados para la gestión de la memoria.

En el capítulo revisión de herramientas, se describen las herramientas encontradas en el mercado para el análisis de los volcados de memoria, se exploran las opciones que manejan, sus alcances y limitaciones sobre lo que pueden analizar.

La segunda parte de Análisis de las herramientas corresponde a la parte práctica del trabajo, es decir se realiza un caso de uso de las herramientas revisadas en la primera parte sobre una muestra de volcado de memoria de sistemas Linux. Finalmente se documentan las pruebas realizadas y se presenta un resumen sobre los resultados obtenidos por las herramientas utilizadas.

2. Investigación y estudio

En este apartado se definen los conceptos y características de los dispositivos, comandos y programas involucrados en el desarrollo del proyecto.

2.1 Dispositivos IoT

El internet de las cosas (IoT) consta de diversos dispositivos conectados por redes de diferentes arquitecturas [5, 6] que les permite comunicarse de manera autónoma para el intercambio de información y datos, características aprovechados por usuarios y empresas que incrementan su uso en diversas aplicaciones como: Smart City, Smart Home, Smart Car, Healthcare, Wearables, Industrial IoT, Smart Farming [7, 8, 9].
Los dispositivos IoT son objetos o cosas con agregados de sistemas embebidos, conexión a redes y capacidad para detectar y actuar, por estas características se les conocen como objetos inteligentes.
Los componentes principales en los dispositivos IoT son los sensores y actuadores que otorgan a los objetos las capacidades para detectar (sentir) y actuar (mover) con su entorno, así como interactuar entre ellos o con los usuarios.

2.1.1 Sensores

Son dispositivos que otorgan capacidad del sentido a los dispositivos IoT al poder medir o registrar magnitudes físicas, químicas o biológicas, los sensores pueden ser activos o pasivos si requieren o no una fuente de energía.
Los sensores están formados por células sensibles que se encargan de medir y recolectar información del entorno para luego transformarlas en señales eléctricas que son magnitudes cuantificables y manipulables.

Algunos de los diversos tipos de sensores son:

- Sensores de temperatura, dispositivos que detectan las variaciones de temperatura y lo transforman en señales eléctricas para su posterior procesado, existen los tipos: termopares, RTD, termistores, infrarrojo.

- Sensores de proximidad, dispositivos que detectan el movimiento de objetos para ello crean un campo magnético a su alrededor, existen los tipos: capacitivos, inductivos, magnéticos.

- Sensores de humedad, dispositivos que detectan la humedad del aire basándose en la conductividad eléctrica del agua.

- Sensores de presión, dispositivos que transforman la fuerza ejercida por unidad de área de gases y líquidos en señales eléctricas, existen los tipos: capacitivos, piezoeléctricos, galgas extensométricas.

- Sensores de aceleración, dispositivos que miden la aceleración o vibración, existen los tipos: acelerómetros piezoeléctricos, acelerómetros de efecto Hall, acelerómetros micromecánicos.

- Sensores fotoeléctricos, dispositivos electrónicos que detectan el cambio de intensidad de la luz de su entorno, constan de componentes emisor y receptor de luz, existen los tipos: fotodiodo, fototransistor, célula fotoeléctrica, sensor CCD, sensor CMOS.

- Sensores de sonidos, dispositivos que detectan los niveles de ruido en decibeles (dB) si la sensibilidad se ajusta al sonido que puede escuchar el oído humano se mide en decibeles ajustados (dBA).

2.1.2 Actuadores

Son dispositivos que otorgan capacidad de respuesta (mover) a los dispositivos IoT por la acción de alguna fuente de energía.

Tipos de actuadores según su fuente de energía:

- Actuadores neumáticos, usan aire comprimido para producir movimiento.
- Actuadores hidráulicos, usan líquido para generar algún movimiento.
- Actuadores eléctricos, producen movimiento con energía eléctrica.
- Actuadores térmicos, usan energía calorífica o magnética para producir movimiento.

Los actuadores eléctricos más usados son:

- Relé o relevador, dispositivo electromagnético que funciona como un interruptor al abrir o cerrar contactos mediante un electroimán, es capaz de controlar un circuito de salida de mayor potencia al de entrada.

- Motores, dispositivos compuestos de bobinas que generan campos magnéticos para convertir la energía eléctrica en energía mecánica de rotación también existen los que pueden realizar el proceso inverso los motores de tracción pueden realizar ambas acciones.

- Motor paso a paso (Stepper), dispositivo electromecánico que convierte una serie de impulsos eléctricos en desplazamientos angulares discretos significa que es capaz de girar una cantidad de grados (pasos) se comporta igual que un conversor digital-analógico (D/A).

- Servomotor, dispositivo con la capacidad de posicionarse de forma inmediata en cualquier posición dentro de un rango de operación que generalmente es de 180°.

2.2 Memoria RAM

Es la memoria principal de un dispositivo es un espacio donde se guarda de forma temporal los programas y datos que serán ejecutados por el procesador el contenido de la memoria se elimina si se reinicia o apaga el dispositivo.

2.2.1 Jerarquía de memoria

Se establece en base a los tiempos de acceso y capacidad disponible, existen memorias que tienen mayor capacidad de almacenamiento y otras el menor tiempo de acceso para realizar operaciones de lectura y escritura estas se usan donde los accesos son más frecuentes.

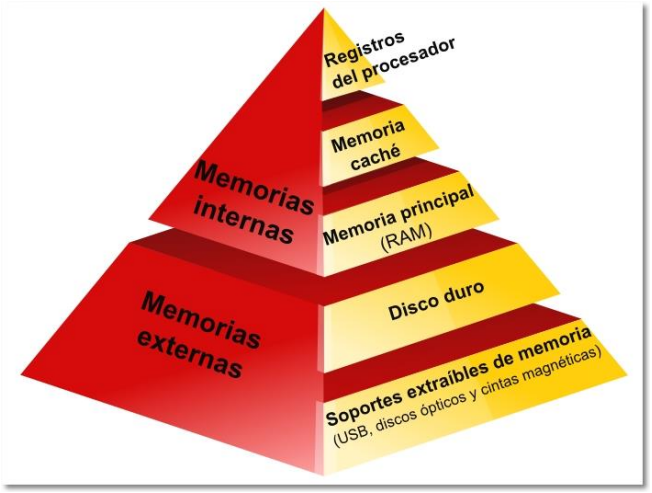


Ilustración 2: jerarquía de memoria
(Fuente: https://xxyzz.webcindario.com/jpg/2.4JerarquiaMemoria.jpg)

2.2.2 Tipos de memoria

Existen dos tipos de memoria RAM:

- SRAM (Static RAM), son memorias formadas por flip-flop construidos con transistores MOSFET o con transistores bipolares que no necesitan el proceso de refresco de datos, son las memorias más rápidas pero costosas de fabricar, se encuentran integradas dentro del procesador como caché y en los discos duros se utilizan como buffer para hacerlos más eficientes.

- DRAM (Dynamic RAM), son memorias construidas con condensadores que necesitan refrescar constantemente los datos almacenados son más sencillas de fabricar por lo que permite construir memorias de mayor capacidad este es del tipo utilizado en la memoria principal.

  La memoria DRAM tiene los siguientes tipos:

- FPM RAM (Fast Page Mode RAM), usado con Intel Pentium.
- EDO RAM (Extended Data Output RAM), mejora del FPM RAM.
- BEDO (Burst Extended Data Output), memoria que transfiere ráfagas de datos (burst) al procesador en cada ciclo de reloj.
- SDRAM (Synchronous DRAM), usa reloj interno para sincronizar la frecuencia de la memoria con la del procesador mejorando los tiempos de acceso y la eficiencia en la comunicación es el tipo de memoria RAM que se usa hoy en día en sus versiones:
  DDR SDRAM (Double Data Rate SDRAM)
  DDR2 SDRAM
  DDR3 SDRAM
  DDR4 SDRAM
  DDR5 SDRAM
  DDR6 SDRAM

2.3 Comandos Linux

Son herramientas que permiten interactuar en los sistemas Linux para realizar operaciones y obtener resultados deseados, ver anexo 7.1 para más detalle.

2.4 Revisión de herramientas

En esta sección se describen características de las herramientas que existen para el análisis de volcado de memoria, así como la manera de obtenerlo, su instalación y prueba de su operatividad.
En el anexo 7.2 se muestran las sintaxis del uso de las herramientas.

2.4.1 Volatility [10]

Publicado el año 2007 como Volatools en Black Hat DC [11] hasta entonces la investigación digital estaba centrado en el análisis de las imágenes del disco duro, Volatility introduce la capacidad de analizar el estado en tiempo de ejecución de un sistema usando los datos que se encuentran en la memoria principal (RAM).
Actualmente, la organización independiente sin fines de lucro The Volatility Foundation [12] es la encargada del desarrollo y la promoción de Volatility para el análisis de memoria dentro de la comunidad forense.
Volatility es un framework con una colección de herramientas completamente abiertas desarrolladas en Python bajo Licencia Pública General GNU para la extracción de artefactos digitales en muestras de volcado de memoria RAM.

Características

- Permite realizar el análisis de volcados de memoria de los sistemas operativos Windows, Linux, Mac y Android de 32 y 64 bits. El diseño

modular de Volatility le permite soportar fácilmente nuevos sistemas operativos y arquitecturas a medida que son lanzados.

- Es de código abierto (GPLv2), significa que se puede aprender cómo funciona la herramienta, entenderlo, ajustarla y mejorarla.
- Esta escrito en lenguaje Python que permite la carga de bibliotecas que se integran fácilmente con Volatility.
- Se ejecuta en sistemas de análisis de Windows, Linux o Mac donde es necesario la ejecución de Python.
- La API extensible y programable permite utilizar Volatility para crear una interfaz web o una interfaz gráfica de usuario personalizada, manejar su caja de arena de malware, realizar una introspección de la máquina virtual o explorar la memoria del núcleo de forma automatizada.
- Conjuntos de características basados en la ingeniería inversa y la investigación especializada. Volatility proporciona capacidades que el propio depurador del kernel de Microsoft no permite.
- Analiza diversos formatos de archivos: raw dumps, crash dumps, archivos de hibernación, archivos VMware (.vmem), archivos de estado guardados y suspendidos de VMware (.vmss/.vmsn), VirtualBox core dumps, LiME (Linux Memory Extractor), expert witness (EWF) y memoria física directa sobre Firewire, incluso se pueden convertir entre formatos.
- Contiene algoritmos rápidos y eficientes que permiten analizar los volcados de memoria de los grandes sistemas sin gastos innecesarios o consumo de memoria.
- Tiene una comunidad seria y poderosa de profesionales e investigadores que trabajan en los campos forenses, respuesta a incidente (IR) y análisis de malware, reúne a colaboradores de empresas comerciales, organismos encargados de aplicación de ley e instituciones académicas de todo el mundo.

Formatos soportados

Volatility admite una variedad de formatos de archivos de muestra de memoria y la capacidad de conversión entre estos formatos:

- Muestra Raw linear (dd)
- Archivo de hibernación (Windows 7 y anteriores)
- Archivo Crash dump
- VirtualBox ELF64 core dump
- Archivos de estado guardado e instantánea de VMware
- Formato EWF (E01)
- Formato LiME
- Formato de archivo Mach-O
- QEMU virtual machine dumps
- Firewire
- HPAK (FDPro)

Requerimientos

Para su funcionamiento se necesita:

- Un computador con sistema operativo Windows, Linux o Mac OS X.
- Python versión 2.6 o superior (pero no 3.x).

Algunos complementos requieren bibliotecas de terceros como:

- Distorm3, biblioteca desensamblador para x86/AMD64.
- Yara, herramienta de identificación y clasificación de malware.
- PyCrypto, herramientas de criptografía de Python.
- OpenPyxl, representación xlsx para todos los complementos.
- PIL, biblioteca de imágenes de Python.
- UJson, biblioteca de análisis de JSON ultrarrápida.

Manejo de la herramienta

La herramienta se puede descargar desde su sitio web pero antes se deben considerar los requisitos para la instalación existen algunas distribuciones Linux que contienen la herramienta en su repositorio facilitando su instalación junto con los programas necesario para su operatividad, al ser una herramienta de software libre se encuentra documentación, guía de instalación, etc.

La herramienta trabaja haciendo uso de perfiles para el análisis del volcado de memoria un perfil contiene información sobre la estructura de la memoria del sistema operativo esto significa que existe un perfil específico para cada tipo de sistema operativo.

En Volatility ya vienen preinstalados perfiles para todas las versiones del sistema operativo Windows pero no trae ningún perfil para los sistemas operativos Linux o Mac por esta razón la primera tarea a realizar antes del análisis de volcados de memoria de sistemas Linux será necesario generar el perfil para la versión del sistema operativo Linux de donde se obtuvo la muestra de volcado de memoria.

Con el perfil correspondiente más los comandos predefinidos de la herramienta se puede realizar el análisis del volcado de memoria obteniendo información sobre los procesos, procesador, variables de entorno, conexiones de red, historial de comandos ejecutados, interfaces de red, etc.

2.4.2 Volatilitux [13]

Inspirado en el desafío SSTIC 2010 [14] cuyo propósito era analizar la memoria física de un teléfono Android. Volatilitux es un framework de Python que ayuda a extraer artefactos digitales de volcados de memoria física (RAM) sin procesar de sistemas Linux.

Características

Estas son las principales características de la herramienta:

- Detección automática de compensaciones de estructura del núcleo.
- Detección manual de esas compensaciones utilizando un módulo de kernel cargable (LKM).
- Soporta múltiples arquitecturas:
  ARM
  x86
  x86 con PAE habilitado
- Puede trabajar como un módulo de Python para automatizar tareas e integrarse con otros proyectos.

Requerimientos

- Computador con sistema operativo Windows, Linux o Mac OS X.
- Python versión 2.6 o superior usa módulos incorporados y no depende de ninguna biblioteca externa.

  Si se planea usar el LKM proporcionado se debe tener:

- GCC
- Make
- Los archivos de encabezado del Kernel (ubicados en /usr/src)

Manejo de la herramienta

La herramienta se puede obtener desde su sitio web no necesita de instalación solo se descomprime y ya puede usarse.

Esta herramienta es semejante a Volatility necesita que se genere un archivo de configuración con las características del sistema operativo de donde se obtuvo la muestra de volcado de memoria.
Tiene pocos comandos predefinidos para el análisis del volcado de memoria que permiten obtener el listado de procesos, lista de archivos abiertos, mapa de memoria del proceso, etc.

2.4.3 Red Hat Crash Utility [15]

El sistema operativo Linux carecía de recurso de volcado por kernel crash como en UNIX que fueron abordados por el parche de kernel Mission Critical Linux Mcore y el parche de kernel LKCD (Linux Kernel Crash Dump) de SGI en 1999 luego por la instalación de Red Hat Netdump en 2002 y la instalación de Red Hat Diskdump en 2004 finalmente la comunidad de Linux adopta en 2006 la instalación de volcado por caída de Kdump.
Pero el archivo de volcado por kernel crash es solo la mitad de la imagen que requiere una utilidad para poder reconocer el formato del archivo de volcado para leerlo y ofrecer un conjunto útil de comandos para darle sentido.
Para examinar el contenido de componentes internos del núcleo de un sistema en vivo desde el espacio del usuario, la única opción disponible fue usar gdb en /proc/kcore pero distribuciones como Red Hat Enterprise Linux han limitado su

acceso inutilizándolo como fuente de memoria del núcleo, además gdb no puede leer archivos de volcado LKCD, Mcore, Xen o s390/s390x.
La utilidad Crash se desarrolla como un medio conveniente para cubrir todos los formatos de archivo de volcado como los sistemas en vivo.

Características

- Está basado en el comando crash de SVR4 UNIX, pero fue ampliamente mejorado con la fusión completa del depurador GNU GDB, combinando la naturaleza específica del kernel de la utilidad crash de UNIX con las capacidades de depuración a nivel de código fuente de GDB.
- Analiza de forma interactiva el estado del sistema Linux mientras está en ejecución.
- Analiza volcados de memorias creadas por netdump, diskdump, LKCD, kdump, xendump or kvmdump.
- Los comandos incorporados se pueden ampliar o mejorar y los módulos de comandos externos se pueden escribir y adjuntar dinámicamente.
- La utilidad Crash se desarrolla y prueba activamente en:
  x86
  x86_64
  ia64
  ppc64
  arm
  s390
  s390x
  Soporte heredado para plataformas Alpha y PowerPC de 32 bits pero no se mantiene activamente.

Formatos soportados

La utilidad puede utilizarse para investigar:

- Volcado de núcleo Linux kernel creado por la instalación de Kdump
- Volcado comprimido de núcleo Linux kernel creado por el comando makedumpfile (de kdump dumpfiles)
- Volcado de núcleo Linux kernel creado por la instalación de Red Hat Netdump
- Volcado de núcleo Linux kernel creado por la instalación de Red Hat Diskdump
- Volcado de núcleo Linux kernel comprimido creado por la instalación de Red Hat Diskdump
- Volcado de núcleo Linux kernel de host Xen creado por la instalación de Kdump
- Volcado de núcleo Linux kernel de invitado Xen creado por la instalación original de xendump
- Volcado de núcleo Linux kernel de invitado Xen creado por la instalación de xendump en formato ELF
- Volcado de núcleo de hypervisor Xen creado por la instalación Kdump

- Volcado de núcleo Linux kernel de invitado KVM creado por la instalación de volcado de virsh
- Volcado de núcleo Linux kernel de s390 creado por la instalación de volcado de núcleo IBM standalone
- Volcado de núcleo Linux kernel de s390x creado por la instalación de volcado de núcleo de IBM standalone
- Volcado de núcleo Linux kernel creado por el proyecto Sourceforge de LKCD (Linux Kernel Crash Dumps)
- Volcado de núcleo Linux kernel creado por el parche Mcore ofrecido por Mission Critical Linux

Requerimientos

- Un computador con sistema operativo Linux.
- Archivo de objeto del kernel *vmlinux* creado con la bandera -g C para que contenga los datos de depuración necesarios para la depuración simbólica, forma parte del paquete kernel debuginfo y se encuentra en el directorio correspondiente /usr/lib/debug/lib/modules/<release>
- Imagen de memoria que puede ser un archivo de volcado por kernel crash generado o memoria del sistema en vivo a la que se accede mediante /dev/mem o su reemplazo el controlador /dev/crash.

Manejo de la herramienta

Esta herramienta forma parte de los programas desarrollados para las distribuciones Linux es decir se encuentra disponible en el gestor de paquete de cada versión de Linux facilitando su disponibilidad para la instalación y configuración de la herramienta en los sistemas Linux.

Al ejecutar la herramienta se carga el ultimo volcado de memoria disponible pero también se puede especificar la ruta de cualquier otra muestra, al cargar el volcado de memoria la herramienta entrega una interfase para escribir los comandos que permiten su análisis.

Viene con varios comandos predefinidos que nos permiten obtener información del estado de los procesos, memoria virtual, archivos abiertos, mensajes del sistema, lista de tareas en ejecución, tareas en espera, archivos de usuarios, lista de dispositivos, etc.

2.4.4 Rekall [16]

Es un framework con una colección de herramientas completamente abierta, implementada en Python bajo la Licencia Pública General de Apache y GNU para la extracción y análisis de sistemas informáticos de artefactos digitales.

Características

- Proporciona una solución de extremo a extremo para los respondedores de incidentes y los analistas forenses.

- Aprovecha la información de depuración proporcionado por proveedores del sistema operativo para localizar con precisión estructuras de datos de kernel.
- Mientras otras herramientas dependen de la heurística y firmas, Rekall pretende ser el marco de análisis de memoria más estable y confiable.
- Mantiene el repositorio de perfil público más grande para diversas versiones de sistemas operativos.

Requerimientos

- Computador con sistema operativo Windows, Linux o Mac OS X.
- Python versión 2.6 o superior.

Manejo de la herramienta

La herramienta está disponible como un programa Python por lo que primero se debe instalar el lenguaje de programación Python en el sistema operativo luego se crea un entorno virtual donde se procede a instalar la herramienta con los programas y dependencias necesarias para su funcionamiento aislando de esta manera cualquier conflicto con los programas del servidor.

La herramienta hace uso de perfiles para realizar el análisis del volcado de memoria tiene pocos comandos que permiten obtener la lista de procesos, archivos abiertos, consulta sobre los procesos, etc.

2.4.5 Draugr [17]

Herramienta para análisis forense de memoria en vivo en Linux, al usar /dev/(k)mem o un volcado de memoria puede usarse para acceder fácilmente con python a la memoria, reproducir (leer, escribir, desensamblar, buscar) con ella y se puede encontrar información del sistema (procesos) por diferentes métodos.
Puede encontrar símbolos del núcleo (haciendo coincidencia de patrones en un archivo XML o con EXPORT_SYMBOL), procesos (información y secciones por la lista vinculada al núcleo o fuerza bruta) y desmontar/volcar la memoria.

Manejo de la herramienta

La herramienta está disponible desde su sitio web pero no existe información sobre su instalación, configuración y manera de uso por tanto no fue posible probar como funciona.

2.4.6 DAMM (Differential Analysis of Malware in Memory) [18]

Herramienta de código abierto para el análisis de memoria construida sobre Volatility pone a disposición de la comunidad técnicas con el intento de acelerar el proceso de investigación mediante la reducción de datos y la codificación de algunos conocimientos expertos.

Características

- Aproximadamente 30 complementos de volatilidad combinados en alrededor de 20 complementos DAMM (pslist, psxview y otros elementos se combinan en complemento de 'procesos').
- Capacidad de ejecutar varios complementos en una sola invocación.
- Opción para almacenar resultados de complementos en bases de datos SQLite para la conservación o para análisis en caché.
- Un sistema de filtrado/tipo que permite filtrar fácilmente atributos como PIDs para ver toda la información relacionada con algún proceso y coincidencia exacta o parcial para cadenas, etc.
- Capacidad de mostrar las diferencias entre dos bases de datos de resultados para máquinas iguales o similares y manipular desde la línea de comandos cómo funciona la diferencia.
- Capacidad de advertencia sobre ciertos tipos de comportamiento sospechoso.

Manejo de la herramienta

Esta herramienta se puede descargar desde su sitio web contiene una carpeta con Volatility porque se basa en dicha herramienta para el análisis del volcado de memoria está disponible para su uso sin instalación.

No fue posible realizar el análisis de muestras de memoria porque presenta errores de configuración que no reconocen los perfiles.

2.4.7 inVtero.net [19]

Análisis forense de alta velocidad (Gbps) integridad y garantía con volcados de la memoria física, incluye capacidades de memoria ofensiva y defensiva.

Características

- Ampliación de la detección de procesos a la introspección de hipervisor independiente de microarquitectura.
- Aislamiento y extracción de la memoria del proceso, incluida la introspección recursiva de VM/hipervisor que puede estar ejecutándose.
- Reducciones rápidas quickdumps es una demostración de la API una pequeña herramienta para extraer rápidamente todo lo que puede ver en una jerarquía de directorios sensible.

Formatos soportados

- VMWARE
- XEN
- Crash dump (archivos de volcado PAGEDUMP64 / Blue Screen)
- Tipo simbólico extracción/vinculación

Scripting DLR (Python)
- Linux básico:
  BSD
  Hyper-V
  Windows
  Genérico (tablas de páginas de puntero automático)

Manejo de la herramienta

Esta herramienta según se observa en su sitio web es para sistemas Windows no existe mucha información sobre instalación y modo de uso.

2.4.8 Foriana (FOrensic Ram Image ANAlyzer) [20]

La mayoría de las herramientas de análisis usa el enfoque de coincidencia de patrones es decir recuerdan las firmas de ciertas estructuras del sistema sin embargo en el mundo Linux/Unix con muchas distribuciones esto no funciona.
Foriana trata de resolver este problema de una manera más general y basa su enfoque en relaciones lógicas y en la búsqueda de listas de elementos definidos por alguna característica, incluso esto puede permitir que el algoritmo no falle con las versiones más recientes de los sistemas operativos que se desconocen al momento de compilar el programa.

Características

- Soporta las arquitecturas:
  ARM
  x86_64
  i386
- Soporta los sistemas operativos:
  Windows
  Linux
  BSD

Manejo de la herramienta

La herramienta está disponible desde su sitio web en formato comprimido también se encuentra ahí una guía para la instalación de la herramienta y la generación de volcado de memoria.

La herramienta permite obtener del volcado de memoria mediante comandos predefinidos información sobre los procesos, archivos, módulos, mapas de memoria, etc.

2.4.9 Forensic Analysis Toolkit (FATKit) [21]

Es un framework de investigación digital multiplataforma, modular y extensible que analiza la memoria volátil del sistema, está destinado a investigadores,

profesionales encargados de hacer cumplir la ley y analistas forenses que estén interesados en extraer e interpretar información relevante a raíz de un delito o incidente.
Automatiza la extracción y visualización de objetos digitales que se encuentran en la memoria física, liberando al analista forense de los tediosos aspectos de obtener los datos a bajo nivel, facilitando la extracción, análisis, agregación y visualización de datos forenses en varios niveles de abstracción y complejidad de datos.
Incluye módulos para la reconstrucción del espacio de direcciones virtuales, la traducción de direcciones virtuales a físicas y emplea una serie de técnicas de visualización y minería de datos para mejorar el análisis y facilitar la búsqueda a través de grandes cantidades de datos.

Características

- Soporta los sistemas operativos Windows y Linux.
- Soporte para espacios de direcciones virtuales basados en x86 y tipos de datos nativos.
- Incluye enumeración de procesos/tareas, enumeración de módulos y detección de código malicioso residente en memoria.
- El sistema de tipo basado en perfiles permite asignar tipos de bajo nivel a construcciones de nivel superior y distribuirlas para varias compilaciones de software.
- Las herramientas automatizadas de generación de perfiles permiten la extracción de formatos de objetos de bajo nivel cuando el código fuente está disponible.
- Los módulos de análisis programables permiten implementar fácilmente técnicas de extracción especializadas o patentadas utilizando un lenguaje de alto nivel, en lugar de rutinas codificadas a mano.
- El diseño modular permite la fácil extensión a nuevas arquitecturas y sistemas operativos.
- Explorador de objetos que permite interpretar objetos de memoria binaria en el nivel de abstracción del lenguaje de alto nivel del código fuente, con el soporte para aplicaciones que están escritas en el lenguaje C el navegador permite manejar objetos en memoria y sus campos anidados, seguir punteros y convertir objetos a otros formatos de datos.
- El visor de espacio de direcciones permite visualizar datos tal como aparecen en un espacio de direcciones físico o virtual, incluye objetos codificados por colores, representación de datos hexadecimales y ASCII

Manejo de la herramienta

No se encuentra mucha información sobre esta herramienta en su sitio web tampoco existe opción para descargarlo.

2.4.10 Memoryze [22]

Software forense libre que puede adquirir y/o analizar imágenes de memoria y en los sistemas en vivo puede incluir el archivo de paginación en su análisis ayudando a los respondedores de incidentes en su búsqueda.

Características

- Soporta los sistemas operativos Windows y Mac OS X
- Imagen de rango completo de la memoria del sistema (sin depender de llamadas API).
- Imagen de un proceso cargado, DLL, EXE y pilas.
- Imagen de un controlador especifico o todos los controladores cargados en la memoria al disco.
- Enumera todos los procesos en ejecución (incluyendo los ocultos por rootkits) para cada proceso:
  - Informa todos los identificadores abiertos en un proceso (incluidos todos los archivos, claves de registro, etc.)
  - Enumera el espacio de direcciones virtuales de un proceso dado incluye las DLL cargadas y las partes asignadas del montón y la pila
  - Lista todos los sockets de red que el proceso ha abierto, incluidos los ocultos por rootkits.
  - Especifica las funciones importadas/exportadas por el EXE y las DLL.
  - Hash el EXE y las DLL en el espacio de direcciones del proceso (MD5, SHA1, SHA256. Esto está basado en disco).
  - Verifica firmas digitales de los EXEs y las DLL (basadas en disco).
  - Salida de todas las cadenas en la memoria por proceso.
- Identifica todos los controladores cargados en la memoria, incluidos los ocultos por rootkits. Para cada controlador:
  - Especifica las funciones que el controlador importa/exporta.
  - Hash del controlador (MD5, SHA1 y SHA256. Basado en disco).
  - Verifica la firma digital del controlador (basada en disco).
  - Salida de todas las cadenas en la memoria por controlador.
- Informa las capas de dispositivos y controladores, que se pueden usar para interceptar paquetes de red, pulsaciones de teclas y actividad de archivos.
- Identifica todos los módulos del kernel cargados recorriendo una lista vinculada.
- Identifica ganchos (a menudo utilizados por rootkits) en la tabla de llamadas del sistema, las tablas de descriptores de interrupciones (IDT) y las tablas de funciones del controlador.

Manejo de la herramienta

Esta herramienta de análisis de memoria es solo para los sistemas Windows requiere de registro para su descarga.

2.4.11 BlackLight [23]

Herramienta forense digital para el análisis de volúmenes de computadoras y dispositivos móviles, tiene licencia y se debe pagar para usar la herramienta, pero puede solicitarse una prueba gratuita por un tiempo determinado.

Características

- Proporciona detalles de las acciones del usuario e informes del análisis de imágenes de memoria.
- Permite el análisis a diversos tipos de archivos de memoria inclusive archivos de hibernación, pagefile.sys, volcados sin formato y volcados por caída.
- Contiene una poderosa vista de filtro de archivos y es más rápido que las herramientas forenses de código abierto.
- Analiza datos de las plataformas:
  Windows
  MacOS X
  Android
  iOS
- Maneja vistas de memoria que analiza procesos, bibliotecas, sockets, controladores y controladores.
- La vista de comunicaciones tiene un registro completo de llamadas, correo de voz, actividades de redes sociales y más.

Requerimientos

- Computador con sistema operativo Windows 7 o superior, Mac OS X Yosemite (10.10) o superior
- Procesador de 2.6 GHz Intel Dual Core i5
- Memoria de 8 GB 1067 MHz DDR3
- 25GB de espacio en disco
- Resolución de pantalla de 1024 x 768 o superior

Formatos soportados

- Imágenes de disco:
  E01 (variants)
  L01
  Raw (.dd)
  DMG, VMDK
  .sparsebundle, .sparseimage
  .img, .iso
- Imágenes de memoria Windows:
  Raw
  hiberfil.sys (Vista y posteriores)
  pagefile.sys
  Crash Dumps (Full)
- Imágenes iOS de terceros:

MPE+
Cellebrite
ElcomSoft
Lantern

Manejo de la herramienta

Herramienta que permite realizar análisis de los sistemas operativos Windows, Mac y Android es software propietario y debe pagarse para su uso también se puede obtener una prueba gratuita con ciertas limitaciones de la herramienta previo registro.

3. Análisis de las herramientas

Este apartado consta de dos partes la primera muestra un cuadro comparativo de las herramientas y la segunda es la parte práctica del trabajo al realizar un caso de análisis a una muestra de volcado de memoria de sistemas Linux.

3.1 Comparación de las herramientas

Herramienta	Tipo de aplicación	Disponible para S.O.	Análisis de muestras	Requisitos	Características
Volatility	Software Libre	Windows Linux Mac	Windows Linux Mac	Python Perfiles de S.O.	Disponible. Fácil instalación y modo de uso.
Volatilitux	Software Libre	Windows Linux Mac	Windows Linux Mac	Python Volatility	Disponible. No requiere instalación.
Crash	Software Libre	Linux	Linux		Disponible. Fácil instalación y modo de uso.
Rekall	Software Libre	Linux	Linux	Python	Disponible. Instalación en entorno virtual de Python.
Draugr	Software Libre	Linux	Linux	No se especifica.	Disponible. Sin información para su uso e instalación.
DAMM	Software Libre	Windows Linux	Windows Linux	Volatility	Disponible. Hay problema de uso por su configuración.
InVtero.net	Software Libre	Windows	Windows	Python	Disponible. Presenta poca información de su instalación y el modo de uso.
Foriana	Software Libre	Linux	Linux Windows	Ninguna	Disponible. Información de modo de uso e instalación
FATKit	Software Libre	Windows Linux	Windows Linux	No existe información.	No disponible Escasa información sobre su uso.
Memoryze	Software Libre	Windows Mac	Windows	Ninguno	Disponible previo registro.
BlackLigth	Propietario	Windows MAc	Windows Mac	Ninguno	Disponible con limitaciones previo registro.

Tabla 2: Comparación de herramientas

3.2 Caso de uso de herramientas

En esta sección se describen las pruebas de las herramientas sobre una muestra de volcado de memoria de sistema Linux.

3.2.1 Ambiente de prueba

Para la realización del análisis de memoria se implementa el ambiente de prueba con el programa de virtualización VMware [24] donde se crea un servidor virtual con el sistema operativo Linux Centos 7.

En este servidor virtual se proceden a instalar las herramientas Volatility v2.6, Volatilitux, Red Hat Crash Utility v7.2, Rekall y Foriana para cada herramienta se siguen las especificaciones dadas en sus respectivos sitios de descarga además de búsquedas de información en Internet para superar algunos errores que se presentaban para la correcta operatividad de las herramientas.

3.2.2 Muestra de memoria

La muestra proporcionada según se indica es un volcado estándar obtenido de un sistema GNU/Linux ARMv7.

Para el uso de la herramienta Volatility en el análisis de la muestra de memoria es necesario saber la versión del sistema operativo de donde se obtuvo la muestra para generar el perfil respectivo y usarlo con la herramienta.

Haciendo uso de comandos del sistema Linux (ver anexo 7.3) se pudo obtener los siguientes datos de la muestra de memoria:

- Dispositivo origen: Raspberry Pi 4 Model B Rev 1.1
- Usa arquitectura ARM
- Tiene CPU versión 7
- Versión del hardware: Pi 4 Model B v1.1
- Sistema Operativo: Raspbian
- Versión de kernel: Linux version 4.19.75-v7l+

De los resultados obtenidos se observa que la muestra de volcado de memoria a analizar corresponde a un dispositivo Raspberry Pi [25] versión 4 modelo B que consta de una arquitectura ARM versión 7 ejecutando el sistema operativo Raspbian que es una distribución del sistema operativo GNU/Linux basado en Debian con la versión de kernel 4.19.75.

3.2.3 Generación de perfil con Raspberry Pi Desktop

Puesto que no se dispone de una arquitectura ARM como primera opción se procede a usar la versión Raspberry Pi Desktop del sistema operativo Raspbian para su instalación en los sistemas x86 de la PC y que se encuentra disponible en el sitio de descarga [26].

Se obtiene la última versión disponible de Raspbian Desktop basado en Debian Buster con la versión de kernel 4.19 que es una versión semejante al de la muestra de memoria se procede a realizar su instalación en una máquina virtual luego se genera el perfil del sistema operativo (ver anexo 7.4).

Con el perfil ya cargado en la herramienta Volatility se procede a analizar la muestra del volcado de memoria pero no se obtienen los resultados esperados por lo que el perfil generado no le corresponde a la muestra.

Revisando la versión de kernel del Raspbian utilizado se observa que es la 4.19.0-8 basado en Debian 4.19.98 se vuelve a realizar la prueba esta vez usando un perfil generado de una versión anterior 4.19.0-6 basado en Debian 4.19.67 pero tampoco se logra obtener información de la muestra.

La otra opción a probar es generar el perfil con la misma versión de kernel de Raspbian al de la muestra de memoria pero no existe el Raspbian Desktop con la versión de kernel requerido por lo que primero se procede a actualizar el Raspbian Desktop con kernel 4.9.0-8 a la versión de kernel 4.19.75 (ver anexo 7.5) terminada la actualización se genera el perfil para esta nueva versión y junto con la herramienta Volatility se intenta nuevamente obtener información de la muestra de memoria sin resultado favorable.

3.2.4 Generación de perfil con Raspberry Pi emulado

Debido al fallo de los perfiles generados con el Raspberry Pi Desktop como segunda opción se procede a usar el programa QEMU [27] que permite realizar la emulación del dispositivo Raspberry Pi con su arquitectura ARM sobre una arquitectura x86.

Mediante VMware se crea la máquina virtual con el sistema operativo Debian 9.8 donde se instala el programa de emulación QEMU se descarga la última imagen del sistema operativo Raspbian además de los programas necesarios para emular el Raspberry Pi (ver anexo 7.6).

Para generar el perfil fue necesario descargar de otros sitios el archivo de mapa del sistema porque con la arquitectura ARM no se encuentra en el sistema operativo como en los sistemas con arquitectura x86.

Se carga el perfil generado en la herramienta Volatility donde se le realiza el test de verificación y se observa que es reconocido como perfil de arquitectura ARM pero al realizar el análisis a la muestra de memoria falla por lo que el perfil generado no le corresponde a la muestra.

Revisando lo utilizado en esta ocasión se observa que la versión del Raspbian es la 4.19.97+ el kernel para la emulación tiene la versión 4.19.50 además los archivos System.map obtenidos son de otras fuentes.

Como ya se puede emular la arquitectura ARM se busca probar con la misma versión del kernel 4.19.75-v7l+ como la de la muestra de memoria pero al no existir esta versión de kernel se procede a crearlo (ver anexo 7.7) esto además nos permite obtener una nueva versión del archivo System.map

Ya teniendo uniformizadas las versiones de todos los elementos se procede a emular el Raspberry Pi con la versión del kernel 4.19.75-v7l+ y se genera un nuevo perfil que agregándolo a la herramienta Volatility se intenta analizar una vez más la muestra del volcado de memoria pero nuevamente el perfil no le corresponde por lo que su análisis no es posible con esta herramienta.

3.3 Uso de otras herramientas

La herramienta utilizada es Autopsy [28] plataforma forense digital de código abierto que también tiene la capacidad de analizar imágenes de memorias se realiza su instalación en el ambiente de prueba (ver anexo 7.8) y se somete a la muestra de memoria para su análisis.

Autopsy reconoce al volcado de la memoria pero su análisis no logra disgregar la información que contiene por el contrario lo maneja como un todo mostrando en este caso su contenido en formato hexadecimal y en formato de texto para este último hace uso del comando Strings.

Otra opción disponible es Bulk Extractor [29] herramienta para el análisis forense de volcados de memoria también realiza tareas como la investigación de malware e intrusión en el ambiente de prueba se procede con su instalación (ver anexo 7.9) donde se realiza el análisis a la muestra de memoria.

El resultado obtenido con bulk extractor fue satisfactorio esta herramienta logra extraer información del volcado de memoria almacenándolas separadamente en archivos de textos entre los datos obtenidos se tiene:

- Números de tarjetas de crédito
- Direcciones de Correo Electrónico
- Direcciones de dominios
- Direcciones MAC de tarjetas de red
- Direcciones de IP
- Json descargados o de código fuente
- Cabeceras de correos
- Urls de cache del navegador web
- Información de archivos comprimidos zip

Existe un archivo packets.pcap que con un analizador de protocolos y paquetes se obtiene un análisis del tráfico de red existente en el dispositivo.

4. Conclusiones

En el presente trabajo se han revisado y estudiado las diversas herramientas que existen para realizar el análisis de volcados de memoria RAM.

En su estudio se observa la importancia para las herramientas en contar con el perfil del sistema donde se obtuvo la muestra de volcado de memoria puesto que determina el éxito o fracaso en el análisis que realiza la herramienta.

Esto se entiende porque un perfil contiene características propias del sistema como son la estructura de datos del kernel y la tabla de direcciones de memoria con las funciones que contiene dichas direcciones.

Se trabajó en realizar el análisis del volcado de memoria RAM de una muestra obtenida desde un dispositivo Raspberry Pi con arquitectura ARM la primera dificultad estuvo en la obtención del perfil para la versión del sistema operativo de la muestra sumándosele que en esta arquitectura el sistema operativo no contiene los archivos para el perfil por lo que fue necesario generarlos.

Los resultados del análisis de la muestra de memoria con el perfil obtenido no fueron satisfactorios concluyendo que los archivos generados para el perfil no tienen las características del kernel o del mapa de memoria del sistema de donde obtuvo la muestra de memoria por lo que se requiere realizar un estudio más profundo para la creación del perfil de esta arquitectura.

El problema del perfil para el análisis de la muestra de memoria RAM impide el uso de las herramientas estudiadas se intentó usar los comandos strings y grep de Linux que en conjunto permiten obtener de la muestra de memoria algunos datos como el sistema operativo utilizado, su arquitectura, la versión de kernel, etc. pero es muy limitado y poco práctico para un análisis más extenso.

Se buscan y prueban nuevas herramientas como Autopsy y bulk_extractor siendo esta última la más efectiva en el análisis de la muestra de volcado de memoria RAM obteniendo información como tarjetas de créditos, direcciones IP, direcciones de correos, direcciones de dominios, captura de paquetes de red, cabeceras de correo, archivos comprimidos, etc.

Del análisis realizado se confirma que en la memoria RAM se almacenan toda la información que manejamos y los datos del propio sistema, claro estos son volátiles pero mientras no se apague el sistema o se llene la memoria los datos están disponibles, por eso es necesario contar con medidas de seguridad para evitar que agentes externos ingresen a nuestro sistema realicen un volcado de memoria remotamente y roben nuestra información.

5. Glosario

- **AMD**: Advanced Micro Devices
- **Android**: Sistema operativo móvil desarrollado por Google
- **API**: Application Programming Interface
- **ARM**: Advanced RISC Machine
- **ASCII**: American Standard Code for Information Interchange
- **BEDO**: Burst Extended Data Output
- **Bit**: Binary digit
- **BSD**: Berkeley Standard Distribution
- **CCD**: Charge Coupled Device
- **CMOS**: Complementary Metal Oxide Semiconductor
- **CPU**: Central Processing Unit
- **D/A**: Digital to Analog
- **dB**: Decibelio
- **dBA**: Decibelio ponderado A
- **dd**: Dataset Definition
- **DDR SDRAM**: Double Data Rate SDRAM
- **DDR2 SDRAM**: Double Data Rate type two SDRAM
- **DDR3 SDRAM**: Double Data Rate type three SDRAM
- **DDR4 SDRAM**: Double Data Rate type four SDRAM
- **DDR5 SDRAM**: Double Data Rate type five SDRAM
- **DDR6 SDRAM**: Double Data Rate type six SDRAM
- **DLL**: Dynamic Link Library
- **DLR**: Dynamic Language Runtime
- **DMA**: Direct Memory Access
- **DRAM**: Dynamic RAM
- **EDO RAM**: Extended Data Output RAM
- **ELF**: Executable and Linkable Format
- **EWF**: Expert Witness Format
- **EXE**: Archivo ejecutable
- **FDPro**: Fast Dump Pro
- **FPM RAM**: Fast Page Mode RAM
- **Gbps**: Gigabits por segundo
- **GCC**: GNU Compiler Collection
- **GDB**: GNU Project Debugger
- **GNU**: GNU's Not Unix
- **GPL**: General Public License
- **Hyper-V**: Programa de virtualización de Microsoft
- **IBM**: International Business Machines
- **IDT**: Interrupt Descriptor Table
- **IEEE**: Institute of Electrical and Electronics Engineers
- **iOS**: Sistema operativo móvil desarrollado por Apple
- **IoT**: Internet of Things
- **IR**: Incident Response
- **JSON**: JavaScript Object Notation

- **JTAG**: Joint Test Action Group
- **KVM**: Kernel-based Virtual Machine
- **LiME**: Linux Memory Extractor
- **LKCD**: Linux Kernel Crash Dump
- **LKM**: Loadable Kernel Module
- **Mac OSX**: Macintosh Operating System X
- **Mach-O**: Mach Object
- **MD5**: Message-Digest Algorithm 5
- **MOSFET**: Metal Oxide Semiconductor Field Effect Transistor
- **PAE**: Physical Address Extension
- **PC**: Personal Computer
- **PCI**: Peripheral Component Interconnect
- **PCX**: PiCture eXchange
- **PIL**: Python Imaging Library
- **PID**: Process IDentifier
- **PyCrypto**: Python Cryptography Toolkit
- **Python**: Lenguaje de programación
- **QEMU**: Quick EMUlator
- **RAM**: Random Access Memory
- **RTD**: Resistance Temperature Detector
- **SCI**: Serial Communication Interface
- **SDRAM**: Synchronous DRAM
- **SGI**: Silicon Graphics Inc.
- **SHA**: Secure Hash Algorithm
- **SRAM**: Static RAM
- **SVR4**: System V Release 4.0
- **TFM**: Trabajo de Fin de Máster
- **UNIX**: Sistema Operativo
- **Virsh**: Virtual Shell
- **WinXPSP2**: Windows XP Service Pack 2

6. Bibliografía

[1] Waits Cal, Akinyele Joseph Ayo, Nolan Richard, Rogers Larry. (2008). *Computer Forensics: Results of Live Response Inquiry vs. Memory Image Analysis.* [en línea] https://apps.dtic.mil/dtic/tr/fulltext/u2/a488423.pdf

[2] Hidalgo Cajo Iván Mesias, Yasaca Pucuna Saul, Hidalgo Cajo Byron Geovanny. (2019) *Evidencias Digitales en la Investigación Forense Informática.* http://cimogsys.espoch.edu.ec/direccion-publicaciones/public/docs/books/2019-09-19-211754-91%20Evidencias%20Digitales%20en%20la%20Investigacio%CC%81n%20Forense%20Informa%CC%81tica.pdf

[3] Cai Liming, Sha Jing, Qian Wei. (2013). *Study on Forensic Analysis of Physical Memory.* [en línea] http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.1032.5847&rep=rep1&type=pdf

[4] Huber1 Manuel, Taubmann Benjamin, Wessel1 Sascha, Reiser Hans P., Sigl Georg. EURASIP Journal on Information Security (2016) 2016:17. *A flexible framework for mobile device forensics based on cold boot attacks.* DOI 10.1186/s13635-016-0041-4

[5] Cirani Simone, Ferrari Gianluiggi, Picone Marco, Veltri Luca. (2018). *Internet of Things: Architectures, Protocols and Standards.*

[6] Mohamed Ahmed Hail. (2019). *IoT-NDN: An IoT Architecture via Named Data Netwoking (NDN).* [en línea] https://ieeexplore-ieee-org.biblioteca-uoc.idm.oclc.org/document/8784859

[7] Khan Jamil Y., Yuce Mehmet R. (2019). *Internet of Things (IoT): Systems and Applications.*

[8] López i Seuba Manel. (2019). *Internet de las cosas La transformación digital de la sociedad.*

[9] González García Antonio Jesús. (2017). *IoT: Dispositivos, tecnologías de transporte y aplicaciones.* [en línea] Barcelona: UOC http://openaccess.uoc.edu/webapps/o2/bitstream/10609/64286/3/agonzalezgarcia0TFM0617memoria.pdf. [Trabajo de fin de máster]

[10] *Volatility* [Fecha de consulta: 28 de febrero de 2020] <https://github.com/volatilityfoundation/volatility>

[11] Walters Aaron, Petroni Nick L. (2007). *Volatools: Integrating Volatile Memory Forensics into the Digital Investigation Process.* [en línea] https://www.blackhat.com/presentations/bh-dc-07/Walters/Paper/bh-dc-07-Walters-WP.pdf

[12] *Volatility foundation* [Fecha de consulta: 20 marzo de 2020] <https://www.volatilityfoundation.org/>

[13] Volatilitux [Fecha de consulta: 28 de febrero de 2020] <https://code.google.com/archive/p/volatilitux/>

[14] Challenge 2010. *Symposium sur la sécurité des technologies de l'information et des communications* [Fecha de consulta: 20 marzo de 2020] <https://www.sstic.org/2010/challenge/>

[15] Red Hat Crash Utility [Fecha de consulta: 28 de febrero de 2020] <http://people.redhat.com/anderson/>

[16] Rekall [Fecha de consulta: 28 de febrero de 2020] <http://www.rekall-forensic.com/>

[17] Draugr [Fecha de consulta: 28 de febrero de 2020] <https://code.google.com/archive/p/draugr/>

[18] DAMM - Differential Analysis of Malware in Memory [Fecha de consulta: 28 de marzo de 2020] <https://github.com/504ensicsLabs/DAMM>

[19] inVtero.net [Fecha de consulta: 28 de marzo de 2020] <https://github.com/ShaneK2/inVtero.net>

[20] Foriana [Fecha de consulta: 28 de febrero de 2020] <http://hysteria.cz/niekt0/fmem/doc/foriana.pdf>

[21] Forensic Analysis Toolkit (FATKit) [Fecha de consulta: 28 de febrero de 2020] <http://4tphi.net/fatkit/>

[22] Memoryze / AuditViewer [Fecha de consulta: 28 de febrero de 2020] <https://www.fireeye.com/services/freeware/memoryze.html>

[23] BlackLight [Fecha de consulta: 28 de marzo de 2020] <https://www.blackbagtech.com/products/blacklight>

[24] VMware Workstation Pro [Fecha de consulta: 08 de abril de 2020] <https://www.vmware.com>

[25] Raspberry Pi [Fecha de consulta: 08 de abril de 2020] <https://www.raspberrypi.org>

[26] Raspberry Pi Desktop [Fecha de consulta: 10 de abril de 2020] <https://www.raspberrypi.org/downloads/raspberry-pi-desktop/>

[27] QEMU [Fecha de consulta: 15 de abril de 2020] <https://www.qemu.org>

[28] Autopsy [Fecha de consulta: 25 de Mayo de 2020]
<https://www.autopsy.com/>
<http://www.sleuthkit.org/autopsy/>

[29] Garfinkel, Simson, Digital media triage with bulk data analysis and bulk_extractor. Computers and Security 32: 56-72 (2013) https://simson.net/clips/academic/2013.COSE.bulk_extractor.pdf

[30] Raspberry Pi Desktop images [Fecha de consulta: 10 de abril de 2020]
<https://downloads.raspberrypi.org/rpd_x86/images/>

[31] Upgrade Kernel [Fecha de consulta: 10 de abril de 2020]
<https://linuxhint.com/how-to-upgrade-kernel-of-debian-9-stretch-from-source/>

[32] Actualizar kernel [Fecha de consulta: 10 de abril de 2020]
<https://www.creadpag.com/2018/07/actualizar-kernel-de-debian.html>

[33] Kernel Linux [Fecha de consulta: 10 de abril de 2020]
<https://cdn.kernel.org/pub/linux/kernel/v4.x/>

[34] Raspbian [Fecha de consulta: 20 de abril de 2020]
<https://www.raspberrypi.org/downloads/raspbian/>

[35] Qemu rpi kernel [Fecha de consulta: 20 de abril de 2020]
<https://github.com/dhruvvyas90/qemu-rpi-kernel>

[36] System.map para Raspberrypi [Fecha de consulta: 20 de abril de 2020]
<https://github.com/raspberrypi/firmware/tree/master/extra>
<https://github.com/RPi-Distro/firmware/tree/debian/extra>

[37] Raspberry Pi raspbian images [Fecha de consulta: 10 de Mayo de 2020]
<https://downloads.raspberrypi.org/raspbian_lite/images>

[38] Raspberry kernel [Fecha de consulta: 10 de Mayo de 2020]
<https://github.com/raspberrypi/linux/tags>

[39] Archivo Raspberry Pi [Fecha de consulta: 10 de Mayo de 2020]
<http://archive.raspberrypi.org/debian/pool/main/r/raspberrypi-firmware/>

[40] Identificar tarjeta de crédito [Fecha de consulta: 25 de Mayo de 2020]
<https://www.thebalance.com/what-do-the-numbers-on-your-credit-card-mean-4588401>

7. Anexos

7.1 Comandos Linux

Descripción de comandos de línea más utilizados

Comando	Utilidad / ejemplo
hostname	Muestra el nombre de la estación de trabajo $ hostname
whoami	Muestra el usuario actual de trabajo $ whoami
uname	Muestra información del sistema $ uname -a
uptime	Informa cuanto tiempo el sistema está activo $ uptime
date	Muestra la fecha y hora del sistema $ date
env	Muestra las variables de entorno del sistema $ env
pwd	Muestra el directorio donde se encuentra actualmente $ pwd
ls	Lista el contenido de directorio $ ls -l
lsof	Lista los archivos abiertos $ lsof
ps	Lista todos los procesos que se están ejecutando $ ps -aux
pstree	Lista procesos que están ejecutándose en forma de árbol $ pstree
pmap	Informa mapa de memoria de un proceso usando su PID $ pmap 1718
top	Información activa de procesos ejecutándose, salir con q $ top
vmstat	Muestra estadísticas de uso de memoria, cpu, disco, etc. $ vmstat
sar	Entrega reporte de actividad del sistema $ sar
kill	Termina un proceso indicando su PID $ kill 4955
vi(m)	Herramienta de edición de archivos $ vi /etc/hosts
nano	Herramienta de edición de archivos $ nano /etc/fstab
cat	Muestra el contenido de un archivo $ cat /etc/hosts
more	Muestra el contenido de un archivo por páginas $ more /etc/hosts

file	Muestra el tipo del archivo $ file /etc/hosts
mkdir	Crear directorios $ mkdir prueba
cp	Copia archivo o directorio (opción -r para directorios) $ cp -r prueba dir2
rm	Elimina archivos o directorios (opción -r para directorios) $ rm -r dir2
mv	Mueve y/o renombra archivos $ mv archivo1 nuevonombre
free	Muestra información sobre la memoria del sistema $ free -m
df	Muestra información del disco $ df -h
head	Muestra las primeras 10 líneas del archivo $ head /proc/meminfo
tail	Muestra las ultimas 10 líneas del archivo $ tail /proc/meminfo
wget	Permite realizar descarga de archivos $ wget https://google.com/index.html
ifconfig	Muestra información sobre la tarjeta de red $ ifconfig -a
iptables	Administra el filtrado de paquetes $ iptables -L
netstat	Muestra conexiones de red, tablas de ruteo, etc $ netstat -plnt
lsmod	Muestra el estado de los módulos cargados $ lsmod
lscpu	Muestra información sobre la arquitectura del CPU $ lscpu
lsblk	Muestra las unidades de disco y sus particiones $ lsblk
find	Busca un archivo en el sistema $ find / -name hosts
grep	Busca una palabra dentro de los archivos $ grep -r centos /boot/
yum	Gestor de programas (instala, borra, actualiza, etc.) $ yum install htop
strings	Muestra caracteres en formato legible de cualquier archivo $ strings /bin/ls
zip	Empaqueta y comprime archivos $ zip comprimido.zip archivo1 archivo2
unzip	Extrae los archivos comprimidos del formato zip $ unzip comprimido.zip
tar	Empaqueta o extrae archivos con o sin compresión $ tar -xf linux-4.19.75.tar.xz

Tabla 3: Comandos Linux para el análisis de la ejecución de memoria

7.2 Sintaxis de herramientas

7.2.1 Uso de Volatility

Los comandos básicos se construyen como se muestra:

$ python vol.py *[plugin]* -f *[image]* --profile=*[profile]*

$ volatility *[plugin]* -f *[image]* --profile=*[profile]*

Donde:

[plugin] complemento que señala la acción a realizar sobre la imagen.
[image] es la ruta de archivo con la imagen de la memoria.
[profile] indica el tipo de sistema operativo del cual proviene el volcado de memoria para saber qué estructuras de datos, algoritmos y símbolos usar internamente el perfil por defecto es WinXPSP2x86.

```
[root@larmasc ~]# volatility linux_banner -f memdump.dd --profile=LinuxCentOS76x64
Volatility Foundation Volatility Framework 2.6.1
Linux version 3.10.0-957.el7.x86_64 (mockbuild@kbuilder.bsys.centos.org) (gcc version
4.8.5 20150623 (Red Hat 4.8.5-36) (GCC) ) #1 SMP Thu Nov 8 23:39:32 UTC 2018
```

7.2.2 Uso de Volatilitux

La sintaxis principal es:

$ volatilitux.py -f <*dumpfile*> [-c <*configfile*>] [-o] [-d] <*command*> [opciones]

Donde:

<*dumpfile*> ruta del archivo de volcado de memoria a analizar.
<*configfile*> archivo de configuración sobre la imagen a analizar.
<*command*> comando a ejecutar sobre la imagen.
[opciones] parámetros adicionales para algunos comandos.

Comandos

Soporta los siguientes comandos:

- filedmp volcar un archivo abierto.
- filelist imprime la lista de todos los archivos abiertos para proceso determinado.
- memdmp volcar la memoria direccionable de un proceso.
- memmap imprime el mapa de memoria de un proceso.
- pslist imprime la lista de todos los procesos.

```
[root@larmasc trunk]# python volatilitux.py -f /home/centos/Downloads/challv2 pslist
Name                 PID       PPID
swapper              0         0
init                 1         0
kthreadd             2         0
ksoftirqd/0          3         2
events/0             4         2
khelper              5         2
suspend              6         2
kblockd/0            7         2
cqueue               8         2
kseriod              9         2
kmmcd                10        2
pdflush              11        2
pdflush              12        2
kswapd0              13        2
aio/0                14        2
mtdblockd            21        2
hid_compat           22        2
rpciod/0             23        2
mmcqd                24        2
sh                   25        1
servicemanager       26        1
vold                 27        1
....
...
```

7.2.3 Uso de Red Hat Crash Utility

Para ingresar a la utilidad Crash se debe ejecutar:

$ crash /usr/lib/debug/lib/modules/<*release*>/vmlinux \
/var/crash/<*registro*>/vmcore
crash>

Donde:
<*release*> es el valor del núcleo liberado
<*registro*> es la IP y la fecha de creación del archivo de imagen

```
[root@larmasc ~]# crash
      KERNEL: /usr/lib/debug/lib/modules/3.10.0-957.el7.x86_64/vmlinux
    DUMPFILE: /dev/crash
        CPUS: 1
        DATE: Fri May 15 20:43:31 2020
      UPTIME: 17:09:07
LOAD AVERAGE: 0.30, 0.38, 0.60
       TASKS: 477
    NODENAME: larmasc
     RELEASE: 3.10.0-957.el7.x86_64
     VERSION: #1 SMP Thu Nov 8 23:39:32 UTC 2018
     MACHINE: x86_64  (2712 Mhz)
      MEMORY: 1 GB
         PID: 22667
     COMMAND: "crash"
        TASK: ffff95fa82046180  [THREAD_INFO: ffff95fa8f190000]
         CPU: 0
       STATE: TASK RUNNING (ACTIVE)
crash> net
   NET_DEVICE     NAME   IP ADDRESS(ES)
ffff95fabc6ba000  lo     127.0.0.1
ffff95fabb370000  ens33
ffff95fabb352000  virbr0 192.168.122.1
ffff95fa8f0dc000  virbr0-nic
```

7.2.4 Uso de Rekall

La sintaxis de la herramienta es:

$ rekal -f <*image*> <*command*> *[opcion]*

Donde:

<*image*> ruta del archivo de volcado de memoria.
<*command*> comando a ejecutar sobre el volcado de memoria.
[opcion] parámetro opcional para el comando.

7.2.5 Uso de DAMM

La sintaxis de la herramienta es:

$ python damm.py --profile *[profile]* -f *[image]* -p *[plugin]*

Donde:

[profile] tipo de perfil para la imagen.
[image] ruta de archivo con la imagen de la memoria.
[plugin] complemento que señala la acción a realizar sobre la imagen.

7.2.6 Uso de Foriana

La sintaxis de la herramienta es:

$ foriana *[options] [image]*

Donde:

[options] parámetros varios a ejecutar sobre la imagen.
[image] ruta de archivo con la imagen de la memoria.

7.3 Información sobre la muestra de memoria

Aplicando el comando file de Linux sobre la muestra de memoria se obtiene que el archivo es de tipo PCX que es un formato de imagen digital.

file standard_IoT.memdump

```
[root@larmasc ~]# file standard_IoT.memdump
standard_IoT.memdump: PCX ver. 2.5 image data
```

Para obtener información del origen de la muestra de memoria se usan los comandos Linux strings que permite mostrar en caracteres legibles el contenido de cualquier archivo y el comando grep que permite filtrar solo las líneas que contengan determinada palabra.

strings standard_IoT.memdump | grep -i architecture

```
[root@larmasc ~]# strings standard_IoT.memdump | grep -i architecture
cimplementor %02x architecture %d part %02x variant %x rev %x
CPU architecture: %s
6hw-breakpoint: debug architecture 0x%x unsupported.
Architecture: all
Architecture: all
Architecture: armhf
Architecture: armhf
CPU architecture: 7
CPU architecture: 7
Detected architecture arm.
[root@larmasc ~]#
```

strings standard_IoT.memdump | grep -i hardwareversion

```
[root@larmasc ~]# strings standard_IoT.memdump | grep -i hardwareversion
      "hardwareVersion"
        "hardwareVersion": "Pi 4 Model B v1.1",
        "hardwareVersion": "Pi 4 Model B v1.1",
```

strings standard_IoT.memdump | grep "Raspberry"

```
[root@larmasc ~]# strings standard_IoT.memdump | grep "Raspberry"
Raspberry Pi 4 Model B Rev 1.1
OF: fdt: Machine model: Raspberry Pi 4 Model B Rev 1.1
Raspberry Pi Foundation
Model           : Raspberry Pi 4 Model B Rev 1.1
```

strings standard_IoT.memdump | grep "raspbian"

```
[root@larmasc ~]# strings standard_IoT.memdump | grep "raspbian"
://raspbian.raspberrypi.org/raspbian/dists/buster/
://raspbian.raspberrypi.org/raspbian
aspbian.raspberrypi.org/raspbian/dists/buster/
/var/lib/apt/lists/raspbian.raspberrypi.org_raspbian_dists_buster_non-free_binary-armhf_Packages
/etc/dpkg/origins/raspbian
/usr/share/keyrings/raspbian-archive-keyring.gpg
/usr/share/doc/raspbian-archive-keyring
```

strings standard_IoT.memdump | grep -i "linux version"

```
[root@larmasc ~]# strings standard_IoT.memdump | grep -i "linux version"
Linux version 4.19.75-v7l+ (dom@buildbot) (gcc version 4.9.3 (crosstool-NG crosstool-ng-
1.22.0-88-g8460611)) #1270 SMP Tue Sep 24 18:51:41 BST 2019
```

7.4 Raspberry Pi Desktop

Terminada la instalación de Raspberry Pi Desktop en una máquina virtual obtenemos información del sistema operativo:

```
root@raspberry:~# uname -a
Linux raspberry 4.19.0-8-amd64 #1 SMP Debian 4.19.98-1 (2020-01-26) x86_64 GNU/Linux
root@raspberry:~#
root@raspberry:~# cat /etc/debian_version
10.3
root@raspberry:~# cat /proc/version
Linux version 4.19.0-8-amd64 (debian-kernel@lists.debian.org) (gcc version 8.3.0 (Debian 8.3.0-6))
 #1 SMP Debian 4.19.98-1 (2020-01-26)
root@raspberry:~#
root@raspberry:~# cat /proc/cpuinfo
processor       : 0
vendor_id       : GenuineIntel
cpu family      : 6
model           : 142
model name      : Intel(R) Core(TM) i5-7200U CPU @ 2.50GHz
stepping        : 9
microcode       : 0xc6
cpu MHz         : 2712.005
cache size      : 3072 KB
physical id     : 0
siblings        : 1
core id         : 0
cpu cores       : 1
apicid          : 0
initial apicid  : 0
fpu             : yes
fpu_exception   : yes
cpuid level     : 22
wp              : yes
```

Usando el gestor de paquetes se instala la herramienta Volatility para buscar e instalar todas las dependencias necesarias
apt install volatility

Se crea el archivo module.dwarf con las estructuras de datos del kernel

```
root@raspberry:/home/pi# cd /usr/share/volatility/tools/linux/
root@raspberry:/usr/share/volatility/tools/linux#
root@raspberry:/usr/share/volatility/tools/linux# make
make -C //lib/modules/4.19.0-8-amd64/build CONFIG_DEBUG_INFO=y M="/usr/share/volatility/tools/linu
x" modules
make[1]: Entering directory '/usr/src/linux-headers-4.19.0-8-amd64'
  CC [M]  /usr/share/volatility/tools/linux/module.o
  Building modules, stage 2.
  MODPOST 1 modules
WARNING: modpost: missing MODULE_LICENSE() in /usr/share/volatility/tools/linux/module.o
see include/linux/module.h for more information
  CC      /usr/share/volatility/tools/linux/module.mod.o
  LD [M]  /usr/share/volatility/tools/linux/module.ko
make[1]: Leaving directory '/usr/src/linux-headers-4.19.0-8-amd64'
dwarfdump -di module.ko > module.dwarf
make -C //lib/modules/4.19.0-8-amd64/build M="/usr/share/volatility/tools/linux" clean
make[1]: Entering directory '/usr/src/linux-headers-4.19.0-8-amd64'
  CLEAN   /usr/share/volatility/tools/linux/.tmp_versions
  CLEAN   /usr/share/volatility/tools/linux/Module.symvers
make[1]: Leaving directory '/usr/src/linux-headers-4.19.0-8-amd64'
```

Se genera el perfil con nombre Raspberry41908.zip juntando el archivo creado module.dwarf y el archivo System.map que contiene la tabla de símbolos del sistema operativo para usarlo con la herramienta Volatility

```
root@raspberry:/usr/share/volatility/tools/linux# zip Raspberry41908.zip module.dwarf /boot/System
.map-4.19.0-8-amd64
  adding: module.dwarf (deflated 91%)
  adding: boot/System.map-4.19.0-8-amd64 (deflated 79%)
root@raspberry:/usr/share/volatility/tools/linux#
root@raspberry:/usr/share/volatility/tools/linux# mv Raspberry41908.zip /usr/lib/python2.7/dist-pa
ckages/volatility/plugins/overlays/linux/
```

Se verifica que la herramienta Volatility esta considerando el perfil agregado

```
root@raspberry:~# volatility --info | grep Linux
Volatility Foundation Volatility Framework 2.6
linux_aslr_shift            - Automatically detect the Linux ASLR shift
linux_banner                - Prints the Linux banner information
linux_yarascan              - A shell in the Linux memory image
LinuxAMD64PagedMemory          - Linux-specific AMD 64-bit address space.
LinuxRaspberry41908x64 - A Profile for Linux Raspberry41908 x64
```

Se descarga el volcado de memoria para analizarlo con el perfil generado pero no es posible obtener información

```
root@raspberry:~# volatility --profile=LinuxRaspberry41908x64 -f standard_IoT.memdump linux_banner
Volatility Foundation Volatility Framework 2.6
No suitable address space mapping found
Tried to open image as:
 MachOAddressSpace: mac: need base
```

Se copia el perfil en el ambiente de prueba y tampoco funciona

```
[root@larmasc ~]# volatility --profile=LinuxRaspberry41908x64 -f standard_IoT.memdump linux_banner
Volatility Foundation Volatility Framework 2.6.1
No suitable address space mapping found
Tried to open image as:
 MachOAddressSpace: mac: need base
```

7.5 Raspberry Pi Desktop versión 4.19.75

Se usa Raspberry Pi Desktop con kernel 4.9.0-8 [30] porque las pruebas con versiones de kernel más recientes daban problemas con la actualización

```
root@raspberry:~# uname -a
Linux raspberry 4.9.0-8-amd64 #1 SMP Debian 4.9.144-3.1 (2019-02-19) x86_64 GNU/Linux
root@raspberry:~#
root@raspberry:~# cat /etc/debian_version
9.8
root@raspberry:~# cat /proc/version
Linux version 4.9.0-8-amd64 (debian-kernel@lists.debian.org) (gcc version 6.3.0 20170516 (Debian 6
.3.0-18+deb9u1) ) #1 SMP Debian 4.9.144-3.1 (2019-02-19)
```

Siguiendo los tutoriales de internet sobre la actualización de kernel [31, 32] se procede a instalar las actualizaciones, librerías y paquetes necesarios
apt update
apt install git fakeroot build-essential ncurses-dev xz-utils libssl-dev bc
apt install gcc libncurses5-dev dpkg-dev
apt install bison flex libelf-dev

Se descarga desde [33] el kernel Linux versión 4.19.75 y se descomprime
wget https://cdn.kernel.org/pub/linux/kernel/v4.x/linux-4.19.75.tar.xz
tar -xf linux-4.19.75.tar.xz
cd linux-4.19.75

Se copia el archivo de configuración del sistema para usarlo en la actualización
cp -v /boot/config-4.9.0-8-amd64 .config
make menuconfig

Se realiza la compilación del kernel para generar los archivos .deb que son los que instalamos al sistema para actualizar el kernel concluido se reinicia el sistema y se verifica la actualización realizada
make -j 5 KDEB_PKGVERSION=1.test deb-pkg
dpkg -i ../linux*.deb
reboot

```
root@raspberry:~# uname -a
Linux raspberry 4.19.75 #1.test SMP Thu Apr 30 11:01:56 HDT 2020 x86_64 GNU/Linux
root@raspberry:~#
root@raspberry:~# cat /etc/debian_version
9.8
root@raspberry:~# cat /proc/version
Linux version 4.19.75 (root@raspberry) (gcc version 6.3.0 20170516 (Debian 6.3.0-18+deb9u1)) #1.te
st SMP Thu Apr 30 11:01:56 HDT 2020
```

Usando la herramienta Volatility se genera el perfil como Raspbian49-419.zip usando los archivos module.dwarf y el archivo de mapa del sistema operativo

```
root@raspberry:/usr/share/volatility/tools/linux# make
make -C //lib/modules/4.19.75/build CONFIG_DEBUG_INFO=y M="/usr/share/volatility/tools/linux" modu
les
make[1]: Entering directory '/usr/src/linux-headers-4.19.75'
  CC [M]  /usr/share/volatility/tools/linux/module.o
  Building modules, stage 2.
  MODPOST 1 modules
WARNING: modpost: missing MODULE_LICENSE() in /usr/share/volatility/tools/linux/module.o
see include/linux/module.h for more information
  CC      /usr/share/volatility/tools/linux/module.mod.o
  LD [M]  /usr/share/volatility/tools/linux/module.ko
make[1]: Leaving directory '/usr/src/linux-headers-4.19.75'
dwarfdump -di module.ko > module.dwarf
make -C //lib/modules/4.19.75/build M="/usr/share/volatility/tools/linux" clean
make[1]: Entering directory '/usr/src/linux-headers-4.19.75'
  CLEAN   /usr/share/volatility/tools/linux/.tmp_versions
  CLEAN   /usr/share/volatility/tools/linux/Module.symvers
make[1]: Leaving directory '/usr/src/linux-headers-4.19.75'
root@raspberry:/usr/share/volatility/tools/linux#
root@raspberry:/usr/share/volatility/tools/linux# ll
total 2852
drwxr-xr-x 2 root root    4096 Apr 30 17:36 kcore
-rw-r--r-- 1 root root     384 Dec 27  2016 Makefile
-rw-r--r-- 1 root root     314 Dec 27  2016 Makefile.enterprise
-rw-r--r-- 1 root root   16061 Dec 27  2016 module.c
-rw-r--r-- 1 root root 2889008 Apr 30 17:37 module.dwarf
root@raspberry:/usr/share/volatility/tools/linux# zip Raspbian49-419.zip module.dwarf /boot/System
.map-4.19.75
  adding: module.dwarf (deflated 91%)
  adding: boot/System.map-4.19.75 (deflated 79%)
```

Se realiza el análisis de memoria usando el nuevo perfil en el ambiente de prueba pero no es satisfactorio

```
[root@larmasc ~]# volatility linux_pslist -f standard_IoT.memdump --profile=LinuxRaspbian49-419x64
Volatility Foundation Volatility Framework 2.6.1
Offset            Name                 Pid             PPid            Uid             Gid    DTB                Start Time
------------------ -------------------- --------------- --------------- --------------- ------ ------------------ ----------
No suitable address space mapping found
Tried to open image as:
```

7.6 Raspberry Pi emulado

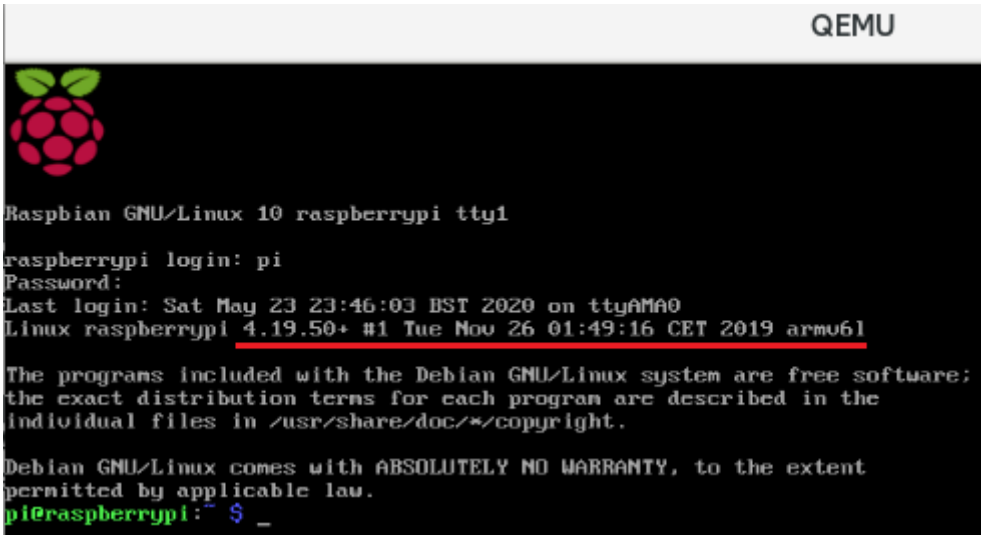
En una máquina virtual se instala el programa de emulación Qemu
apt install qemu

Se descarga la última versión de Raspbian [34] y al descomprimirlo se obtiene la imagen 2020-02-13-raspbian-buster-lite.img
wget https://downloads.raspberrypi.org/raspbian_lite_latest
unzip raspbian_lite_latest

Se descargan desde [35] el kernel Linux 4.19.50 y el archivo del árbol de dispositivos (.dtb) ya compilados para usarlo en la emulación
wget https://github.com/dhruvvyas90/qemu-rpi-kernel/raw/master/kernel-qemu-4.19.50-buster
wget https://github.com/dhruvvyas90/qemu-rpi-kernel/raw/master/versatile-pb-buster.dtb

Para realizar la emulación de Raspberry Pi se ejecuta el siguiente comando donde se junta Raspbian y todos los archivos descargados

qemu-system-arm -kernel kernel-qemu-4.19.50-buster -append "rw console=ttyAMA0 root=/dev/sda2 rootfstype=ext4" -drive file=2020-02-13-raspbian-buster-lite.img,index=0,media=disk,format=raw -dtb versatile-pb-buster.dtb -cpu arm1136 -m 256 -M versatilepb -no-reboot -serial stdio -redir tcp:5022::22



Se verifica la versión del kernel del sistema operativo levantado

```
root@raspberrypi:~# uname -a
Linux raspberrypi 4.19.50+ #1 Tue Nov 26 01:49:16 CET 2019 armv6l GNU/Linux
root@raspberrypi:~#
root@raspberrypi:~# cat /etc/debian_version
10.3
root@raspberrypi:~# cat /proc/version
Linux version 4.19.50+ (niklas@ubuntu) (gcc version 9.2.1 20191008 (Ubuntu 9.2.1
-9ubuntu2)) #1 Tue Nov 26 01:49:16 CET 2019
```

Se comprueba que la versión de la arquitectura ARM emulada es la v6

```
root@raspberrypi:~# cat /proc/cpuinfo
processor       : 0
model name      : ARMv6-compatible processor rev 3 (v6l)
BogoMIPS        : 706.45
Features        : swp half thumb fastmult vfp edsp java tls
CPU implementer : 0x41
CPU architecture: 6TEJ
CPU variant     : 0x1
CPU part        : 0xb36
CPU revision    : 3

Hardware        : ARM-Versatile (Device Tree Support)
Revision        : 0000
Serial          : 0000000000000000
```

Se instala la herramienta Volatility para crear el archivo module.dwarf con la estructura correspondiente al sistema operativo.

Fue necesario descargar de los sitios [36] el archivo de mapa del sistema porque la arquitectura ARM no lo tiene en el sistema operativo como los sistemas con arquitectura x86 juntando ambos archivos se generan los perfiles con nombres Arm41950.zip y Arm41950*.zip para cada archivo System.map

Se mueven los perfiles al ambiente de prueba y se realiza la verificación con la herramienta Volatility reconociéndolos como perfiles de arquitectura ARM

```
Volatility Foundation Volatility Framework 2.6.1
LinuxArm41950ARM                - A Profile for Linux Arm41950 ARM
LinuxArm41950rdARM              - A Profile for Linux Arm41950rd ARM
LinuxArm41950v7ARM              - A Profile for Linux Arm41950v7 ARM
LinuxArm41950v7lARM             - A Profile for Linux Arm41950v7l ARM
LinuxArm41950v7lrdARM           - A Profile for Linux Arm41950v7lrd ARM
LinuxArm41950v7rdARM            - A Profile for Linux Arm41950v7rd ARM
LinuxArm41950v8rdx64            - A Profile for Linux Arm41950v8rd x64
LinuxArm41950v8x64              - A Profile for Linux Arm41950v8 x64
```

Se realiza el análisis del volcado de memoria pero no son satisfactorios

```
[root@larmasc ~]# volatility -f standard_IoT.memdump --profile=LinuxArm41950ARM linux_banner
Volatility Foundation Volatility Framework 2.6.1
No suitable address space mapping found
```

7.7 Raspberry Pi emulado versión 4.19.75-v7l+

Se descarga la versión 4.19.75-v7+ de Raspbian [37] al descomprimirlo se obtiene la imagen 2019-09-26-raspbian-buster-lite.img que fue renombrado como buster-lite-2019-09-26.img

wget https://downloads.raspberrypi.org/raspbian_lite/images/raspbian_lite-2019-09-30/2019-09-26-raspbian-buster-lite.zip
unzip 2019-09-26-raspbian-buster-lite.zip
mv 2019-09-26-raspbian-buster-lite.img buster-lite-2019-09-26.img

Se descarga desde [38] el kernel Raspberry versión 4.19.75 y se descomprime
wget https://github.com/raspberrypi/linux/archive/raspberrypi-kernel_1.20190925-1.zip
unzip raspberrypi-kernel_1.20190925-1.zip
cd linux-raspberrypi-kernel_1.20190925-1

Se exportan las variables de entornos
export ARCH=arm
export CROSS_COMPILE=arm-linux-gnueabihf-

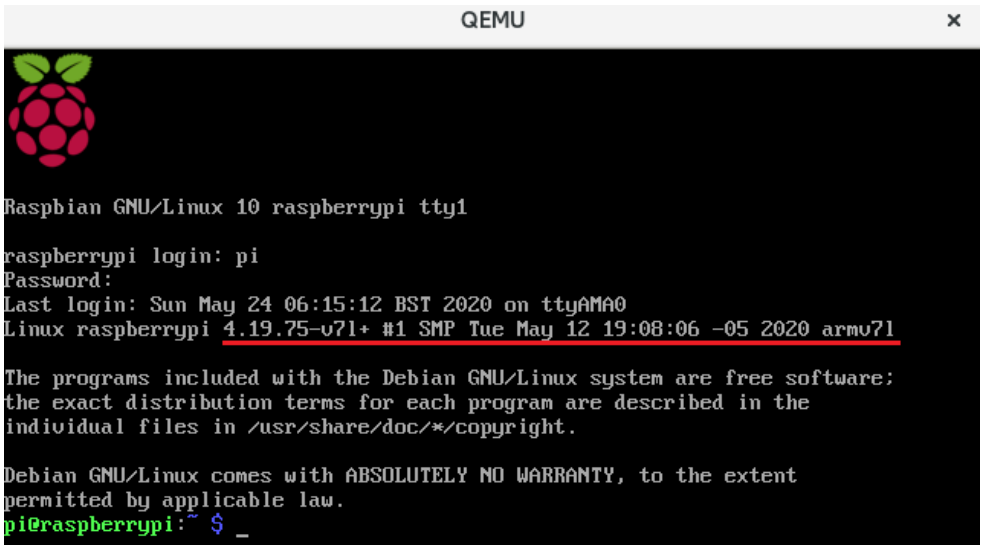
Se agrega en el archivo Makefile el valor -v7l+ en el campo extraversión
VERSION = 4
PATCHLEVEL = 19
SUBLEVEL = 75
EXTRAVERSION = -v7l+

Se genera el archivo de configuración .config para vexpress y se compila
make vexpress_defconfig
make all

Se extraen los archivos necesarios para realizar la emulación de Raspberry Pi con la arquitectura ARMv7l y el archivo System.map creado
cp arch/arm/boot/zImage /root/rpi/
cp arch/arm/boot/dts/vexpress-v2p-ca15_a7.dtb /root/rpi/
cp arch/arm/boot/dts/vexpress-v2p-ca9.dtb /root/rpi/
cp System.map /root/rpi/
cd /root/rpi/
mv zImage kernel-qemu-4.19.75v7l-buster

Se realiza la emulación de Raspberry Pi ejecutando el siguiente comando

qemu-system-arm -M vexpress-a15 -m 512 -kernel kernel-qemu-4.19.75v7l-buster -dtb vexpress-v2p-ca15_a7.dtb -sd buster-lite-2019-09-26.img -append "console=ttyAMA0 root=/dev/mmcblk0p2" -serial stdio -net nic -net user,hostfwd=tcp::2222-:22



Se verifica la versión del kernel del sistema operativo

```
root@raspberrypi:~# uname -a
Linux raspberrypi 4.19.75-v7l+ #1 SMP Tue May 12 19:08:06 -05 2020 armv7l
GNU/Linux
root@raspberrypi:~#
root@raspberrypi:~# cat /etc/debian_version
10.1
root@raspberrypi:~# cat /proc/version
Linux version 4.19.75-v7l+ (root@Leoncio) (gcc version 6.3.0 20170516 (Deb
ian 6.3.0-18)) #1 SMP Tue May 12 19:08:06 -05 2020
```

Se comprueba que la versión de la arquitectura ARM emulada es la v7

```
root@raspberrypi:~# cat /proc/cpuinfo
processor       : 0
model name      : ARMv7 Processor rev 1 (v7l)
BogoMIPS        : 125.00
Features        : half thumb fastmult vfp edsp neon vfpv3 tls vfpv4 idiva
idivt vfpd32 lpae evtstrm
CPU implementer : 0x41
CPU architecture: 7
CPU variant     : 0x2
CPU part        : 0xc0f
CPU revision    : 1

Hardware        : ARM-Versatile Express
Revision        : 0000
Serial          : 0000000000000000
Model           : V2P-CA15_CA7
```

Se descarga desde [39] el kernel-headers versión 4.19.75 y se instala
wget http://archive.raspberrypi.org/debian/pool/main/r/raspberrypi-firmware/raspberrypi-kernel-headers_1.20190925-1_armhf.deb
dpkg -i raspberrypi-kernel-headers_1.20190925-1_armhf.deb

Del sistema operativo se crea el archivo System.map usando el contenido del archivo kallsyms que contiene el listado de todos los símbolos del kernel
cat /proc/kallsyms > System.map

Se instala la herramienta Volatility para crear el archivo module.dwarf luego se crean dos perfiles uno usando el archivo System.map creado del archivo kallsyms y el otro usando el archivo System.map generado en la compilación del kernel.

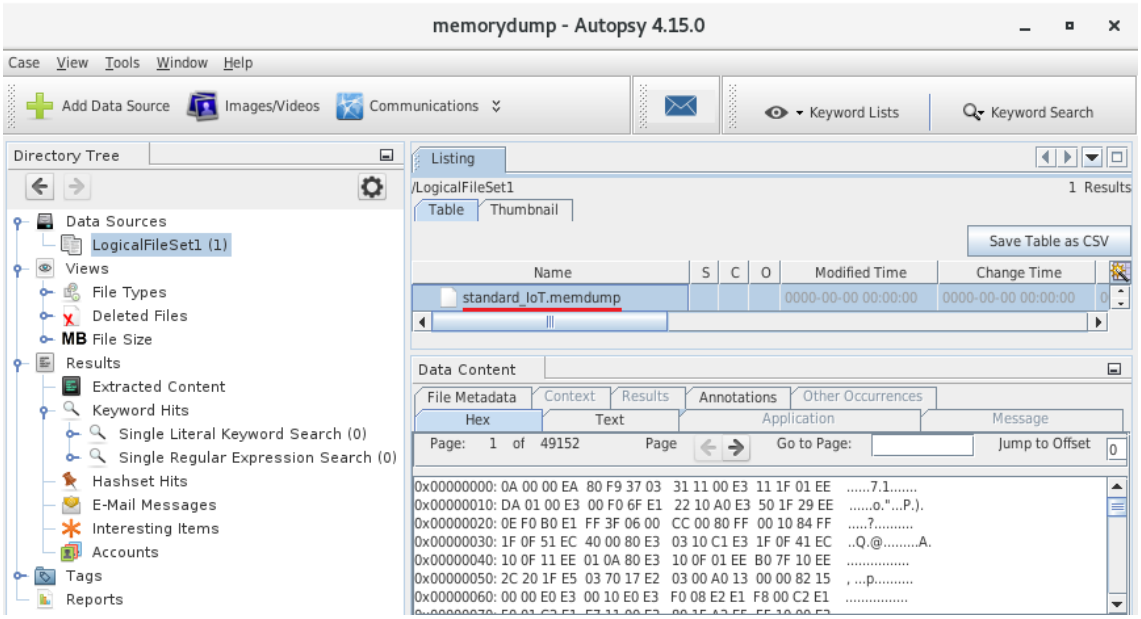
Se mueven los perfiles al ambiente de prueba para realizar el análisis de la muestra de volcado de memoria usando Volatility pero se obtiene el mensaje de siempre: No suitable address space mapping found por lo tanto los perfiles generados no corresponden a la muestra.

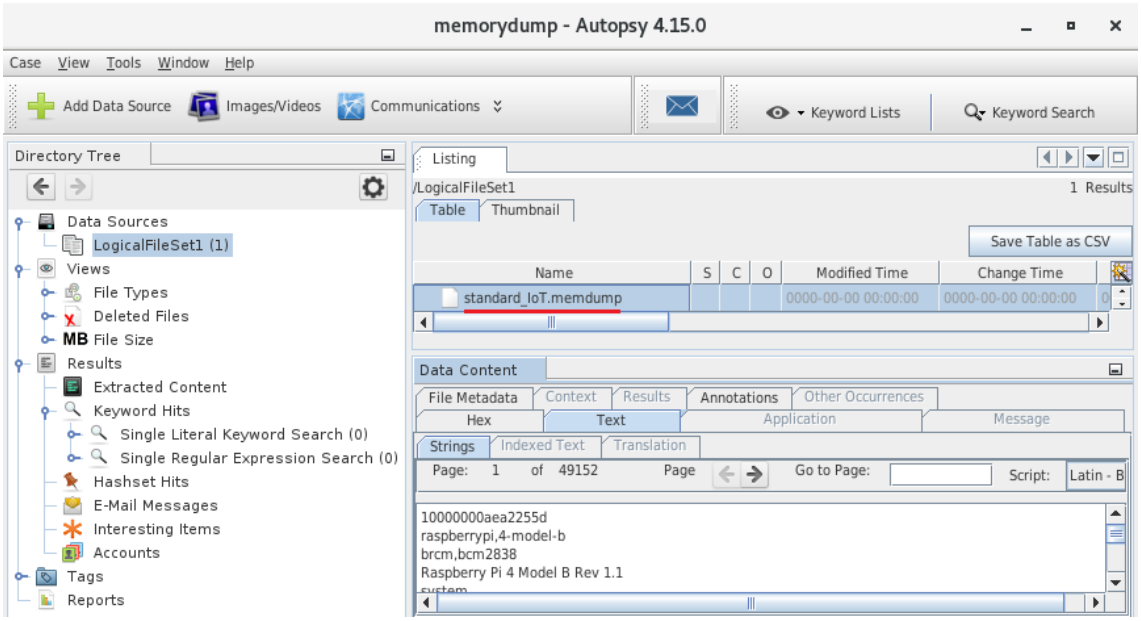
7.8 Herramienta Autopsy

En el servidor de prueba se instala la herramienta Autopsy
yum install autopsy

La herramienta usa una interfaz gráfica, se debe crear un caso donde se indica el nombre y el directorio base, el siguiente paso es agregar la fuente de dato para ello se selecciona el tipo de dato como Logical Files luego se selecciona y agrega el archivo de volcado de memoria.

Terminada la carga de la muestra se observa que la herramienta no alcanza a realizar el análisis detallado de su contenido sino que lo maneja como un todo proporcionando solo información en formato hexadecimal y de texto.





7.9 Herramienta Bulk Extractor

La sintaxis de la herramienta es:
bulk_extractor *[options] [imagefile]*

Donde:

[options] opciones varias a ejecutar sobre la imagen.
[imagefile] archivo de imagen de la memoria y especificar el directorio que se creará para guardar los archivos con la información obtenida.

En el servidor de prueba se instala la herramienta
yum install bulk_extractor

Se realiza la extracción de información del volcado de memoria especificando con -o el nombre de la carpeta que la herramienta creará para guardar todos los archivos y carpetas con los datos obtenidos

```
[root@larmasc ~]# bulk_extractor -o bulk standard_IoT.memdump
bulk_extractor version: 1.6.0
Hostname: larmasc
Input file: standard_IoT.memdump
Output directory: bulk
Disk Size: 805306368
Threads: 1
Attempt to open standard_IoT.memdump
20:40:24 Offset 67MB (8.33%) Done in  0:02:48 at 20:43:12
20:40:42 Offset 150MB (18.75%) Done in  0:02:26 at 20:43:08
20:41:03 Offset 234MB (29.17%) Done in  0:02:12 at 20:43:15
20:41:26 Offset 318MB (39.58%) Done in  0:01:58 at 20:43:24
20:41:46 Offset 402MB (50.00%) Done in  0:01:38 at 20:43:24
20:42:13 Offset 486MB (60.42%) Done in  0:01:21 at 20:43:34
20:42:38 Offset 570MB (70.83%) Done in  0:01:01 at 20:43:39
20:42:59 Offset 654MB (81.25%) Done in  0:00:39 at 20:43:38
20:43:20 Offset 738MB (91.67%) Done in  0:00:17 at 20:43:37
All data are read; waiting for threads to finish...
Time elapsed waiting for 1 thread to finish:
     (timeout in 60 min.)
All Threads Finished!
Producer time spent waiting: 168.214 sec.
Average consumer time spent waiting: 2.37028 sec.
*****************************************
** bulk_extractor is probably CPU bound. **
**    Run on a computer with more cores  **
**      to get better performance.       **
*****************************************
MD5 of Disk Image: 902c6088606eb9d9e7b9db90e217118e
Phase 2. Shutting down scanners
Phase 3. Creating Histograms
Elapsed time: 207.785 sec.
Total MB processed: 805
Overall performance: 3.87567 MBytes/sec (3.87567 MBytes/sec/thread)
Total email features found: 2117
```

En la carpeta bulk creada por la herramienta existen varios archivos de textos con diversos tipos de información recopilada de la muestra de memoria

```
[root@larmasc ~]# ls bulk/
aes_keys.txt                email.txt           json.txt            sqlite_carved.txt         url.txt
alerts.txt                  ether_histogram.txt kml.txt             telephone_histogram.txt   vcard.txt
ccn_histogram.txt           ether.txt           ntfsusn_carved.txt  telephone.txt             windirs.txt
ccn_track2_histogram.txt    exif.txt            packets.pcap        unrar_carved.txt          winlnk.txt
ccn_track2.txt              find_histogram.txt  pii_teamviewer.txt  unzip_carved.txt          winpe_carved.txt
ccn.txt                     find.txt            pii.txt             url_facebook-address.txt  winpe.txt
domain_histogram.txt        gps.txt             rar.txt             url_facebook-id.txt       winprefetch.txt
domain.txt                  httplogs.txt        report.xml          url_histogram.txt         zip.txt
elf.txt                     ip_histogram.txt    rfc822.txt          url_microsoft-live.txt
email_domain_histogram.txt  ip.txt              sin.txt             url_searches.txt
email_histogram.txt         jpeg_carved.txt     sqlite_carved       url_services.txt
```

El archivo ccn.txt contiene información de las tarjetas de crédito, se encuentra el número de una tarjeta Mastercard porque según [40] el primer digito es 5

```
# BANNER FILE NOT PROVIDED (-b option)
# BULK_EXTRACTOR-Version: 1.6.0 ($Rev: 10844 $)
# Feature-Recorder: ccn
# Filename: standard IoT.memdump
# Feature-File-Version: 1.1
612334098       5[redacted]6        us.com/features/5[redacted]6.\x00The
```

Los archivos domain_histogram.txt y domain.txt contienen información sobre los dominios visitados

```
# BANNER FILE NOT PROVIDED (-b option)
# BULK_EXTRACTOR-Version: 1.6.0 ($Rev: 10844 $)
# Feature-Recorder: domain
# Filename: standard IoT.memdump
# Histogram-File-Version: 1.1
n=69666 web.lfr
n=57513 api.endpoint
n=1762  127.0.0.1
n=1479  192.168.138.1
n=1344  192.168.10.53
n=1108  openssh.com
n=263   api.endp
n=205   lists.alioth.debian.org
n=169   debian.org
n=160   lists.debian.org
n=144   example.com
n=128   web
n=123   github.com
n=116   www.freedesktop.org
n=103   gmail.com
n=100   httpbin.org
n=87    www.google.com
n=84    accounts.google.com
n=80    google.com
n=76    www.w3.org
n=75    www.python.org
```

El archivo elf.txt contiene información de archivos ejecutables elf del sistema

```
# Feature-Recorder: elf
# Filename: standard IoT.memdump
# Feature-File-Version: 1.1
12587008        f82985adebf0ff2f419c0e6119cec781        <ELF class="ELFCLASS32" data="ELFDATA2LSB" osabi="ELFOSABI_NONE"
version="0" ><ehdr type="ET_DYN" machine="EM_ARM" version="1" entry="768" phoff="52" shoff="2176" flags="83886082" ehsize
2" phentsize="32" phnum="5" shentsize="40" shnum="15" shstrndx="14" /><sections><section name="" type="SHT_NULL" addr="0x
offset="0" size="0" link="0" info="0" addralign="0" shentsize="0"><flags></flags></section><section name=".hash" type="SH
ASH" addr="0xd4" offset="d4" size="28" link="2" info="0" addralign="4" shentsize="4"><flags><SHF_ALLOC /></flags></sectio
section name=".dynsym" type="SHT_DYNSYM" addr="0xfc" offset="fc" size="50" link="3" info="2" addralign="4" shentsize="10"
lags><SHF_ALLOC /></flags></section><section name=".dynstr" type="SHT_STRTAB" addr="0x14c" offset="14c" size="44" link="0
nfo="0" addralign="1" shentsize="0"><flags><SHF_ALLOC /></flags></section><section name=".gnu.version" type="SHT_GNU_vers
 addr="0x190" offset="190" size="a" link="2" info="0" addralign="2" shentsize="2"><flags><SHF_ALLOC /></flags></section><
tion name=".gnu.version_d" type="SHT_GNU_verdef" addr="0x19c" offset="19c" size="38" link="3" info="2" addralign="4" shen
ze="0"><flags><SHF_ALLOC /></flags></section><section name=".note" type="SHT_NOTE" addr="0x1d4" offset="1d4" size="24" li
"0" info="0" addralign="4" shentsize="0"><flags><SHF_ALLOC /></flags></section><section name=".dynamic" type="SHT_DYNAMIC
ddr="0x1f8" offset="1f8" size="80" link="3" info="0" addralign="4" shentsize="8"><flags><SHF_WRITE /><SHF_ALLOC /></flags
section><section name=".ARM.extab" type="SHT_PROGBITS" addr="0x278" offset="278" size="48" link="0" info="0" addralign="4
hentsize="0"><flags><SHF_ALLOC /></flags></section><section name=".ARM.exidx" type="" addr="0x2c0" offset="2c0" size="40"
nk="a" info="0" addralign="4" shentsize="0"><flags><SHF_ALLOC /><SHF_LINK_ORDER /></flags></section><section name=".text"
pe="SHT_PROGBITS" addr="0x300" offset="300" size="480" link="0" info="0" addralign="4" shentsize="0"><flags><SHF_ALLOC />
F_EXECINSTR /></flags></section><section name=".got.plt" type="SHT_PROGBITS" addr="0x780" offset="780" size="c" link="0"
o="0" addralign="4" shentsize="4"><flags><SHF_WRITE /><SHF_ALLOC /></flags></section><section name=".comment" type="SHT_P
```

Los archivos email.txt, email_histogram.txt y email_domain_histogram.txt guardan información relacionada a las direcciones de correos dominios de correos y registros del tráfico generado

```
# BANNER FILE NOT PROVIDED (-b option)
# BULK_EXTRACTOR-Version: 1.6.0 ($Rev: 10844 $)
# Feature-Recorder: email
# Filename: standard IoT.memdump
# Histogram-File-Version: 1.1
n=1108  @openssh.com
n=205   @lists.alioth.debian.org
n=168   @debian.org
n=160   @lists.debian.org
n=103   @gmail.com
n=45    @libssh.org
n=40    @lemburg.com
n=33    @proxy.example.com
n=27    @python.org
n=23    @tracker.debian.org
n=17    @conectiva.com
n=12    @lfw.org
n=11    @openssl.org
n=10    @lysator.liu.se
n=9     @red-dove.com
n=8     @qa.debian.org
n=7     @redivi.com
n=7     @ubuntu.com
n=7     @vger.kernel.org
n=6     @chambersign.org
n=6     @cwi.nl
```

Los archivos ether_histogram.txt y ether.txt guardan información de direcciones MAC de las tarjetas de red y su relación entre ellas

```
# BANNER FILE NOT PROVIDED (-b option)
# BULK_EXTRACTOR-Version: 1.6.0 ($Rev: 10844 $)
# Feature-Recorder: ether
# Filename: standard_IoT.memdump
# Feature-File-Version: 1.1
2301    DC:A6:32:2F:27:87       msc95xx.macaddr=DC:A6:32:2F:27:87 vc_mem.mem_base
17480554        DC:A6:32:2F:27:87       msc95xx.macaddr=DC:A6:32:2F:27:87 vc_mem.mem_base
22104977        DC:A6:32:2F:27:87       \x00sc95xx\x00macaddr=DC:A6:32:2F:27:87\x00vc_mem\x00mem_base
44600209        DC:A6:32:2F:27:87       \x00sc95xx\x00macaddr=DC:A6:32:2F:27:87\x00vc_mem\x00mem_base
201422594       DC:A6:32:2F:27:87        (ether_dhost)
201422594       18:B1:69:9D:DE:0A        (ether_shost)
201458114       00:10:75:29:55:5C        (ether_shost)
201458434       10:E7:C6:29:C6:15        (ether_shost)
201460034       00:10:75:29:55:5C        (ether_shost)
```

Destination host

Source host

El archivo httplogs.txt contiene registro de todo el tráfico http generado

```
# BANNER FILE NOT PROVIDED (-b option)
# BULK_EXTRACTOR-Version: 1.6.0 ($Rev: 10844 $)
# Feature-Recorder: httplogs
# Filename: standard_IoT.memdump
# Feature-File-Version: 1.1
1024260 127.0.0.1 - - [10/Apr/2020:10:52:09 +0000] "GET //reader/battery_info?key=7f7b31dff64e4ca8d0f818cfca06f777 HTTP/1
 200 198 "https://web.lfr:444/" "Mozilla/5.0 (X11; Linux armv7l) AppleWebKit/537.36 (KHTML, like Gecko) Raspbian Chromium
.0.3904.108 Chrome/78.0.3904.108 Safari/537.36" 127.0.0.1 - - [10/Apr/2020:10:52:09 +0000] "GET //reader/battery_info?key
7b31dff64e4ca8d0f818cfca06f777 HTTP/1.1" 200 198 "https://web.lfr:444/" "Mozilla/5.0 (X11; Linux armv7l) AppleWebKit/537.
(KHTML, like Gecko) Raspbian Chromium/78.0.3904.108 Chrome/78.0.3904.108 Safari/537.36"
1024548 127.0.0.1 - - [10/Apr/2020:10:52:09 +0000] "GET //reader/system_date_time?key=7f7b31dff64e4ca8d0f818cfca06f777 HT
1.1" 200 137 "https://web.lfr:444/" "Mozilla/5.0 (X11; Linux armv7l) AppleWebKit/537.36 (KHTML, like Gecko) Raspbian Chro
m/78.0.3904.108 Chrome/78.0.3904.108 Safari/537.36"     127.0.0.1 - - [10/Apr/2020:10:52:09 +0000] "GET //reader/system_d
_time?key=7f7b31dff64e4ca8d0f818cfca06f777 HTTP/1.1" 200 137 "https://web.lfr:444/" "Mozilla/5.0 (X11; Linux armv7l) Appl
bKit/537.36 (KHTML, like Gecko) Raspbian Chromium/78.0.3904.108 Chrome/78.0.3904.108 Safari/537.36"
1024840 127.0.0.1 - - [10/Apr/2020:10:52:10 +0000] "GET //reader/reader_info?key=7f7b31dff64e4ca8d0f818cfca06f777 HTTP/1.
200 541 "https://web.lfr:444/" "Mozilla/5.0 (X11; Linux armv7l) AppleWebKit/537.36 (KHTML, like Gecko) Raspbian Chromium/
0.3904.108 Chrome/78.0.3904.108 Safari/537.36"  127.0.0.1 - - [10/Apr/2020:10:52:10 +0000] "GET //reader/reader_info?key=
b31dff64e4ca8d0f818cfca06f777 HTTP/1.1" 200 541 "https://web.lfr:444/" "Mozilla/5.0 (X11; Linux armv7l) AppleWebKit/537.3
KHTML, like Gecko) Raspbian Chromium/78.0.3904.108 Chrome/78.0.3904.108 Safari/537.36"
1025127 127.0.0.1 - - [10/Apr/2020:10:52:11 +0000] "GET //reader/battery_info?key=7f7b31dff64e4ca8d0f818cfca06f777 HTTP/1
 200 198 "https://web.lfr:444/" "Mozilla/5.0 (X11; Linux armv7l) AppleWebKit/537.36 (KHTML, like Gecko) Raspbian Chromium
.0.3904.108 Chrome/78.0.3904.108 Safari/537.36" 127.0.0.1 - - [10/Apr/2020:10:52:11 +0000] "GET //reader/battery_info?key
7b31dff64e4ca8d0f818cfca06f777 HTTP/1.1" 200 198 "https://web.lfr:444/" "Mozilla/5.0 (X11; Linux armv7l) AppleWebKit/537.
(KHTML, like Gecko) Raspbian Chromium/78.0.3904.108 Chrome/78.0.3904.108 Safari/537.36"
```

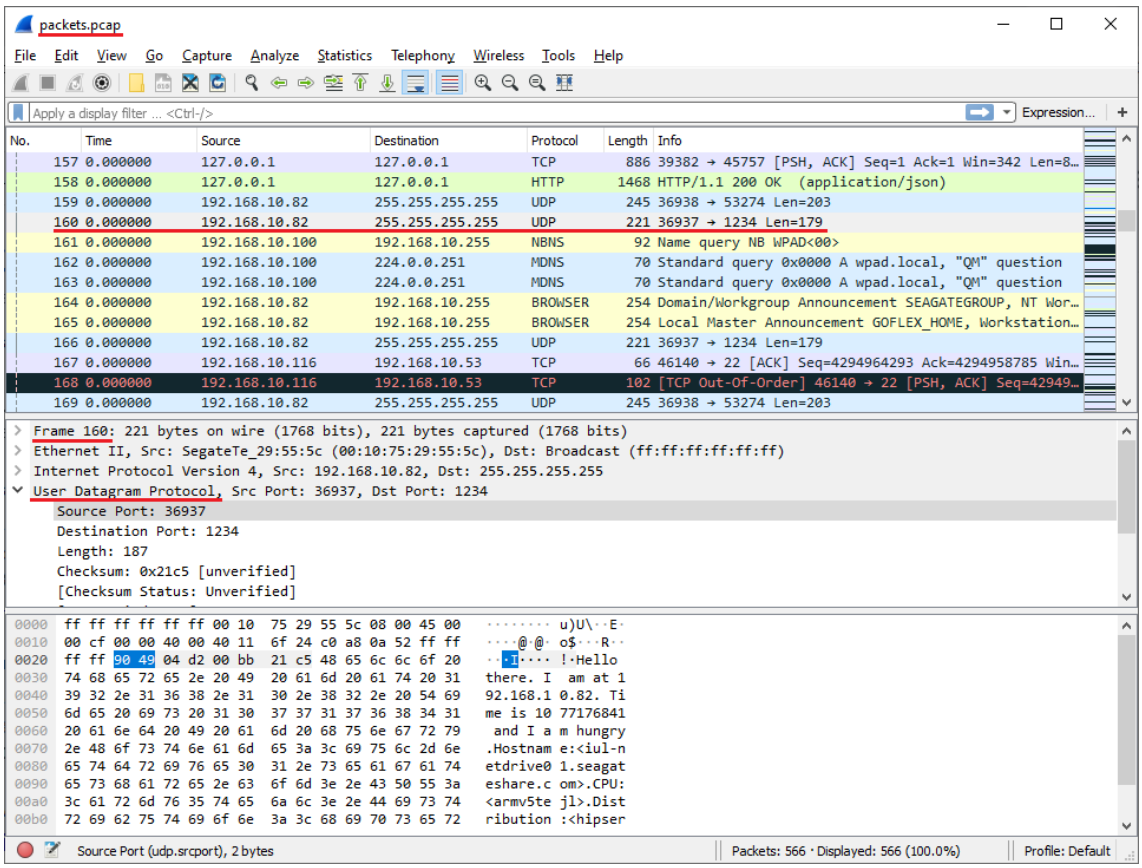
Los archivos ip_histogram.txt y ip.txt guardan información de las direcciones IP

```
# BANNER FILE NOT PROVIDED (-b option)
# BULK_EXTRACTOR-Version: 1.6.0 ($Rev: 10844 $)
# Feature-Recorder: ip
# Filename: standard_IoT.memdump
# Feature-File-Version: 1.1
201422608       8.8.8.8 struct ip R (src) cksum-ok
201422608       192.168.10.53   struct ip L (dst) cksum-ok
201458128       192.168.10.82   struct ip L (src) cksum-ok
201458128       255.255.255.255 struct ip R (dst) cksum-ok
201458448       192.168.10.55   struct ip L (src) cksum-ok
201458448       192.168.10.255  struct ip R (dst) cksum-ok
201460048       192.168.10.82   struct ip L (src) cksum-ok
201460048       255.255.255.255 struct ip R (dst) cksum-ok
201460816       192.168.10.116  struct ip L (src) cksum-ok
201460816       192.168.10.53   struct ip R (dst) cksum-ok
201463184       192.168.10.116  struct ip L (src) cksum-ok
201463184       192.168.10.53   struct ip R (dst) cksum-ok
```

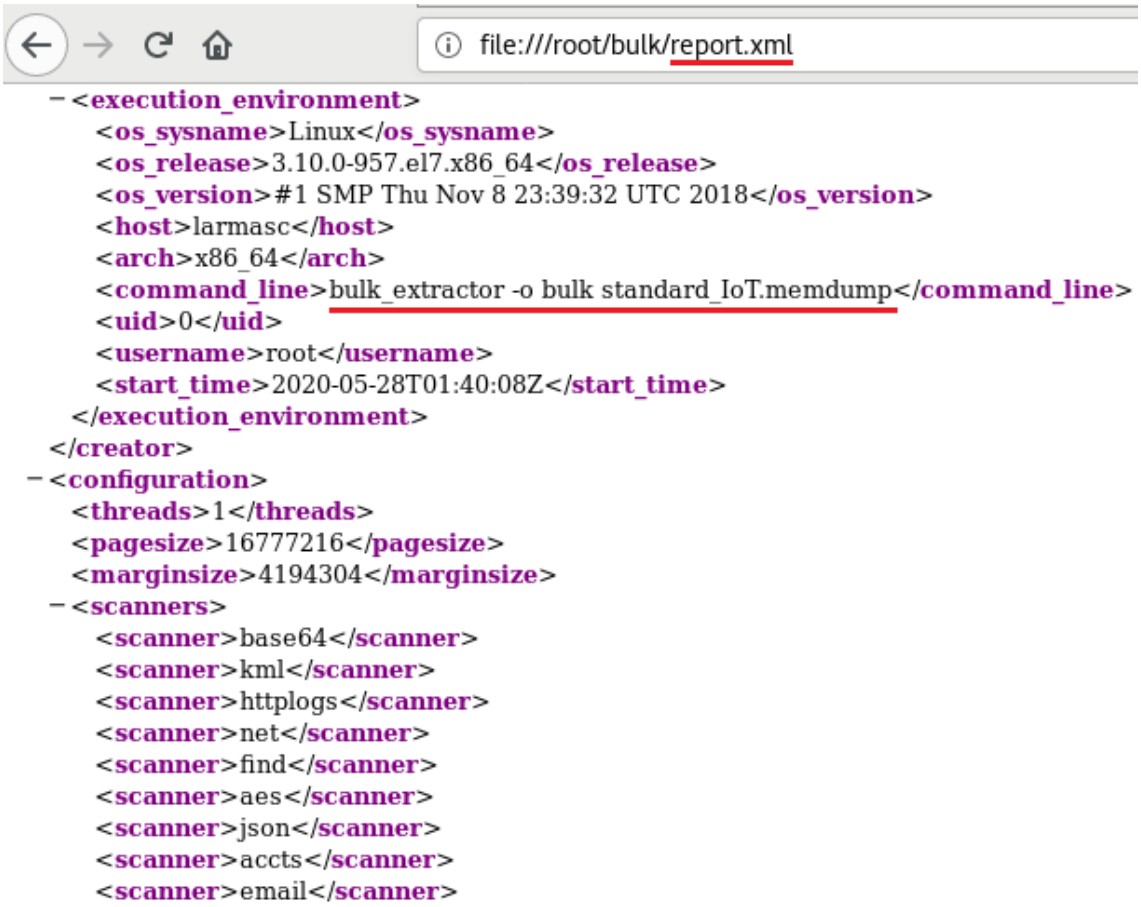
El archivo json.txt contiene los archivos Json encontrados en la memoria

```
# BANNER FILE NOT PROVIDED (-b option)
# BULK_EXTRACTOR-Version: 1.6.0 ($Rev: 10844 $)
# Feature-Recorder: json
# Filename: standard_IoT.memdump
# Feature-File-Version: 1.1
18399063        [1, 2, 3, 4, 5] 6675982b5d76c7fdd870c9e439c0b413
33418454        {"limit_of_stored_cassettes": 20, "sample_id_dup_check": true, "check_database_full": false, "cassette_ex
ation_required": true, "sample_id_required": true, "current_limit_of_stored_results": "500", "cassette_confirmation_requi
": "none", "uqc_period": 295200000, "cassette_settings_persistent": false, "uqc_fail_lock": false, "genericTimeOut": 0, "
_scan_lock": false, "limit_of_stored_results": "500", "userSettFileName": "/opt/iul/data/user_set.json", "process_counter
0, "current_limit_of_stored_cassettes": 20}     a0843ff36cb12d2b8a9ab6093a0b88ac
34936621        {"data": {"batteryVoltage": 4.195064011596857, "batteryUnderVoltage": false, "batteryAlarm": false, "batt
Charging": true, "error": "", "batteryPercent": 66.72265625, "batteryError": false}, "error": {"code": 0, "description":
ccess: "}}      9a8b389aae0912f853b9dd3c203df5ad
44771592        [1,3,5,7]       bd1b88be9bc6866a7bfaaa28f4094aeb
44771603        [0,2,4,8]       9b2bfca0ddd4d29866e56c23fb1569b9
44771617        [5,10,15,20]    cac944667dfae77d5c90bf8af80d0d63
44771677        [0, 1, 2, 3, 4, 5, 5, 7, 8, 10, 15, 20, 25]     83ab387f5193827018121f38ccbd97dd
65600726        {"limit_of_stored_cassettes": 20, "sample_id_dup_check": true, "check_database_full": false, "cassette_ex
ation_required": true, "sample_id_required": true, "current_limit_of_stored_results": "500", "cassette_confirmation_requi
": "none", "uqc_period": 295200000, "cassette_settings_persistent": false, "uqc_fail_lock": false, "genericTimeOut": 0, "
_scan_lock": false, "limit_of_stored_results": "500", "userSettFileName": "/opt/iul/data/user_set.json", "process_counter
0, "current_limit_of_stored_cassettes": 20}     a0843ff36cb12d2b8a9ab6093a0b88ac
```


También se encuentra el archivo packets.pcap que guarda los paquetes de red capturados haciendo uso de un programa analizador de tráfico de red como Wireshark se puede revisar la comunicación a nivel de red del dispositivo



Se encuentra el archivo report.xml que contiene el reporte de lo realizado por la herramienta usando el navegador Firefox se puede ver su contenido



El archivo rfc822.txt guarda datos de las cabeceras de correo

```
# BANNER FILE NOT PROVIDED (-b option)
# BULK_EXTRACTOR-Version: 1.6.0 ($Rev: 10844 $)
# Feature-Recorder: rfc822
# Filename: standard_IoT.memdump
# Feature-File-Version: 1.1
12498684        host: log_buf   brcm,debug\x00\x00\x016sdhost: log_buf @ %p (%llx)\x0A\x00\x00\x01
12498716        host: failed    %p (%llx)\x0A\x00\x00\x013sdhost: failed to remap timer\x0A
12498752        host: failed    p timer\x0A\x00\x00\x00\x00\x013sdhost: failed to allocate log
27677887        Sun, 06 Nov 1994 08:49:37 GMT   standards.\x0A\x0A    Sun, 06 Nov 1994 08:49:37 GMT  ; RFC 822, upd
177618109       Wed, 09 Feb 1994 22:23:32 GMT   g formats:\x0A\x0A    Wed, 09 Feb 1994 22:23:32 GMT       -- HTTP f
270141864       Host: api.endpoint      .1\x0D\x0AX-Forwarded-Host: api.endpoint\x0D\x0AHost: 127.0.0.
270141884       Host: 127.0.0.1 : api.endpoint\x0D\x0AHost: 127.0.0.1:8881\x0D\x0Aaccept: a
270150048       Host: api.endpoint      .1\x0D\x0AX-Forwarded-Host: api.endpoint\x0D\x0AHost: 127.0.0.
270150068       Host: 127.0.0.1 : api.endpoint\x0D\x0AHost: 127.0.0.1:8882\x0D\x0Aaccept: a
270156193       Host: api.endpoint      .1\x0D\x0AX-Forwarded-Host: api.endpoint\x0D\x0AHost: 127.0.0.
270156213       Host: 127.0.0.1 : api.endpoint\x0D\x0AHost: 127.0.0.1:8882\x0D\x0Aaccept: a
275587665       HOST: api.endpoint      .1\x0D\x0AX_FORWARDED_HOST: api.endpoint\x0D\x0AHOST: 127.0.0.
275587685       HOST: 127.0.0.1 : api.endpoint\x0D\x0AHOST: 127.0.0.1:8882\x0D\x0AACCEPT: a
276042336       HOST: api.endpoint      .1\x0D\x0AX_FORWARDED_HOST: api.endpoint\x0D\x0AHOST: 127.0.0.
276042356       HOST: 127.0.0.1 : api.endpoint\x0D\x0AHOST: 127.0.0.1:8882\x0D\x0AACCEPT: a
289584852       host:dir        e title to user@host:dir\x0Acase "$TERM" in
```

El archivo sin.txt contiene información sobre normas y estándares utilizados

```
# BANNER FILE NOT PROVIDED (-b option)
# BULK_EXTRACTOR-Version: 1.6.0 ($Rev: 10844 $)
# Feature-Recorder: sin
# Filename: standard_IoT.memdump
# Feature-File-Version: 1.1
720477212       120-100-100     um-r-normal--17-120-100-100-p-122-iso10646-
720494832       120-100-100     um-r-normal--16-120-100-100-c-80-jisx0201.1
720496304       120-100-100     um-r-normal--16-120-100-100-c-80-iso8859-1\x00
720496624       170-100-100     um-r-normal--24-170-100-100-c-120-iso8859-1
720497824       120-100-100     um-r-normal--16-120-100-100-c-80-jisx0201.1
720497920       170-100-100     um-r-normal--24-170-100-100-c-120-jisx0201.
720498008       170-100-100     um-r-normal--24-170-100-100-c-120-jisx0201.
720540623       100-100-100     micondensed--13-100-100-100-c-60-iso8859-1\x00
```

El archivo sqlite_carved.txt guarda registro de las bases de datos generados en formato SQLite 3.x estas se encuentran dentro de la carpeta sqlite_carved y contienen información del contenido del volcado de memoria en formato txt

```
[root@larmasc ~]# ll bulk/sqlite_carved/000/
total 150776
-rw-r--r--. 1 root root  9888704 May 27 20:41 313072704.sqlite3
-rw-r--r--. 1 root root  5116616 May 27 20:41 317844792.sqlite3
-rw-r--r--. 1 root root  9838744 May 27 20:41 328556696.sqlite3
-rw-r--r--. 1 root root  6441784 May 27 20:41 409058104.sqlite3
-rw-r--r--. 1 root root  7041024 May 27 20:42 476774400.sqlite3
-rw-r--r--. 1 root root  7511256 May 27 20:42 477236440.sqlite3
-rw-r--r--. 1 root root  7976160 May 27 20:42 494486752.sqlite3
-rw-r--r--. 1 root root  8592976 May 27 20:42 495111760.sqlite3
-rw-r--r--. 1 root root  9739632 May 27 20:42 497771152.sqlite3
-rw-r--r--. 1 root root  5378648 May 27 20:42 508678744.sqlite3
-rw-r--r--. 1 root root  5408976 May 27 20:42 508709072.sqlite3
-rw-r--r--. 1 root root  1886792 May 27 20:42 521957960.sqlite3
-rw-r--r--. 1 root root  5801968 May 27 20:42 525858800.sqlite3
-rw-r--r--. 1 root root  5956984 May 27 20:42 526017912.sqlite3
-rw-r--r--. 1 root root  6099296 May 27 20:42 526074208.sqlite3
-rw-r--r--. 1 root root 10409968 May 27 20:42 530466800.sqlite3
-rw-r--r--. 1 root root  5182096 May 27 20:42 542032528.sqlite3
-rw-r--r--. 1 root root  1969864 May 27 20:42 555597512.sqlite3
-rw-r--r--. 1 root root  8821536 May 27 20:42 582575328.sqlite3
-rw-r--r--. 1 root root  6422096 May 27 20:42 593604176.sqlite3
-rw-r--r--. 1 root root  8968992 May 27 20:43 683091168.sqlite3
-rw-r--r--. 1 root root  9897968 May 27 20:43 731281392.sqlite3
```