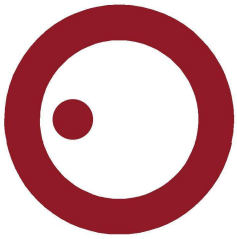


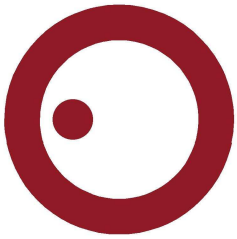
escola superior de tecnologia e gestão
instituto politécnico de leiria

Instituto Politécnico de Leiria
Escola Superior de Tecnologia e Gestão
Departamento de Engenharia Informática
Mestrado em Cibersegurança e Informática Forense

CAPTURA E ANÁLISE DE CONTEÚDOS DE MEMÓRIA EM SISTEMAS COMPUTACIONAIS LIGADOS

PATRÍCIA DOS SANTOS SILVA

Leiria, Novembro de 2020



escola superior de tecnologia e gestão
instituto politécnico de leiria

Instituto Politécnico de Leiria
Escola Superior de Tecnologia e Gestão
Departamento de Engenharia Informática
Mestrado em Cibersegurança e Informática Forense

CAPTURA E ANÁLISE DE CONTEÚDOS DE MEMÓRIA EM SISTEMAS COMPUTACIONAIS LIGADOS

PATRÍCIA DOS SANTOS SILVA
Número: 2180068

Dissertação realizada sob orientação do Professor Doutor Patrício Rodrigues Domingues (patricio.domingues@ipleiria.pt) e do Professor Doutor Miguel Monteiro de Sousa Frade (miguel.frade@ipleiria.pt).

Leiria, Novembro de 2020

AGRADECIMENTOS

Gostaria de agradecer aos professores orientadores desta dissertação, o Professor Doutor Patrício Rodrigues Domingues e o Professor Doutor Miguel Monteiro de Sousa Frade, por todo o apoio e disponibilidade que tiveram ao longo do processo de desenvolvimento da dissertação. Agradeço ainda a oportunidade que esta dissertação me deu em trabalhar e explorar a área de análise forense de memória.

RESUMO

De modo a perceber o que ocorreu num sistema que foi alvo de um incidente de segurança é necessário efetuar uma investigação digital. Numa investigação digital, se possível, é importante recolher os dados que estejam presentes na memória do sistema. Dada a relevância que a informação presente em memória poderá ter, é fundamental a existência de ferramentas que permitam visualizar os dados adquiridos da memória. Nesta dissertação é apresentada uma solução de visualização da informação presente em imagens de memória de sistemas em análise. O objetivo consiste em apresentar o conteúdo da imagem de memória, executando-se o Volatility (uma ferramenta de análise da memória) sobre a mesma. Pretende-se apresentar alguma informação essencial de forma automática, de modo a otimizar o trabalho do investigador. Planeia-se ainda que os resultados obtidos sejam apresentados em forma de relatório. Esta solução será integrada com o *software* Autopsy.

ABSTRACT

In order to understand what happened in a system that was the target of a security incident, a digital investigation is required. In a digital investigation, if possible, it is important to collect the data that is present in the system's memory. Given the relevance that the information present in memory may have, it is essential to have tools to visualize the data acquired from memory. In this dissertation is presented a solution for visualization of information present in memory images of systems under analysis. The goal is to present the contents of the memory image by running Volatility (a memory analysis tool) on it. It is intended to present some essential information automatically, in order to optimize the researcher's work. It is also planned that the results obtained will be presented in the form of a report. This solution will be integrated with the Autopsy software.

ÍNDICE

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS

AFF	Advanced Forensic Format.
bit	Digito binário.
CLI	Command-Line Interface.
CPU	Central Processing Unit.
CSS	Cascading Style Sheets.
DLL	Dynamic Link Library.
EWF	Expert Witness Format.
exFAT	Extended File Allocation Table.
FAT	File Allocation Table.
GB	Gigabyte.
GiB	GibiByte.
GPL	General Public License.
IP	Internet Protocol.
JSON	JavaScript Object Notation.
KiB	Kibibyte.
KML	Keyhole Markup Language.
MiB	MebiByte.
NTFS	New Technology File System.

PCI Peripheral Component Interconnect.

PEB Process Environment Block.

RAM Random Access Memory.

SID Security Identifier.

SMB Server Message Block.

SQL Structured Query Language.

SSD Solid-State Drive.

TiB Tebibyte.

TLS Transport Layer Security.

TSK The Sleuth Kit.

UI User Interface.

URL Uniform Resource Locator.

USB Universal Serial Bus.

VNC Virtual Network Computing.

XML Extensible Markup Language.

1

INTRODUÇÃO

No âmbito do 2ºano do Mestrado de Cibersegurança e Informática Forense da Escola Superior de Tecnologia e Gestão de Leiria (ESTG) do Instituto Politécnico de Leiria (IPLeiria), foi elaborada a presente dissertação intitulada de 'Captura e Análise de Conteúdos de Memória em Sistemas Computacionais Ligados' - inserida nas áreas científicas de Análise Digital Forense e Engenharia Informática - e tem como proponentes o Professor Doutor Patrício Rodrigues Domingues e o Professor Doutor Miguel Monteiro de Sousa Frade.

No contexto de uma investigação digital, uma das primeiras etapas a ser concretizada consiste na recolha dos dados do(s) sistema(s) abrangido(s) na referida investigação, mais concretamente das respetivas fontes de dados. Outrora a fonte de recolha usual correspondia ao disco rígido, logo, os métodos aplicados tinham como foco a preservação do conteúdo presente no mesmo. Contudo, a adição de outros focos para busca de informação pertinente fez com que ocorresse uma notável evolução nas investigações digitais. Assim, os dados presentes na memória Random Access Memory (RAM) tornaram-se também um outro alvo importante das investigações (Case e Richard III, 2017).

A RAM é uma memória volátil do computador, ou seja, armazena os seus dados temporariamente. O seu conteúdo permanece acessível enquanto o sistema tiver energia ou até o seu conteúdo ser sobrescrito durante a utilização do computador uma vez que o conteúdo da memória muda constantemente enquanto o sistema operativo está em funcionamento.

O facto de os dados da RAM serem armazenados temporariamente faz com que esta seja utilizada para armazenar, por exemplo, dados relacionados com ações com intenção maliciosa e incidentes, levando a que a análise à memória se torne um processo cada vez mais frequente (Ligh Hale et al., 2014). De modo a efetuar uma análise forense à memória de uma sistema é necessário, em primeiro lugar, efetuar a aquisição dos dados presentes na mesma. Todavia, o processo de aquisição da memória não é um processo trivial, ao qual se deve sempre considerar um certo risco relacionado, exigindo uma prévia avaliação do cenário.

Para a análise do conteúdo obtido na aquisição é necessário recorrer a uma das várias ferramentas de análise existentes. Com estas é possível visualizar os dados presentes no ficheiro correspondente à imagem de memória e analisar os mesmos de modo a perceber o ocorrido no sistema e assim reportar as evidências encontradas. Com a utilização destas ferramentas cabe ao investigador efetuar a análise dos dados obtidos do ficheiro, processo este que se pode revelar demorado. É de referir que, em certas situações, é complicado para o investigador entender qual deverá ser o seu ponto de partida em todo o processo de análise (Case e Richard III, 2017).

1.1 OBJETIVOS

O objetivo desta dissertação consiste na exploração e estudo do processo de aquisição de memória de sistemas bem como da análise dos dados adquiridos numa aquisição. Consiste ainda no desenvolvimento de módulos/*scripts* que permitirão efetuar análises forense à memória adquirida, mais precisamente, aos dados presentes numa imagem de memória, apresentando os mesmos ao utilizador. Pretende-se ainda que seja disponibilizada alguma informação adicional ao utilizador através da análise automática dos próprios resultados obtidos. O trabalho desenvolvido irá abranger o sistema operativo *Windows*. A importância do trabalho a ser realizado consiste em permitir ao investigador efetuar o processo de análise da memória, permitindo auxiliar e otimizar o trabalho em curso do investigador. É de referir que o trabalho que se espera desenvolver pode-se tornar importante especialmente para empresas e organizações, nomeadamente, para profissionais da área da segurança.

1.2 CONTRIBUTOS

Nesta dissertação foi efetuada uma análise e exploração às fases de aquisição e análise aos dados da memória de sistemas, abordando-se conceitos e ferramentas inerentes a ambas as fases.

Para além disso, desenvolveu-se o *data source ingest module* RAMAnalysis que permite realizar a análise da memória de sistemas através da execução dos *plugins* do Volatility. Foi ainda desenvolvido o RAMAnalysis Report que é um *report module* que apresenta os dados obtidos pelo *data source ingest module* e alguma informação

adicional. Para a utilização de ambos é necessário recorrer ao *software* Autopsy. Os módulos RAMAnalysis e o RAMAnalysis Report encontram-se disponíveis em [1].

1.3 ORGANIZAÇÃO DO DOCUMENTO

O presente documento encontra-se dividido em oito capítulos. O capítulo 1 é a introdução e apresenta o enquadramento ao tema da dissertação, a apresentação do problema a ser resolvido, a importância da dissertação desenvolvida e ainda a estrutura da mesma. O capítulo 2 apresenta alguns conceitos inerentes ao tema e ainda algumas das ferramentas da área da análise forense à memória. Ao longo da dissertação foram efetuadas explorações a ferramentas de aquisição da memória, sendo que a exploração realizada encontra-se no capítulo 3. Dada a importância forense que o conteúdo do *clipboard* de um sistema poderá apresentar, o capítulo 4 contempla uma análise ao mesmo e a ferramentas que permitam analisar o seu conteúdo. Já no capítulo 5 é exposta a exploração realizada a algumas ferramentas de análise de memória. O capítulo 6 introduz o *software* RAMAnalysis, um módulo do tipo *data source ingest* que se destina a ser executado no *software* de análise forense Autopsy. Por sua vez, o capítulo 7 é relativo ao outro módulo inerente a esta dissertação, nomeadamente, ao *report module* designado de RAMAnalysis Report. Tanto no capítulo 6 como no capítulo 7 são apresentados alguns aspetos relativos à elaboração de ambos os módulos e aos resultados obtidos com os mesmos. Por fim, o capítulo 8 contempla as principais conclusões da dissertação, sugerindo ainda trabalho futuro.

1 https://github.com/patriciaSSilva/RAMAnalysis-Module

2

ESTADO DA ARTE

A análise forense a dispositivos de memória persistente, nomeadamente discos, é o procedimento habitualmente adotado para a procura de evidências necessárias para um dado caso. Para tal exploração, primeiro torna-se necessário adquirir a informação presente em disco para depois esta ser analisada. É de ter em conta a existência de algumas ferramentas que permitem realizar essas mesmas fases referidas. Este tipo de análise apresenta quatro fases, mais precisamente, a identificação, a preservação, a análise e, por fim, a apresentação (Casey, 2009). Na fase de identificação é necessário, por exemplo, realizar o acordo de confidencialidade e um levantamento de dados acerca do estado do sistema em causa. Já na fase de preservação, o investigador deverá proceder à cópia das evidências, seguindo-se a fase de análise na qual as evidências são examinadas com o objetivo de se encontrar dados relevantes. Por fim, o investigador terá que apresentar um relatório com os resultados obtidos.

Uma análise forense a um sistema apresenta alguns conceitos que se encontram relacionados com esta e com algumas das tarefas que são habitualmente realizadas. Por conseguinte, neste capítulo ir-se-á efetuar uma apresentação de alguns conceitos, como, por exemplo, dispositivos de armazenamento persistentes. Exceto quando indicado em contrário, o ambiente computacional considerado é o sistema operativo *Windows 10*.

Com o surgimento de técnicas/procedimentos adequados começou a ser explorada a informação presente em memória (Ruff, 2008). Assim, a captura de conteúdo em memória tornou-se um processo ocorrente em investigações e relevante para entender algumas das ações executadas. Antes de ser efetuar a aquisição é necessário garantir a preservação do ambiente bem como tomar algumas decisões. A aquisição à memória poderá ser com recurso a *software* ou com recurso a *hardware*. No caso da aquisição com recurso a *software*, pode ser realizada uma aquisição local ou uma aquisição remota. A decisão entre estes dois tipos de aquisição depende das características do ambiente com o qual o investigador se depara. Esta é uma das razões pela quais a avaliação dos fatores do ambiente no qual o(s) sistema(s) se insere ser tão relevante.

Com a avaliação do cenário no qual o sistema se insere concluída e com todas as decisões necessárias tomadas, dá-se início ao processo de aquisição da memória de um sistema com recurso a uma ferramenta. Relativamente a ferramentas de aquisição, atualmente, um investigador tem várias disponíveis no mercado, existindo ferramentas que capturam toda a memória do sistema - como a MAGNET RAM Capture[1], AcessData FTK Imager[2] e Winpmem[3] - e outras somente a memória dos processos pertinentes ao utilizador, como, por exemplo, a MAGNET Process Capture[4] e o Gestor de Tarefas do *Windows*. Assim, as ferramentas de aquisição da memória anteriormente mencionadas, serão apresentadas neste capítulo.

Após a aquisição dos sistema estar concluída, o investigador inicia a fase de análise do conteúdo adquirido. No que diz respeito à fase de análise dos dados existentes na memória, existem dois caminhos possíveis, mais especificamente, uma aquisição por *snapshot* ou uma *live analysis*. O Rekall[5] é uma ferramenta que permite a realização de uma *live analysis*, enquanto que o Volatility[6] permite a realização de uma aquisição por *snapshot*. É de ter em conta que existe ainda um módulo do Volatility para o Autopsy[7] para se efetuar análises a dados presentes na memória.

Todos estes aspetos mencionados que serão explorados neste capítulo, devem ser abordados no decurso de uma investigação. Assim, o estudo destes mesmos fatores e a realização de certas decisões podem ser determinantes para o sucesso de uma investigação.

2.1 ANÁLISE FORENSE A DISCOS RIGÍDOS

O foco principal de uma investigação digital consiste em encontrar evidências digitais relacionadas com o incidente ou ação maliciosa. Durante muito tempo, o enfoque principal da informática forense eram os dispositivos de memória persistente, especialmente os discos, e mais recentemente pens e cartões de memória. De uma forma sucinta, para se analisar um disco, é necessário adquirir os dados presentes neste para, de seguida, estes serem analisados (Casey, 2009). Obviamente não se pode ignorar o facto de ser necessário, antes de mais, avaliar o estado do ambiente no qual o sistema se insere.

1 https://www.magnetforensics.com/resources/magnet-ram-capture/
2 http://marketing.accessdata.com/ftkimager3.1.1
3 https://github.com/google/rekall/tree/master/tools/windows/winpmem
4 https://www.magnetforensics.com/resources/magnet-process-capture/
5 http://www.rekall-forensic.com/
6 https://www.volatilityfoundation.org/
7 https://github.com/markmckinnon/Autopsy-Plugins/tree/master/Volatility

Para o investigador adquirir os dados presentes em disco terá que recorrer a uma ferramenta que permita realizar tal tarefa. A ferramenta AccessData FTK (*Forensic Toolkit*) Imager é uma ferramenta de aquisição disponível para sistema operativo *Windows* que permite criar imagens de discos sem alterar os dados originais. Esta permite também visualizar ficheiros, pastas e o conteúdo das imagens forenses guardadas ou na máquina local ou numa unidade de rede (AcessData, 2020; S. D. Forensics e Blog, 2009).

Com a imagem do disco em causa criada, segue-se a fase da análise da mesma, na tentativa de encontrar alguma informação pertinente. Para tal, duas ferramentas disponíveis no momento e que podem ser utilizadas é o Autopsy[8] e o RegReport[9] . O Autopsy é uma ferramenta de código aberto, e de banda larga, útil na análise de imagens de discos permitindo, por exemplo, analisar conteúdo multimédia e ficheiros recentemente eliminados que permaneçam em espaço não alocado no disco. Já a ferramenta RegReport (mais específica da *Registry*) permite obter, por exemplo, informações sobre o sistema operativo *Windows*, o *software* instalado e utilizadores existentes. Esta ferramenta, para tal, recorre aos ficheiros *SAM*, *SOFTWARE*, *SYSTEM* e *NTUSER.DAT* do *registry* do *Windows*. Assim, o objetivo desta ferramenta é a criação de relatórios da *Registry* que revele dados sobre o sistema. Este *software* não permite obter os dados da *Registry* do sistema operativo em execução (Brinkmann, 2009; Gaijin, 2019).

Finalizada a fase de análise, é necessário, por fim, criar o relatório de análise forense com as conclusões obtidas na fase anterior.

É de referir que as ferramentas referidas são algumas das que são utilizadas seja pela sua interface objetiva e organizada, bem como pelo facto de as suas funcionalidades suportarem as necessidades usuais de uma investigação ao disco de um sistema. Para além disso, todas as ferramentas apresentadas são gratuitas.

O sistema IPED[10] (Indexador e Processador de Evidências Digitais) indexa e processa evidências digitais, procurando e organizando os dados relevantes que estejam em ficheiros visíveis, ocultos, apagados e fragmentados. Esses mesmos ficheiros podem-se encontrar, por exemplo, em discos rígidos, Solid-State Drive (SSD), *pens*, cartões de memória e CDs. Deste modo, o utilizador poderá, por exemplo, procurar pela informação que pretende ou até mesmo visualizar imagens e vídeos. O sistema IPED apresenta uma interface simples e intuitiva e pode ser executado nos sistemas *Windows*, *Linux* e *MacOS*. Este sistema foi desenvolvido

8 https://www.autopsy.com/
9 https://www.gaijin.at/en/software/registryreport
10 https://github.com/sepinf-inc/IPED

com a linguagem de programação *Java*. Alguns aspetos do sistema IPED a realçar é o facto de permitir o processamento de dados em vários computadores, identifica automaticamente ficheiros codificados, cria relatórios e permite a recuperação de ficheiros apagados (IPOG, 2018).

A ferramenta KAPE[11] (*Kroll Artifact Parser and Extractor*) é uma ferramenta gratuita de triagem. Este *software* recolhe ficheiros e, de seguida, processa esses mesmos ficheiros de modo a tentar encontrar artefactos úteis e interessantes a nível forense. Os dados criados pelo KAPE podem, por exemplo, ser visualizados através de *timelines* (Zimmerman, 2019).

2.2 CONCEITOS

2.2.1 *Memória de um Processo*

A Figura 1 corresponde a um diagrama de alto-nível do conteúdo habitualmente presente na memória virtual de um dado processo (Ligh Hale et al., 2014).

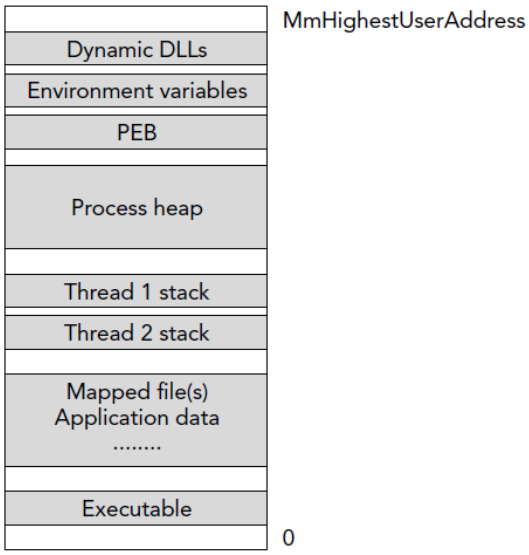


Figura 1: Diagrama com os dados usualmente presentes na memória de um processo (Ligh Hale et al., 2014) (Process Environment Block (PEB))

Sabendo-se que um processo tem a sua própria visão privada da memória nesse intervalo, neste diagrama, 'MmHighestUserAddress' representa o limite superior mais elevado do espaço de endereçamento do utilizador. No diagrama da Figura 1

11 https://www.kroll.com/en/services/cyber-risk/incident-response-litigation-support/kroll-artifact-parser-extractor-kape

verifica-se que na memória correspondente a um processo é possível aceder-se, por exemplo, às variáveis de ambiente do processo e ao executável em causa (Ligh Hale et al., 2014).

2.2.2 *Ficheiros `pagefile.sys`, `hiberfil.sys` e `swapfile.sys`*

A nível forense, em sistemas *Windows*, no que toca a dados em memória, é relevante considerar os ficheiros `pagefile.sys`, `hiberfile.sys` e `swapfile.sys`. Estes ficheiros localizam-se na raíz da partição onde se encontra instalado o *Windows*, usualmente, mas não obrigatoriamente, no '`C:`'.

O ficheiro `pagefile.sys` é o ficheiro de paginação de memória do *Windows*. Quando o sistema necessita de mais espaço na RAM do que aquela que se encontra disponível no momento, as páginas presentes na RAM que não pertencem ao processo em execução no momento podem ser enviadas para o ficheiro `pagefile.sys` (Ribeiro et al., 2019).

O ficheiro `hiberfil.sys` é o ficheiro de hibernação e este armazena os dados que estavam na memória do sistema no momento em que o sistema entrou em hibernação (Scott, 2016). Assim, quando o sistema hiberna, os dados em memória são guardados no ficheiro `hiberfil.sys`, sendo este mesmo ficheiro reescrito cada vez que o sistema em causa hiberna (A. Singh et al., 2016). Já no caso de uma máquina ter sido desligada, quando esta arrancar novamente, o ficheiro `hiberfil.sys` é colocado a zero (Hunt, 2019).

O ficheiro `swapfile.sys` é um ficheiro de memória virtual presente desde a versão 8.1 do *Windows* e tem tamanho limitado a 16384 Kibibyte (KiB), ou seja, 16 MebiByte (MiB) (Ajinkya, 2020; ExeFiles, 2020). Este ficheiro é utilizado para 'trocar' o novo estilo de aplicação da *Microsoft*, sendo que a *Microsoft* designou estas, por exemplo, de *universal apps*, *Windows Store apps*, *Metro apps* e *Modern apps*. Assim sendo, com este ficheiro o sistema *Windows* escreve eficazmente todo o conjunto de trabalho (privado) de uma *app* no disco, com o objetivo de conseguir obter espaço em memória (A. Singh et al., 2016).

Na Figura 2 encontra-se uma captura de ecrã ao sistema com a visualização dos ficheiros `pagefile.sys`, `hiberfil.sys` e `swapfile.sys` no *Windows Explorer*, no `C:`. Para o *Windows Explorer* mostrar estes ficheiros é necessário garantir que os ficheiros protegidos do sistema operativo não são ocultados.

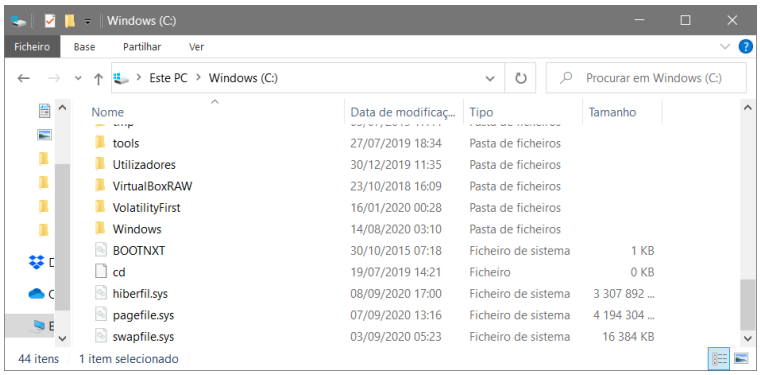


Figura 2: Visualização dos ficheiros `pagefile.sys`, `hiberfil.sys` e `swapfile.sys` no *Windows Explorer*

A obtenção dos ficheiros `pagefile.sys`, `hiberfil.sys` e `swapfile.sys` pode ser realizada através de ferramentas como a AccessData FTK Imager. Na Figura 3 pode-se ver a interface do AcessData FTK Imager com os três ficheiros.

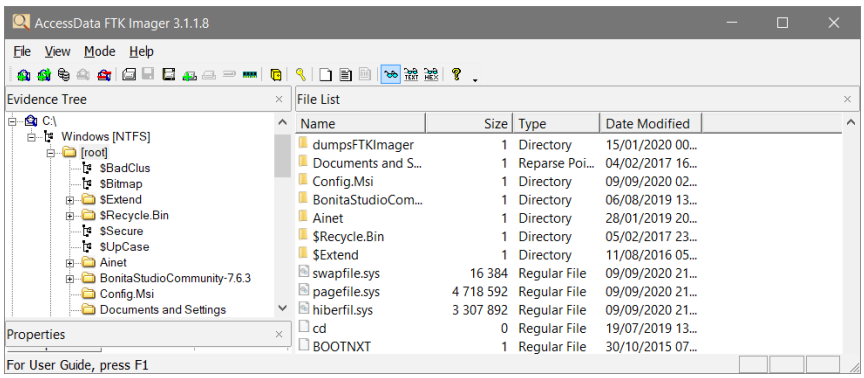


Figura 3: Interface do AccessData FTK Imager após a obtenção dos ficheiros `pagefile.sys`, `hiberfil.sys` e `swapfile.sys`

Nesta, após se efetuar a captura da memória, para se obter os três ficheiros, é necessário efetuar-se a exportação dos mesmos. Já a análise deste três ficheiros pode ser realizada, por exemplo, com recurso à ferramenta Belkasoft Evidence Center[12] (Chandel, 2015b).

2.2.3 *Memória Compactada*

Dado o facto de o sistema operativo implícito a esta dissertação ser o *Windows 10*, um conceito necessário de ser explorado é o de memória compactada.

12 https://belkasoft.com/ec

Um componente importante do *Windows 10* é o *Windows 10 Memory Manager*. De uma forma sucinta, este é o responsável por criar a ilusão de um dado processo ter disponível toda a memória do sistema. Por conseguinte, o *Windows 10 Memory Manager* transfere páginas entre a memória física (RAM), o disco (mais precisamente, os ficheiros `pagefile.sys` e `swapfile.sys`) e a *virtual store* (Omar Sardar, 2019a).

A *virtual store* permite que aplicações guardem, leiam/escrevam ficheiros nesta localização virtual que não exige privilégios de administrador para tais ações serem executadas. Estes ficheiros parecem estar virtualmente na sua localização original (networkinghowtos, 2013). No sistema *Windows* a localização da *virtual store* é a seguinte:

‘`C:\Users\<username>\AppData\Local\VirtualStore`‘

A memória compactada consiste em armazenar parte da memória no formato compactado, mantendo-se os dados compactados na memória. Os dados só são compactados quando é necessário e/ou útil para o sistema. Esta surgiu nos sistemas a partir do *Windows 8.1*, juntamente com uma nova *virtual store*. A memória compactada permitiu um aumento no desempenho do sistema dado que permite uma utilização mais eficiente da memória (Hoffman, 2017b; Stancill et al., 2019). Sabendo-se que o acesso a dados em disco é mais lento que o em memória, a compactação da memória torna-se vantajosa pois, por exemplo, permite que mais dados sejam guardados em memória reduzindo-se assim a necessidade de ir ao disco obtê-los (Omar Sardar, 2019b). Mais precisamente, a compactação/descompactação de dados da memória torna-se um processo mais rápido do que guardar os dados no *pagefile* em disco e, posteriormente, aceder aos mesmos quando necessário. Uma desvantagem apontada à compactação/descompactação de memória são os recursos de Central Processing Unit (CPU) consumidos no processo, dado que na prática se está a trocar CPU por memória e velocidade (Hoffman, 2017b).

Uma forma de verificar se a funcionalidade de compactação de memória está ativa num dado sistema é através da *Windows PowerShell*. Assim, com o comando ‘`get-mmagent`‘, se o parâmetro ‘MemoryCompression‘ possuir o valor ‘*True*‘ então significa que a compactação de memória está ativa (Figura 4). Contudo, se se pretender desativar a funcionalidade de memória compactada basta executar em modo administrador o comando ‘`Disable-MMAgent -mc`‘, e, de seguida, reiniciar o sistema. Para reativar a memória compactada, executa-se o comando ‘`Enable-MMAgent -mc`‘ (Hub, 2018).

```
PS C:\WINDOWS\system32> get-mmagent

ApplicationLaunchPrefetching : True
ApplicationPreLaunch         : True
MaxOperationAPIFiles         : 256
MemoryCompression            : True
OperationAPI                 : True
PageCombining                : True
PSComputerName               :
```

Figura 4: Funcionalidade de memória compactada ativa

Na Tabela 1 encontram-se os comandos da *Windows PowerShell* referentes à funcionalidade de compactação de memória mencionados anteriormente.

Tabela 1: Comandos da *Windows PowerShell* referentes à funcionalidade de compactação de memória

Comando	**Funcionalidade**
`get-mmagent`	Detetar se a compactação de memória está ativa
`Disable-MMAgent -mc` e *reboot*	Desativar a funcionalidade de memória compactada
`Enable-MMAgent -mc` e *reboot*	Ativar a funcionalidade de memória compactada

No *Windows 10* é ainda possível, no Gestor de Tarefas (em 'Desempenho', 'Memória'), obter a quantidade de memória que se encontra compactada. Na Figura 5 verifica-se que 391 MiB dos 5,6 GibiByte (GiB) é memória compactada. É de referir que o valor de quantidade de memória compactada varia ao longo do tempo, consoante a abertura e fecho de aplicações no sistema. Já ao passar o rato sobre o gráfico 'Composição da memória' é possível obter mais alguns detalhes acerca da memória compactada. Na Figura 6 verifica-se que o sistema em causa está a utilizar 5,9 GiB da sua memória física, sendo que 412 MiB é memória compactada que, por sua vez, armazena aproximadamente 1,4 GiB de dados. Todo este processo permite que o sistema economize cerca de 945 MiB de memória, pois, sem compactação da memória, o sistema em causa estaria a usar aproximadamente 6,8 GiB de memória.

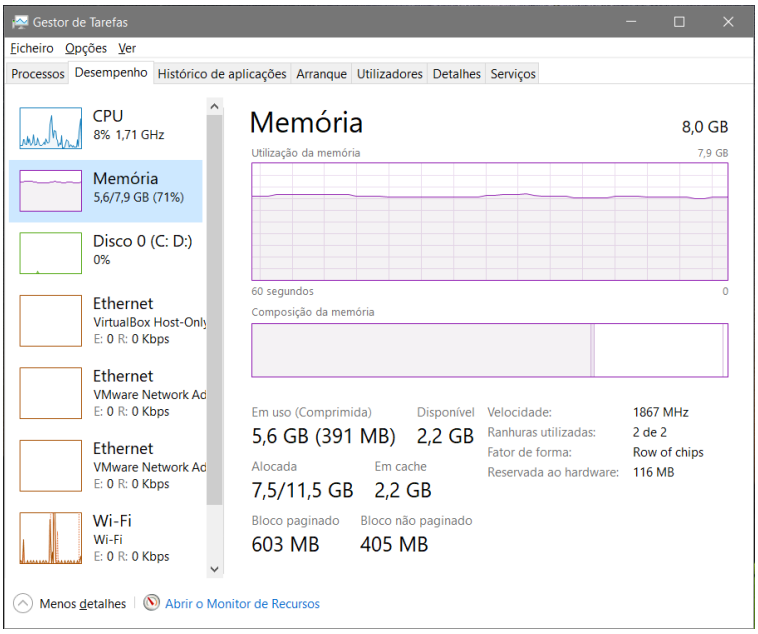


Figura 5: Visualização da quantidade de memória compactada no sistema

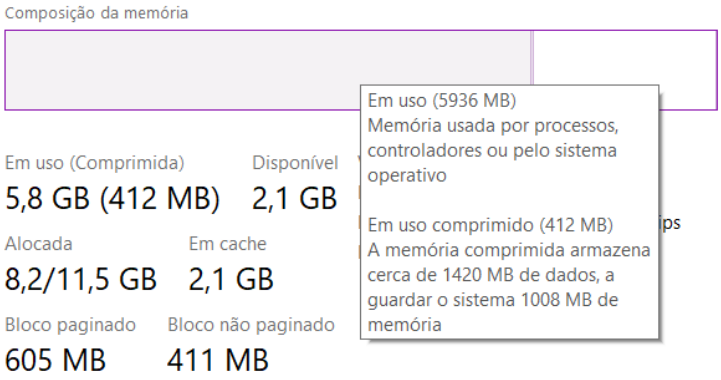


Figura 6: Visualização de detalhes acerca da memória compactada no sistema

A nível forense, o problema que se colocou com a memória compactada prendeu-se com a questão de as ferramentas de análise forense não conseguirem ler páginas compactadas, mais precisamente, com o facto de alguns artefactos forenses - como, por exemplo, programas maliciosos - não serem detetados (Stancill et al., 2019). Para além disso, na aquisição não seriam adquiridos os dados expectáveis, pois uma parte encontra-se compactada em memória (Omar Sardar, 2019b).

Quando surgiu a memória compactada no *Windows 10*, os módulos das ferramentas de análise ao conteúdo da memória com o Volatility e o Rekall não estavam preparados para interpretar conteúdo mapeado para memória compactada. Assim, os resultados dos módulos eram incompletos sempre que envolviam conteúdo guardado

como compactado. As novas versões do Volatility e do Rekall já são capazes de interpretar memória compactada.

Antes de mais, é necessário apresentar sucintamente duas ferramentas, o Volatility e o Rekall. Ambas são ferramentas de análise de dados da memória, contudo, o Rekall também permite efetuar aquisições de memória. Para efetuar a análise da memória com estas duas ferramentas basta recorrer a um conjunto de *plugins* em que, cada um, apresenta o seu devido *output*, com um dado conjunto de informação. Dada a existência da memória compactada nos sistemas, nas ferramentas Volatility e Rekall, o *output* de alguns *plugins* passaram a ser ligeiramente incompletos, como, por exemplo, o *plugin* modules (Stancill et al., 2019). Este apresenta as *drivers* carregadas no sistema no momento da captura da memória. Por conseguinte, com a memória compactada, alguns dos caminhos para os ficheiros estão paginados para a memória compactada, logo, o caminho não surge nos resultados. Tal deve-se ao facto de a ferramenta não conseguir efetuar a leitura da memória compactada, impedindo o utilizador de aceder aos dados armazenados na mesma. Alguns desses caminhos ocultos no referido *output* poderão ser interessantes pois, por exemplo, podem estar de algum modo relacionados com *malware*.

Atualmente, as ferramentas Volatility e Rekall já contornaram a questão da memória compactada, conseguindo descompactar os dados em páginas compactadas, ou seja, são capazes de ler páginas compactadas e, assim, apresentar a devida informação nos seus resultados (Omar Sardar, 2019a; Omar Sardar, 2019b; Stancill et al., 2019).

2.2.4 *Sistemas de Ficheiros*

O sistema de ficheiros é utilizado para armazenar e organizar dados num dispositivo. Um sistema de ficheiros consiste numa hierarquia de diretorias para organizar ficheiros. Contudo, é de referir que um sistema de ficheiros não armazena só os ficheiros em si, mas também informações sobre os mesmos - os chamados metadados - como o tamanho do ficheiro, atributos, o seu nome ou ainda a localização do ficheiro no disco (Fisher, 2019). Um disco é dividido em partições, sendo que para ser utilizado deve ter pelo menos uma partição. Após a criação de uma partição, esta é formatada com um sistema de ficheiros (Hoffman, 2017a). É de ter em conta que cada partição de um dispositivo tem um só sistema de ficheiros. Sistemas de ficheiros empregues em sistema operativo *Windows* a mencionar são o File Allocation Table (FAT) – FAT16, FAT32, FAT64 (mais conhecido por Extended File Allocation Table

(exFAT)) - e o New Technology File System (NTFS). O sistema de ficheiros FAT apresentou uma evolução. Depois do FAT16 surgiu o FAT32, suportando ficheiros com o tamanho máximo de 4 *GibiByte* (GiB) e partições até 32 GiB no caso do *Windows*, embora o limite teórico da FAT32 seja 4 Tebibyte (TiB) (Agarwal, 2012; Sieber, 2019).

O sistema de ficheiros NTFS é o mais comum em computadores com o sistema operativo *Windows* sendo tipicamente utilizado no disco ou partição onde o sistema operativo *Windows* é instalado (Explorer, 2019; Tróia, 2017). O NTFS existe desde o sistema operativo *Windows NT* (1993) tendo surgido novas versões desde então (Carpenter, 2011). O NTFS, comparativamente com o sistema de ficheiros FAT, apresenta algumas melhorias que levam, por sua vez, ao aumento tanto da segurança como do desempenho (Diffen, 2019). Relativamente às compatibilidades, desde o *Windows XP* que o NTFS é suportado por todas as versões do *Windows* (Tróia, 2017).

2.2.5 *Windows Clipboard*

O *Windows clipboard* (ou Área de Transferência do *Windows*) é o mecanismo utilizado pelos sistemas operativos *Windows* para partilhar conteúdo entre as aplicações do sistema, tendo surgido no *Windows 3.1* (Okolica e Peterson L., 2011). Este é um dos mecanismos mais básicos e comuns de um utilizador de um sistema, tendo sido popularizado com o 'copiar/colar' ou 'cortar/colar'. Muito frequentemente, o utilizador ao fazer uso do sistema sente a necessidade de copiar/cortar texto ou imagens de uma aplicação para outra. Deste modo, esta utilização tão regular do *clipboard* pelos utilizadores fez com que esta funcionalidade sofresse uma evolução.

Assim, com a atualização de outubro de 2018 do *Windows 10* surgiu um novo *clipboard* com novas funcionalidades que permitem facilitar o fluxo de trabalho do utilizador no sistema. Entre as novas funcionalidades destaca-se a existência de um histórico do *clipboard* (opção 'Histórico da área de transferência') e ainda a funcionalidade de partilha dos itens do *clipboard* em todos os dispositivos *Windows 10* (opção 'Sincronizar entre dispositivos').

2.2.5.1 *Histórico do Clipboard*

Para fazer uso do histórico do *clipboard* é necessário nas definições do sistema, em 'Sistema', em 'Área de Transferência' ativar a opção 'Histórico da área de transferência' (Figura 7) (Hoffman, 2018).

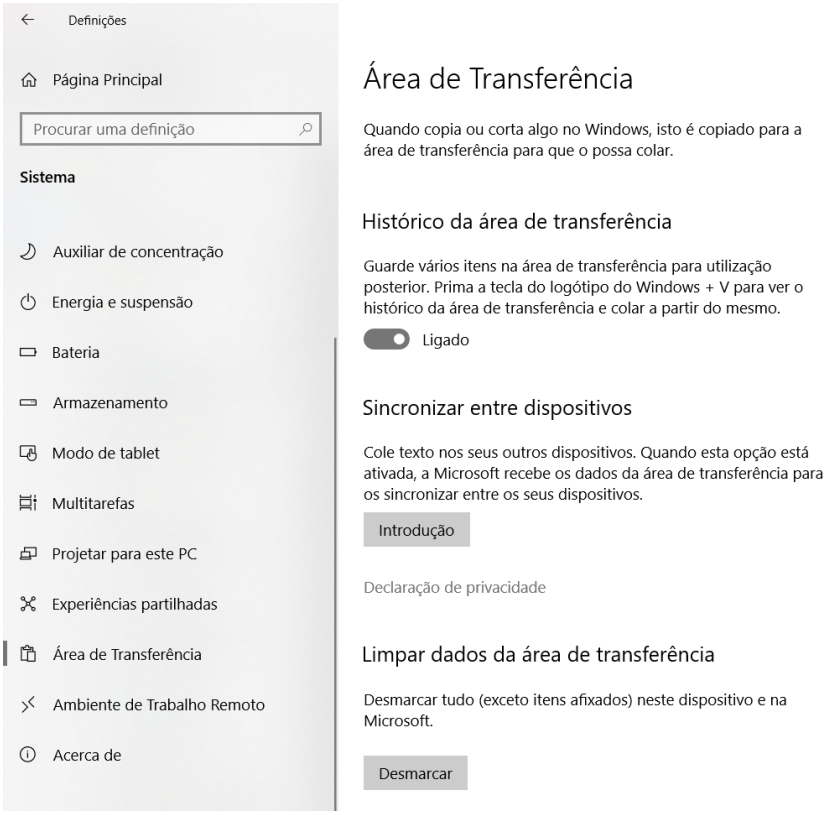


Figura 7: Ativação do histórico no novo *clipboard* no *Windows 10*

Após a ativação, para abrir o *clipboard* basta executar as teclas `Windows+V`, o que fará com que surja o painel do *clipboard* (Figura 8).

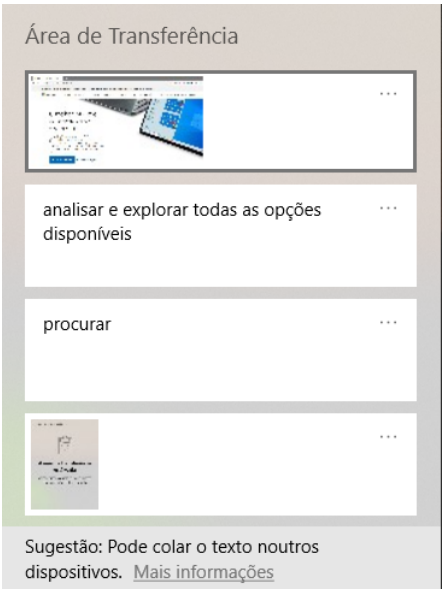


Figura 8: Painel do *clipboard* com os elementos copiados ou cortados

Assim, e tal como se pode ver na Figura 8, o novo *clipboard* apresenta uma interface gráfica com o histórico dos elementos copiados ou cortados, armazenando texto, *HyperText Markup Language* (HTML) e imagens até 4MiB. É de ter em conta que, por exemplo, a cópia de ficheiros não é registada no histórico do *clipboard*. De modo a copiar um dos elementos presentes no *clipboard*, basta clicar no elemento pretendido. No painel do *clipboard* o elemento mais recentemente copiado ou cortado encontra-se no topo da lista, sendo que o utilizador pode ainda fixar um elemento no topo da mesma.

O histórico do *clipboard* consegue armazenar até 25 elementos. No caso de ser necessário mais espaço para novos elementos, os elementos mais antigos são automaticamente eliminados desde que não estejam fixados (Hoffman, 2018; Microsoft, 2020). Existe ainda a possibilidade de o utilizador eliminar um dado elemento da lista individualmente. Para além disso, poderá ainda eliminar todos os elementos presentes na lista (opção 'Desmarcar tudo') com exceção dos elementos fixados que permanecem na área de transferência (Hoffman, 2018).

Uma das vantagens deste histórico é, por exemplo, no caso de o utilizador repetir uma operação de cópia anterior, basta recorrer ao *clipboard* em vez de efetuar novamente o processo de cópia da imagem, o que permite uma poupança de tempo na realização de tarefas. Um dos aspetos negativos do *clipboard* é o facto de este armazenar palavras-passe em *plain-text*, bastando aceder ao painel do *clipboard* para a visualizar. Este risco torna-se ainda maior, se a funcionalidade 'Sincronizar entre dispositivos' estiver ativa. Para além disso, no caso de no sistema existir alguma

aplicação que monitorize os dados que são copiados para o *clipboard*, a sua utilização pode-se tornar num risco para a segurança do utilizador.

No contexto da segurança na utilização do *clipboard* aquando a cópia de palavras-passe é relevante analisar a ferramenta KeePass[13]. O KeePass é um gestor de palavras-passe gratuito e *open-source*, que armazena numa base de dados encriptada as palavras-passe que se pretenda. Esta mesma base de dados está bloqueada com uma chave-mestre, ou seja, o utilizador só terá que introduzir a chave-mestre para aceder às suas palavras-passe (Reichl, 2020b). O KeePass é uma ferramenta que tem a funcionalidade '*Timed clipboard clearing*', ou seja, algum tempo após ter ocorrido a cópia de uma das palavras-passe existentes na base de dados do KeePass, a ferramenta limpa o *clipboard* automaticamente. Para além disso, quando o KeePass está a executar as palavras-passe estão sempre encriptadas, logo, mesmo que seja realizado um *dump* do processo relativo ao KeePass as palavras-passe estão protegidas (Reichl, 2020a). Deste modo, verifica-se que estas características do KeePass não só fornecem segurança mas também reduzem os riscos inerentes ao *clipboard* do *Windows*.

2.2.5.2 *Sincronização dos Elementos entre Dispositivos*

Na Figura 7 anteriormente apresentada, é possível verificar que existe ainda mais uma nova funcionalidade no *clipboard*, nomeadamente, a opção 'Sincronizar entre dispositivos'. Esta oferece a possibilidade de se aceder ao conteúdo copiado/cortado em vários dispositivos *Windows 10*, sendo efetuado o *upload* do histórico do *clipboard* para a *cloud*. É de ter em conta que para este processo é necessário que nos dispositivos em causa esteja configurada a mesma conta da nuvem Microsoft, por exemplo, uma conta do domínio outlook.com.

Na ativação da funcionalidade de sincronização, um aspeto a realçar é o facto de a Microsoft exigir a verificação da entidade, através do envio de uma mensagem de texto com um código para o contacto telefónico relativo à conta Microsoft que se encontra ativa no dispositivo em causa, o que garante alguma segurança a este processo.

Na Figura 9 verifica-se que existem duas opções de sincronização disponíveis, nomeadamente, a sincronização automática e a sincronização manual.

13 https://keepass.info/

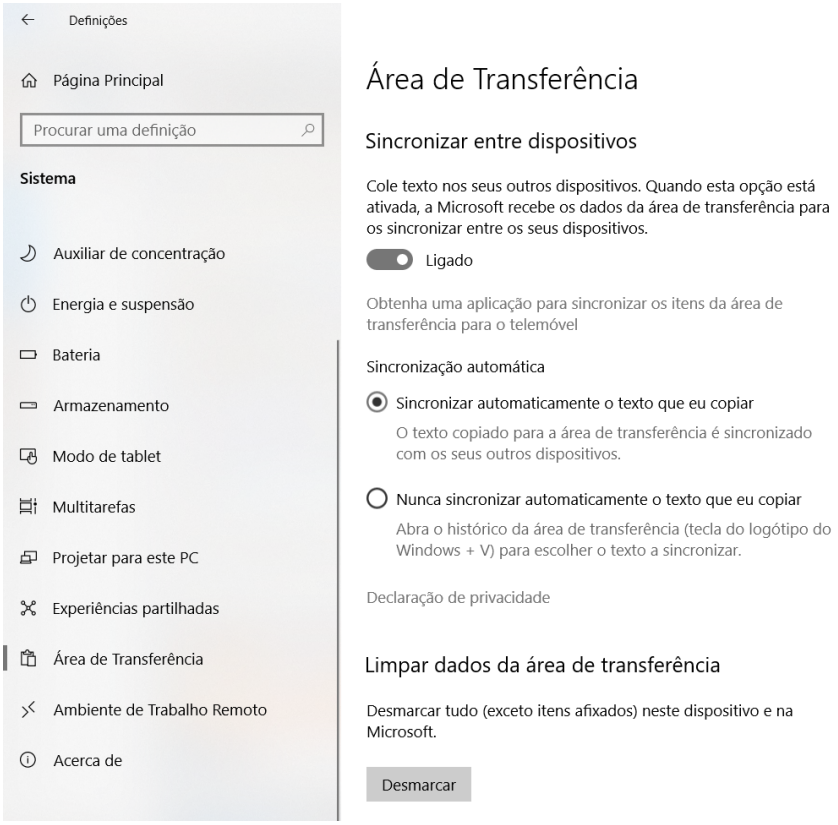


Figura 9: Sincronização entre dispositivos ativa, com sincronização automática

Por omissão, o sistema define a sincronização automática, que por sua vez, consiste em sincronizar automaticamente todo o texto que for copiado, levando a que o que seja copiado num dispositivo apareça de imediato no histórico do *clipboard* do outro dispositivo. Contudo, também é possível realizar uma sincronização manual, em que só é sincronizado com outros dispositivos os elementos que forem selecionados no painel do *clipboard* para tal (Figura 10). Note-se contudo que o *software* KeePass não permite que uma palavra-passe copiada para o *clipboard* seja disponibilizada a sistemas externos que estejam a sincronizar o *clipboard* com a máquina local.

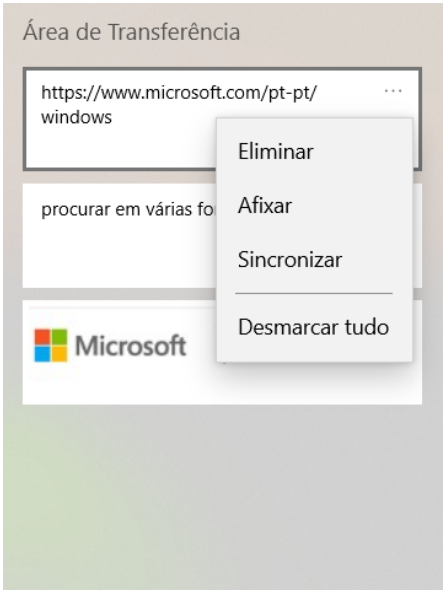


Figura 10: Visualização da opção de sincronização manual de um elemento

Comparando a sincronização manual com a automática a nível de segurança, a sincronização manual torna-se um processo mais seguro, pois assim não são automaticamente sincronizados dados sensíveis, tais como, palavras-passe (Hoffman, 2018).

2.2.5.3 *Importância Forense do Clipboard*

A nível forense, este novo *clipboard* pode-se tornar relevante para uma análise forense à memória. Uma vez que os dados do *clipboard* são armazenados na memória do sistema, torna-se possível extrair os seus dados da memória. O *clipboard* poderá conter dados relevantes ao nível forense, tais como, Uniform Resource Locator (URL) suspeitos e palavras-passe em *plain-text*, que, por sua vez, poderão auxiliar a investigação em curso (Okolica e Peterson L., 2011).

Contudo, constata-se que existem poucas ferramentas que permitem obter os dados do *clipboard*, contrariamente ao vasto número de ferramentas para obter, por exemplo, processos e atividade de rede em memória. Este aspeto revela que a análise de dados do *clipboard* ainda não é uma área muito explorada (Okolica e Peterson L., 2011). Mesmo assim, duas ferramentas disponíveis que permitem a análise dos dados presentes no *clipboard* de um sistema são a InsideClipboard (versão 1.15)[14] da Nirsoft e a Free Clipboard Viewer (versão 3.0)[15] da Comfort

14 https://www.nirsoft.net/utils/inside_clipboard.html
15 https://www.freeclipboardviewer.com/

Software Group. Posteriormente, será apresentada a análise destas duas ferramentas de visualização do conteúdo do *clipboard* (ver Capítulo 4).

2.3 CAPTURA DE CONTEÚDO DE MEMÓRIA

O modo como a primeira intervenção ao sistema é realizada é determinante para a preservação do ambiente digital. Os *first responders* (ou 'primeiros intervenientes') são os primeiros a interagir com o sistema, logo, acabam por desempenhar um papel relevante na preservação do ambiente digital e nos dados presentes neste. Assim, para os *first responders* torna-se relevante ter conhecimento das ações que devem ser tomadas, ações essas variáveis consoante os fatores que influenciam o estado do sistema em causa. Com a aquisição da memória - realização da cópia do conteúdo da memória para um dispositivo de armazenamento não volátil - o papel dos *first responders* ganhou ainda mais destaque. Tal deve-se, por exemplo, à volatilidade da informação presente na memória algo que a torna uma informação que pode ser perdida/alterada facilmente, por exemplo, simplesmente desligando o dispositivo.

Outrora - aquando a principal fonte forense correspondia ao suporte persistente - os procedimentos aplicáveis e aceites consistiam em desligar o sistema e fazer uma cópia dos dados presentes no disco para posterior análise. Estes tinham como objetivo minimizar a probabilidade de distorção dos dados do sistema de ficheiros. Contudo, estes procedimentos tradicionais não podem ser aplicados se se pretender a recolha de dados da memória. Assim, atualmente, em primeiro lugar, é necessário efetuar-se uma avaliação do ambiente digital tendo por base alguns dos fatores mais usuais.

O fluxograma presente na Figura 11 apresenta alguns dos fatores mais comuns com que um investigador se depara 'em campo' antes de se realizar o processo de aquisição da memória. Se o sistema for uma máquina virtual, então, o investigador poderá utilizar alguns dos recursos de uso de memória que o *hypervisor* fornece para, por exemplo, efetuar um *snapshot* ou usar a *introspection*. No caso de ser uma máquina física, ter-se-á que verificar se o sistema está ou não em execução. Se não estiver em execução o investigador efetuará uma aquisição de dados da RAM que estão armazenados em fontes persistentes como, por exemplo, os ficheiros de hibernação. Por outro lado, com o sistema alvo em execução, significa que é possível efetuar uma aquisição dos dados que estejam naquele momento na RAM do sistema. Assim, na presença de um sistema em execução, segue-se a questão acerca da existência ou não de privilégios *root*. Na ausência de privilégios *root*

terá que se realizar uma aquisição por *hardware*. Este tipo de aquisição exige avaliar se o sistema tem uma RAM superior a 4GiB, pois tal aspeto vai auxiliar na escolha da tecnologia utilizada para a aquisição. Com uma RAM superior a 4GiB, o investigador irá utilizar a tecnologia Peripheral Component Interconnect (PCI), que tem a desvantagem de ser uma tecnologia com um preço elevado. Já se a RAM for menor que 4GiB irá ser utilizada a tecnologia *Firewire*. Todavia, se existirem privilégios *root*, o investigador efetuará uma aquisição por *software*, aquisição esta que exige que se avalie pelo menos cinco fatores como, por exemplo, se será realizada uma aquisição local ou remota (Ligh Hale et al., 2014).

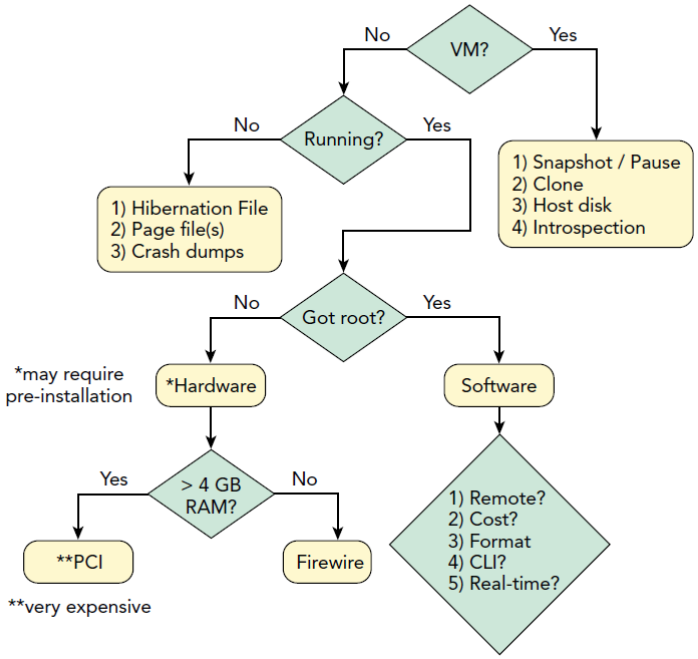


Figura 11: Fluxograma com os fatores iniciais a considerar antes da aquisição de memória (Ligh Hale et al., 2014)

Deve-se evitar a recolha de dados do sistema em determinados momentos nos quais se verificam alterações do sistema que ocorrem inclusive na memória. Exemplos desses momentos são, por exemplo, o iniciar e desligar do sistema e durante a execução de *backups* do sistema. Mesmo assim, é necessário considerar que o processo de aquisição acarreta alguns riscos, que podem, por exemplo, levar à corrupção de evidências. Um desses riscos está relacionado com o facto de a aquisição da memória não ser um processo atómico. Tal deve-se ao facto de o conteúdo desta mudar constantemente durante a utilização do sistema. Por conseguinte, mesmo durante a aquisição, por exemplo, outros processos poderão escrever em memória e poderão ainda ser estabelecidas novas ligações de rede (Ligh Hale et al., 2014).

2.3.1 *Aquisição Local e Remota*

Ao se realizar a aquisição com recurso a *software*, um dos pontos que se coloca em avaliação é se se realiza uma aquisição local ou uma aquisição remota. A aquisição local passa por se realizar o *dumping* da memória para uma unidade externa Universal Serial Bus (USB), ESATA ou *Firewire* conectada ao sistema alvo da aquisição. Neste tipo de aquisição é necessário ter algumas precauções como, por exemplo, não utilizar a unidade externa em uso em mais de um computador com o objetivo de se evitar o risco de *malware* se espalhar e não conectar uma unidade externa potencialmente infetada ao computador no qual a análise será realizada (Ligh Hale et al., 2014).

Ainda relativamente às unidades externas USB, ESATA, ou *Firewire* referidas anteriormente, dada a diversidade destes dispositivos presente no mercado, é necessária uma prévia avaliação antes da seleção final ser realizada. Um dos fatores a analisar é o seu sistema de ficheiros pois este deve suportar tamanhos de ficheiro pelo menos iguais à quantidade de RAM do sistema (Frade e Rodrigues, 2019). Dado o tamanho de ficheiros suportado por unidades NTFS ser teoricamente considerável e dado o tamanho da RAM dos computadores atuais, uma vantagem no processo de aquisição numa investigação é a garantia de que a unidade que guardará os dados da aquisição esteja formatada com NTFS.

Com a necessidade de efetuar-se a aquisição da RAM num número considerável de sistemas, uma aquisição local acaba por se revelar atualmente um processo demorado e não muito prático. Assim, em investigações digitais, começaram a ser realizadas aquisições remotas. O procedimento para a realização desta aquisição começa pelo envio da ferramenta de aquisição pela rede para o sistema seguindo-se a execução da ferramenta em causa através de uma tarefa ou serviço agendado. Com a aquisição da RAM pretendida concluída, procede-se ao envio da aquisição pela rede para o investigador efetuar a devida análise. Este mesmo referido envio pode ser executado através de um *network share*, mas isto só em último recurso pois utilizadores indevidos podem ter acesso ao mesmo e até mesmo pelo facto de os *worms* se espalharem através de *network shares*. Outra possível solução passa pelo envio dos dados via um *stream* com o *netcat* mas neste processo é preciso ter alguns cuidados para garantir a segurança do envio. Assim, um dos primeiros cuidados a tomar é evitar a exposição das credenciais de administrador e dos dados da aquisição sendo que, para tal, deve-se criar não só uma conta de administrador temporária mas também utilizar um canal encriptado (com uma aplicação que suporte Transport Layer Security (TLS)). Uma outra ação a ser

tomada é a configuração de uma *firewall* para limitar o tráfego entre o sistema alvo da aquisição e o sistema para o qual se envia os dados. Em todo este processo executado pela rede, torna-se necessário efetuar o cálculo dos valores de *hash* de integridade tanto antes como após o envio da aquisição. Deste modo, torna-se possível verificar que os dados não mudaram durante a transmissão, garantindo-se assim a fiabilidade dos dados adquiridos ao longo da aquisição (Ligh Hale et al., 2014). Para tal, o investigador pode recorrer a algumas ferramentas, tais como, a *Windows Powershell* e a ferramenta HashMyFiles da NirSoft[16] (Phillips, 2019). É de referir que posteriormente, neste mesmo documento, será apresentada a exploração e análise destes dois métodos de cálculo dos valores de *hash* (ver Capítulo 3).

2.3.2 *Snapshot vs. Live Analysis*

De modo a se analisar o conteúdo presente na memória de um sistema, um investigador poderá realizar uma aquisição por *snapshot* ou então uma *live analysis*. Assim sendo, a aquisição por *snapshot* é usualmente utilizada no momento da apreensão, sendo frequentemente empregue pela entidade encarregue da aquisição. Esta ao chegar ao local no qual se encontra o sistema efetua a aquisição da memória do mesmo no momento. Posteriormente, o investigador realiza a análise do conteúdo adquirido.

Relativamente à *live analysis*, este método de análise da RAM consiste em se efetuar uma triagem rápida do sistema, sem ser necessária a criação das usuais imagens de memória, imagens essas que normalmente têm um tamanho significativo. Deste modo, pode-se considerar que, antes da aquisição em si, é realizada uma análise prévia ao conteúdo presente na memória, diretamente no sistema alvo. O Rekall é uma ferramenta que permite realizar uma *live analysis*, podendo ser executada na mesma plataforma que está a analisar (R. Forensics, 2019b).

2.4 FERRAMENTAS DE AQUISIÇÃO DE MEMÓRIA

Neste subcapítulo serão apresentadas duas ferramentas de aquisição de memória gratuitas, desenvolvidas pela *Magnet Forensics Inc.*, a MAGNET RAM Capture e a MAGNET Process Capture. Para além destas, as ferramentas Winpmem e AcessData

16 https://www.nirsoft.net/utils/hash_my_files.html

FTK Imager também serão apresentadas, e ainda um método que permite adquirir um dado processo que esteja em memória.

2.4.1 *Captura da RAM*

Para se adquirir a memória de um sistema pode-se recorrer a ferramentas de aquisição, tal como, a MAGNET RAM Capture. A MAGNET RAM Capture é uma ferramenta que suporta sistemas *Windows* de 32 e 64 *bits* e efetua a captura da memória do sistema (M. Forensics, 2019a). Tal permite que, de seguida, se efetue a análise da mesma numa ferramenta de análise. Esta ferramenta não requer instalação, estando disponível no formato de executável. A versão utilizada nesta dissertação foi a 1.20 (24 de julho de 2019). Uma característica a realçar é o facto de a ferramenta deixar um *footprint* pequeno no sistema no qual se está a realizar a aquisição, ou seja, são minimizados os dados que são escritos na memória (M. Forensics, 2019a). A execução da ferramenta pode ser efetuada a partir da máquina local alvo ou a partir de um dispositivo USB. Este aspeto oferece uma certa comodidade ao investigador, podendo este escolher como pretende efetuar a aquisição.

Uma outra ferramenta de aquisição da RAM a mencionar é a Winpmem. A Winpmem realiza a captura da memória de sistemas *Windows*, *Linux* e *MacOS* e possui uma *Apache License* (Zollner et al., 2019). É de ter em conta que a Winpmem é executada em linha de comandos, não sendo necessário instalar a mesma. No que toca a sistemas *Windows*, esta ferramenta suporta várias versões deste sistema, tais como, o *Windows XP* e o *Windows 10* (SecTechno, 2018). A versão utilizada da Winpmem nesta dissertação corresponde à 2.1. A ferramenta pode ser executada no próprio sistema em que se pretende efetuar a aquisição, executada a partir de um dispositivo USB ou ainda a partir de um *shared cloud storage* (SecTechno, 2018).

Em investigações, de modo a se realizar uma aquisição da memória, poderá ser ainda utilizada a ferramenta AccessData FTK Imager, em sistemas de 32 *bits* e de 64 *bits*. Esta permite ainda que sejam obtidos os ficheiros `pagefile.sys`, `hiberfil.sys` e `swapfile.sys` do sistema em causa. Nesta dissertação analisa-se a versão 3.1.1.8 (Chandel, 2015a; Chandel, 2015b).

2.4.2 *Captura de Processos*

Relativamente à captura de processos em específico que estejam em memória, em sistema *Windows*, no Gestor de Tarefas do *Windows* (ou *Windows Task Manager*) (em 'Desempenho', através da opção 'Criar ficheiro de captura') é possível realizar-se o *dump* de um processo que esteja em memória no sistema. Assim, o Gestor de Tarefas do *Windows* permite criar um ficheiro de captura relativo a um dado processo (Figura 12). O formato do ficheiro criado é `.dmp`. Este método de aquisição de processos específicos em memória é prático e fácil de ser executado para o investigador e não requer permissão de administrador.

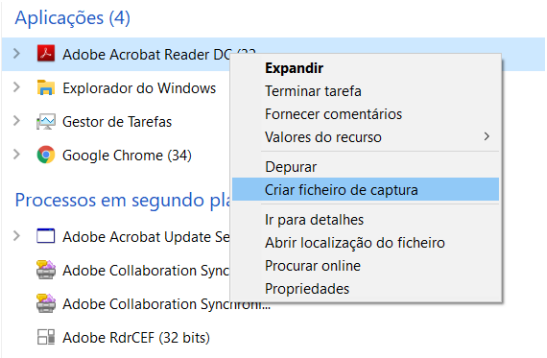


Figura 12: Criação de um ficheiro de captura relativo a um processo no Gestor de Tarefas do *Windows*

Por outro lado, a captura de processos que estejam em memória também pode ser realizada através de ferramentas, como, por exemplo, a ferramenta Magnet Process Capture. A MAGNET Process Capture é uma ferramenta que permite efetuar a captura de memória de processos em execução individuais. Por outras palavras, ao invés de se adquirir toda a memória do sistema alvo é simplesmente adquirida a memória referente aos processos em execução pretendidos. Esta ferramenta torna-se útil quando se está interessado só em algum processo em específico ou ainda em situações nas quais não existe muito tempo para se realizar a captura da memória do sistema. Nesta dissertação é empregue a versão 1.3 (15 de janeiro de 2020) (M. Forensics, 2019b).

2.5 FERRAMENTAS DE ANÁLISE DA MEMÓRIA

Neste subcapítulo serão apresentadas *frameworks* de análise de conteúdo de memória. Assim, primeiramente, será apresentada a *framework* Volatility da The Volatility

Foundation, seguindo-se a *framework* Rekall. Será ainda apresentado o módulo do Volatility para o Autopsy. Deste modo, serão apresentados e analisados alguns aspetos destas ferramentas de análise de dados da memória, de modo a se prosseguir, posteriormente, para a fase de exploração das ferramentas.

2.5.1 *Volatility*

A *framework* Volatility da The Volatility Foundation é uma ferramenta *open-source* (licença GNU General Public License (GPL)), desenvolvida em *Python* e que pode ser utilizada em todos os sistemas operativos que apresentem suporte para *Python*. O Volatility analisa os dados presentes na RAM de um sistema, sendo assim possível observar o *runtime state* num dado momento de um sistema (Ligh Hale et al., 2014; Volatility Foundation, 2019a).

É de ter em conta que esta ferramenta permite analisar *dumps* de memória de sistemas de 32 ou 64 *bits* com os sistemas operativos *Windows*, *Linux* e *MacOS*. Contudo, esta ferramenta não realiza aquisições. Importa notar que, fora a análise de aquisições, também é possível efetuar a análise de ficheiros de paginação, *crash dumps* e ficheiros de hibernação. O Volatility suporta ainda a análise de vários tipos de ficheiros como, por exemplo, VirtualBox ELF64 core dump e o formato Expert Witness Format (EWF) (E01) (Balapure, 2018; Volatility Foundation, 2019d).

De uma forma sucinta, o Volatility efetua a extração de artefactos digitais de uma aquisição de RAM; as técnicas de extração que são aplicadas são realizadas de forma independente do sistema alvo (Volatility Foundation, 2019d). O Volatility é bastante versátil e intuitivo, permitindo uma utilização simples e objetiva ao utilizador em investigações digitais.

2.5.1.1 *Formatos Disponíveis*

O Volatility é uma ferramenta de linha de comandos que se encontra disponível para os sistemas operativos *Windows*, *Linux* e *Mac OS X*. Esta *framework* é distribuída em vários formatos sendo um deles o executável autónomo do *Windows*. Um outro formato disponível é o *Windows Python module installer* que é utilizado quando se pretende analisar ou até mesmo efetuar alterações no código-fonte; este modo de utilização exige a instalação do *Python 2.7 interpreter* e das dependências. Por último, a ferramenta é ainda distribuída no formato de *packages* do código-fonte

(em ficheiros *zip* e *gzip/tarball*), sendo este o formato mais versátil pois pode ser aplicado em *Windows*, *Linux*, ou *Mac* (Ligh Hale et al., 2014).

Nesta dissertação, o método de utilização da ferramenta selecionado foi o formato executável para *Windows* pois, para além de o sistema operativo da máquina utilizada ser o *Windows*, o executável torna-se numa maneira rápida e simplista de se explorar esta ferramenta. Ao contrário dos outros métodos - que exigem a instalação do *Python 2.7 interpreter* e das dependências - a utilização do executável é direta sem a necessidade de efetuar-se alguma instalação prévia antes da sua exploração. Uma outra razão da escolha efetuada foi o facto de o objetivo consistir na exploração das funcionalidades e não na alteração/análise do código-fonte. A versão utilizada nesta dissertação foi a versão 2.6.

Posteriormente, neste documento, será ainda alvo de análise a versão *beta* 3.0 do Volatility[17].

2.5.1.2 *Aspetos da Ferrramenta*

De modo a um utilizador recorrer a esta ferramenta para efetuar uma análise a uma imagem de memória será necessário, em primeiro lugar, identificar o perfil adequado a aplicar à imagem em causa. Assim, de seguida, o utilizador especifica no comando o perfil e o *plugin* pretendido, de modo a obter um dado conjunto de dados presente em memória.

Relativamente aos perfis, o Volatility é uma ferramenta que apresenta vários perfis em que, ao ser selecionado um deles para a amostra em questão, é alterado o modo de análise que a ferramenta aplicará. A seleção de um dos perfis consiste em comunicar à ferramenta qual é o sistema operativo do sistema no qual se efetuou a aquisição. No caso do sistema operativo ser desconhecido então terá que se fazer uso do *plugin* `imageinfo` que irá sugerir alguns perfis adequados à amostra de RAM em questão. A razão de serem sugeridos vários perfis deve-se ao facto de versões do mesmo sistema operativo apresentarem, compreensivelmente, muitas semelhanças. Nas Tabelas 2 e 3 encontra-se a designação dos perfis juntamente com a descrição de cada um deles, sendo que a Tabela 2 é relativa aos perfis para *Windows* versão *desktop* e a Tabela 3 aos perfis para *Windows* versão *server* (Darknet, 2016; Volatility Foundation, 2019d).

17 https://github.com/volatilityfoundation/volatility3

Tabela 2: Perfis do Volatility para *Windows* versão *desktop* e uma breve descrição dos mesmos (Win=Windows; Win7=Windows 7; Win8=Windows 8; Win10=Windows 10).

Perfil	**Descrição**
WinXPSP1x64	*Win XP* SP1 x64
WinXPSP2x64	*Win XP* SP2 x64
WinXPSP2x86	*Win XP* SP2 x86
WinXPSP3x86	*Win XP* SP3 x86
VistaSP0x64	Perfil para *Win Vista* SP0 (*Service Pack 0*) x64
VistaSP0x86	*Win Vista* SP0 x86
VistaSP1x64	*Win Vista* SP1 x64
VistaSP1x86	*Win Vista* SP1 x86
VistaSP2x64	*Win Vista* SP2 x64
VistaSP2x86	*Win Vista* SP2 x86
Win7SP0x64	*Win7* SP0 x64
Win7SP0x86	*Win7* SP0 x86
Win7SP1x64	*Win7* SP1 x64
Win7SP1x64_23418	*Win7* SP1 x64 (6.1.7601.23418 / 2016-04-09)
Win7SP1x86	*Win7* SP1 x86
Win7SP1x86_23418	*Win7* SP1 x86 (6.1.7601.23418 / 2016-04-09)
Win8SP0x64	*Win8* x64
Win8SP0x86	*Win8* x86
Win81U1x64	*Win8.1 Update 1* x64
Win81U1x86	*Win8.1 Update* 1 x86
Win8SP1x64	*Win8.1* x64
Win8SP1x64_18340	*Win8.1* x64 (6.3.9600.18340 / 2016-05-13)
Win8SP1x86	*Win8.1* x86
Win10x64	*Win10* x64
Win10x64_10586	*Win10* x64 (10.0.10586.306 / 2016-04-23)
Win10x64_14393	*Win10* x64 (10.0.14393.0 / 2016-07-16)
Win10x86	*Win10* x86
Win10x86_10586	*Win10* x86 (10.0.10586.420 / 2016-05-28)
Win10x86_14393	*Win10* x86 (10.0.14393.0 / 2016-07-16)

Tabela 3: Perfis do Volatility para *Windows* versão *server* e uma breve descrição dos mesmos (Win=Windows).

Perfil	**Descrição**
Win2003SP0x86	*Win* 2003 SP0 x86
Win2003SP1x64	*Win* 2003 SP1 x64
Win2003SP1x86	*Win* 2003 SP1 x86
Win2003SP2x64	*Win* 2003 SP2 x64
Win2003SP2x86	*Win* 2003 SP2 x86
Win2008R2SP0x64	*Win* 2008 R2 SP0 x64
Win2008R2SP1x64	*Win* 2008 R2 SP1 x64
Win2008R2SP1x64_23418	*Win* 2008 R2 SP1 x64 (6.1.7601.23418 / 2016-04-09)
Win2008SP1x64	*Win* 2008 SP1 x64
Win2008SP1x86	*Win* 2008 SP1 x86
Win2008SP2x64	*Win* 2008 SP2 x64
Win2008SP2x86	*Win* 2008 SP2 x86
Win2012x64	*Win Server* 2012 x64
Win2016x64_14393	*Win Server* 2016 x64 (10.0.14393.0 / 2016-07-16)

O Volatility oferece ainda o *plugin* `kdbgscan` que visa identificar corretamente o perfil do sistema da aquisição e o endereço *kernel debugger block* (KDBG) correto, localizando e analisando as características do bloco de dados do *kernel debugger*. É de ter em conta que é com base na funcionalidade fornecida pelo *plugin* `kdgscan` que o `imageinfo` obtém as sugestões de perfis. É importante considerar que os *plugins* `imageinfo` e `kdbgscan` são exclusivos do *Windows*, existindo mais formas alternativas para *Mac OS* e *Linux* (Balapure, 2018; Ligh Hale et al., 2014). No que toca ao *Linux*, deve-se criar um perfil para o sistema operativo alvo da investigação. O *Linux* tem várias versões e subversões do *kernel* e ainda *kernels* personalizados. Deste modo, o número considerável de versões do *kernel* faz com que seja praticamente impossível que o Volatility disponibilize perfis para todas as *builds* possíveis do kernel do *Linux* (Ligh Hale et al., 2014). No caso de análises de aquisições de memória a sistemas *Mac OS* com o Volatility, será necessário efetuar o *download* do perfil apropriado ou então criar o perfil adequado. É de se referir que tanto para o *Linux* como para o *Mac OS*, na página do *GitHub* do Volatility [18], existem perfis para várias versões desses sistemas operativos (Ligh Hale et al., 2014).

18 https://github.com/volatilityfoundation/profiles

De modo a se concretizar uma análise a uma imagem de memória, o Volatility tem disponíveis vários *plugins* sendo que cada um deles permite obter uma dada informação relativa aos dados adquiridos (Volatility Foundation, 2019b).

2.5.1.3 *Volatility 3.0 Public Beta*

O Volatility 3.0 *Public Beta* foi lançada em outubro de 2019 e a sua *release* está agendada para outubro de 2020. Já em agosto de 2021 é esperado que o desenvolvimento e suporte do Volatility seja unicamente para a versão 3.x ao mesmo tempo que ocorrerá o fim do desenvolvimento e suporte para a versão 2.x. O Volatility 3.0 tem a licença *Volatility Software License 1.0* (Mike Auty, 2019).

Ao observar-se a evolução da análise forense da memória, verifica-se que, ao longo dos últimos anos, têm ocorrido algumas mudanças, como, por exemplo, o número de tarefas de análise (*plugins*) disponíveis e utilizados em investigações, bem como o aumento do tamanho das imagens de memória a serem analisadas. Tais mudanças levaram à criação da versão 3 da *framework* Volatility de modo a satisfazerem-se as necessidades atuais e futuras da análise de memória.

A versão 3 do Volatility foi escrita em *Python 3* sendo que toda a *framework* (por exemplo, *plugins*) foi reescrita e redesenhada. É de realçar que esta versão da *framework* apresentará, segundo os autores, um aumento do desempenho. No que toca ao tempo de execução dos *plugins*, no Volatility 3 verifica-se uma certa diminuição do tempo de execução comparativamente com o Rekall e com o Volatility 2 (Mike Auty, 2019).

Uma importante inovação da versão 3 é o facto de deixar de ser necessário, em qualquer sistema operativo, mencionar no comando o perfil adequado à imagem de memória em análise dado que ocorrerá a deteção automática de perfis. Esta nova versão da ferramenta apresenta ainda a avaliação automatizada de código na memória, de modo a que no processo de análise seja evitado o processo de *reverse engineering*. Nesta versão os *plugins* podem chamar diretamente outros *plugins*. É ainda de referir que se encontra disponível ao utilizador uma documentação abrangente da *Application Programming Interface* (API) (Mike Auty, 2019).

2.5.2 *Rekall*

A *framework* Rekall nasceu como uma *branch* alternativa do Volatility, tendo se tornado, posteriormente, uma ferramenta independente. O Rekall foi desenvolvido

em *Python* e é *open-source* (possui uma licença *GNU General Public License v2.0*) (Rekall, 2019c). Esta ferramenta permite não só a realização da análise de uma aquisição de memória, mas também a da aquisição da memória de um sistema. Esta pode ser utilizada para se efetuar uma *live analysis* de uma memória, sendo, para isso, executada na mesma plataforma que está a analisar.

No que toca ao processo de análise, nesta ferramenta é possível analisar *dumps* de memória de sistemas de 32 ou 64 *bits* com os sistemas operativos *Windows*, *Linux* ou *MacOS*. Esta ferramenta pode ser utilizada em todos os sistemas operativos desde que estes consigam executar *Python*.

O Rekall apresenta três interfaces, nomeadamente, a linha de comandos, a consola interativa e a *Webconsole GUI*, esta última apenas disponível nas versões mais antigas (Ribeiro et al., 2019).

Supondo-se a existência de vários computadores numa rede, com a ferramenta *Rekall Agent* é possível, em tempo real, efetuar análises remotas aos dispositivos. Esta característica do Rekall automatiza o trabalho dos investigadores, por exemplo, em grandes organizações. O *Rekall Agent* utiliza a plataforma de *cloud computing* da *Google* (*Google Cloud Platform*) e apresenta uma arquitetura cliente/servidor, sendo os clientes as máquinas que são monitorizadas. Assim, sempre que for necessário, o servidor pode solicitar a execução de certos *plugins* às máquinas que monitoriza para, de seguida, analisar o *output* destes (Michael Cohen, 2017).

É de referir que nesta dissertação será alvo de exploração a interface de linha de comandos do Rekall e ainda a consola interativa, não sendo explorada a funcionalidade remota da ferramenta nem a sua capacidade de aquisição. A versão da ferramenta que foi utilizada corresponde à 1.7.2.rc1 (*Hurricane Ridge*) (Rekall, 2016).

A SIFT Workstation[19] é um conjunto de ferramentas gratuitas e *open-source* forenses e de resposta a incidentes. Estas mesmas ferramentas já se encontram devidamente instaladas na SIFT Worktstation, algo que facilita o trabalho do investigador dado que não se torna necessário instalar as ferramentas. Exemplos de ferramentas presentes na SIFT Workstation são o Autopsy, o log2timeline e ainda as *frameworks* Volatility e Rekall. Deste modo, nesta dissertação, foi explorada a *framework* Rekall na SIFT Workstation (S. D. Forensics e Response, 2019). A SIFT Workstation permite analisar discos *raw* e vários sistemas de ficheiros. Esta suporta ainda vários formatos de evidências, tais como, `raw` (*Single raw file* (dd)) e Advanced Forensic Format (AFF). Esta ferramenta tem ainda estabelecidas

19 https://digital-forensics.sans.org/community/downloads

diretrizes/roteiros que definem como as evidências são analisadas (só leitura), verificando se as evidências não foram alteradas (S. D. Forensics e Response, 2019).

2.5.2.1 *Alguns Aspetos acerca da Ferramenta*

Relativamente à utilização do Rekall, esta é bastante simples e intuitiva.

Nesta ferramenta o utilizador não precisa identificar o perfil a aplicar, pois o Rekall efetua essa tarefa automaticamente. Mesmo assim, e tal como na *framework* Volatility, é possível especificar o perfil a utilizar na análise. É de ter em conta que o Rekall apresenta um repositório público de perfis[20].

Já no que diz respeito às técnicas de análise, estas são selecionadas consoante o sistema operativo examinado. Tal como no Volatility, para o utilizador obter a informação presente na imagem de memória, é necessário executar um conjunto de *plugins*.

O Rekall apresenta a linguagem EFliter que permite filtrar o *output* dos *plugins* do Rekall, tornando-se possível que o utilizador determine que dados são relevantes que surjam em *output*. Para tal, basta efetuar pesquisas, recorrendo-se, por vezes, a operadores, como, por exemplo, o operador *where* (B. R. Forensics, 2016).

2.5.2.2 *Trabalho Relativo à framework Rekall*

No documento (Gomes et al., 2019) são apresentados alguns cenários de testes que foram colocados em prática no Rekall. Os alvos desses testes foram a procura de senhas e chaves de cifragem e ainda de processos maliciosos que estivessem a correr no sistema.

Um dos testes teve como alvo o gerenciador de senhas LastPass de modo a encontrar em memória as palavra-passe mestre e as palavras-passe do cofre. Após a análise verificou-se que se pode aceder às senhas que são armazenadas pelo LastPass se o utilizador saber quais são, ou seja, apenas pesquisando pela própria palavra-passe é que se pode encontrá-la. Concretamente, não é possível encontrar a palavra-passe mestre nem as palavras-passe do cofre sem as conhecer previamente. Um outro teste realizado foi relativo a programas de compressão, extração e encriptação, no qual foram utilizados os programas 7zip, winrar, zip e veracrypt numa máquina virtual *Windows XP*. Este teste permitiu verificar que não é possível obter as chaves destes mesmos programas por análise da memória.

20 https://github.com/google/rekall-profiles

No que toca a testes relacionados com processos maliciosos em execução foram executados três, nomeadamente, um com um *keylogger*, outro com um *backdoor* e ainda um com o *wannacry*.

No teste com o *keylogger* local - captura os *inputs* do utilizador no teclado mas só os guarda localmente - verificou-se que o processo relativo a este não estava escondido na lista de processos, não tendo sido contudo detetado como um processo malicioso. Deste modo, um *keylogger* é algo difícil de ser detetado se este conseguir se apresentar na lista de processos de uma maneira discreta e ainda se não realizar quaisquer comunicações remotas.

Para o teste com o *backdoor*, foi criado um ficheiro executável (`exe`) com um *payload* malicioso que instanciasse um servidor Virtual Network Computing (VNC) na máquina alvo, para, remotamente, transmitir o conteúdo gráfico da máquina alvo para a máquina atacante. Na análise, o Rekall permitiu visualizar a relação entre o processo escondido relativo ao *backdoor* com a conexão remota estabelecida entre o sistema e a máquina atacante. Assim, o Rekall revelou-se útil para detetar a execução de *backdoors*.

Por fim, foi efetuado um teste com o *software* malicioso *Wannacry*, (fabricmagic72, 2019) que encripta os ficheiros do utilizador em troca da realização de um pagamento e propaga-se automaticamente pela rede de modo a infiltrar-se em máquinas com o sistema operativo *Windows* que tenham a versão 1 do Server Message Block (SMB) ativa. Com as tarefas realizadas na análise foi obtido o executável do *Wannacry* e ainda *links* utilizados pelo *Wannacry*, por exemplo, para o *download* do *TOR browser*. Apesar disso, não foi possível, por exemplo, descobrir a chave para realizar a desencriptação e ainda prever qual o comportamento seguinte do *Wannacry*.

2.5.2.3 *Rekall Session*

A *Session* é um objeto que o Rekall utiliza para encapsular a análise de uma dada imagem, ou seja, contém dados acerca da análise da imagem de memória atual. Quando é dado início à análise de uma imagem de memória utilizando uma consola interativa, a *session* persiste em memória. Assim, a informação encontra-se disponível em memória para outros módulos que necessitem da mesma. É por este mesmo aspeto que o Rekall se torna uma ferramenta de análise rápida, permitindo tornar operações futuras sobre uma imagem mais rápidas. Com o comando '`print session`' é possível visualizar-se o conteúdo desse mesmo objeto para uma dada imagem. Como se pode ver na Listagem 1, a *Session* tem duas partes, nomeadamente, a '*configuration (Config)*' e a '*cache*'. Um exemplo de informação que é disponibilizada

é, por exemplo, a designação da imagem de memória em análise (em *Config*). Já em *Cache* verifica-se a existência de objetos que foram guardados na *Session* após o *plugin* `pslist` ser executado pela primeira vez, sendo que estes ficheiros podem ser reutilizados na próxima execução desse mesmo *plugin* (R. Forensics, 2019c).

Listagem 1: *Output* da execução do comando '`print session`'

```
[1] ch2.dmp 12:55:01> print session
--------------------> print(session)
Rekall Memory Forensics session Started on Wed Dec 18 12:51:12 2019.

Config:
{
  __dummy = False
  autodetect = ['nt_index', 'pe', 'rsds', 'windows_kernel_file', 'linux_index',
  ↪  'linux', 'osx', 'ntfs', 'tsk']
  autodetect_build_local = basic
  autodetect_build_local_tracked = {'nt', 'tcpip', 'win32k', 'ntdll'}
  autodetect_scan_length = 18446744073709551616
  autodetect_threshold = 1.0
  base_filename = ch2.dmp
  buffer_size = 20971520
  (...)
  filename = C:\Users\santo\Desktop\ch2.dmp
  (...)
  live = None
  (...)
  timezone = UTC
  verbose = False
  version = False
}

Cache (<FileCache @ C:/Users/santo/.rekall_cache/sessions/v1.0/sessions/a04a3d2f
↪  2563239bb961e361d7d05a4b76fe6315>):
{
  (...)
  pslist_CSRSS = {2276337792, 2276049544, 2307598224, 2307205400, 2307898136,
  ↪  2504612888, 2276197920, 2277941920, 227 ...
  pslist_Handles = {2295347712, 2276337792, 2276049544, 2307598224, 2307898136,
  ↪  2307205400, 2504612888, 2306563104, 230 ...
  pslist_PsActiveProcessHead = {2295347712, 2276337792, 2276049544, 2307598224,
  ↪  2307205400, 2504539416, 2307898136, 2504612888, 230 ...
  pslist_PspCidTable = {2295347712, 2276337792, 2276049544, 2504652408,
  ↪  2307598224, 2504539416, 2307898136, 2307205400, 250 ...
  pslist_Sessions = {2295347712, 2276337792, 2276049544, 2307598224, 2307205400,
  ↪  2307898136, 2504539416, 2504612888, 227 ...
}
```

2.5.2.4 *Automatização de Tarefas*

O Rekall permite a automação de tarefas, sendo que para tal basta incorporar as funcionalidades pretendidas da ferramenta num *script* em *Python*. Tal permite, por exemplo, criar um *script* com os *plugins* que se pretendem aplicar a um determinada imagem de memória e até mesmo definir o ficheiro para o qual se pretende enviar o *output* da execução de um dado *plugin*. Assim, a automação de tarefas possibilita a simplificação de tarefas usuais na ferramenta (R. Forensics, 2019c).

Listagem 2: *Script* exemplo de automatização de tarefas

```
import logging

logging.basicConfig(level=logging.DEBUG)

from rekall import session
from rekall import plugins

s = session.InteractiveSession(filename="nomeFicheiroImagemMemoria")

fich = open("fichResultados.txt", "a")

renderer=text.TextRenderer(session=s, fd=fich)

with renderer.start():
  plugin=s.plugins.pslist()
  if plugin!=None:
      plugin.render(renderer)
```

Os passos necessários para a automatização de tarefas passam por importar os módulos do Rekall, criar um objeto de sessão, criar um objeto de renderização, instanciar os *plugins* e, por fim, a execução do *plugin* através do objeto de renderização. O objeto de sessão é o que irá estabalecer ligação com o Rekall e, por sua vez, apresentará toda a informação relativa à imagem de memória em causa. Já o objeto de renderização tem como finalidade a definição do formato de saída dos dados, sendo possível o envio do *output* para a consola. É de referir que a automação de tarefas no Rekall não foi explorada na prática nesta dissertação (R. Forensics, 2019c).

Mesmo assim, na Listagem 2 encontra-se um exemplo de automatização de tarefas no Rekall. Neste exemplo, os resultados da execução do *plugin* `pslist` são enviados para um ficheiro. Assim, foi necessário criar o objeto de sessão, no exemplo designado por '`s`'. Na função '`open()`' é definido o ficheiro que irá guardar os dados e, de seguida, define-se em que modo se pretende abrir o ficheiro. Segue-se a criação do objeto de renderização (designado por '`renderer`'), que, por sua vez, se encontra associado ao ficheiro que guardará os resultados obtidos. Posteriormente, é iniciado o '`renderer`', permitindo que o *output* do *plugin* seja armanezado no ficheiro.

2.5.3 *Módulo Volatility Para o Autopsy*

Foi ainda alvo de análise e exploração o módulo do Volatility para o Autopsy desenvolvido por Mark McKinnon. Este módulo foi desenvolvido em *Python* e *Jython*[21] com o objetivo de executar o Volatility sobre uma dada imagem de

21 http://www.jython.org

memória. Deste modo, a existência deste módulo permite realizar análises forenses à memória na ferramenta Autopsy. É de referir que este módulo tem uma licença *GNU General Public License v3.0* (McKinnon, 2019).

Este módulo permite que possa ser executado sobre várias imagens de memória, mostrando o *output* do *plugin* para cada imagem de memória.

Há que mencionar a existência de três módulos deste *plugin*, nomeadamente, o 'Volatility Module', o 'Volatility Dump File Module' e o 'Volatility Convert Hiber/Crash Module'. O módulo designado por 'Volatility Module' permite executar *plugins* do Volatility sobre imagens de memória, enquanto que o 'Volatility Dump File Module' efetua o *dump* de ficheiros de imagens de memória. Já o 'Volatility Convert Hiber/Crash Module' permite converter ficheiros de hibernação/*crash*, usando para tal o Volatility. Neste trabalho serão alvo de exploração o 'Volatility Module' e o 'Volatility Dump File Module'.

2.6 AUTOPSY

O Autopsy é uma plataforma forense digital *open-source* e é utilizado por investigadores no contexto de investigações digitais. O *software* Autopsy é a interface gráfica para a ferramenta The Sleuth Kit (TSK) e outras ferramentas forense (TSK, 2019a). O TSK é *open-source* e é um conjunto de ferramentas de linha de comandos que permite efetuar análises forenses, sendo utilizado pelo Autopsy bem como por outras ferramentas. Dado o facto de o Autopsy apresentar uma interface gráfica, o TSK ao ser utilizado por este, permite que sejam simplificados os processos para encontrar as evidências forenses pertinentes (Infosec, 2019; TSK, 2019b).

Assim, o principal objetivo do Autopsy consiste precisamente na procura de evidências digitais que permitam perceber o que aconteceu num sistema, podendo-se comprovar, por exemplo, um dado incidente ou ação maliciosa executada. O Autopsy tem a particularidade de poder ser utilizado, por exemplo, para análises forenses a discos rígidos de computadores, dispositivos móveis como também à memória de sistemas.

Este *software* pode ser utilizado em sistemas operativos *Windows* (como o caso desta dissertação), *Linux* e *Mac OS*. A sua instalação é simples e rápida e ocorre através de um simples executável disponível no *website* oficial, tal como foi realizado nesta dissertação. É de mencionar que nesta dissertação será utilizada a versão 4.16.0 do Autopsy.

As suas funcionalidades são interessantes e úteis no contexto de uma investigação digital, auxiliando os investigadores. O *software* permite a integração de *plugins*, sendo permitido o desenvolvimento de módulos em *Java* ou *Python*. No que toca ainda à sua utilização, esta é simples e intuitiva para o utilizador. Uma das vantagens do Autopsy é a disponibilização de documentação tanto para utilizadores bem como para *developers*/programadores.

2.6.1 *Utilização do Autopsy*

No contexto de uma investigação digital, o fluxo de trabalho de um utilizador no Autopsy inicia-se pela criação de um caso, adicionando-se, de seguida, a este uma ou mais fontes de dados. O tipo de fonte de dados poderá ser uma imagem (formato E01 e imagens em raw(/imagens em bruto) (dd) são os formatos suportados), um disco local ou ficheiros lógicos e pastas. O próximo passo diz respeito à configuração dos *ingest modules*, que, por sua vez, vão ser executados em *background*, analisando os dados e, se existirem resultados pertinentes, estes serão apresentados (TSK, 2018b). Um destes módulos é o *PhotoRec Carver Module* que permite recuperar ficheiros de espaços não alocados. Outro módulo é o *Keyword Search Module* que, com base numa lista de palavras-chave, procura por ficheiros com palavras em específico nos mesmos; é de ter em conta que pode ser configurada a lista de palavras a utilizar (TSK, 2018d).

Com o término da execução dos *ingest modules*, passa-se à fase de análise dos dados. No lado esquerdo da interface, é apresentada uma árvore em que cada fonte de dados do caso encontra-se no nó '*Data Source*'. É de referir que o nó '*Data Source*' mostra todos os dados presentes no caso. Já no nó '*Views*' os dados são devidamente organizados por categorias de informações; é no sub-nó '*Results*' que são apresentados os resultados da execução dos *ingest modules*. Selecionando-se um dos nós da árvore, no lado direito da interface gráfica visualiza-se uma lista de ficheiros e, ao selecionar-se um deles, na parte inferior do ecrã - na secção '*Content Viewer*' - pode-se explorar o item selecionado, surgindo, por exemplo, o conteúdo do mesmo e os dados hexadecimais. Na análise que é efetuada, por exemplo, poderá ser possível encontrar artefactos da *web* como atividade recente na rede (em '*Results*', '*Extracted Data*') e conteúdo multimédia (em '*Views*', '*File Types*') (TSK, 2018d).

Uma funcionalidade útil do Autopsy é a *Timeline* (em '*Tools*', '*Make Timeline*') que em investigações permite identificar, por exemplo, quando uma dada atividade ocorreu e para verificar que ações foram executadas antes e após uma dada ação.

Faz uso da data e hora de, por exemplo, ficheiros, artefactos da *web* e dados EXIF. Apresenta dois modos, sendo um deles o gráfico de barras (em '*Mode*', selecionar '*Counts*') (Figura 13) que não se foca tanto no detalhe das ações mas sim no quanto ocorreu; assim, por exemplo, pode ser útil para se saber a quantidade de dados ocorridos num determinado período de tempo. Já o outro modo da *Timeline* (em '*Mode*', selecionar '*Details*') (Figura 14) fornece detalhes sobre os dados (A. D. Forensics, 2013; TSK, 2019c).

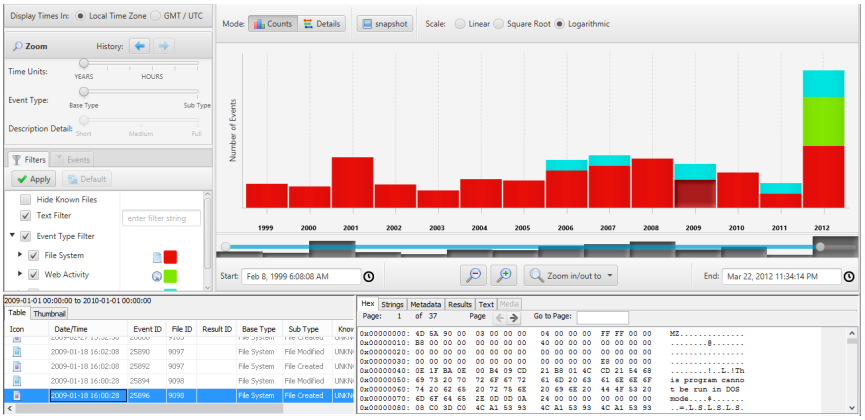


Figura 13: Interface da *Timeline* do Autopsy em modo gráfico de barras (TSK, 2019c)

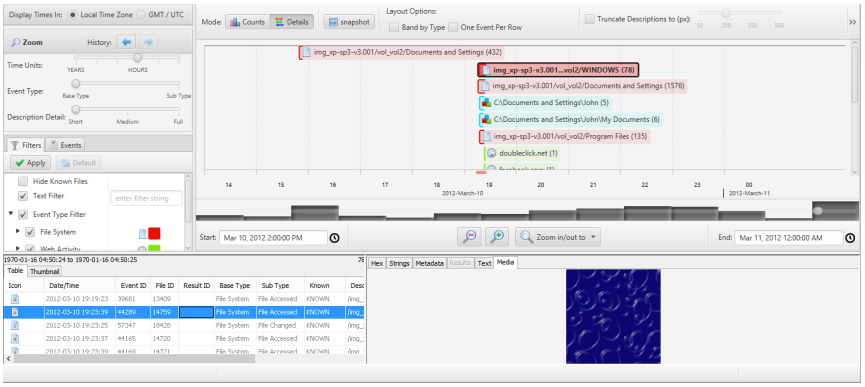


Figura 14: Interface da *Timeline* do Autopsy com os detalhes dos dados em análise (TSK, 2019c)

Também pode ser obtido um relatório relativo ao caso, que poderá ser configurado pelo utilizador. Ainda relativamente ao relatório, se o caso apresentar mais do que uma fonte de dados e se pretender-se um relatório relativo a uma só dessas mesmas fontes é necessário criar um caso para cada fonte de dados (TSK, 2018d).

É de referir ainda que o Autopsy permite a realização de *live triages*. Este *software* permite também que vários utilizadores em computadores diferentes tenham o mesmo caso aberto (TSK, 2018c; TSK, 2018e).

Na Figura 15 encontra-se a janela principal do Autopsy, no decurso de uma análise de um caso.

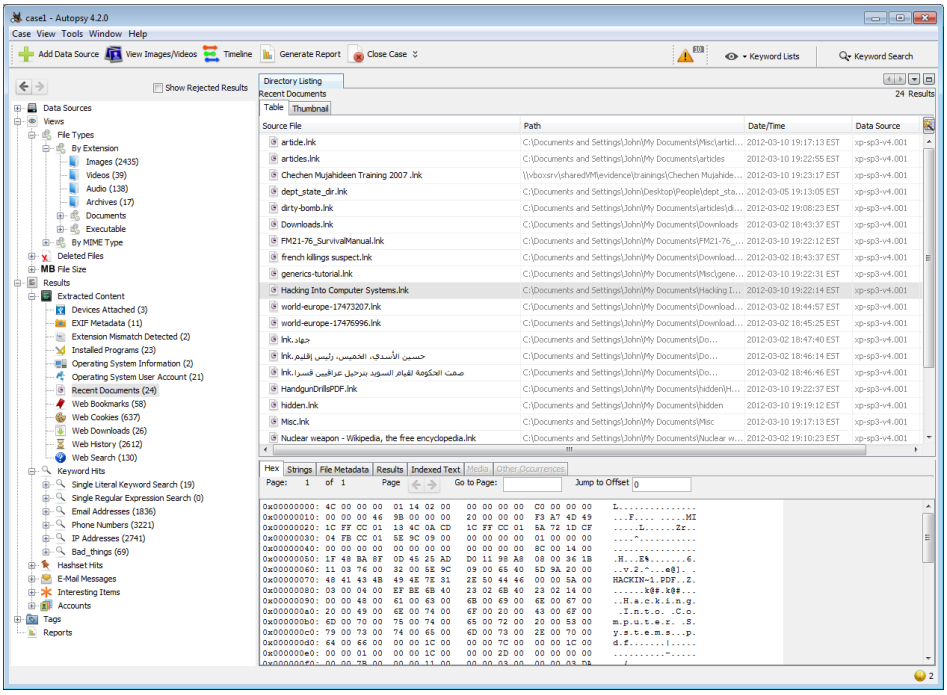


Figura 15: Janela Principal do Autopsy (TSK, 2018d)

2.6.2 *Tipos de Módulos*

O Autopsy é uma ferramenta que permite que módulos sejam adicionados. É de ter em conta que existem vários tipos de módulos, os quais podem ser aplicados a locais diferentes na ferramenta de acordo com o objetivo implícito. Assim, existem os *ingest modules*, os *report modules*, os *content viewers* e os *result viewers* (TSK, 2018a).

Os *ingest modules* são módulos que são executados ao se adicionar uma fonte de dados a um caso com o objetivo de analisar os dados desta. Estes apresentam dois tipos, nomeadamente, os '*File Ingest Modules*' e os '*Data Source Ingest Modules*'. Os '*File Ingest Modules*' - como o '*file type identification*' - são módulos que ao serem executados permitem explorar o conteúdo dos ficheiros da fonte de dados, disponibilizando um método que é chamado para cada um dos ficheiros da imagem forense. Já os '*Data Source Ingest Modules*' são módulos que podem recorrer à base de dados do Autopsy que indexa todos os ficheiros e diretórios existentes na imagem forense, para consultar ficheiros e analisá-los consequentemente. Estes são chamados por imagens e conjuntos de ficheiros lógicos uma só vez, sendo um

exemplo a análise de artefactos da *web*. É de referir que este tipo de módulo não tem acesso ao conteúdo de ficheiros *zip*.

Os *report modules* tem como objetivo tanto a criação de um relatório com os resultados mas também, em alguns casos, a realização da análise em si. O Autopsy permite que sejam criados diferentes tipos de relatórios, por exemplo, relatório HTML, relatório em *Excel* e ainda relatório Keyhole Markup Language (KML), este último para dados de localização geográfica. É possível criar módulos adicionais de modo a serem obtidos relatórios com formatos personalizados (TSK, 2018f).

Por sua vez, os *content viewers* são módulos gráficos que dizem respeito à parte inferior direita da interface, na qual é possível visualizar, por exemplo, as *strings* extraídas do ficheiro. Assim sendo, estes módulos são responsáveis por apresentar um dado ficheiro selecionado pelo utilizador.

Na Figura 16 encontra-se a interface do Autopsy com a devida identificação das suas principais áreas, tais como, as áreas '*Result Viewer*' e '*Content Viewer*' (The Sleuth Kit - TSK, 2018).

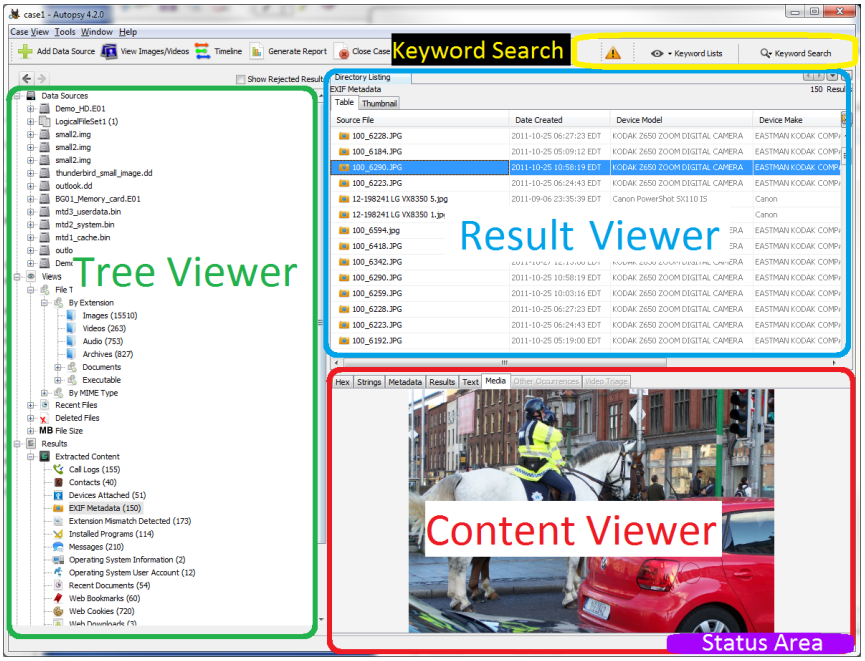


Figura 16: Interface do Autopsy com a indentificação das áreas '*Result Viewer*' e '*Content Viewer*' (The Sleuth Kit - TSK, 2018)

Por fim, os *result viewers* correspondem aos módulos responsáveis pela parte superior direita da interface, sendo estes os *viewers* que permitem visualizar os ficheiros implícitos em tabela ou *thumbnail*. Sendo assim, estes são então os módulos responsáveis por mostrar informação sobre um conjunto de ficheiros.

Na Tabela 4 encontram-se os módulos acima apresentados e uma breve descrição de cada um dos tipos de módulos.

Tabela 4: Módulos do Autopsy com a respetiva descrição de cada um dos mesmos

Módulo	**Descrição**
Ingest modules	Módulos executados sobre uma fonte de dados que seja adicionada a um caso.
Report modules	Módulos que permitem a criação de diferentes tipos de relatórios.
Content viewers	Módulos responsáveis por apresentar informações sobre um dado ficheiro selecionado.
Result viewers	Módulos que permitem apresentar dados acerca de um conjunto de ficheiros.

2.6.3 *Blackboard*

Um outro conceito relevante a ser compreendido é o *blackboard*. Este permite que os módulos comuniquem não só uns com os outros mas também com a interface do utilizador. Consequentemente, um dado módulo ao colocar os seus resultados no *blackboard*, um outro módulo pode aceder aos mesmos. Por exemplo, os *report modules* solicitam ao *blackboard* o que se deve reportar no relatório a criar. Assim sendo, em termos práticos, considera-se que o *blackboard* é um conjunto de artefactos, cada um com um dado tipo e com atributos que, por sua vez, são conjuntos de pares nome-valor (TSK, 2018a).

2.7 SÍNTESE

Neste capítulo, inicialmente foi abordada a análise forense a discos rígidos - procedimentos a executar e ferramentas a utilizar - e foram ainda apresentados alguns dos conceitos inerentes à análise de um sistema.

Para se adquirir a memória de um sistema é necessário tomar algumas precauções para garantir o sucesso do processo de aquisição. Para além disso, é importante que a aquisição da memória só ocorra após o investigador analisar o sistema em causa e

o ambiente em que este se insere. Uma das razões para tal é, por exemplo, avaliar se será realizada uma aquisição remota ou local. Com a aquisição realizada, inicia-se a fase de análise da memória, podendo ser realizada uma aquisição por *snapshot* ou uma *live analysis*. Tanto a aquisição como a análise da memória de um sistema requerem a utilização de ferramentas, sendo que algumas delas foram apresentadas neste capítulo.

Antes de se iniciar uma aquisição ou análise da memória o investigador deve explorar quais as melhores ferramentas para o caso em questão, mais precisamente, observar e comparar as suas funcionalidades, verificando as ferramentas que melhor se adaptam para o cenário em causa.

O Autopsy é um *software* utilizado em investigações digitais, permitindo descobrir evidências digitais. Esta ferramenta pode ser utilizada para a análise forense de memórias de sistemas. Neste capítulo, foram apresentados alguns aspetos e conceitos do Autopsy, como, por exemplo, o *blackboard* e os tipos de módulos. Algo a realçar, é o facto de esta ferramenta permitir a integração de *plugins*.

3

FERRAMENTAS DE AQUISIÇÃO DA MEMÓRIA

De modo a se aceder à informação presente na memória de um sistema, é necessário recorrer a uma das várias ferramentas de aquisição de conteúdo em memória existentes. O utilizador tem a possibilidade de adquirir toda a memória do sistema ou então unicamente um processo em específico, existindo ferramentas para ambas as situações.

Assim, este capítulo contém as explorações realizadas a ferramentas de aquisição da memória, tal como, a Magnet RAM Capture. De seguida, são apresentadas as análises realizadas às ferramentas Winpmem e AccessData FTK Imager. No contexto da exploração da ferramenta AccessData FTK Imager, é apresentada a forma de se obterem os ficheiros `pagefile.sys`, `hiberfil.sys` e `swapfile.sys`. No que toca à captura de processos em específico que estejam na memória, é apresentada a exploração efetuada à ferramenta Magnet Process Capture. É ainda apresentada uma comparação entre as ferramentas de aquisição analisadas, sendo também apresentada uma tabela comparativa entre as ferramentas.

Quando se realiza uma análise remota é necessário garantir que os dados não foram alterados no processo de envio. Assim, e apesar de não ser explorada a aquisição remota nesta dissertação, foram analisadas ferramentas de verificação de valores de *hash*, mais precisamente, a *Windows Powershell* no contexto em causa e a ferramenta HashMyFiles da NirSoft.

Nesta dissertação, as análises efetuadas às ferramentas de aquisição de memória e os consequentes processos de aquisição foram executados no sistema operativo *Microsoft Windows 10 Home* versão 10.0.17763 Compilação 17763, com 8 GiB de RAM e com um processador Intel(R) Core(TM) i7-7500U CPU @ 2.70GHz.

3.1 MAGNET RAM CAPTURE

A MAGNET RAM Capture é uma ferramenta que realiza a captura de memória de um computador, sendo que a memória adquirida pode ser exportada nos formatos `.dmp`, `.raw` ou `.bin` (M. Forensics, 2019a).

O executável desta ferramenta utilizado nesta dissertação encontra-se disponível em [1], sendo que, só se obtem acesso à ferramenta após o preenchimento do formulário presente no referido *link*. Após a obtenção do ficheiro executável desta ferramenta, este foi executado a partir da máquina local alvo da aquisição que se pretendia realizar, que, por sua vez, correspondia a um sistema *Windows*.

A interface gráfica desta ferramenta é simples e objetiva, até mesmo para utilizadores que não são peritos na área, como se pode observar na Figura 17. Para dar início à aquisição é necessário indicar a designação do ficheiro que armazenará a aquisição realizada ao mesmo tempo que se define o caminho onde este mesmo ficheiro será guardado. O estado da aquisição é fornecido pela barra de progresso que surgirá na interface.

Um aspeto a referir é ainda a opção '*Segment size*' que permite definir se os ficheiros devem ou não ser fragmentados. Nesta opção existem alguns valores que correspondem a tamanhos de fragmentação, tais como, 1GiB e 4GiB. Por omissão, a opção selecionada é '*Don't split*', ou seja, a aquisição será realizada sem qualquer fragmentação dos ficheiros. A possibilidade de fragmentação fornecida por esta ferramenta, em investigações, permite alguma comodidade ao utilizador. Por exemplo, numa dada investigação, se se utilizar um dispositivo USB com o sistema de ficheiros FAT32 para executar a ferramenta, no caso de o sistema alvo da aquisição apresentar uma RAM maior que 4 Gigabyte (GB), deve-se fragmentar os ficheiros. A razão da fragmentação deve-se à capacidade máxima do tamanho de ficheiro do sistema de ficheiros FAT32 (M. Forensics, 2015).

Nesta dissertação foi realizada, por exemplo, como teste da ferramenta uma captura à RAM do sistema que criasse um ficheiro `.raw`. Na Figura 17 encontra-se a interface da ferramenta num dado momento da captura. A captura efetuada foi rápida (1 minuto e 51 segundos) e não exigiu a interação do utilizador com a mesma no seu decurso. Após a aquisição estar concluída é então criado um ficheiro com os dados da memória do sistema. Um outro teste ainda realizado consistiu na criação de um ficheiro `.dmp` com os dados do sistema alvo de aquisição, cuja duração desta foi semelhante à do teste anterior.

1 https://www.magnetforensics.com/resources/magnet-ram-capture/

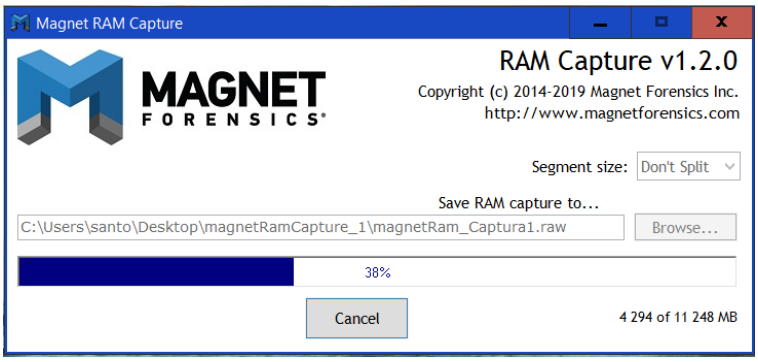


Figura 17: Interface da ferramenta durante uma aquisição

No que diz respeito à análise dos ficheiros obtidos através desta ferramenta, esta não pode ser realizada com as ferramentas Rekall e Volatility. Assim, deve ser empregue pelo investigador, por exemplo, a ferramenta Magnet AXIOM[2] (M. Forensics, 2017).

3.2 WINPMEM

A Winpmem é uma ferramenta linha de comandos que realiza a captura do conteúdo presente na memória de um dado sistema. Relativamente ao formato do ficheiro com os dados adquiridos da memória, este poderá ser, por exemplo, `.elf`, `.aff4` e `.raw`. O executável desta ferramenta encontra-se disponível em [3], não sendo efetuada qualquer instalação desta ferramenta no sistema.

Assim, para executar a Winpmem é necessário, na linha de comandos, mencionar o nome do ficheiro executável, seguindo-se a indicação da opção pretendida, juntamente com o seu respetivo valor. Na Listagem 3 encontra-se o comando a executar para visualizar a informação de utilização da ferramenta Winpmem, com parte do seu resultado.

Para se efetuar a aquisição dos dados presentes na memória do sistema, basta, com a opção `-o`, indicar o nome do ficheiro no qual esses mesmos dados capturados serão armazenados e ainda o caminho correspondente à localização onde esse mesmo ficheiro será guardado. Na Listagem 4 encontra-se a execução do comando para a obtenção do conteúdo da memória do sistema, sendo que este será armazenado no ficheiro `dump1.aff4`, o qual ficará guardado na diretoria '`C:\dumpsWinpmem`' (B. R. Forensics, 2015).

2 https://www.magnetforensics.com/products/magnet-axiom/
3 https://github.com/google/rekall/releases

Listagem 3: Visualização de parte da informação de utilização da ferramenta Winpmem

```
C:\>winpmem-2.1.post4.exe -h

USAGE:

 winpmem-2.1.post4.exe  [-l] [-u] [--write-mode] [--mode <MmMapIoSpace,
                        PhysicalMemory, PTERemapping>] [--driver <Path to
                        driver.>] [--format <map, elf, raw>] [-m] [-p
                        </path/to/pagefile>] ...  [-V] [-d] [-v] [-t] [-i
                        </path/to/file/or/device>] ...  [-e <string>] [-o
                        </path/to/file>] [-c <zlib, snappy, none>] [--]
                        [--version] [-h] </path/to/aff4/volume> ...


Where:

 -l,  --load-driver
   Load the driver and exit

 -u,  --unload-driver
   Unload the driver and exit

 --write-mode
   Enable write mode. You must have the driver compiled with write
   support and be on a system with test signing enabled.

 --mode <MmMapIoSpace, PhysicalMemory, PTERemapping>
   Select the acquisition mode. Default is PTERemapping.

 --driver <Path to driver.>
   Use this driver instead of the included one. This option is rarely
   used.
```

Listagem 4: Aquisição da memória de um sistema, através da ferramenta Winpmem

```
C:\>winpmem-2.1.post4.exe -o C:\dumpsWinpmem\dump1.aff4
Driver Unloaded.
CR3: 0x00001AD002
6 memory ranges:
Start 0x00001000 - Length 0x00057000
Start 0x00059000 - Length 0x00045000
Start 0x00100000 - Length 0x21F3C000
Start 0x2203E000 - Length 0x17C11000
Start 0x3AEFF000 - Length 0x00001000
Start 0x100000000 - Length 0x1BF000000
Creating output AFF4 ZipFile.
Dumping Range 0 (Starts at 1000, length 57000)
Dumping Range 1 (Starts at 59000, length 45000)
Dumping Range 2 (Starts at 100000, length 21f3c000)
Dumping Range 3 (Starts at 2203e000, length 17c11000)
Dumping Range 4 (Starts at 3aeff000, length 1000)
Dumping Range 5 (Starts at 100000000, length 1bf000000)
    Reading 0x8000   0MiB / 8075MiB 0MiB/s
    Reading 0xd60000  13MiB / 8075MiB 52MiB/s
    Reading 0x1c70000  28MiB / 8075MiB 60MiB/s
    Reading 0x2ad0000  42MiB / 8075MiB 57MiB/s
    Reading 0x3708000  55MiB / 8075MiB 48MiB/s
(...)
```

A captura realizada demorou aproximadamente 2 minutos e 25 segundos.

Na ferramenta Winpmem, é ainda possível adquirir o *pagefile* do sistema, utilizando-se, para tal, a opção '`-p`' de modo a especificar o caminho para o *pagefile* em causa.

Ao longo da exploração desta ferramenta foram obtidos ficheiros com os dados da RAM em formatos diferentes, mais precisamente, com os formatos `.elf`, `.raw` e `.aff4`. No Rekall tentou-se proceder à análise desses mesmos ficheiros, contudo a ferramenta não conseguiu analisar nenhum dos ficheiros.

3.3 ACCESSDATA FTK IMAGER

A AccessData FTK Imager é uma ferramenta que permite efetuar a aquisição da memória de um dado sistema. A sua interface gráfica é simples e intuitiva para o utilizador. É de ter em conta que a ferramenta pode ser obtida em [4], sendo de seguida necessário proceder à sua instalação.

Para dar início a aquisição da memória do sistema, basta selecionar a opção '*Capture Memory*', a qual irá apresentar uma janela com aspetos a configurar para a aquisição. Deste modo, é necessário indicar, por exemplo, a diretoria na qual serão guardados os ficheiros extraídos e ainda se é ou não incluído o *pagefile*. Na Figura 18 encontra-se as opções definidas antes de se iniciar a aquisição realizada, e, como

4 http://marketing.accessdata.com/ftkimager3.1.1

se pode verificar, foi também adquirido o *pagefile*. Após se iniciar a aquisição, o estado desta é indicado através de uma barra de progresso (Figura 19).

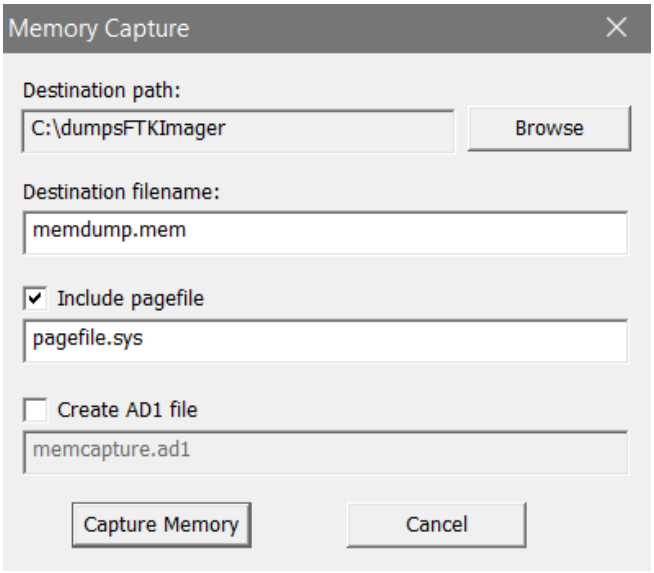


Figura 18: Definição de certas opções antes de iniciar a aquisição da memória com a ferramenta AccessData FTK Imager

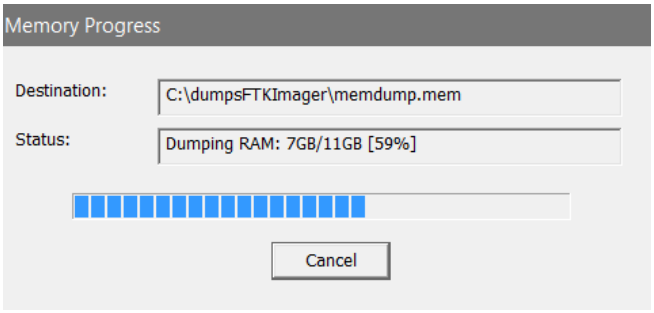


Figura 19: Estado da aquisição da memória na ferramenta AccessData FTK Imager

O processo de aquisição realizado como teste à ferramenta demorou cerca de 1 minuto e 12 segundos.

Com o processo de aquisição terminado, na diretoria especificada são criados dois ficheiros, um com o *dump* realizado à memória e outro correspondente ao ficheiro `pagefile.sys` adquirido (Figura 20).

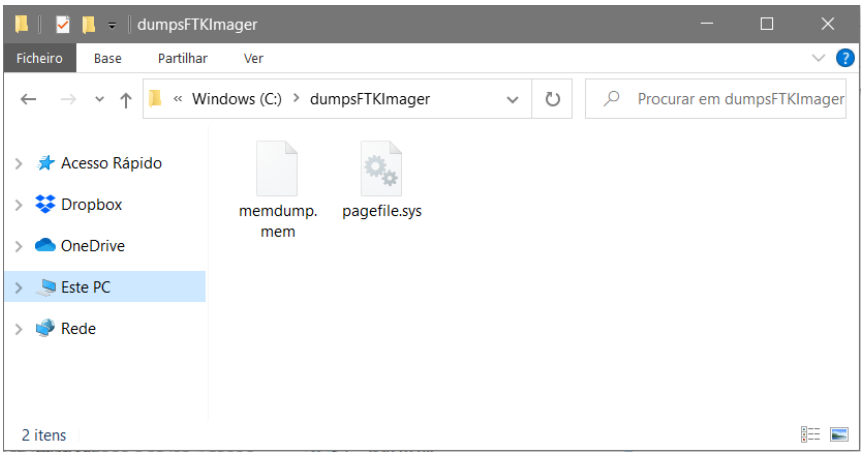


Figura 20: Visualização dos dois ficheiros criados com a aquisição realizada na ferramenta AccessData FTK Imager

Para além do referido, esta ferramenta permite obter os ficheiros `hiberfil.sys` e `swapfile.sys` do sistema no qual a ferramenta está a ser executada. Para tal, na ferramenta, em '*File*', seleciona-se a opção '*Add All Attached Devices*', o que faz com que surja na secção '*Evidence Tree*' as fontes existentes no sistema. Assim, no caso da exploração efetuada, os ficheiros referidos foram encontrados em '`C:\[root]`', tal como se pode constatar na Figura 21 (Chandel, 2015b).

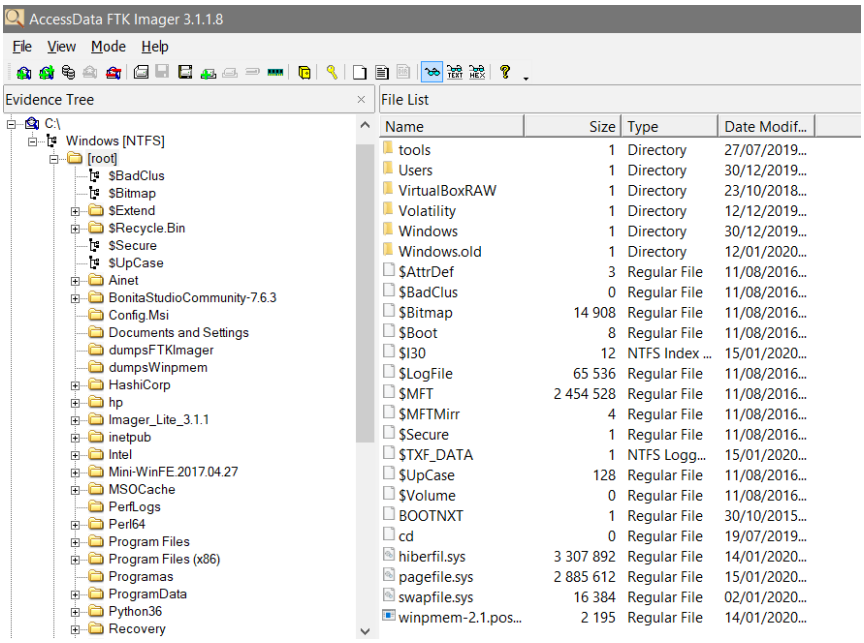


Figura 21: Ficheiros `pagefile.sys`, `hiberfil.sys` e `swapfile.sys` do sistema na ferramenta AccessData FTK Imager

3.4 MAGNET PROCESS CAPTURE

A MAGNET Process Capture realiza a aquisição de processos em execução individuais, ou seja, permite selecionar que processos se pretende obter da memória, permitindo ainda a aquisição de toda a memória da máquina alvo (M. Forensics, 2019b).

Esta ferramenta pode ser obtida em [5], sendo necessário preencher um formulário para obter o ficheiro executável da ferramenta. Este mesmo ficheiro é depois executado no sistema no qual se pretende efetuar a aquisição, não sendo necessário proceder à sua instalação. O utilizador pode fazer uso desta ferramenta a partir da sua interface gráfica ou executando-a a partir da linha de comandos. Nesta dissertação, foram explorados os dois modos de execução possíveis desta ferramenta.

3.4.1 *Interface Gráfica*

A interface gráfica da MAGNET Process Capture é simples e acessível, permitindo que esta seja rapidamente compreendida pelo utilizador (Figura 22). Nesta existe uma lista com os processos em execução naquele momento no sistema, na qual são selecionados os processos de interesse. Esta mesma lista pode ser atualizada através da opção '*Refresh List*'. Antes de iniciar a captura terá ainda que se definir a pasta na qual serão guardados os ficheiros da memória do processo (opção '*Select an Output Folder*'). É ainda possível guardar a memória dos processos selecionados num ficheiro *zip* (com/sem *password*).

Em algumas investigações torna-se interessante visualizar as alterações ocorridas na memória, por exemplo, após um dado conjunto de ações serem executadas. Para tal, esta ferramenta tem a opção '*Monitor Mode*' que, por sua vez, poderá revelar-se bastante útil para o investigador, nomeadamente quando está a analisar *software* malicioso. Esta permite que seja realizada uma captura da memória ligeiramente diferente dado que, a cada número de segundos, é capturada continuamente a memória dos processos selecionados. O número de segundos é um parâmetro cujo valor é configurado pelo utilizador (M. Forensics, 2018).

Esta versão da ferramenta apresenta uma funcionalidade que permite exportar listas de processos. Assim, torna-se possível selecionar um dado conjunto de processos na lista de processos em execução e exportar a lista dos processos selecionados

5 https://www.magnetforensics.com/resources/magnet-process-capture/

para um ficheiro de texto (opção '*Export List*'). Já na opção '*Import Ignore List*', é possível importar um ficheiro de texto com uma lista de processos a ignorar no processo de aquisição. Tal torna-se útil quando, numa aquisição de processos em memória, se pretende ignorar um dado conjunto de processos já conhecidos, de modo a serem capturados unicamente os processos que não constam na lista de processos importada.

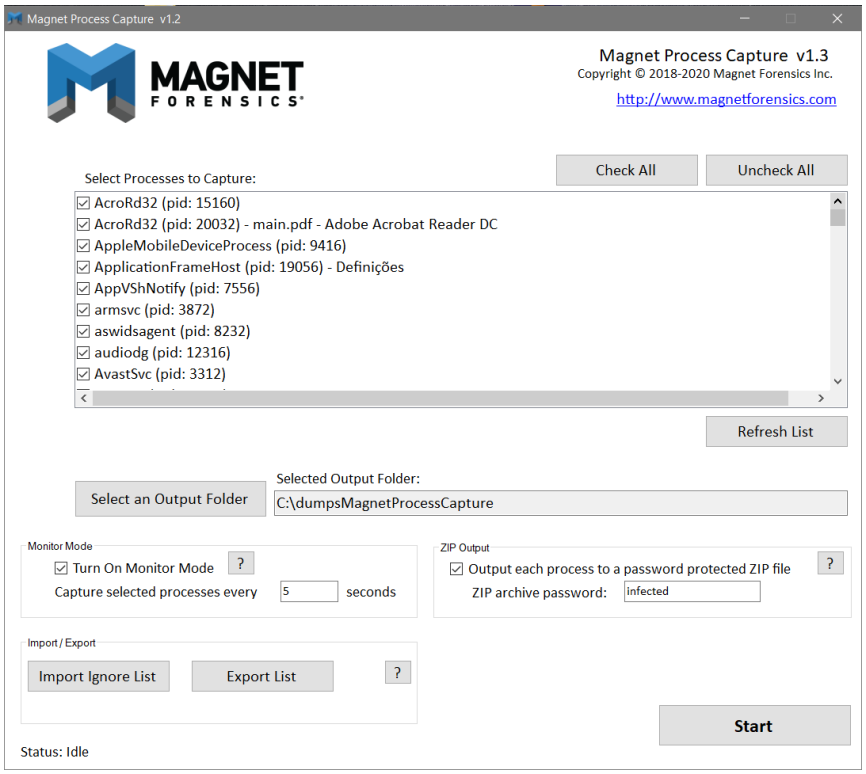


Figura 22: Interface da ferramenta Magnet Process Capture

Nesta dissertação, foram realizadas algumas aquisições de processos de forma a se perceber melhor o funcionamento da ferramenta. Assim, inicialmente, foi realizada a simples captura de alguns processos, por exemplo, um processo '*chrome*' e um processo '*svchost*', tendo-se obtido um ficheiro com a aquisição realizada por cada processo selecionado.

Posteriormente foram ainda realizadas aquisições de processos com recurso à funcionalidade de importar/exportar listas de processos. Assim, foi exportado um ficheiro de texto com todos os processos em execução selecionados, com exceção do processo ' *explorer*'. De seguida, este mesmo ficheiro foi importado, dando-se início à captura (Figura 23). Com este mesmo ficheiro de texto importado, o objetivo passa por adquirir somente a memória referente ao processo '*explorer*'. Com o término da

aquisição, constata-se que foi capturada a memória referente ao processo '*explorer*' e ainda a referente ao processo '*SystemSettings*'.

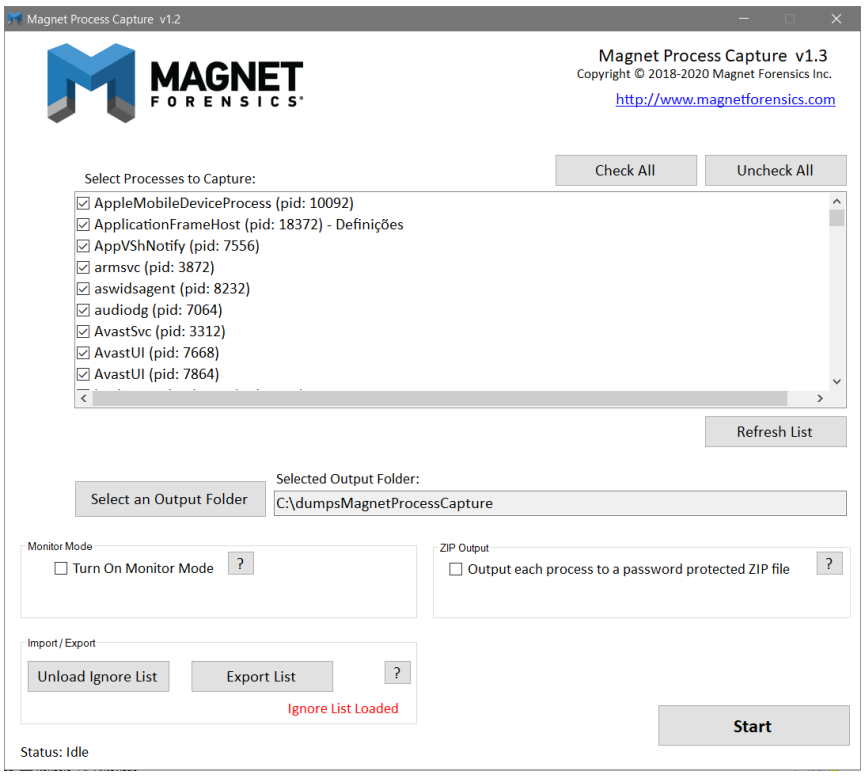


Figura 23: Configuração da ferramenta Magnet Process Capture, com a importação do ficheiro de texto com os processos a ignorar no processo de aquisição

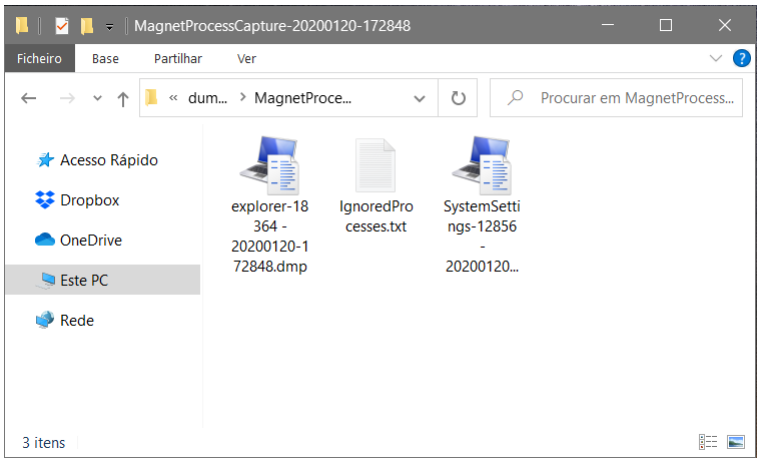


Figura 24: Ficheiros criados com a aquisição do processo '*explorer*', com recurso à funcionalidade '*Import Ignore List*'

No que toca à análise dos ficheiros obtidos nesta ferramenta, esta não pode ser realizada com as ferramentas Rekall e Volatility. Deste modo, o investigador deve utilizar, por exemplo, a ferramenta Magnet AXIOM (M. Forensics, 2018).

3.4.2 *Interface Linha de Comandos*

A MAGNET Process Capture pode ser utilizada a partir da linha de comandos do sistema. Assim, para perceber o funcionamento desta ferramenta ao ser executada deste modo basta, na linha de comandos, executar o seguinte comando:

```
MagnetProcessCapture.exe /help (ou MagnetProcessCapture.exe /?)
```

Este comando apresenta ao utilizador a janela informativa presente na Figura 25 que contém as opções de linha de comandos disponíveis para a MAGNET Process Capture.

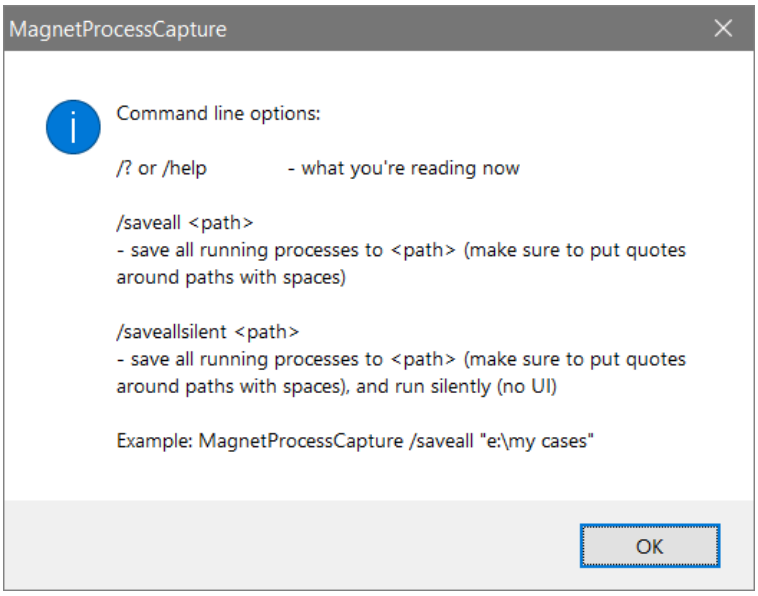


Figura 25: Opções da linha de comandos da ferramenta MAGNET Process Capture

Assim, inicialmente, realizou-se uma captura de todos os processos em execução no sistema, guardando os mesmos na diretoria especificada. Para tal executou-se o comando abaixo:

```
MagnetProcessCapture.exe /saveall C:\dumpsMagnetProcessCapture
```

Logo, após se ter dado início à execução deste comando, surge a interface gráfica da ferramenta, informando no canto inferior esquerdo desta que processo está a ser capturado no momento (Figura 26). Quando a aquisição termina, automaticamente esta janela é fechada.

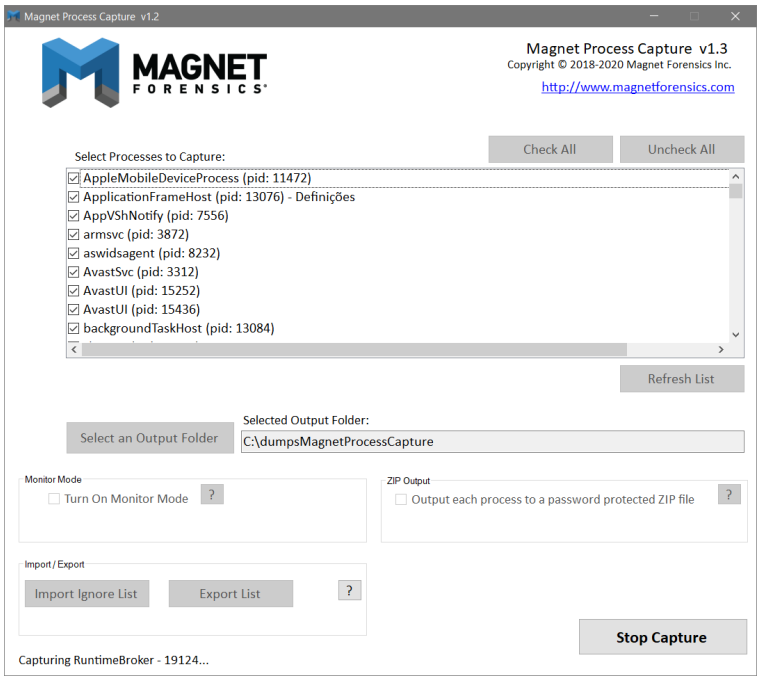


Figura 26: Interface gráfica da MAGNET Process Capture com a informação do processo a ser capturado, após se ter iniciado a aquisição na linha de comandos

No caso de o utilizador pretender realizar a aquisição por linha comandos sem que apareça a interface gráfica da ferramenta, ao invés da opção '`/saveall`', terá que utilizar a opção '`/saveallsilent`'. Contudo, com esta opção o utilizador não sabe que processos estão a ser capturados nem quando a aquisição ao sistema termina.

É de ter em conta que os ficheiros obtidos nas aquisições realizadas a partir da interface de linha de comandos desta ferramenta não podem ser analisados com as ferramentas Rekall e Volatility.

A interface de linha de comandos da MAGNET Process Capture, comparativamente com a sua interface gráfica, não permite armazenar os ficheiros resultantes da aquisição num ficheiro *zip*. Para além disso, a opção '*Monitor Mode*' não se encontra disponível ao utilizador quando este faz uso da interface de linha de comandos. Assim, verifica-se que a interface de linha comandos desta ferramenta pode ser útil quando o objetivo passa por se efetuar uma aquisição simples e rápida de todos os processos na memória do sistema.

3.5 COMPARAÇÃO DAS FERRAMENTAS DE AQUISIÇÃO EXPLORADAS

Com o término da fase de análise e exploração das ferramentas de aquisição da memória selecionadas foi possível realizar uma análise comparativa entre as mesmas.

O facto de existirem ferramentas que permitem obter toda a memória do sistema e outras só alguns dos processos presentes na memória oferece uma certa comodidade ao utilizador. Assim, através da utilização da ferramenta MAGNET Process Capture, se unicamente um dos processos for relevante para o caso em questão, o utilizador não terá que adquirir todos os dados em memória.

No que se refere à interface das ferramentas, a MAGNET RAM Capture e a AcessData FTK Imager apresentam interface gráfica, enquanto que a Winpmem é uma ferramenta com uma interface linha de comandos. Já a MAGNET Process Capture apresenta tanto interface gráfica como interface linha de comandos (ou Command-Line Interface (CLI)). Mesmo assim, e indepentemente do tipo de interface, todas as ferramentas de aquisição analisadas apresentam uma interface simples e objetiva, permitindo uma utilização fácil.

Com as ferramentas de aquisição que capturam toda a memória de um sistema foi possível comparar o tempo que demora até ao processo de aquisição estar concluído, sendo que a MAGNET RAM Capture foi a ferramenta mais rápida.

No que toca às funcionalidades destas ferramentas, existem algumas diferenças. A ferramenta MAGNET RAM Capture permite fragmentar os ficheiros resultantes da aquisição, enquanto que as ferramentas Winpmem e AcessData FTK Imager não apresentam esta funcionalidade. As ferramentas AcessData FTK Imager e Winpmem permitem obter o ficheiro `pagefile.sys`. Já a ferramenta AcessData FTK Imager permite ainda obter os ficheiros `hiberfil.sys` e `swapfile.sys` do sistema no qual a ferramenta está a ser executada. Para a análise de *malware*, a ferramenta MAGNET Process Capture pode-se tornar útil.

Na Tabela 5 é apresentado um sumário da comparação entre as ferramentas de aquisição exploradas, com base em alguns dos aspetos referidos acima.

Tabela 5: Tabela Comparativa entre as Ferramentas de Aquisição

	MAGNET RAM Capture	**Winpmem**	**AcessData FTK Imager**	**MAGNET Process Capture**
Formatos explorados	Interface gráfica	Interface gráfica	Interface gráfica	Interface gráfica e CLI
Interface(s)	Simples e objetiva	Simples e objetiva	Simples e objetiva	Simples e objetiva
Formato do ficheiro criado	`.dmp`, `.raw` ou `.bin`	`.elf`, `.aff4` ou `.raw`	`.mem`	`.dmp`
Aquisição completa da memória	Sim	Sim	Sim	Não
Aquisição de processos específicos	Não	Não	Não	Sim
Fragmentação dos ficheiros da aquisição	Sim	Não	Não	Não
Obtenção do ficheiro `pagefile.sys`	Não	Sim	Sim	Não
Obtenção dos ficheiros `hiberfil.sys` e `swapfile.sys`	Não	Não	Sim	Não
Tempo da aquisição	1'51"	2'25"	1'12"	-

3.6 FERRAMENTAS DE VERIFICAÇÃO DE VALORES DE *hash*

Numa investigação forense a dispositivos digitais, ao ser realizada uma aquisição remota é necessário verificar se os dados não foram alterados durante o processo de envio destes pela rede. Deste modo, o investigador terá que calcular os valores de *hash* dos dados resultantes da aquisição, tanto antes como após o envio destes. Assim sendo, para esse mesmo cálculo, é necessário utilizar ferramentas de verificação de valores de *hash*.

De seguida será descrita a exploração efetuada a ferramentas de verificação de valores de *hash*. Por conseguinte será apresentada a *Windows Powershell* no contexto referido, mencionando-se os passos necessários para tal. Posteriormente é ainda descrita a exploração realizada à ferramenta HashMyFiles da NirSoft.

3.6.1 *Windows Powershell para Verificação de Valores de Hash*

A *Windows PowerShell* possui comandos que permitem calcular os valores de *hash* de ficheiros. Deste modo, para se calcular os valores de *hash*, basta executar o comando seguinte (Phillips, 2019):

```
get-filehash FILEPATH
```

Na Listagem 5 encontra-se um exemplo da execução do comando anteriormente referido; é possível visualizar o valor de *hash* criado de de um ficheiro de um *dump* de memória, sendo empregue o algoritmo SHA256.

Listagem 5: Cálculo do valor de *hash* de uma aquisição de memória na *Windows PowerShell*

```
PS C:\ch2> get-filehash ch2.dmp

Algorithm
---------
SHA256

Hash
----
4EDD6B2BE99075E652AD1B2A70470BB897C27096BCE1B5BD5067BA28885BB8BA

Path
----
C:\ch2\ch2.dmp
```

Por omissão, a *Windows PowerShell* efetua o cálculo do valor de *hash* com o algoritmo SHA256. Contudo, também se pode utilizar os algoritmos SHA1, SHA256, SHA384, SHA512, MACTripleDES, MD5, e ainda o RIPEMD-160. Para tal, basta executar o comando que se segue (Phillips, 2019):

```
get-filehash -Algorithm [HASH TYPE] FILEPATH
```

Assim, na Listagem 6 encontra-se o cálculo do valor de *hash* de um ficheiro de um *dump* de memória, utilizando-se o algoritmo RIPEMD-160.

Listagem 6: Cálculo do valor de *hash* de uma aquisição de memória na *Windows PowerShell*, com o algoritmo RIPEMD-160

```
PS C:\ch2> get-filehash -Algorithm RIPEMD160 ch2.dmp

Algorithm       Hash                                                   Path
---------       ----                                                   ----
RIPEMD160       C750F681CA10D3ADDE210D8D5499E1262C992883               C:\ch2\ch2.dmp
```

3.6.2 *HashMyFiles*

A ferramenta HashMyFiles da NirSoft permite efetuar o cálculo de valores de *hash* de um ou mais ficheiros. Nesta dissertação foi explorada a versão 2.36 desta ferramenta[6]. Assim, o utilizador ao adicionar um determinado ficheiro, é lhe apresentado os valores de *hash* com os algoritmos MD5, SHA1, CRC32, SHA-256, SHA-512 e o SHA-384 (Figura 27).

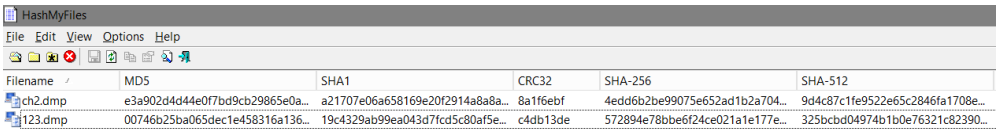


Figura 27: Alguns dos valores de *hash* de dois ficheiros de aquisição da memória na ferramenta HashMyFiles

É ainda de mencionar que esta ferramenta permite que seja adicionado um ficheiro com os dados relativos a um dado processo do sistema no qual a ferramenta está a ser executada (em '*File*', '*Add Process Files*'). Por exemplo, na Figura 28, verifica-se que ao ser adicionado o ficheiro do processo '*chrome.exe*', também foram adicionados os ficheiros Dynamic Link Library (DLL) relativos a esse mesmo processo (itens `chrome_elf.dll` e `chrome_child.dll`).

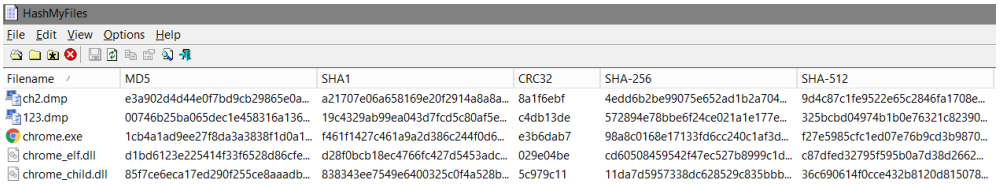


Figura 28: Alguns dos valores de *hash* do ficheiro do processo '*chrome.exe*' e dos ficheiros relacionados com este na ferramenta HashMyFiles

6 https://www.nirsoft.net/utils/hash_my_files.html

Para obter um ou vários dos valores de *hash* calculados para um dado ficheiro, basta que o utilizador no menu '*Edit*' selecione que valor de *hash* calculado por um dado algoritmo pretende copiar. Na ferramenta HashMyFiles é ainda possível obter um relatório em HTML dos itens selecionados ou de todos eles.

Algo relevante ainda a mencionar é o facto de nesta ferramenta ser possível analisar um dado ficheiro selecionado no *website* VirusTotal[7] (em '*File*', '*Open In VirusTotal Web site*'). Este *website* utiliza vários antivírus e *antimalwares* para fornecer um *feedback* acerca de ficheiros que o utilizador considere suspeitos. No VirusTotal é possível efetuar-se o envio do ficheiro ou o envio da *hash* do ficheiro. Na Figura 29 encontra-se parte do relatório fornecido pelo VirusTotal como resultado do envio da imagem de memória 'ch2.dmp'. Como se pode constatar, esta imagem de memória não apresenta conteúdo malicioso. No relatório do VirusTotal presente na Figura 29 encontra-se o valor da *hash* SHA-256 do ficheiro calculada pela ferramenta HashMyFiles e, por sua vez, apresentada, na interface da ferramenta (Figura 30).

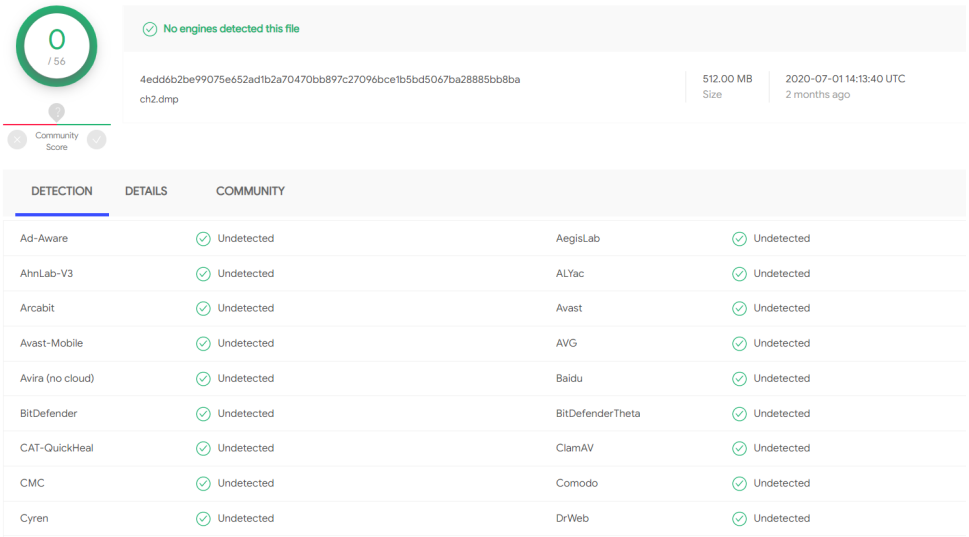


Figura 29: Parte do relatório do VirusTotal acerca de uma imagem de memória

Figura 30: Alguns dos valores de *hash* referentes à imagem de memória em causa

7 https://www.virustotal.com/gui/

3.7 SÍNTESE

Ao longo deste capítulo foram exploradas e analisadas algumas ferramentas de aquisição da memória de um sistema.

De ferramentas de aquisição da memória, foram analisadas duas ferramentas com interface gráfica - a MAGNET RAM Capture e AcessData FTK Imager - e a ferramenta de linha de comandos Winpmem. Das três ferramentas a única que requer instalação é a AcessData FTK Imager. O resultado do processo de aquisição destas ferramentas é um ficheiro com o conteúdo da memória (por exemplo, com extensão `.dmp`). Ao contrário da MAGNET RAM Capture, a Winpmem e a AcessData FTK Imager permitem ainda extrair o *pagefile*.

A MAGNET Process Capture é a ferramenta de aquisição de processos em específico que foi analisada neste capítulo. Esta ferramenta pode ser executada no modo interface gráfica ou no modo linha de comandos. A MAGNET Process Capture é uma ferramenta útil para o investigador quando este só pretende adquirir um dos processos em específico.

Com ferramentas de verificação de valores de *hash* como a HashMyFiles, numa aquisição remota, torna-se possível garantir a segurança do processo e, consequentemente, a dos dados em causa, permitindo uma análise bem sucedida da aquisição realizada.

Com a aquisição da memória concluída com sucesso, o investigador terá que iniciar a fase de análise do conteúdo obtido da memória, de modo a encontrar evidências digitais que sejam relevantes para o caso. Para realizar o procedimento de análise, é necessário recorrer a uma ferramenta de análise da memória.

4

FERRAMENTAS DE ANÁLISE DO CLIPBOARD

Este capítulo descreve as ferramentas de análise e exploração do conteúdo do *clipboard* do sistema operativo *Windows*. Assim, neste capítulo serão analisadas as ferramentas InsideClipboard da Nirsoft e a Free Clipboard Viewer da Comfort Software Group, de modo a explorar alguns dos seus aspetos e funcionalidades.

4.1 O *clipboard* DO WINDOWS

O *clipboard* do *Windows*, quando surgiu, permitia que o utilizador executasse os tradicionais métodos de copiar/colar ou cortar/colar entre as aplicações do sistema.

A versão 18.10 do *Windows 10* acrescentou duas funcionalidades à área de transferência desse sistema operativo, nomeadamente, o histórico de *clipboard* e a partilha de *clipboard* entre dispositivos *Windows 10*. O histórico do *clipboard* disponibiliza uma lista com os conteúdos que foram copiados/colados através da área de transferência pelo utilizador. A lista pode ser visualizada através da tecla `Windows+V` (Figura 31).

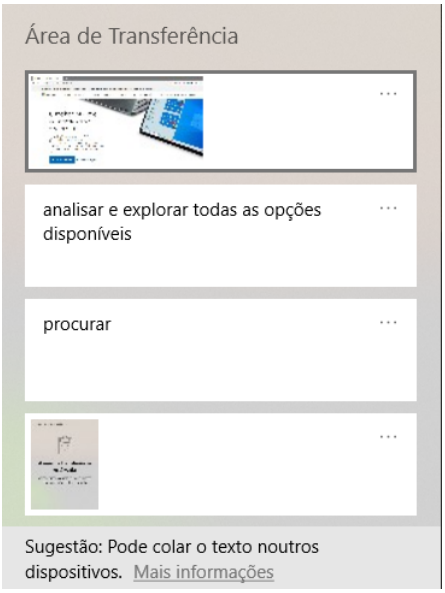


Figura 31: Lista do *clipboard* com os elementos copiados/cortados

A partilha de *clipboard* permite que o conteúdo copiado para a área de transferência de um determinado dispositivo seja acedido em outros dispositivos que se encontrem sincronizados.

A nível forense o conteúdo do *clipboard* é relevante pois neste poderão ser encontrados URLs suspeitos e até mesmo palavras-passe em *plain-text* ou ainda endereços de carteiras de criptomoedas, dado que é frequente o uso do *clipboard* para evitar copiar as longas *strings* dos endereços, como por exemplo, o endereço Bitcoin `1TghdBoXPPBkRGW7WW1qmryi1KjyCCcUT`.

4.2 INSIDECLIPBOARD

A InsideClipboard é uma ferramenta desenvolvida pela Nirsoft que apresenta o conteúdo binário de todos os formatos de um elemento que, até ao momento, se encontram no *clipboard*. Ou seja, apresenta o último elemento que tenha sido copiado em vários formatos. Esta ferramenta permite ainda guardar o conteúdo de um formato em específico num arquivo binário (NirSoft, 2020).

A ferramenta InsideClipboard encontra-se disponível em [1], num ficheiro executável, logo, não é necessário proceder à instalação da ferramenta.

Ao analisar-se a interface da ferramenta constata-se que esta é objetiva e de fácil compreensão para o utilizador.

Na Figura 32 encontra-se a janela da ferramenta InsideClipboard, após a cópia da palavra ‘qualquer‘ a partir do *browser* Google Chrome, verificando-se a existência de quatro formatos diferentes para esse mesmo elemento. Já na Figura 33 encontra-se o resultado obtido após a cópia de uma imagem, não sendo possível visualizar a imagem copiada. Assim, verifica-se que os formatos apresentados pela ferramenta variam de acordo com o tipo do elemento copiado.

1 https://www.nirsoft.net/utils/inside_clipboard.html

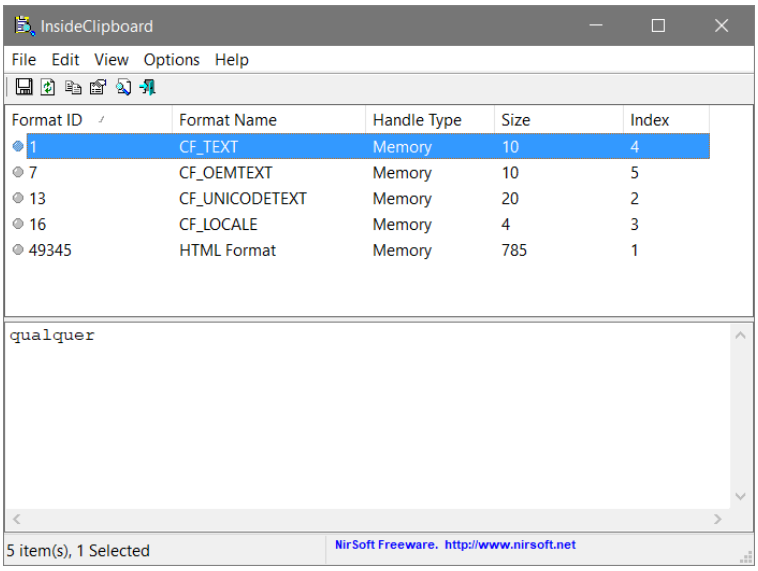


Figura 32: Janela da ferramenta InsideClipboard após a cópia de uma palavra

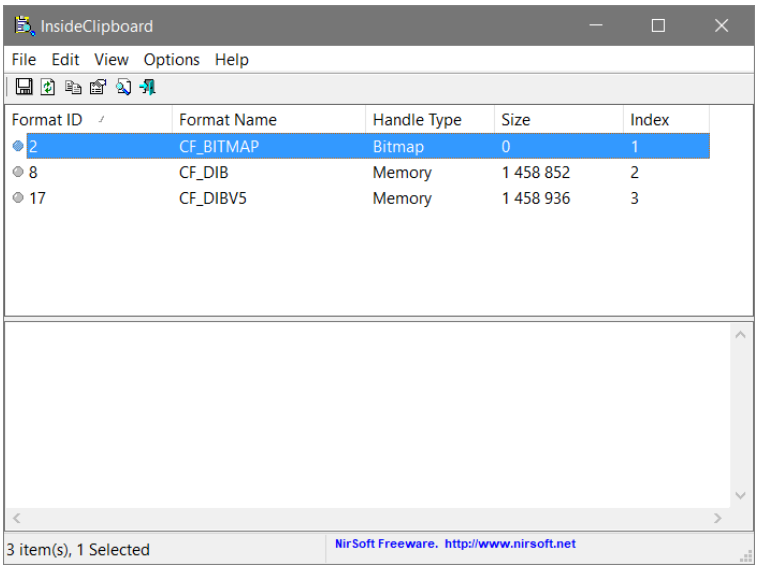


Figura 33: Formatos apresentados após a cópia de uma imagem

Com a utilização da ferramenta, verificou-se que a InsideClipboard só funciona num sistema *live*. Constatou-se ainda que esta ferramenta, a cada elemento novo que seja copiado e inserido no *clipboard* do sistema, elimina o elemento anterior. Logo, a InsideClipboard não armazena um histórico dos formatos dos elementos anteriormente copiados, o que poderá não ser prático para o investigador. Assim, de modo a armazenar os dados recolhidos, poderá ser gerado um relatório HTML ou então guardar os formatos obtidos num ficheiro `.Cpl` (*control panel item*) ou num *image file*.

Foram ainda realizados alguns testes de exploração da ferramenta e verificou-se que os formatos apresentados podem também variar de acordo com a ferramenta/programa de onde o elemento copiado provém. Assim, copiou-se a mesma palavra (em *plain-text*) - nomeadamente 'comando' - da ferramenta Sublime Text (versão 3)[2] e da ferramenta Adobe Acrobat Reader DC[3]. Tal como se pode ver nas Figuras 34 e 35, os formatos obtidos foram distintos.

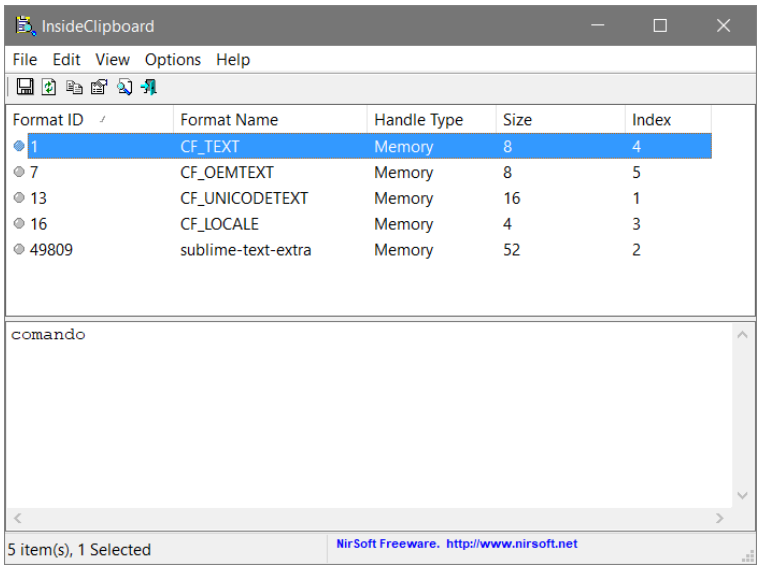


Figura 34: Formatos apresentados após a cópia de texto da ferramenta Sublime Text

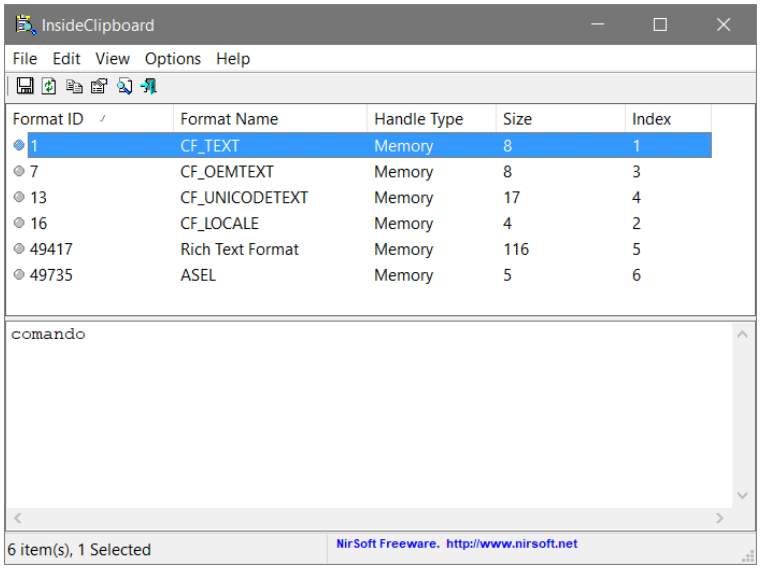


Figura 35: Formatos apresentados após a cópia de texto da ferramenta Adobe Acrobat Reader DC

2 https://www.sublimetext.com/3
3 https://www.adobe.com/pt/

4.3 FREE CLIPBOARD VIEWER

A ferramenta Free Clipboard Viewer da Comfort Software Group permite visualizar o último elemento copiado em vários formatos. Tal como a ferramenta InsideClipboard não permite aceder aos formatos dos elementos anteriormente copiados, a não ser que se armazene os dados obtidos num ficheiro. Nesta ferramenta não é possível gerar relatórios HTML. A Free Clipboard Viewer só funciona num sistema *live* e apresenta uma interface simples e intuitiva ao utilizador.

A ferramenta Free Clipboard Viewer pode ser utilizada no formato executável ou então o investigador poderá instalá-la no sistema. Nesta dissertação foi alvo de exploração a versão do ficheiro executável, que pode ser obtido em [4], não sendo necessário proceder à instalação da ferramenta.

Na Figura 36 encontra-se a interface da ferramenta após a cópia de uma imagem da ferramenta Adobe Acrobat Reader DC. Como se pode verificar, é possível visualizar a imagem em causa, sendo que no lado esquerdo da interface são apresentados os formatos para este elemento.

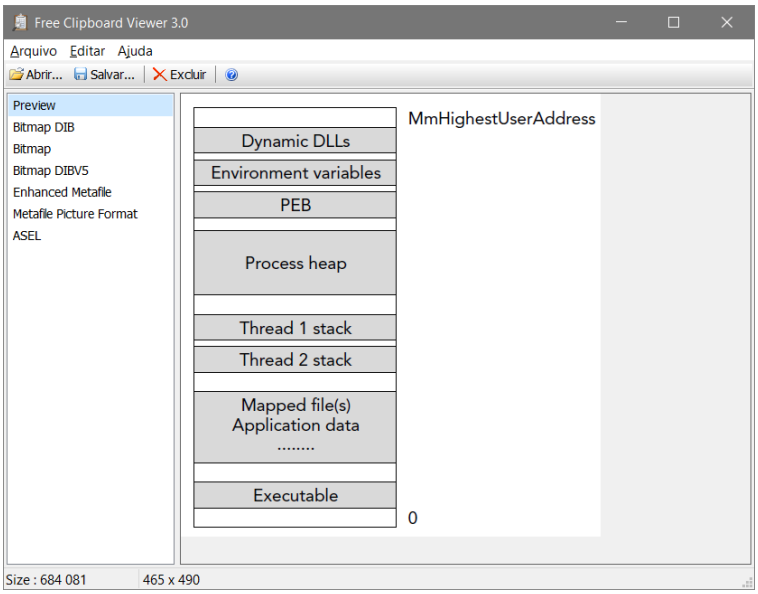


Figura 36: Visualização dos formatos apresentados após a cópia de uma imagem a partir da ferramenta Adobe Acrobat Reader DC

Já na Figura 37 encontra-se os formatos apresentados após a cópia da palavra 'usualmente' (em *plain-text*) da ferramenta Sublime Text.

4 https://www.freeclipboardviewer.com/

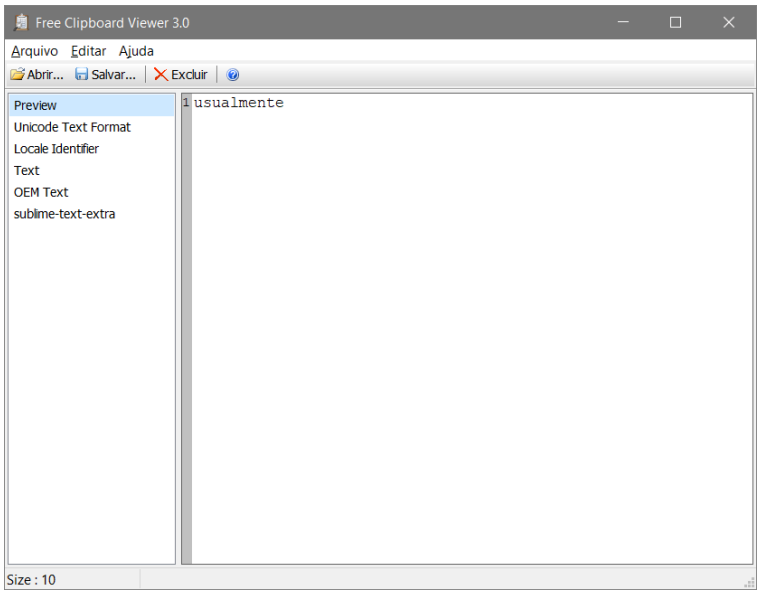


Figura 37: Visualização dos formatos apresentados após a cópia da palavra 'usualmente' a partir da ferramenta Sublime Text

Como seria de esperar, nesta ferramenta os formatos apresentados por esta variam de acordo com o tipo do elemento copiado. Nesta ferramenta também foi realizado o teste de cópia do mesmo *plain-text* de ferramentas/programas distintos. Tal como na ferramenta InsideClipboard, verificou-se que copiando o mesmo *plain-text* de ferramentas/programas distintos podem surgir formatos diferentes, variando de acordo com a ferramenta/programa de onde o elemento copiado provém.

4.4 SÍNTESE

A observação do conteúdo presente no *clipboard* poderá ser, em alguns casos, um auxílio na análise ao sistema, podendo ser uma fase benéfica de o investigador realizar.

Neste capítulo foram exploradas ferramentas que permitem a análise *live* do conteúdo do *clipboard* do *Windows* que se encontra em memória. Tanto a InsideClipboard como a Free Clipboard Viewer só permitem visualizar o último elemento que foi copiado, ou seja, o elemento mais recentemente adicionado ao histórico do *clipboard*. Em ambas as ferramentas os elementos copiados podem ser visualizados em diferentes formatos.

Uma das limitações observadas nestas duas ferramentas são o facto de estas não permitirem visualizar os elementos anteriormente copiados.

5

FERRAMENTAS DE ANÁLISE DA MEMÓRIA

Com a aquisição da memória do sistema concluída é necessário analisar os dados resultantes, de modo a verificar se de facto a memória do sistema apresenta alguma informação relevante. Assim, será necessário recorrer a ferramentas que permitam analisar aquisições de memória. Nesta dissertação foram exploradas algumas ferramentas existentes de modo a se perceber as suas funcionalidades bem como alguns aspetos das mesmas.

Deste modo, neste capítulo serão apresentadas as explorações e análises realizadas às ferramentas Volatility - versões 2.6 e 3.0 *beta* - e Rekall. Segue-se uma comparação entre estas, acompanhada por uma tabela comparativa com alguns dos aspetos mais relevantes destas *frameworks*. Posteriormente é ainda apresentada a exploração ao módulo do Volatility para o Autopsy.

Todas estas ferramentas de análise do conteúdo da memória exploradas nesta dissertação foram instaladas numa máquina que apresenta o sistema operativo *Microsoft Windows 10 Home* versão 10.0.17763 Compilação 17763, com 8 GiB de RAM e com um processador Intel(R) Core(TM) i7-7500U CPU @ 2.70GHz. Foram ainda utilizadas para explorar as ferramentas de análise a imagem de memória 'ch2.dmp' que se obtém no site Root Me, no *challenge* '*Command & Control -level 2*'[1] (opção '*Start the challenge*') e a imagem de memória 'memdump.mem'[2] que está relacionada com o concurso *Capture The Flag* organizado pela 13cubed[3].

5.1 VOLATILITY 2.6

O Volatility é uma ferramenta que permite efetuar análises a imagens de memória. Esta apresenta um conjunto de perfis a serem aplicados às imagens de memória em análise e ainda *plugins* para o utilizador obter os dados presentes na memória dos sistema no momento da aquisição.

1 https://www.root-me.org/en/Challenges/Forensic/Command-Control-level-2
2 https://drive.google.com/drive/folders/1E-i2RTUBXBGUd_Xz0k67kFOpHcr6WX8J
3 https://www.13cubed.com/

Para efetuar esta exploração, não foi necessário efetuar qualquer instalação. Ao invés disso, primeiramente, foi preciso obter o ficheiro executável da ferramenta, sendo que o mesmo encontra-se disponível no *website* do Volatility[4].

O modo de execução desta ferramenta é simples e direto e passa por, na linha de comandos, primeiro, referir o nome do ficheiro executável, seguindo-se o nome do ficheiro do *dump* em análise, o perfil adequado ao caso e, por fim, a opção/*plugin* que permitirá obter a informação pretendida. O comando-base para a execução desta ferramenta é o seguinte:

```
volatility.exe -f nomeFichDump --profile=perfilAdequado plugin
```

No comando-base acima 'nomeFichDump' corresponde à designação do ficheiro correspondente à imagem de memória em análise (opção '`-f`') e 'perfilAdequado' ao perfil a aplicar à imagem de memória em causa (opção '`--profile`'). Já 'plugin' refere-se ao *plugin* que se pretende executar sobre a imagem de memória.

Um *plugin* que inicialmente poderá ser um suporte é o `imageinfo` que fornece um resumo de informações sobre a imagem de memória, tais como, o número de processadores e o perfil a aplicar. Refere ainda a hora/data em que a amostra foi recolhida (parâmetro '*Image local date and time*') (Ligh Hale et al., 2014; Volatility Foundation, 2019b).

No Volatility, uma das primeiras informações que se pretende analisar corresponde aos processos existentes no sistema no momento em que foi realizada a aquisição, existindo, para tal, alguns *plugins* que fornecem um conjunto de informação acerca destes. O *plugin* `pslist` permite obter a lista dos processos em execução e indica, por exemplo, o PID (*Process ID*), o PPID (*Parent Process ID*) e a data/hora de quando o processo foi iniciado e terminado. Através deste *plugin* pode ser importante reparar, por exemplo, se algum processo tem um processo pai que não está na lista, ou seja, se o PPID de um dado processo não se encontra na lista. Nesse caso, tanto o processo cujo PPID não consta na lista, bem como os processos gerados por este são considerados suspeitos. É ainda de referir que se deve analisar se a funcionalidade dos processos gerados é ou não necessária ao funcionamento do seu processo pai. Assim sendo, torna-se relevante investigar o processo que tem o PPID que não está na lista (Scott, 2017). O *plugin* `pslist` deteta tanto os processos críticos do sistema - como, por exemplo, '*system*', '*csrss.exe*' e '*winlogon.exe*' - bem como os processos não críticos, tais como, antivírus e ferramentas de *password-cracking*. Esta ferramenta permite visualizar a relação entre processos pai e filhos em árvore (*plugin*

4 https://www.volatilityfoundation.org/26

Listagem 7: Visão parcial da saída do *plugin* `pstree` no Volatility

```
C:\VolatilityFirst\volatility_2.6_win64_standalone>volatility_2.6_win64_standalone.exe -f ch2.dmp --profile=Win7SP1x86_23418 pstree
Volatility Foundation Volatility Framework 2.6
Name                                                  Pid   PPid   Thds   Hnds   Time
-------------------------------------------------- ------ ------ ------ ------ ----
 0x892ac2b8:wininit.exe                               456    396      3     77 2013-01-12 16:38:14 UTC+0000
. 0x896294c0:services.exe                             560    456      6    205 2013-01-12 16:38:16 UTC+0000
.. 0x89805420:svchost.exe                             832    560     19    435 2013-01-12 16:38:23 UTC+0000
... 0x87c90d40:audiodg.exe                           1720    832      5    117 2013-01-12 16:58:11 UTC+0000
.. 0x89852918:svchost.exe                             904    560     17    409 2013-01-12 16:38:24 UTC+0000
... 0x87ad44d0:dwm.exe                               2496    904      5     77 2013-01-12 16:40:25 UTC+0000
.. 0x898b2790:svchost.exe                            1172    560     15    475 2013-01-12 16:38:27 UTC+0000
.. 0x89f3d2c0:svchost.exe                            3352    560      9    141 2013-01-12 16:40:58 UTC+0000
.. 0x898fbb18:SearchIndexer.                         2900    560     13    636 2013-01-12 16:40:38 UTC+0000
.. 0x8986b030:svchost.exe                             928    560     26    869 2013-01-12 16:38:24 UTC+0000
.. 0x8a1d84e0:vmtoolsd.exe                           1968    560      6    220 2013-01-12 16:39:14 UTC+0000
.. 0x8962f030:svchost.exe                             692    560     10    353 2013-01-12 16:38:21 UTC+0000
.. 0x898911a8:svchost.exe                            1084    560     10    257 2013-01-12 16:38:26 UTC+0000
.. 0x898a7868:AvastSvc.exe                           1220    560     66   1180 2013-01-12 16:38:28 UTC+0000
.. 0x89f1d3e8:svchost.exe                            3624    560     14    348 2013-01-12 16:41:22 UTC+0000
.. 0x9542a030:TPAutoConnSvc.                         1612    560      9    135 2013-01-12 16:39:23 UTC+0000
... 0x87ae2880:TPAutoConnect.                        2568   1612      5    146 2013-01-12 16:40:28 UTC+0000
.. 0x88cded40:sppsvc.exe                             1872    560      4    143 2013-01-12 16:39:02 UTC+0000
```

`pstree`), algo que, em certos casos, auxilia na deteção de alguns aspetos que se podem revelar suspeitos como, por exemplo, a existência de processos com o mesmo nome mas que não têm o mesmo processo pai. Na Listagem 7 encontra-se parte do *output* da execução do *plugin* `pstree` sobre uma imagem de memória. Existe ainda a possibilidade de, através do *plugin* `psscan`, se detetar processos escondidos (por exemplo, um *rootkit*), bem como, processos terminados.

O `psxview` é um *plugin* que utiliza vários métodos para detetar processos do sistema em que foi executada a aquisição, permitindo uma pesquisa de processos mais acurada; no seu *output*, *True* mostra que o método detetou o processo e *False* em caso contrário (Figura 38) (Defense, 2017; Volatility Foundation, 2019b).

```
C:\Volatility\volatility_2.6_win64_standalone>volatility_2.6_win64_standalone.exe -f ch2.dmp --profile=Win7SP1x86_23418
 psxview
Volatility Foundation Volatility Framework 2.6
Offset(P)  Name                    PID pslist psscan thrdproc pspcid csrss session deskthrd ExitTime
---------- -------------------- ------ ------ ------ -------- ------ ----- ------- -------- --------
0x1fce2880 TPAutoConnect.         2568 True   False  True     True   True  True    True
0x1eaded40 sppsvc.exe             1872 True   False  True     True   True  True    True
0x1fc9c288 conhost.exe            2600 True   False  True     True   True  True    True
0x00e506b0 conhost.exe            2168 True   False  True     True   True  True    True
0x1deb2790 svchost.exe            1172 True   False  True     True   True  True    True
0x1de52918 svchost.exe             904 True   False  True     True   True  True    False
0x1defbb18 SearchIndexer.         2900 True   False  True     True   True  True    False
0x1d93d2c0 svchost.exe            3352 True   False  True     True   True  True    True
0x1e1b5c20 svchost.exe             764 True   False  True     True   True  True    False
0x1d7f5030 VMUpgradeHelpe          448 True   False  True     True   True  True    True
0x1de98030 cmd.exe                1616 True   False  True     True   True  True    True
0x1de6b030 svchost.exe             928 True   False  True     True   True  True    True
0x1de911a8 svchost.exe            1084 True   False  True     True   True  True    False
0x1e4ac2b8 wininit.exe             456 True   False  True     True   True  True    False
0x1fb4d338 iexplore.exe           3044 True   False  True     True   True  True    True
0x1e02f030 svchost.exe             692 True   False  True     True   True  True    False
0x0c5fbc18 taskmgr.exe            1232 True   False  True     True   True  True    True
0x1fd82438 VMwareTray.exe         2660 True   False  True     True   True  True    True
```

Figura 38: Parte do *output* do *plugin* `psxview` no Volatility

Através da ferramenta Volatility, é possível obter a informação recolhida acerca da *Registry*, mais precisamente, os dados presentes na *Registry* do sistema no momento da aquisição. Pode-se obter as *hives* do sistema, o seu caminho no disco e endereços físicos/lógicos (*plugin* `hivelist`) e ainda as subchaves, valores, dados e tipos de dados de uma dada *key* (*plugin* `printkey` com opção `-K`) ou então de uma dada *hive* (*plugin* `printkey` com opção `-o`), sendo apresentado em *output* a informação relativa à *key* indicada (Volatility Foundation, 2019b).

Outra informação que é possível obter são as variáveis de ambiente presentes numa aquisição e isto através do *plugin* `envars`. Já através do *plugin* `userassist` pode-se obter as chaves do *UserAssist*, que, por sua vez, podem conter informação sobre a atividade do utilizador na máquina, como, por exemplo, quantas vezes uma aplicação foi executada (B. Singh e U. Singh, 2017; Volatility Foundation, 2019b).

A deteção de *malware* na amostra de RAM em análise é algo que esta ferramenta também consegue realizar. Um *plugin* que auxilia nessa deteção é o `malfind` que pesquisa as DLLs/código injetado ou escondido na memória. Normalmente, o *output* deste *plugin* apresenta para cada região de memória suspeita alguns atributos, por exemplo, o processo em causa e o endereço da região em causa. Neste *output* surgem também dados em *hexdump* e em *disassembly*, informação esta que ao ser analisada permitirá descobrir se a região em causa é ou não maliciosa. Contudo, é de referir que ao se analisar a saída produzida pelo *plugin*, terá que se ter em atenção que podem surgir regiões de memória que foram alocadas por progamas de forma legítima, mas que este *plugin* considera suspeitas de igual modo, resultando, portanto, em falsos positivos (Ligh Hale et al., 2014; Volatility Foundation, 2017; Volatility Foundation, 2019b).

O Volatility lista os serviços *Windows* que estão registados na imagem de memória (*plugin* `svcscan`) sendo possível verificar o estado dos serviços e ainda o PID (*Process ID*) do processo associado a cada serviço. Esta ferramenta permite também obter

os comandos executados na linha de comandos localmente ou remotamente por *backdoors* (*plugin* `consoles`), obter as ligações ativas e terminadas no momento da aquisição (*plugin* `connscan`) bem como que privilégios estão presentes, ativados e/ou ativados por omissão em cada processo (*plugin* `privs`). Para verificar processos que tenham escalonamento de privilégios malicioso pode-se visualizar o Security Identifier (SID) e utilizador associados a cada processo, através do *plugin* `getsids` (Balapure, 2018; Ligh Hale et al., 2014; Volatility Foundation, 2017; Volatility Foundation, 2019b).

Outra funcionalidade interessante que o Volatility apresenta é o *plugin* `dlldump` que permite realizar, por exemplo, o *dump* de DLLs de todos os processos em memória e o *dump* de DLLs de um dado processo que esteja em memória (com opção '`--pid=PID`', em que PID é o ID do processo em questão). No Volatility é ainda possível realizar outros tipos de *dump*, tal como, a extração do executável de um processo (*plugin* `procdump`) (Volatility Foundation, 2019b).

Na Tabela 6 encontra-se um resumo dos *plugins* do Volatility descritos no texto, bem como, uma breve descrição para cada um deles.

Tabela 6: *Plugins* do Volatility apresentados no texto com a sua descrição

Plugin	**Descrição**
connscan	Ligações ativas e terminadas no momento da aquisição.
consoles	Comandos executados na linha de comandos (localmente ou remotamente).
dlldump	Extração de DLLs de um dado processo em memória.
envars	Obtenção das variáveis de ambiente dos processos em memória.
getsids	Visualização do SID e utilizador associados a cada processo.
hivelist	Informação acerca da *Registry*, como, por exemplo, as *hives* do sistema e o seu caminho no disco.
imageinfo	Resumo da imagem de memória em análise, permitindo, por exemplo, obter o perfil a aplicar.
kdbgscan	Identifica corretamente o perfil e o endereço KDBG correto.
malfind	Deteta DLLs/código injetado ou escondido que esteja em memória.
printkey	Obtenção de uma *key* da *Registry*, as suas subchaves e valores.
privs	Obtenção dos privilégios de cada processo em memória.
procdump	Extração do executável de um dado processo.
pslist	Lista os processos do sistema em execução.
psscan	Lista os processos escondidos e os processos terminados.
pstree	Lista os processos em árvore, visualizando-se a relação pai-filho entre eles.
psxview	Pesquisa de processos mais aperfeiçoada, utilizando vários métodos para detetar processos do sistema.
svcscan	Visualização dos serviços registados na memória.
userassist	Obtenção das chaves do *UserAssist* presentes em memória.

5.2 VOLATILITY 3.0 PUBLIC BETA

A Volatility 3.0 *Public Beta* corresponde à versão mais recente da *framework* Volatility. A versão 3.0 visa optimizar o desempenho, reduzindo o tempo de execução dos *plugins* (Mike Auty, 2019). Assim, nesta dissertação, foi explorada esta versão da ferramenta de modo a se analisar o seu funcionamento, sendo que, para tal, foi obtido o arquivo *zip* com o código desta ferramenta, na página do GitHub do Volatility 3.0[5].

A utilização da versão 3.0 desta ferramenta é simples e similar à versão anterior. O comando-base a executar na linha de comandos, de modo a utilizar-se esta versão da ferramenta é o seguinte:

```
vol.py -f nomeFichDump plugin
```

No comando-base acima, 'nomeFichDump' corresponde à designação da imagem de memória (opção '`-f`') e 'plugin' ao *plugin* que se pretende executar. É de referir que verifica-se no comando-base a ausência da opção '`--profile`' dado o facto de não ser necessário indicar o perfil no comando para executar o *plugin* em causa.

Tal como na versão anterior, para o utilizador obter a informação adquirida da memória do sistema que foi alvo da aquisição, o utilizador tem que executar *plugins*. Na Figura 39 encontra-se parte do *output* resultante da execução do *plugin* `pslist` sobre uma imagem de memória, utilizando-se para tal o Volatility 3.

```
C:\Volatility\volatility3-master\volatility3-master>vol.py -f C:\Users\santo\Desktop\ch2.dmp windows.pslist
Volatility 3 Framework 1.0.0-beta.1
Progress:  87.11                Scanning primary2 using PdbSignatureScanner
PID     PPID    ImageFileName   Offset(V)       Threads Handles SessionId       Wow64   CreateTime      ExitTime

4       0       System  0x87978b78      103     3257    N/A     False   2013-01-12 16:38:09.000000      N/A
308     4       smss.exe        0x88c3ed40      2       29      N/A     False   2013-01-12 16:38:14.000000      N/A
404     396     csrss.exe       0x8929fd40      9       469     0       False   2013-01-12 16:38:14.000000      N/A
456     396     wininit.exe     0x892ac2b8      3       77      0       False   2013-01-12 16:38:14.000000      N/A
468     448     csrss.exe       0x88d03a00      10      471     1       False   2013-01-12 16:38:14.000000      N/A
500     448     winlogon.exe    0x892ced40      3       111     1       False   2013-01-12 16:38:14.000000      N/A
560     456     services.exe    0x896294c0      6       205     0       False   2013-01-12 16:38:16.000000      N/A
576     456     lsass.exe       0x896427b8      6       566     0       False   2013-01-12 16:38:16.000000      N/A
```

Figura 39: Parte do *output* do *plugin* `pslist` no Volatility 3.0

Como se pode constatar na Figura 39, de facto não é necessário especificar o perfil correspondente à imagem de memória em análise. Tal facilita o trabalho do investigador pois assim o utilizador, antes de iniciar a análise, não necessita encontrar o perfil, ou seja, não é obrigado a executar o *plugin* `imageinfo`, podendo de imediato executar o *plugin* pretendido.

Algo a referir é o facto da nomenclatura dos *plugins* variar consoante o sistema operativo do sistema ao qual o ficheiro da aquisição diz respeito, sendo necessá-

5 https://github.com/volatilityfoundation/volatility3

rio indicar o sistema operativo em causa juntamente à designação do *plugin* a executar. Ou seja, a forma base de se referir um *plugin* num comando é 'sistemaOperativo.designacaoPlugin'. Por exemplo, para executar o *plugin* `pslist` sobre uma imagem de memória de um sistema *Windows*, para indicar o *plugin* coloca-se no comando `windows.pslist`, enquanto se for de um sistema *Linux* coloca-se `linux.pslist` e se for *MacOS* coloca-se `mac.pslist`. Na Tabela 7 encontra-se parte da saída do comando '`vol.py -h`', onde é apresentada a nomenclatura dos *plugins* consoante o sistema operativo.

Tabela 7: Designação de alguns dos *plugins* apresentados pelo comando '`vol.py -h`' no Volatility 3.0, com a sua respetiva descrição

Plugins	**Descrição**
linux.pslist.PsList	Lista os processos presentes numa determinada imagem *Linux*.
linux.pstree.PsTree	*Plugin* para listar os processos em árvore com base no ID do processo pai.
mac.bash.Bash	Histórico do comando *bash* da memória.
mac.check_syscall.Check_syscall	Verifica a tabela de chamadas do sistema em busca de *hooks*.
mac.check_sysctl.Check_sysctl	Verifica os *sysctl handlers* quanto a *hooks*.
mac.check_trap_table.Check_trap_table	Verifica a tabela *mach trap* quanto a *hooks*.
mac.ifconfig.Ifconfig	Lista os módulos do *kernel* carregados.
mac.lsmod.Lsmod	Lista os módulos do *kernel* carregados.
mac.lsof.lsof	Lista todos os descritores de ficheiros abertos para todos os processos.
mac.malfind.Malfind	Lista intervalos de memória de processo que potencialmente contêm código injetado.
mac.netstat.Netstat	Lista todas as ligações de rede para todos os processos.
mac.proc_maps.Maps	Lista intervalos de memória de processo que potencialmente contêm código injetado.
mac.psaux.Psaux	Recupera argumentos da linha de comando do programa.
mac.pslist.PsList	Lista os processos presentes numa determinada imagem de memória *Mac*.
mac.pstree.PsTree	*Plugin* para listar processos em uma árvore com base no ID do processo pai.
mac.tasks.Tasks	Lista os processos presentes numa determinada imagem de memória mac.
mac.trustedbsd.trustedbsd	Verifica se há módulos 'trustedbsd' maliciosos.
timeliner.Timeliner	Executa todos os *plugins* relevantes que fornecem informações relacionadas ao tempo e ordenam os resultados por tempo.
windows.cmdline.CmdLine	Lista os argumentos da linha de comando do processo.
windows.dlldump.DllDump	Realiza *dumps* de intervalos de memória de processo como DLLs.

Na Figura 40 encontra-se parte da saída resultante da execução do *plugin* `hivelist` sobre uma imagem de memória.

```
C:\Volatility\volatility3-master\volatility3-master>vol.py -f C:\Users\santo\Desktop\ch2.dmp windows.registry.hivelist
Volatility 3 Framework 1.0.0-beta.1
Progress:  87.11                Scanning primary2 using PdbSignatureScanner
Offset  FileFullPath

0x8b20c008
0x8b21c008      \REGISTRY\MACHINE\SYSTEM
0x8b23c008      \REGISTRY\MACHINE\HARDWARE
0x8ee66008      \Device\HarddiskVolume1\Boot\BCD
0x8ee66740      \SystemRoot\System32\Config\SOFTWARE
0x90cab9d0      \SystemRoot\System32\Config\DEFAULT
0x9670e9d0      \??\C:\Users\John Doe\ntuser.dat
0x9670f9d0      \??\C:\Users\John Doe\AppData\Local\Microsoft\Windows\UsrClass.dat
0x9aad6148      \SystemRoot\System32\Config\SAM
0x9ab25008      \SystemRoot\System32\Config\SECURITY
0x9aba79d0      \??\C:\Windows\ServiceProfiles\LocalService\NTUSER.DAT
0x9abb1720      \??\C:\Windows\ServiceProfiles\NetworkService\NTUSER.DAT
```

Figura 40: Informação resultante da execução do *plugin* `hivelist` com a ferramenta Volatility 3.0

Ao serem executados alguns *plugins* nesta ferramenta verifica-se que ao nível da saída destes não existe diferença desta versão mais recente para a 2.6.

É de ter em conta que até à data em que esta ferramenta foi explorada nesta dissertação, nem todos os *plugins* presentes na versão anterior desta ferramenta se encontram disponíveis nesta versão mais recente, tais como, o `psxview` e o `consoles`.

5.3 REKALL

O Rekall[6] é uma ferramenta que permite efetuar a análise de aquisições de memória, cuja utilização se revela acessível para o utilizador.

No que toca às interfaces de utilização linha de comandos e consola interativa, estas são bastante similares, embora existam algumas diferenças. A linha de comandos do Rekall torna-se a opção mais viável para o utilizador que pretenda executar um único comando sobre uma dada aquisição de memória, sendo o comando base para tal '`rekal -f nomeFichAquisicaoMem plugin [opcoesplugin]`', em que 'nomeFichAquisicaoMem' representa o ficheiro resultante da aquisição da memória. Contudo, quando se pretende efetuar uma exploração a uma dada aquisição que implique a execução de diversos *plugins* é mais vantajoso recorrer à consola interativa (comando '`rekal -f nomeFichAquisicaoMem`'). Já para se efetuar uma análise à memória ao sistema no qual se executa naquele momento o Rekall, terá que se executar o comando '`rekal -v live`'.

A Tabela 8 resume os modos de execução possíveis da ferramenta Rekall, juntamente com os comandos a executar para cada um dos modos.

6 https://github.com/google/rekall

Tabela 8: Modos de execução do Rekall

Modo de execução	**Comando**
Linha de Comandos	`rekal -f nomeFichAquisicaoMem plugin [opcoesplugin]`
Consola Interativa	`rekal -f nomeFichAquisicaoMem`
Modo *Live*	`rekal -v live`

Tal como já foi referido anteriormente, nesta dissertação, será alvo de análise a interface de linha de comandos do Rekall e a consola interativa.

5.3.1 *Processo de Instalação*

O processo de instalação do Rekall, quer em *Windows* ou em *Linux*, requer a instalação do *Python* (versão 3.6) bem como à instalação de algumas *packages* através do instalador de pacotes *pip*[7].

O Rekall apresenta ainda alguma instabilidade. São exemplo disso, a necessidade de rebaixar a versão do *future* da 0.17 para a 0.16 e do módulo '*pyaff4*' da versão 0.27 para a 0.26.post6 (Figura 41). Para o primeiro caso foi necessário executar o comando '`pip install future==0.16.0`', enquanto que, para o segundo caso, foi executado o comando '`pip install pyaff4==0.26.post6`' (Mike Cohen, 2019; Ribeiro et al., 2019). A execução destes passos possibilitou o funcionamento do Rekall em *Windows 10*.

```
c:\users\santo\dev\lib\site-packages\rekall\plugins\addrspaces\aff4.py in __init__(self,
filename, **kwargs)
    123                                     lexicon.AFF4_FILE_NAME,
    124                                     rdfvalue.XSDString(
--> 125                                         os.path.join(cache_dir, "aff4_cache")))
    126         except IOError:
    127             pass
```

Figura 41: Código envolvido no término não regular do Rekall em *Windows 10*

5.3.2 *Utilização da Framework*

Um dos aspetos desta ferramenta a realçar é o facto de não ser necessário especificar o perfil adequado à imagem de memória, o que permite que o utilizador inicie de imediato a análise pretendida, executando, para tal, *plugins*.

7 https://pypi.org/project/pip/

O Rekall faz uso de um conjunto de *plugins* para o utilizador obter a informação necessária. Alguns dos *plugins* têm a mesma designação, funcionalidade e *output* que os do Volatility. Assim sendo, e tal como no Volatility, por exemplo, é possível obter informações e inferências acerca dos processos existentes em memória, tais como, a listagem de processos em execução no momento da aquisição (*plugin* `pslist`) e também a obtenção dos privilégios destes (*plugin* `privileges`) (R. Forensics, 2019d).

Também é possível utilizar esta ferramenta em contexto de deteção de *malware* tal como o Volatility, por exemplo, com o *plugin* `malfind`. Nesta ferramenta há ainda a possibilidade de se obter artefactos de rede como, por exemplo, com o *plugin* `services` que permite obter os serviços em memória. Através do *plugin* `dlllist` é possível obter uma lista dos ficheiros DLL carregados por cada processo, sendo apresentado, por exemplo, o caminho dos ficheiros DLL. Com o *plugin* `tokens` são apresentados os *tokens* de autorização que cada processo detém, sendo que o Rekall tem a capacidade de resolver automaticamente o SID do *token* para o nome de um utilizador (Listagem 8) (R. Forensics, 2019d).

Listagem 8: Parte do *output* da execução do *plugin* `tokens` sobre uma imagem de memória

```
(Dev) C:\Users\santo>rekal -f C:\Users\santo\Desktop\ch2.dmp tokens
             Process                        Sid            Comment
------------------------------ ------------------------- -------
0x87978b78 System            4 S-1-5-18                  Local System
0x87978b78 System            4 S-1-5-32-544              Administrators
0x87978b78 System            4 S-1-1-0                   Everyone
0x87978b78 System            4 S-1-5-11                  Authenticated Users
0x87978b78 System            4 S-1-16-16384              System Mandatory Level
0x88c3ed40 smss.exe        308 S-1-5-18                  Local System
0x88c3ed40 smss.exe        308 S-1-5-32-544              Administrators
0x88c3ed40 smss.exe        308 S-1-1-0                   Everyone
0x88c3ed40 smss.exe        308 S-1-5-11                  Authenticated Users
```

Nesta ferramenta é possível enumerar os *desktops* e as *threads* de cada *desktop* (Figura 42). O *malware* pode utilizar *desktops*, por exemplo, para executar aplicações em *desktops* alternados de modo a que o utilizador conectado não os visualize (Ligh Hale et al., 2014). Na Figura 42 encontra-se o *desktop* 'WinSta0\Default', cujo os processos que se encontram associados a este foram iniciados pelo utilizador que estava conectado ao sistema (Microsoft, 2018).

```
(Dev) C:\Users\santo>rekall -f C:\Users\santo\Desktop\ch2.dmp desktops
 Win32k/GUID/21B142079AA54C46B81E8016949ED1D82 matched offset 0xfcff3+0x81f00000  tagDESKTOP
Name          Sid Hooks   tagWND   Winds Thrd  _EPROCESS
  ---------- -------------------- --- ----- ---------- ----- ----- ---------
--------------------------------------------------------------------------------
Desktop: 0x892fcf78, Name: WinSta0\Default
Heap: 0xfea00000, Size: 0xc00000, Base: 0xfea00000, Limit: 0xff600000
--------------------------------------------------------------------------------
  0x892fcf78 Default              0   32    0xfea00618 5     2448  0x8a0f9c40 spoolsv.exe  1712
  0x892fcf78 Default              0   32    0xfea00618 5     2440  0x8a0f9c40 spoolsv.exe  1712
  0x892fcf78 Default              0   32    0xfea00618 5     2436  0x8a0f9c40 spoolsv.exe  1712
  0x892fcf78 Default              0   32    0xfea00618 5     1744  0x8a0f9c40 spoolsv.exe  1712
  0x892fcf78 Default              0   32    0xfea00618 5     2240  0x8a1d84e0 vmtoolsd.exe  1968
  0x892fcf78 Default              0   32    0xfea00618 5     1984  0x8a1d84e0 vmtoolsd.exe  1968
  0x892fcf78 Default              0   32    0xfea00618 5     1972  0x8a1d84e0 vmtoolsd.exe  1968
  0x892fcf78 Default              0   32    0xfea00618 5     -     -
  0x892fcf78 Default              0   32    0xfea00618 5     -     -
------------------------------------------------
```

Figura 42: Exemplo de um *desktop* e a sua respetiva informação que surgiu num *output* do *plugin* `desktops`

Um outro *plugin* do Rekall a mencionar é o *plugin* `procinfo` (Listagem 9) que revela informações sobre cada um dos processos em execução. Assim, este *plugin* pode revelar, por exemplo, informações sobre o ficheiro executável relacionado com o processo em questão e ainda as *strings* que correspondem às variáveis de ambiente disponíveis no processo (Microsoft, 2018).

Com a exploração do Rekall verificou-se que alguns *plugins* não funcionavam corretamente, como, por exemplo, os *plugins* `psscan`, `psxview` e `netscan`. Na Figura 43 encontra-se o erro que surge ao executar-se o *plugin* `psscan`. Estes *plugins* ao serem executados apresentam em *output* o mesmo erro, erro este para o qual não se conseguíu encontrar resolução.

```
(Dev) C:\Users\santo>rekall -f C:\Users\santo\Desktop\ch2.dmp psscan
a  offset_p          name         pid   offset_v   ppid     pdb     stat       create_time               exit_time
- ---------- -------------------- ----- ---------- ----- ---------- ---- ------------------------- -------------------------
2019-12-30 17:50:17,284:CRITICAL:rekall.1:Traceback (most recent call last):
  File "c:\users\santo\dev\lib\site-packages\rekall\session.py", line 866, in RunPlugin
    result = plugin_obj.render(ui_renderer) or plugin_obj
  File "c:\users\santo\dev\lib\site-packages\rekall\plugin.py", line 761, in render
    for row in self.collect():
  File "c:\users\santo\dev\lib\site-packages\rekall\plugins\windows\filescan.py", line 419, in collect
    for run in self.generate_memory_ranges():
  File "c:\users\santo\dev\lib\site-packages\rekall\plugins\windows\common.py", line 544, in generate_memory_ranges
    regions = list(self.session.plugins.virt_map())
  File "c:\users\santo\dev\lib\site-packages\rekall\plugin.py", line 379, in __iter__
    for x in self.collect():
  File "c:\users\santo\dev\lib\site-packages\rekall\plugins\windows\misc.py", line 129, in collect
    key=lambda x: (int(x["virt_start"]), int(x["virt_end"]))):
  File "c:\users\santo\dev\lib\site-packages\rekall_lib\utils.py", line 1164, in __iter__
    for item in self._iterator:
  File "c:\users\santo\dev\lib\site-packages\rekall\plugins\windows\misc.py", line 190, in collect_from_MiSystemVaType
    for offset in range(system_range_start, 0xffffffff, va_table_size):
TypeError: 'float' object cannot be interpreted as an integer

2019-12-30 17:50:17,284:CRITICAL:root:'float' object cannot be interpreted as an integer. Try --debug for more information.
Traceback (most recent call last):
  File "c:\python36\Lib\runpy.py", line 193, in _run_module_as_main
    "__main__", mod_spec)
  File "c:\python36\Lib\runpy.py", line 85, in _run_code
    exec(code, run_globals)
  File "C:\Users\santo\Dev\Scripts\rekall.exe\__main__.py", line 9, in <module>
  File "c:\users\santo\dev\lib\site-packages\rekall\rekal.py", line 104, in main
    user_session.RunPlugin(plugin_cls, **config.RemoveGlobalOptions(flags))
  File "c:\users\santo\dev\lib\site-packages\rekall\session.py", line 869, in RunPlugin
    self._HandleRunPluginException(ui_renderer, e)
  File "c:\users\santo\dev\lib\site-packages\rekall\session.py", line 866, in RunPlugin
    result = plugin_obj.render(ui_renderer) or plugin_obj
  File "c:\users\santo\dev\lib\site-packages\rekall\plugin.py", line 761, in render
    for row in self.collect():
  File "c:\users\santo\dev\lib\site-packages\rekall\plugins\windows\filescan.py", line 419, in collect
    for run in self.generate_memory_ranges():
  File "c:\users\santo\dev\lib\site-packages\rekall\plugins\windows\common.py", line 544, in generate_memory_ranges
    regions = list(self.session.plugins.virt_map())
  File "c:\users\santo\dev\lib\site-packages\rekall\plugin.py", line 379, in __iter__
    for x in self.collect():
  File "c:\users\santo\dev\lib\site-packages\rekall\plugins\windows\misc.py", line 129, in collect
    key=lambda x: (int(x["virt_start"]), int(x["virt_end"]))):
  File "c:\users\santo\dev\lib\site-packages\rekall_lib\utils.py", line 1164, in __iter__
    for item in self._iterator:
  File "c:\users\santo\dev\lib\site-packages\rekall\plugins\windows\misc.py", line 190, in collect_from_MiSystemVaType
    for offset in range(system_range_start, 0xffffffff, va_table_size):
TypeError: 'float' object cannot be interpreted as an integer
```

Figura 43: Erro que surge ao ser executado o *plugin* `psscan` no Rekall

Listagem 9: Parte do *output* do *plugin* `procinfo` acerca do processo '*csrss.exe*'

```
 Pid: 404 csrss.exe

Process Environment
   ComSpec=C:\Windows\system32\cmd.exe
   FP_NO_HOST_CHECK=NO
   NUMBER_OF_PROCESSORS=1
   OS=Windows_NT
   Path=C:\Windows\system32;C:\Windows;C:\Windows\System32\Wbem;C:\Windows\Syste
   ↪  m32\WindowsPowerShell\v1.0\
   PATHEXT=.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC
   PROCESSOR_ARCHITECTURE=x86
   PROCESSOR_IDENTIFIER=x86 Family 6 Model 23 Stepping 6, GenuineIntel
   PROCESSOR_LEVEL=6
   PROCESSOR_REVISION=1706
   PSModulePath=C:\Windows\system32\WindowsPowerShell\v1.0\Modules\
   SystemDrive=C:
   SystemRoot=C:\Windows
   TEMP=C:\Windows\TEMP
   TMP=C:\Windows\TEMP
   USERNAME=SYSTEM
   windir=C:\Windows

PE Infomation
          Attribute                                          Value
------------------------------
↪  ------------------------------------------------------------------
Machine                         IMAGE_FILE_MACHINE_I386
TimeDateStamp                   2009-07-13 23:11:09Z
Characteristics                 IMAGE_FILE_32BIT_MACHINE,
↪  IMAGE_FILE_EXECUTABLE_IMAGE
GUID/Age                        -
PDB                             -
MajorOperatingSystemVersion     6
MinorOperatingSystemVersion     1
MajorImageVersion               6
MinorImageVersion               1
MajorSubsystemVersion           6
MinorSubsystemVersion           1

Sections (Relative to 0x4a060000):
Perm   Name     Raw Off      VMA        Size
---- -------- ---------- ---------- ----------
xr-  .text    0x00000400 0x00001000 0x00000800
-rw  .data    0x00000c00 0x00002000 0x00000200
-r-  .rsrc    0x00000e00 0x00003000 0x00000800
-r-  .reloc   0x00001600 0x00004000 0x00000200
```

Na Tabela 9 encontra-se um sumário dos *plugins* do Rekall mencionados no texto, e ainda uma breve descrição de cada um dos mesmos.

Tabela 9: *Plugins* do Rekall apresentados no texto com a descrição de cada um destes

Plugin	**Descrição**
desktops	Lista os *desktops* e as *threads* de cada um dos *desktops*.
dlllist	Apresenta os ficheiros DLL carregados por cada processo.
malfind	Encontra código escondido ou injetado.
netscan	Obtenção das conexões e *sockets* em memória.
procinfo	Apresenta informações sobre os processos em execução.
privileges	Visualização dos privilégios dos processos em execução.
pslist	Lista os processos em execução.
psscan	Obtenção dos processos escondidos e processos terminados.
psxview	Apresenta os processos escondidos utilizando vários métodos.
services	Lista os serviços existentes em memória.
tokens	Lista os *tokens* de autorização que cada processo possuí.

5.3.3 *Linguagem EFilter*

Com a exploração da *framework* Rekall torna-se evidente que a quantidade de dados disponível em *output* é considerável. Assim, por vezes, o utilizador pode sentir necessidade que o *output* seja reduzido simplesmente aos dados pretendidos.

Um exemplo de uma forma rápida e simples de se restringir o *output* de um *plugin* consiste em se fazer uso da opção `proc_regex`, que permite para o efeito o uso de expressões regulares. No exemplo da Listagem 10, podemos ver a saída do *plugin* `pslist`, sendo que foram selecionados os processos cujos nomes iniciavam por 'win', mais precisamente, pelo valor definido como valor de *regex* (R. Forensics, 2019c).

Listagem 10: *Output* do *plugin* `pslist` com a aplicação da *regex* win

```
[1] ch2.dmp 17:38:50> pslist proc_regex="win"
--------------------> pslist(proc_regex="win")
_EPROCESS          name          pid   ppid  thread_count handle_count
↪  session_id wow64    process_create_time       process_exit_time
---------- -------------------- ----- ------ ------------ ------------
↪  ---------- ------ ------------------------ ------------------------
0x892ac2b8 wininit.exe            456    396            3           77
↪  0 False  2013-01-12 16:38:14Z     -
0x892ced40 winlogon.exe           500    448            3          111
↪  1 False  2013-01-12 16:38:14Z     -
0x87cbfd40 winpmem-1.3.1.         3144   3152           1           23
↪  1 False  2013-01-12 16:59:17Z     -
Out<17:38:52> Plugin: pslist (WinPsList)
```

Contudo, em investigações pode ser necessário realizar pesquisas mais complexas, e, por sua vez, cada vez mais precisas. Com esta necessidade presente, surgiu a linguagem EFilter que é uma linguagem de consulta - com semelhanças à linguagem Structured Query Language (SQL) - que permite filtrar, combinar e personalizar o *output* dos *plugins* do Rekall. Por outras palavras, esta linguagem permite que o utilizador determine que dados pretende visualizar em *output* (R. Forensics, 2019a).

Contudo, para que a linguagem EFilter funcione é necessário que os *plugins* produzam um *output* estruturado num formato específico. Assim sendo, é necessário considerar que um *output* terá várias linhas com várias colunas, sendo que cada célula corresponde a um determinado tipo de objeto. Deste modo, para se saber essa informação recorre-se ao comando '`describe(nomePlugin)`', cujo *output* mostrará os tipos de dados de cada coluna e que subcampos pertencem a que campos principais (Listagem 11). É de referir que a informação obtida por este comando torna-se relevante para o utilizador perceber como é que poderá construir as suas consultas (R. Forensics, 2019a).

Listagem 11: Parte do *output* do *plugin* `describe` do *pslist*

```
[1] ch2.dmp 09:11:59> describe(pslist)
                      Field                              Type
---------------------------------------------------- ----
  _EPROCESS                                          _EPROCESS
. AccountingFolded                                   BitField
. ActiveProcessLinks                                 _LIST_ENTRY
.. Blink                                             Pointer
.. Flink                                             Pointer
.. indices                                           tuple
.. obj_end                                           int
.. obj_name                                          str
.. obj_offset                                        int
.. obj_size                                          int
.. obj_type                                          str
.. parents                                           generator
. ActiveThreads                                      unsigned long
. ActiveThreadsHighWatermark                         unsigned long
. AddressCreationLock                                _EX_PUSH_LOCK
```

No Rekall, o funcionamento de um comando com EFilter presente nele inicia-se pela análise da consulta para, de seguida, executar os *plugins* necessários para satisfazer a consulta em questão. De seguida, os dados obtidos são filtrados pelo EFilter, sendo assim criado o *output* de acordo com o pretendido na consulta. Por conseguinte, há que ter em conta que o EFilter executa os *plugins* do Rekall para obter os dados, ou seja, cria dados dinâmicos. Já o contrário ocorre com o SQL que, por sua vez, analisa os dados armazenados (B. R. Forensics, 2016).

Para perceber melhor o funcionamento desta linguagem foram realizadas algumas pesquisas a um ficheiro exemplo de uma aquisição de memória.

Relativamente ao *plugin* `pslist`, uma pesquisa simples e que se pode realizar consiste em se obter em *output* unicamente o ID do processo em questão, o ID do processo-pai e o nome desse mesmo processo (Listagem 12).

Listagem 12: Parte do *output* da pesquisa dos processos com seleção de colunas

```
[1] ch2.dmp 09:18:36> select _EPROCESS.name, _EPROCESS.pid, ppid from pslist()
     name         pid  ppid
--------------- ----- -----
System           4     0
smss.exe         308   4
wlms.exe         336   560
csrss.exe        404   396
VMUpgradeHelpe   448   560
wininit.exe      456   396
csrss.exe        468   448
```

O operador '`where`' permite que o utilizador restrinja o *output* a uma determinada condição, tornando-se algo bastante útil (Listagem 13). É ainda de ter em conta

a existência do operador '`not`' que permite excluir algum valor em especifico do *output* (B. R. Forensics, 2016).

Listagem 13: Pesquisa dos processos, os quais apresentem como ID do processo-pai (*ppid*) o valor 2548

```
[1] ch2.dmp 09:20:37> select * from pslist() where ppid==2548
           _EPROCESS             ppid thread_count handle_count session_id wow64
           ↪  process_create_time  process_exit_time
------------------------------ ---- ------------ ------------ ---------- -----
↪  -------------------- ------------------
0x9549f678 iexplore.exe     1136 2548 18           454          1          False
↪  2013-01-12 16:57:44Z -
0x95495c18 taskmgr.exe      1232 2548 6            116          1          False
↪  2013-01-12 16:42:29Z -
0x87b82438 VMwareTray.exe   2660 2548 5            80           1          False
↪  2013-01-12 16:40:29Z -
0x87aa9220 VMwareUser.exe   2676 2548 8            190          1          False
↪  2013-01-12 16:40:30Z -
0x87b784b0 AvastUI.exe      2720 2548 14           220          1          False
↪  2013-01-12 16:40:31Z -
0x898fe8c0 StikyNot.exe     2744 2548 8            135          1          False
↪  2013-01-12 16:40:32Z -
0x87b6b030 iexplore.exe     2772 2548 2            74           1          False
↪  2013-01-12 16:40:34Z -
0x87bf7030 cmd.exe          3152 2548 1            23           1          False
↪  2013-01-12 16:44:50Z -
0x87c6a2a0 swriter.exe      3452 2548 1            19           1          False
↪  2013-01-12 16:41:01Z -
Out<09:20:38> Plugin: search (Search)
```

Em pesquisas com o EFilter há que ter em conta a diferença entre o operador '`=~`' e a função '`regex_search()`'. Ambas têm como objetivo estabelecer uma correspondência entre uma expressão e os valores de um dado campo. A diferença entre estas é que o operador '`=~`' é *case sensitive* enquanto que a função é *case insensitive*, sendo que em algumas pesquisas - por exemplo, nomes de ficheiros - poderá ser útil a função ao invés do operador. Na Listagem 14 é apresentado um exemplo da utilização da função '`regex_search()`' (B. R. Forensics, 2016).

Listagem 14: Pesquisa dos processos '*avast*' com o nome do processo, ID e caminho do ficheiro do processo

```
[1] ch2.dmp 09:25:20> select _EPROCESS.name, _EPROCESS.pid, _EPROCESS.FullPath
↪  FROM pslist() WHERE regex_search("avast", _EPROCESS.name)
   name     pid                        FullPath
------------ ---- ------------------------------------------------
AvastSvc.exe 1220 C:\Program Files\AVAST Software\Avast\AvastSvc.exe
AvastUI.exe  2720 C:\Program Files\AVAST Software\Avast\AvastUI.exe
Out<09:25:21> Plugin: search (Search)
```

Como se pode ainda constatar na Listagem 14, com o Efilter acedeu-se a uma coluna que não consta no *output* padrão do *plugin* `pslist`, nomeadamente, a coluna

‘`_EPROCESS.FullPath`‘, que se refere ao caminho do ficheiro executável referente ao processo.

Um outro exemplo da utilização da função ‘`regex_search()`‘ consiste em apresentar em *output* os processos nos quais no seu nome exista, por exemplo, o caracter ‘k‘ em qualquer posição na *string* (ver Listagem 15).

Listagem 15: Pesquisa dos processos cujo o nome do processo apresente o caracter ‘k‘

```
[1] ch2.dmp 09:28:05> select _EPROCESS.name, _EPROCESS.pid, _EPROCESS.FullPath
↪  FROM pslist() WHERE regex_search("k", _EPROCESS.name)
    name     pid                      FullPath
------------ ---- --------------------------------------------------
taskmgr.exe  1232 C:\Windows\System32\taskmgr.exe
taskhost.exe 2352 C:\Windows\System32\taskhost.exe
StikyNot.exe 2744 C:\Windows\System32\StikyNot.exe
wmpnetwk.exe 3176 C:\Program Files\Windows Media Player\wmpnetwk.exe
Out<09:28:06> Plugin: search (Search)
```

É ainda possível também ordenar o *output*, por exemplo, por ordem alfabética do nome do processo (ver Listagem 16).

Listagem 16: Pesquisa anterior com ordenação alfabética do nome dos processos

```
[1] ch2.dmp 09:31:06> select _EPROCESS.name, _EPROCESS.pid FROM pslist() WHERE
↪  regex_search("k", _EPROCESS.name) order by _EPROCESS.name asc
    name       pid
------------- -----
StikyNot.exe  2744 ing -
taskhost.exe  2352
taskmgr.exe   1232
wmpnetwk.exe  3176
Out<09:31:07> Plugin: search (Search)
```

Assim, para colocar os valores por ordem crescente ou alfabética terá que se fazer uso do operador ‘`asc`‘, enquanto que, para ordenar os valores de uma determinada coluna por ordem descrescente terá que se utilizar o operador ‘`desc`‘, algo bastante similar ao SQL (B. R. Forensics, 2016).

Uma pesquisa interessante a ser realizada é relativa ao *plugin* `tokens`. Na Listagem 17 encontra-se parte do *output* do *plugin* `tokens`, no qual é possível detetar a existência de um utilizador, designado por ‘John Doe‘. Deste modo, é possível realizar-se pesquisas tendo por base o nome de utilizador encontrado. Um exemplo de uma pesquisa possível é a presente na Listagem 18, na qual se pesquisa por todos os processos inicializados pelo utilizador ‘John Doe‘ (R. Forensics, 2019a).

Listagem 17: Deteção de um utilizador através da execução do *plugin* `tokens`

```
[1] ch2.dmp 09:34:50> tokens
--------------------> tokens()
                Process                                          Sid
                 ↪            Comment
---------------------------------------
↪  ------------------------------------------------- -------
(...)
0x9549f678 iexplore.exe             1136
↪  S-1-5-21-1646808395-2823249608-1159325386-1000     User: John Doe
0x9549f678 iexplore.exe             1136
↪  S-1-5-21-1646808395-2823249608-1159325386-513      Domain Users
0x9549f678 iexplore.exe             1136 S-1-1-0
↪              Everyone
0x9549f678 iexplore.exe             1136 S-1-5-32-544
↪              Administrators
0x9549f678 iexplore.exe             1136 S-1-5-32-545
↪              Users
0x9549f678 iexplore.exe             1136 S-1-5-4
↪              Interactive
0x9549f678 iexplore.exe             1136 S-1-2-1
↪              Console Logon (Users who are logged onto the physical console)
0x9549f678 iexplore.exe             1136 S-1-5-11
↪              Authenticated Users
0x9549f678 iexplore.exe             1136 S-1-5-15
↪              This Organization
0x9549f678 iexplore.exe             1136 S-1-5-5-0-287851
↪              Logon Session
0x9549f678 iexplore.exe             1136 S-1-2-0
↪              Local (Users with the ability to log in locally)
0x9549f678 iexplore.exe             1136 S-1-5-64-10
↪              NTLM Authentication
0x9549f678 iexplore.exe             1136 S-1-16-8192
↪              Medium Mandatory Level
(...)
0x87b6b030 iexplore.exe             2772
↪  S-1-5-21-1646808395-2823249608-1159325386-1000     User: John Doe
```

Listagem 18: Processos inicializados pelo utilizador 'John Doe'

```
[1] ch2.dmp 09:43:09> select * from tokens() where Comment =~ 'User: John Doe'
        Process                          Sid                  Comment
---------------------- ---------------------------------- ----------
0x9549f678 iexplore.exe  1136 S-1-5-21-1646808395-2823249608-1159 User: John
                               325386-1000                          Doe
0x95495c18 taskmgr.exe  1232 S-1-5-21-1646808395-2823249608-1159 User: John
                              325386-1000                          Doe
0x89898030 cmd.exe  1616 S-1-5-21-1646808395-2823249608-1159 User: John
                         325386-1000                          Doe
0x954826b0 conhost.exe  2168 S-1-5-21-1646808395-2823249608-1159 User: John
                              325386-1000                          Doe
0x87ac0620 taskhost.exe  2352 S-1-5-21-1646808395-2823249608-1159 User: John
                               325386-1000                          Doe
0x87ad44d0 dwm.exe  2496 S-1-5-21-1646808395-2823249608-1159 User: John
                         325386-1000                          Doe
0x87ac6030 explorer.exe  2548 S-1-5-21-1646808395-2823249608-1159 User: John
                               325386-1000                          Doe
```

Tal como no SQL, na linguagem EFliter é possível renomear os nomes das colunas com o operador '`as`'. Outro aspeto é o facto de o EFilter suportar subconsultas, ou

seja, permite que os resultados de uma consulta sejam considerados como *input* de uma outra consulta (B. R. Forensics, 2016). É ainda de ter em conta que se pode limitar o número de linhas que surge no *output* adicionando-se à consulta a expressão '`limit numeroLimite`', em que 'numeroLimite' corresponde ao número de linhas que se pretende que estejam em *output*.

Por último, para uma análise *live*, a linguagem EFilter disponibiliza alguns *plugins* que se podem revelar úteis, como, por exemplo, o *plugin* `glob` que pesquisa por ficheiros utilizando uma expressão `glob` (R. Forensics, 2019a).

5.3.4 *Exploração do Rekall na SIFT Workstation*

A SIFT Workstation apresenta diversas ferramentas forenses e de resposta a incidentes. Entre as várias ferramentas forenses que apresenta já instaladas, uma delas corresponde ao Rekall. Assim sendo, foi realizada uma análise do Rekall na SIFT Workstation.

A obtenção e consequente utilização da SIFT Workstation poderá ser efetuada de duas formas, mais precisamente, através do *download* da SIFT Workstation VM Appliance, ou, então, através da SIFT-CLI num sistema *Ubuntu 16.04* ou *Windows 10* (S. D. Forensics e Response, 2019). Nesta dissertação foi obtida a SIFT Workstation VM Appliance, de modo a se executar e explorar a *framework* Rekall nesta.

Inicialmente foi obtida a SIFT Workstation VM Appliance[8], sendo esta posteriormente executada através da ferramenta VMWare Workstation Player (versão 15.5)[9]. Para se aceder à SIFT Workstation terá que se introduzir as credenciais apresentadas na página da ferramenta, nomeadamente, como *login* 'sansforensics' e como *password* 'forensics'.

Para ocorrer o envio do ficheiro do *dump* de memória a analisar para a SIFT Workstation foi criada uma pasta partilhada entre a máquina física e a máquina virtual. Tal como se pode verificar na Figura 44 esta foi configurada nas definições da máquina virtual. Posteriormente, para se verificar se o processo ocorreu com sucesso, na máquina virtual, na diretoria '`/mnt/hgfs`' verificou-se que existia uma pasta com a designação anteriormente atribuída (Figura 45) (vmware, 2019).

8 https://digital-forensics.sans.org/community/downloads
9 https://www.vmware.com/products/workstation-player/workstation-player-evaluation.html

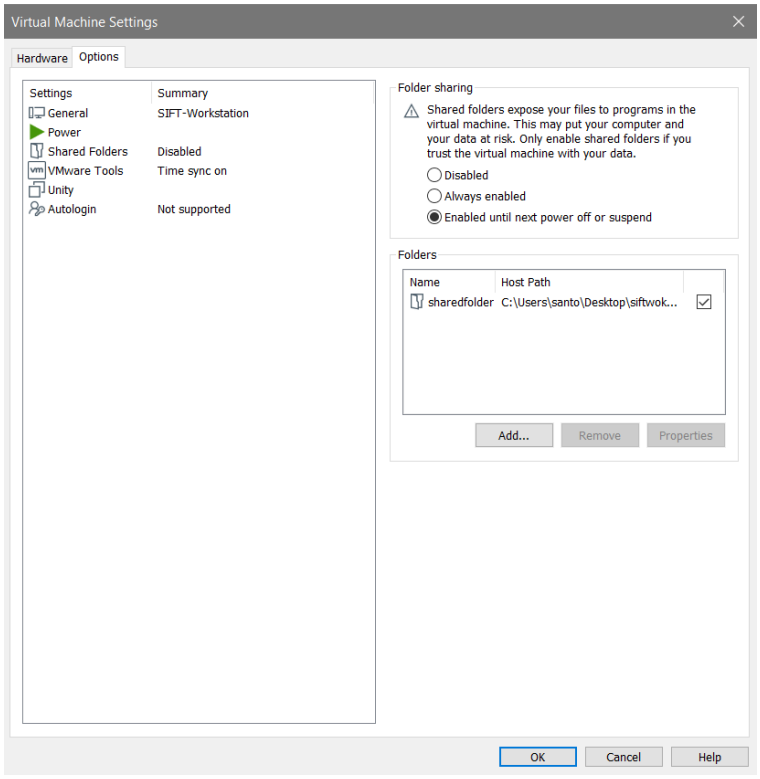


Figura 44: Configuração da pasta partilhada na SIFT Workstation

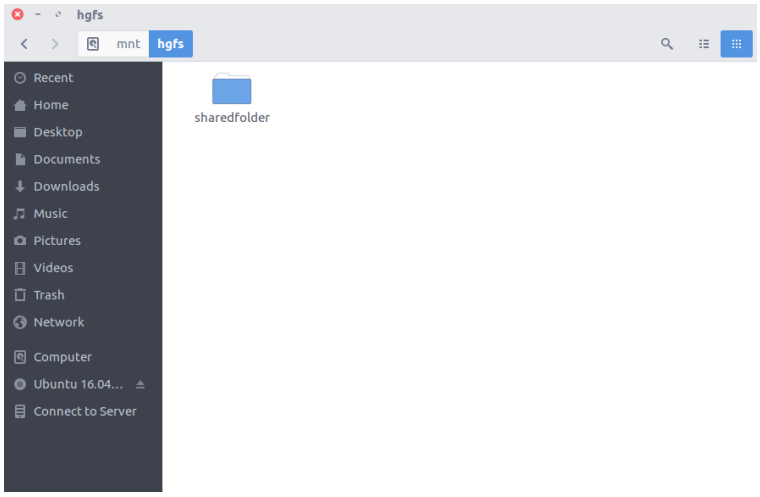


Figura 45: Visualização da pasta partilhada na SIFT Workstation

Nesta máquina virtual, pode-se executar o Rekall em modo consola interativa e linha de comandos. Na Listagem 19 é apresentado parte do *output* da execução do *plugin* `pslist` sobre uma imagem de memória que, por sua vez, está a ser analisada em modo consola interativa.

Listagem 19: Parte do *output* do *plugin* `pslist` no Rekall na SIFT Workstation

```
[1] ch2.dmp 17:01:58> pslist
--------------------> pslist()
_EPROCESS          name           pid   ppid  thread_count handle_count
↪  session_id wow64    process_create_time       process_exit_time
---------- -------------------- ----- ------ ------------ ------------
↪  ---------- ------ ------------------------ -------------------------
0x87978b78 System                    4      0          103         3257
↪  - False  2013-01-12 16:38:09Z     -
0x88c3ed40 smss.exe                308      4            2           29
↪  - False  2013-01-12 16:38:09Z     -
0x9541c7e0 wlms.exe                336    560            4           45
↪  0 False  2013-01-12 16:39:21Z     -
0x8929fd40 csrss.exe               404    396            9          469
↪  0 False  2013-01-12 16:38:14Z     -
0x8a1f5030 VMUpgradeHelpe          448    560            4           89
↪  0 False  2013-01-12 16:39:21Z     -
0x892ac2b8 wininit.exe             456    396            3           77
↪  0 False  2013-01-12 16:38:14Z     -
0x88d03a00 csrss.exe               468    448           10          471
↪  1 False  2013-01-12 16:38:14Z     -
0x892ced40 winlogon.exe            500    448            3          111
↪  1 False  2013-01-12 16:38:14Z     -
```

Na exploração realizada ao Rekall instalado nesta máquina verificou-se que é possível realizar pesquisas através da linguagem EFilter. Na Listagem 20 encontra-se a execução de uma pesquisa EFilter, na qual se procura por todos os processos '*svchost*'.

Listagem 20: Pesquisa pelos processos *svchost*, através da linguagem EFilter, no Rekall na SIFT Workstation

```
[1] ch2.dmp 17:09:56> select * from pslist() where regex_search('svchost',
↪  _EPROCESS.name)
          _EPROCESS           ppid thread_count handle_count session_id
          ↪  wow64       process_create_time  process_exit_time
---------------------------- ---- ------------ ------------ ----------
↪  ---------------- -------------------- -----------------
0x8962f030 svchost.exe    692 560  10           353          0          False
↪          2013-01-12 16:38:21Z -
0x897b5c20 svchost.exe    764 560  7            263          0          False
↪          2013-01-12 16:38:23Z -
0x89805420 svchost.exe    832 560  19           435          0          False
↪          2013-01-12 16:38:23Z -
0x89852918 svchost.exe    904 560  17           409          0          False
↪          2013-01-12 16:38:24Z -
0x8986b030 svchost.exe    928 560  26           869          0          False
↪          2013-01-12 16:38:24Z -
0x898911a8 svchost.exe   1084 560  10           257          0          False
↪          2013-01-12 16:38:26Z -
0x898b2790 svchost.exe   1172 560  15           475          0          False
↪          2013-01-12 16:38:27Z -
0x8a102748 svchost.exe   1748 560  18           310          0          False
↪          2013-01-12 16:38:58Z -
0x89f3d2c0 svchost.exe   3352 560  9            141          0          False
↪          2013-01-12 16:40:58Z -
0x89f1d3e8 svchost.exe   3624 560  14           348          0          False
↪          2013-01-12 16:41:22Z -
Out<17:09:56> Plugin: search (Search)
```

Ao explorar o Rekall nesta máquina, constatou-se ainda que os *plugins* cuja execução falharam na execução do Rekall em *Windows*, também falham na execução do Rekall na SIFT Workstation, sendo exemplos desses *plugins* o `psscan` e o `netscan`. Na Listagem 21 é apresentado o erro que surge em *output* após a execução do *plugin* `psscan`.

Listagem 21: Erro na execução do *plugin* `psscan` do Rekall na SIFT Workstation

```
[1] ch2.dmp 17:12:18> psscan
--------------------> psscan()
a  offset_p          name          pid   offset_v   ppid     pdb      stat
↪  create_time               exit_time
- ---------- -------------------- ----- ---------- ------ ---------- ----
↪  ------------------------ ------------------------
> /opt/rekall/local/lib/python2.7/site-packages/rekall/obj.py(2477)Object()
-> "Type name must be a string, not %s" % name.__class__)
(Pdb)
```

Assim, na SIFT Workstation, o Rekall ao se encontrar já instalado facilita o trabalho do investigador pois não é necessário proceder à sua instalação. No que toca aos modos de execução, na SIFT Workstation, pode-se executar o Rekall tanto em modo consola interativa como em modo linha de comandos. Na execução do Rekall nesta máquina, verificou-se ainda que não é possível executar com sucesso alguns dos *plugins* da ferramenta.

5.4 ATIVIDADE DOS PROJETOS VOLATILITY E REKALL

Nos repositórios do GitHub pode-se analisar as alterações ocorridas nos mesmos através do gráfico '*code frequency*' que apresenta as adições e eliminações de conteúdo por semana no histórico de um repositório (GitHub, 2019).

Nas Figuras 46 e 47 encontram-se os gráficos '*code frequency*' do Rekall e do Volatility respetivamente. Através do gráfico presente na Figura 46 pode-se concluir que o Rekall é uma ferramenta que nos últimos anos tem apresentado uma diminuta atividade de desenvolvimento. Através da Figura 47 constata-se que o Volatility apresenta uma produção mais ativa comparativamente ao Rekall. Para além disso, comparando os dois gráficos verifica-se que o repositório do Rekall apresentou uma escassa atividade ao longo do ano de 2020, enquanto que no repositório do Volatility já foram efetuadas algumas adições de conteúdo no mesmo ano, sendo ainda de realçar o lançamento da versão 3.0 *Public Beta*.

Um outro facto a mencionar relativamente ao nível da produção das duas ferramentas é a data do último *commit* realizado, sendo que, até à presente data, o último *commit* no repositório do Rekall ocorreu a 5 de agosto de 2020 enquanto que o do Volatility foi a 16 de agosto de 2020 (Rekall, 2019b; Volatility Foundation, 2019d).

Figura 46: Gráfico '*code frequency*' do Rekall (Rekall, 2019a)

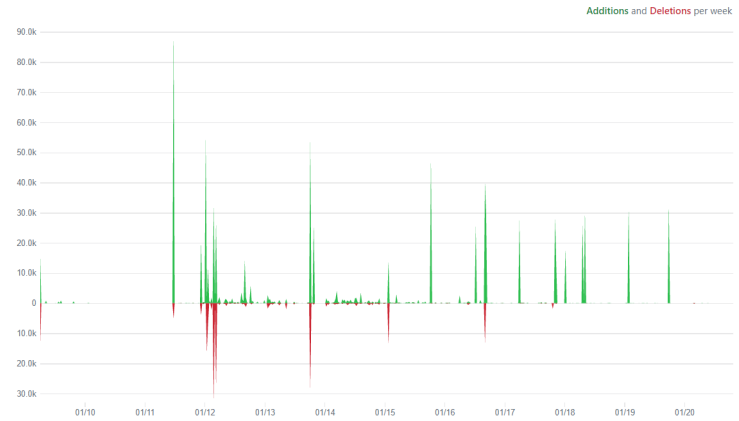


Figura 47: Gráfico '*code frequency*' do Volatility (Volatility Foundation, 2019c)

5.5 COMPARAÇÃO ENTRE VOLATILITY E REKALL

Após a exploração do Volatility e do Rekall, foi possível, por fim, realizar uma análise comparativa entre as mesmas. O Volatility e o Rekall são duas ferramentas que partilham alguns aspetos em comum, contudo apresentam também algumas divergências entre si. É de ter em conta que para esta análise comparativa foram consideradas as duas versões do Volatility exploradas, mais precisamente, a 2.6 e a 3.0 *Public Beta* e a versão 1.7.2.rc1 (*Hurricane Ridge*) do Rekall.

No que toca ao Rekall, na sua instalação e após esta estar concluída, o utilizador depara-se com alguns erros na ferramenta até obter, finalmente, uma ferramenta completamente funcional. Por exemplo, um ponto a realçar é o facto de mesmo após a ferramenta estar instalada, alguns *plugins* não se encontram funcionais. Para além disso, a documentação existente torna-se insuficiente quando se pretende a resolução de tais problemas. É de referir que na SIFT Workstation, o Rekall nesta já se apresenta previamente instalado, o que faz com que o investigador não se depare com a fase de instalação da ferramenta. Contudo, mesmo com a instalação já efetuada, alguns *plugins* permanecem não funcionais. Por outro lado, como a versão 2.6 do Volatility para sistemas *Windows* apresenta disponível um executável, não se torna necessária a sua instalação, algo que torna a sua utilização mais rápida. A versão 3.0 *Public Beta* do Volatility também não requer instalação, bastando executar a ferramenta a partir da linha de comandos do sistema.

De todas as interfaces disponíveis destas duas ferramentas, o foco para análise foi a linha de comandos (CLI) e, em ambas as ferramentas, estas são semelhantes. Com a experimentação realizada, concluíu-se que um utilizador que alterne entre as mesmas não notará uma grande necessidade de adaptação dadas as semelhanças verificadas entre ambas. Uma característica que estas duas ferramentas têm em comum é o facto de ambas apresentarem *plugins* que permitem estender as suas funcionalidades. É ainda de referir que alguns dos *plugins* destas duas ferramentas partilham não só a designação mas também a funcionalidade e informação que é devolvida ao utilizador. Um desses casos é o *plugin* `pslist` que tanto no Volatility como no Rekall listam os processos em execução. De um modo geral, ambas as ferramentas permitem obter informação vasta acerca do sistema no qual se efetuou a aquisição, possibilitando a obtenção de uma boa perceção do que ocorreu no sistema. Assim, em ambas poderá-se encontrar, por exemplo, dados sobre processos em memória e conexões de rede estabelecidas no sistema.

Contudo, apesar das semelhanças, o Volatility e o Rekall apresentam aspetos diferentes entre si. O Rekall permite realizar aquisições, ao invés do Volatility em que é necessário utilizar uma ferramenta de aquisição independente. Esta ferramenta permite também realizar *live analysis*, o que leva a recorrer-se a esta ferramenta quando, por exemplo, se pretende selecionar os dados a adquirir da memória. No Rekall existe um método de autodeteção do perfil a utilizar, o que torna desnecessário, em grande parte das vezes, que o utilizador o especifique no comando. Por outro lado, no Volatility, a autodeteção do perfil para a imagem de memória em análise só está disponível na versão 3.0. Assim, ao utilizar-se a versão 2.6 desta ferramenta, o utilizador tem que encontrar o perfil adequado e especificá-lo de modo a prosseguir

com a análise (R. Forensics, 2019c). No caso de se pretender a execução de diversos *plugins* sobre uma imagem de memória, o Rekall será a ferramenta mais adequada para o utilizador graças à consola interativa. Também no que toca ao Rekall é de mencionar a linguagem EFilter que permite personalizar os *outputs* dos *plugins*, resumindo-os à informação que é de facto pretendida.

Relativamente à automação de tarefas, o Rekall na sua documentação indica esta mesma funcionalidade, apresentando os passos necessários para o utilizador, na prática, a conseguir efetuar (R. Forensics, 2019c). Já no que toca ao Volatility, nos sítios oficiais da plataforma não existe qualquer indício dessa possibilidade. Contudo, dado que o Volatility foi escrito em *Python* é expectável que também permita a criação de *scripts* personalizados (martijnveken, 2012).

Em suma, com as explorações realizadas e com base nos aspetos anteriormente mencionados, conclui-se que o Volatility é uma ferramenta de análise mais funcional para um investigador do que o Rekall.

Na Tabela 10 é apresentado um sumário da comparação entre o Volatility e o Rekall, tendo por base algumas das observações efetuadas e já descritas acima.

Tabela 10: Tabela Comparativa entre as Ferramentas Volatility e Rekall

	Volatility	**Rekall**
Formatos Explorados	CLI	CLI
Interface	Simples e intuitiva	Simples e intuitiva
Aquisição	Não	Sim
Análise por *snapshot*	Sim	Sim
Live Analysis	Não	Sim
Autodeteção de perfil	Sim (só na versão 3.0)	Sim
Interatividade com uma única imagem de memória	Não	Sim
Informação em *output* dos *plugins*	Variada e organizada	Variada e organizada
Automação de tarefas	Sim	Sim
Linguagem para filtragem de *output*	Não	Sim

5.6 MÓDULO DO VOLATILITY PARA O AUTOPSY

O módulo do Volatility para o Autopsy desenvolvido por Mark McKinnon permite que no Autopsy sejam executados os *plugins* do Volatility sobre uma imagem de memória. Este mesmo módulo encontra-se disponível em [10] que, por sua vez, corresponde ao repositório do GitHub do seu criador.

5.6.1 *Preparação da Utilização do Módulo*

De modo a se utilizar este módulo é necessário, em primeiro lugar, efetuar a integração deste no Autopsy que é algo simples de executar. Para tal, no Autopsy tem que se selecionar a opção '*Tools*', '*Python Plugins*' e copiar a pasta do módulo para a diretoria que foi aberta com a execução do passo anterior (TSK, 2015a).

Para executar este módulo sobre uma dada imagem de memória é necessário que primeiro esta seja adicionada ao Autopsy. Assim, terá que ser adicionado uma nova '*Data Source*' de '*Logical Files*' (Figura 48) e, de seguida, selecionar o ficheiro correspondente à imagem de memória. É de referir que para esta exploração foi utilizada uma imagem de memória exemplo no formato `.dmp`.

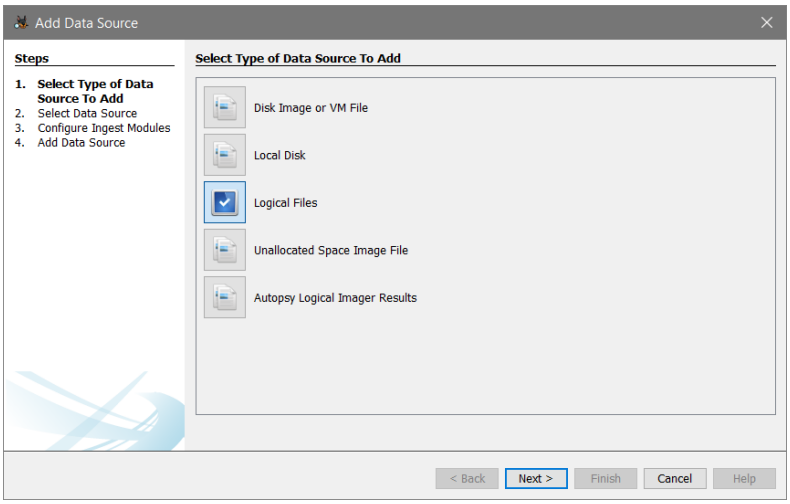


Figura 48: Seleção do tipo de fonte de dados '*Logical Files*' para adicionar a imagem de memória para análise

Com a fonte de dados adicionada pode-se prosseguir à execução do módulo sobre esta. Tal como já foi referido, serão alvo de exploração nesta dissertação dois dos

10 https://github.com/markmckinnon/Autopsy-Plugins/tree/master/Volatility

três módulos deste *plugin*, mais precisamente o 'Volatility Module' e o 'Volatility Dump File Module'.

5.6.2 *Volatility Module*

O 'Volatility Module' permite que sejam executados os *plugins* do Volatility sobre uma imagem de memória que foi adicionada à ferramenta Autopsy.

Tendo em conta que a fonte de dados já está adicionada, para utilizar este módulo, no Autopsy, é necessário selecionar a opção '*Tools*', '*Run Ingest Module*', a qual irá fazer com que surja uma janela idêntica à presente na Figura 49.

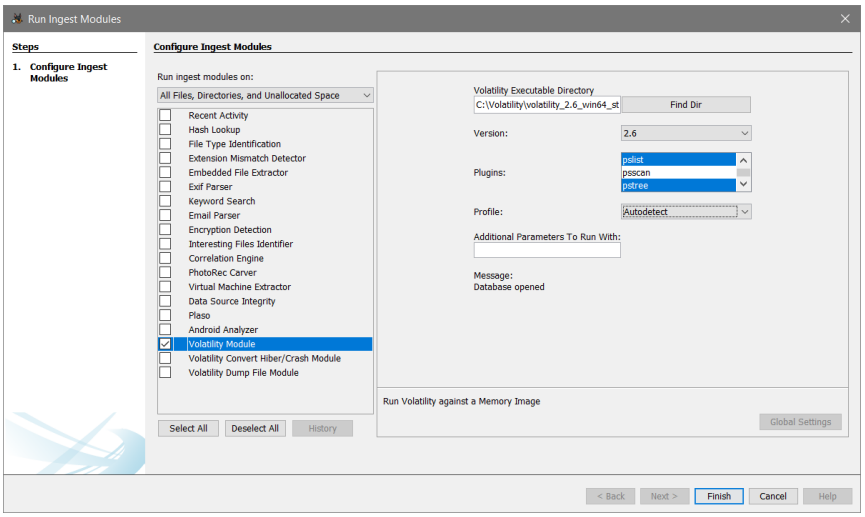


Figura 49: Seleção do 'Volatility Module' e respetiva configuração deste

Tal como se pode ver na Figura 49, é na opção '*Volatility Executable Directory*' que é indicada a diretoria na qual se encontra o executável do Volatility. Assim, conclui-se que para se utilizar este módulo é necessário possuir o executável do Volatility, sendo que na exploração realizada foi utilizada a versão 2.6 do Volatility. Em '*Plugins*' encontra-se uma lista com os *plugins* suportados pelas versões 2.5 e 2.6 do Volatility, sendo que é nesta lista que o utilizador seleciona os *plugins* que pretende executar. É de ter em conta que o nome dos *plugins* encontra-se na base de dados SQLite, que é consultada quando o número da versão do Volatility é modificado. Assim, quando uma nova versão do Volatility fica disponível, os *plugins* que essa versão suporta terão que ser adicionados à base de dados, o que torna fácil o suporte a novas versões (McKinnon, 2017). Já em '*Profile*' o utilizador seleciona o perfil indicado para a imagem de memória em causa. No caso de o utilizador não saber ao certo qual o perfil indicado, este poderá escolher a opção '*Autodetect*' que

irá selecionar o perfil automaticamente, o que levará a que seja executado o *plugin* `imageinfo` para se detetar o perfil.

Ao executar os *plugins* selecionados sobre a imagem de memória, o *output* dos *plugins* é armazenado na base de dados SQLite na diretoria 'ModuleOutput' presente na pasta do caso do Autopsy em causa. De seguida, após a execução destes terminar, o *output* dos *plugins* é importado para a secção '*Extracted Content*' da interface do Autopsy (McKinnon, 2017).

Nesta exploração, em um dos testes ao módulo foram selecionados os *plugins* `envars`, `pslist` e `pstree` e, no que toca ao *Profile*, selecionou-se a opção '*Autodetect*'. Após a conclusão da execução destes, o *output* dos *plugins* encontra-se disponível na secção '*Extracted Content*', na qual se seleciona o *output* pretendido para analisar cada *output* atentamente. Na Figura 50 encontra-se a interface gráfica do Autopsy, após a execução dos três *plugins* selecionados e ainda do *plugin* `imageinfo`, com a seleção do *output* do *plugin* `pslist`. Já na Figura 51 encontra-se parte do *output* resultante da execução do *plugin* `pstree`.

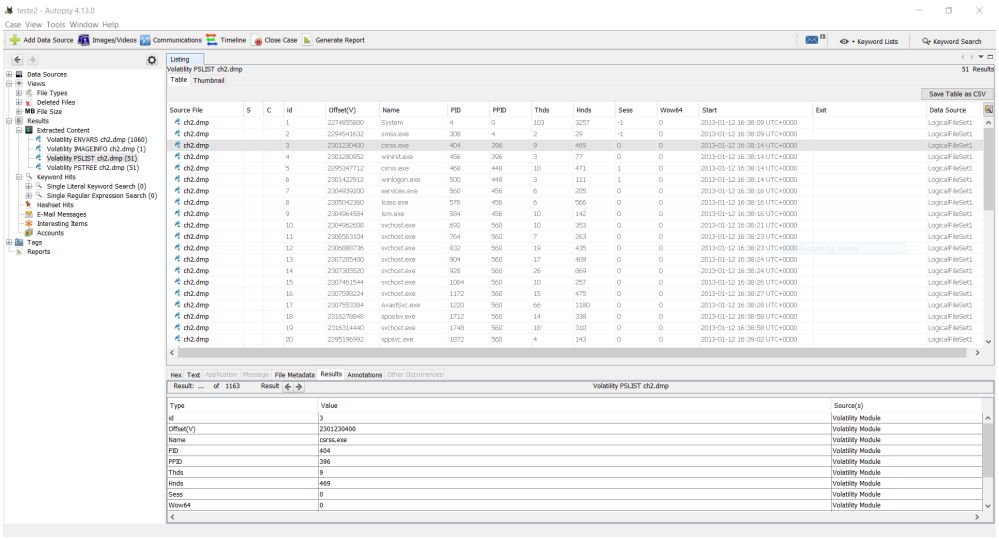


Figura 50: Visualização de parte do *output* do *plugin* `pslist` no Autopsy

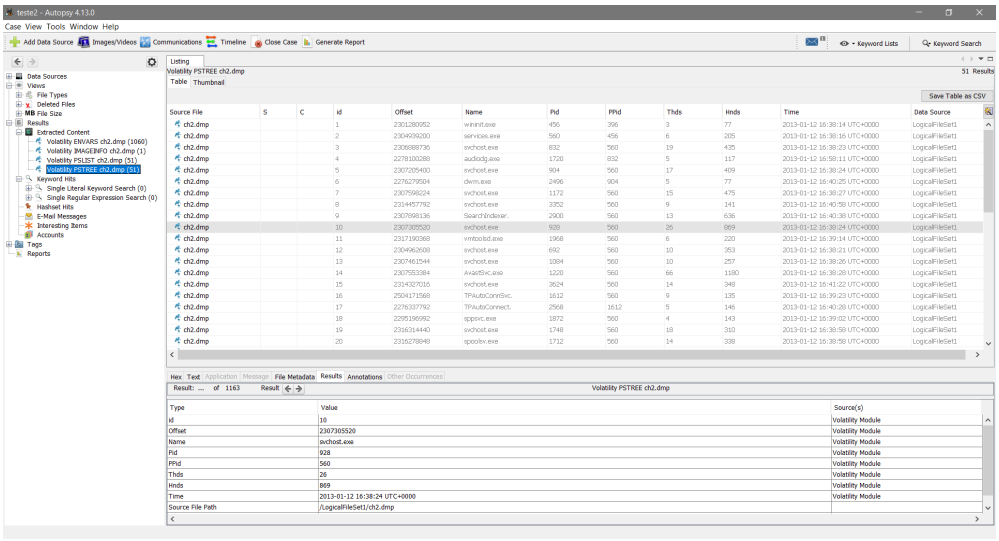


Figura 51: Visualização de parte do *output* do *plugin* `pstree` no Autopsy

É ainda possível obter um relatório com o *output* dos *plugins* executados, sendo este mesmo relatório útil para uma investigação digital. Na Figura 52 encontra-se parte do *output* do relatório HTML gerado, com os resultados obtidos pela execução dos *plugins* `envars`, `pslist` e `pstree`.

Figura 52: Parte do relatório HTML gerado no Autopsy com os resultados obtidos da execução do 'Volatility Module'

A utilização deste módulo que permite a execução do Volatility sobre uma imagem de memória no Autopsy é fácil e simples para o utilizador, tornando uma análise ao conteúdo da memória a partir desta ferramenta um processo interessante e útil a nível forense.

5.6.3 *Volatility Dump File Module*

O 'Volatility Dump File Module' tem como objetivo realizar o *dump* de ficheiros de imagens de memória. É de ter em conta que para executar este módulo também é necessário utilizar o executável do Volatility (versão 2.6).

Neste módulo, os únicos *plugins* que se encontram disponíveis são aqueles que permitem efetuar o *dump* de ficheiros. É de referir que é necessário indicar o perfil indicado para a imagem de memória ou escolher a opção '*Autodetect*'. Este módulo apresenta ainda o parâmetro de configuração '*Process ids to dump (comma seperated list)*' que permite indicar o ID dos processos aos quais se pretende efetuar o *dump*.

Nesta dissertação foi testado o *plugin* dumpregistry, de modo a obter a realizar o *dump* das *hive* da Registry para o disco (Figura 53).

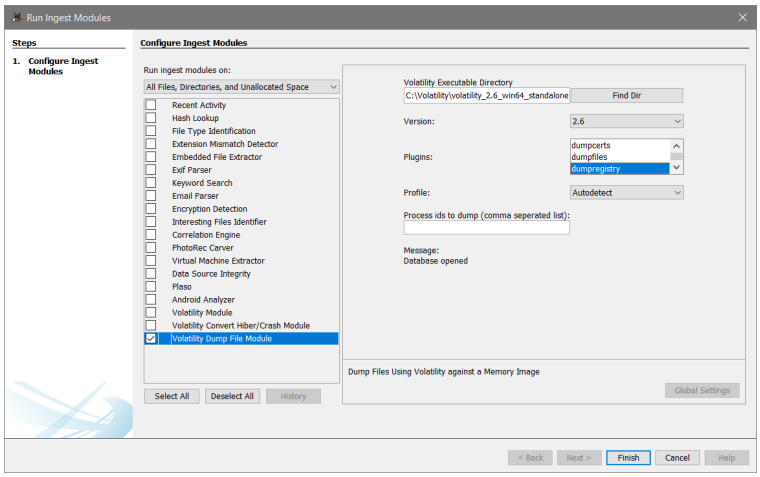


Figura 53: Configuração dos parâmetros necessários para a execução do *plugin* `dumpregistry`

Na execução do processo de *dump* definido, verificou-se a existência de um erro, que, por sua vez, impediu a obtenção dos ficheiros pretendidos, não se encontrando qualquer ficheiro na diretoria '*Module Output*' presente na pasta do caso Autopsy. Na Figura 54 encontra-se o erro que foi apresentado pelo Autopsy acerca da impossibilidade de execução do *plugin* `dumpregistry`.

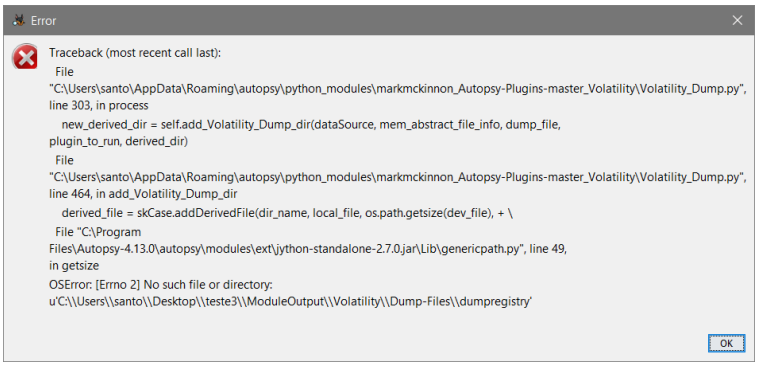


Figura 54: Erro relativo à execução do *plugin* `dumpregistry`

Foram ainda testados outros *plugins* de *dump*, como, por exemplo, o `procdump`, verificando-se neste o mesmo tipo de erro que o apresentado na execução do *plugin* `dumpregistry`.

5.7 AUTOPSY - VOLATILITY DATA SOURCE PROCESSOR

Na versão 4.15 do *software* Autopsy surgiu a funcionalidade '*Volatility Data Source Processor*'. Esta funcionalidade da ferramenta permite efetuar análises de memória no Autopsy através da execução do Volatility sobre uma imagem de memória. (TSK, 2020b)

Esta opção não surge no Autopsy por omissão. Para que a opção '*Volatility Data Source Processor*' fique disponível ao utilizador é necessário ativar o *plugin* '*Experimental*', em '*Tools*', '*Plugins*' (Figura 55). O módulo '*Experimental*' apresenta funcionalidades que ainda não fazem parte da *release* oficial do Autopsy podendo as mesmas terem menos documentação que as restantes; este módulo apresenta funcionalidades estáveis, mas que ainda podem sofrer alterações. (TSK, 2020a) Com o processo de ativação do *plugin* concluído, a opção '*Volatility Data Source Processor*' fica pronta a ser utilizada.

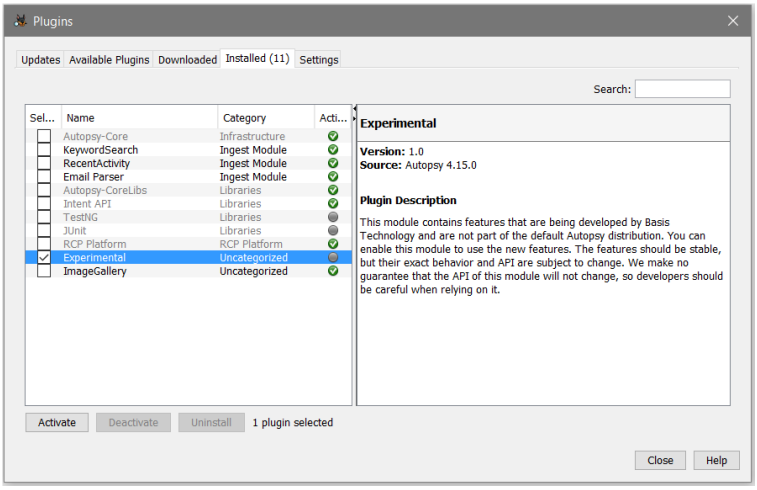


Figura 55: Ativação do *plugin* '*Experimental*'

Este *plugin* só funciona no sistema operativo *Windows*, sendo que no sistema operativo *Linux* dá erro.

Assim, para se executar a funcionalidade '*Volatility Data Source Processor*' é necessário adicionar a imagem de memória em causa, ou seja, terá que se adicionar uma '*Data Source*' de '*Memory Image File (Volatility)*' (Figura 56).

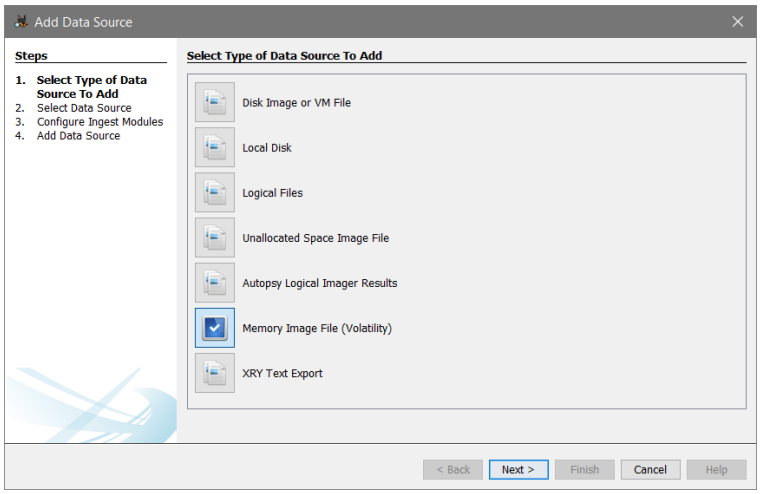


Figura 56: Seleção do tipo de fonte de dados '*Memory Image File (Volatility)*' de modo a se adicionar a imagem de memória pretendida

De seguida, seleciona-se a imagem de memória que se pretende analisar, sendo que para a exploração desta funcionalidade foi utilizada uma imagem de memória exemplo no formato `.dmp`. Por fim, efetua-se a seleção do(s) *plugin(s)* do Volatility que se pretende que sejam executados (Figura 57). Nesta exploração, em um dos testes a esta opção foram selecionados os *plugins* `pslist` e `pstree`.

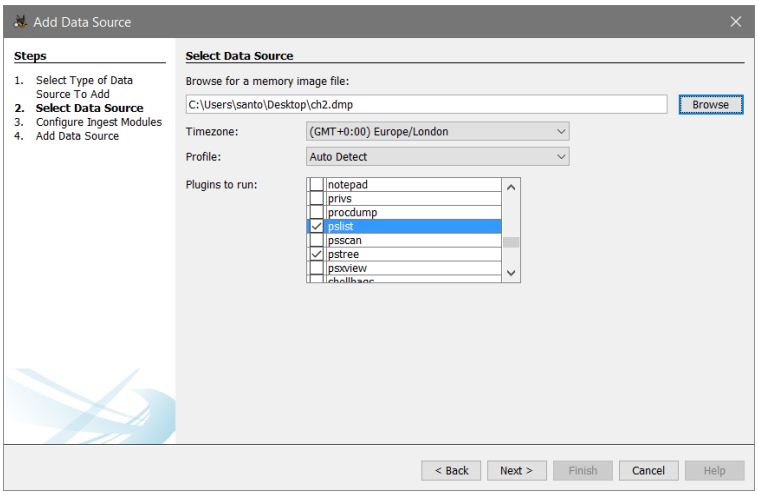


Figura 57: Seleção do ficheiro correspondente à imagem de memória e dos *plugins* do Volatility a executar

Após a execução do Volatility sobre a imagem de memória estar concluída, os resultados obtidos são disponibilizados ao utilizador na secção '*Module Output*' da árvore do lado direito da interface gráfica (Figura 58). Como se pode verificar na Figura 58, o *plugin* `imageinfo` é executado automaticamente pelo Volatility de modo a ser detetado automaticamente o perfil, o que justifica o facto de serem apresentados ao utilizador os resultados obtidos pelo *plugin* `imageinfo`.

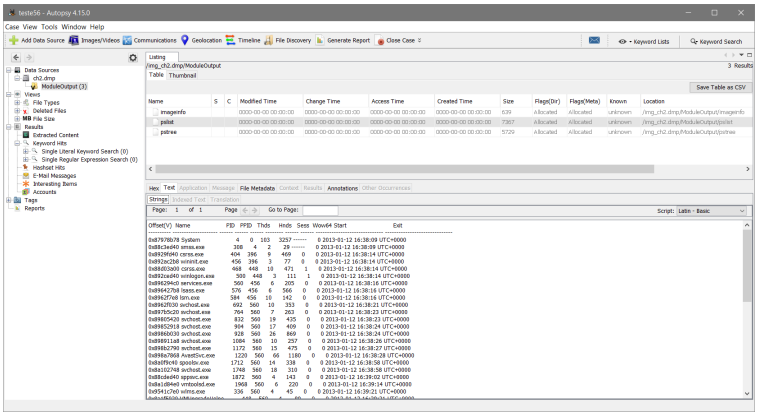


Figura 58: Visualização dos resultados obtidos da execução do *plugin* `pslist`

O utilizador também tem disponível em ficheiros de texto o *output* de cada *plugin* do Volatility, sendo que esses mesmos ficheiros encontram-se na pasta '*ModuleOutput/Volatility*' na pasta do caso do Autopsy em causa.

5.8 SÍNTESE

Após a aquisição da memória, o investigador poderá iniciar a fase de análise dos dados recolhidos, recorrendo, para tal, a uma ferramenta de análise. Neste capítulo foram exploradas e analisadas as ferramentas Volatility, Rekall e ainda um módulo do Volatility para o Autopsy. Por fim, foi ainda explorada a funcionalidade '*Volatility Data Source Processor*' do Autopsy.

Entre o Volatility e o Rekall verificam-se algumas semelhanças no que toca à designação e função dos *plugins*, bem como nos *outputs* destes. Contudo, enquanto que no Volatility todos os *plugins* testados funcionaram corretamente, no Rekall alguns deles não funcionam. Ao se comparar a atividade dos projetos Volatility e Rekall, verificou-se que o Rekall não apresentou atividade no ano 2019. Foi ainda efetuada uma comparação entre o Rekall e o Volatility, de forma a se analisar as principais diferenças entre as duas ferramentas.

O módulo do Volatility para o Autopsy analisado permite analisar uma imagem de memória no Autopsy, sendo que os resultados obtidos da execução do Volatility são apresentados na interface gráfica do Autopsy. Dos dois módulos analisados deste *plugin*, o 'Volatility Dump File Module' apresentou um erro ao ser executado, para o qual não se encontrou solução.

A funcionalidade '*Volatility Data Source Processor*' disponibilizada pelo Autopsy através do módulo '*Experimental*' permite que seja executado o Volatility sobre uma imagem de memória, permitindo assim a realização de uma análise à memória na ferramenta.

6

DESENVOLVIMENTO DO DATA SOURCE INGEST MODULE PARA AUTOPSY

Um dos objetivos no estudo do Volatility passa pelo pelo desenvolvimento de um módulo *Python 2.7.0* para Autopsy que execute o Volatility sobre imagens de memória de modo a apresentar no *software* Autopsy os resultados obtidos. Isto vai permitir ao investigador tentar procurar por evidências digitais que sejam relevantes para o caso em questão. O *data source ingest module* desenvolvido designa-se por RAMAnalysis.

Para o desenvolvimento do módulo, foi utilizado o *software* IntelliJ IDEA Community Edition versão 2020.1.4[1] desenvolvido pela JetBrains. Ao longo do desenvolvimento do módulo foram ainda utilizadas outras ferramentas, de acordo com as necessidades, como por exemplo a ferramenta JsonEditor versão 3.2.3[2] da CompuClever Systems Inc. para visualizar ficheiros JavaScript Object Notation (JSON).

O desenvolvimento do módulo ocorreu numa máquina que apresenta o sistema operativo *Microsoft Windows 10 Home* versão 10.0.17763 Compilação 17763, com 8 GiB de RAM e com um processador Intel(R) Core(TM) i7-7500U CPU @ 2.70GHz. Para testar o módulo desenvolvido foram utilizadas as imagens de memória 'ch2.dmp' e 'memdump.mem' já anteriormente apresentadas.

6.1 ARQUITETURA DO MÓDULO

A arquitetura do módulo permite perceber o funcionamento do módulo e a interação que irá existir entre os diversos componentes presentes na sua execução. Na Figura 59 encontra-se o diagrama relativo à arquitetura do *data source ingest module* desenvolvido.

1 https://www.jetbrains.com/idea/
2 https://www.microsoft.com/en-us/p/jsoneditor/9nwmlcgc25x3?activetab=pivot:overviewtab

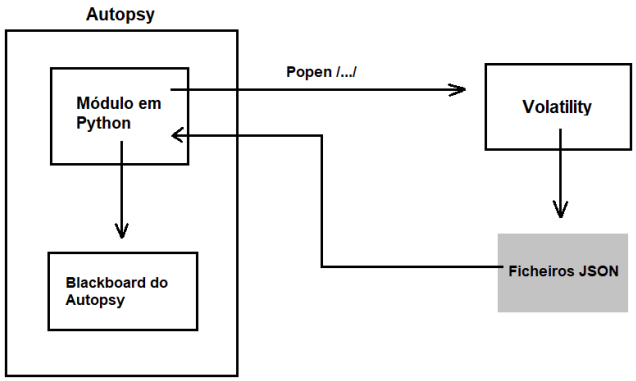


Figura 59: Diagrama com a arquitetura do módulo desenvolvido

O Volatility é executado através da função '`Popen`' e armazena os resultados num ficheiro em formato JSON. Deste modo, o *data source ingest module* não necessita de se preocupar em converter para JSON os resultados obtidos. De seguida, os resultados produzidos são enviados para o *data source ingest module* que é responsável por processar os dados presentes no ficheiro JSON. Por fim, o *data source ingest module* irá colocar os resultados devidamente no Autopsy.

6.2 FASE INICIAL

Tal como já foi referido no Capítulo 2, o Autopsy permite a integração de módulos que, por sua vez, permitem extender as funcionalidades da ferramenta.

Como se pretende analisar os dados presentes numa fonte de dados que seja adicionada ao caso no Autopsy, primeiramente, foi criado um *ingest module*, mais precisamente um *data source ingest module*.

A aprendizagem do processo de desenvolvimento de um *data source ingest module* para Autopsy passou pelo estudo de tutoriais para o efeito[3]. Essa abordagem tornou possível um melhor entendimento de conceitos próprios da programação de módulos para Autopsy, tais como, *blackboard*, *attribute* e *case*.

Alguns aspetos e características acerca do módulo e do seu funcionamento foram definidos e estruturados antes de se dar início à fase de desenvolvimento do mesmo.

Antes de mais, é importante ter em conta que o utilizador deste *plugin* terá que criar no seu dispositivo uma pasta com as imagens de memória a analisar.

3 https://www.autopsy.com/python-autopsy-module-tutorial-2-the-data-source-ingest-module/

No Autopsy irá definir como fonte de dados para o caso essa mesma pasta que anteriormente criou.

Este módulo desenvolvido ao ser executado irá criar automaticamente uma pasta na diretoria do caso do Autopsy que irá guardar os ficheiros gerados ao longo da execução do módulo. Essa mesma pasta que é criada designa-se por 'RAMAnalysisModule' e é criada na pasta 'ModuleOutput' do caso do Autopsy. No caso de o utilizador, no mesmo caso do Autopsy, executar novamente o módulo, a pasta 'RAMAnalysisModule' e o seu conteúdo são eliminados sendo criada uma nova pasta com a mesma designação.

Como já se verificou na exploração da ferramenta Volatility (Capítulo 5), para se executar o mesmo é necessário indicar o perfil adequado à imagem de memória em análise e indicar o mesmo no comando a ser executado. Assim, no sentido de simplificar a utilização do módulo e a análise que está a ser efetuada, o utilizador não precisa de escolher que perfil aplicar uma vez que é o próprio módulo que irá obter o perfil.

Para o desenvolvimento do *data source ingest module* utilizou-se o *template* do mesmo disponibilizado na página do GitHub do Autopsy[4] (designação do ficheiro do *template*: `dataSourceIngestModule.py`.

6.3 INTERFACE GRÁFICA

Os *data source ingest modules* permitem que seja criada uma interface gráfica que permite configurar alguns aspetos antes de iniciar a execução do módulo.

O módulo desenvolvido nesta dissertação apresenta uma interface gráfica que apresenta algumas opções para o utilizador definir de modo a executar o módulo de análise da memória.

Primeiramente, o utilizador terá que indicar a localização do ficheiro executável do Volatility.

De seguida, o utilizador terá que selecionar o(s) *plugin(s)* que pretende que sejam executados de modo a apresentar resultados no *blackboard* do Autopsy. A designação dos *plugins* do Volatility que podem ser executados neste módulo encontram-se num ficheiro `.json` que o módulo processa (ficheiro `plugins.json`). Os *plugins* presentes nesta lista são os apresentados no *output* do comando '`volatility.exe -h`', sendo

4 https://github.com/sleuthkit/autopsy/tree/develop/pythonExamples

que a esses foi adicionado o plugin `netscan` apresentado em [5]. De modo a que fosse possível apresentar as designações dos *plugins* foi necessário colocá-las numa `JList`.

Na interface existe ainda uma outra lista com os *plugins* do Volatility cujo o *output* é guardado em formato `.dot`. A linguagem DOT é uma linguagem de descrição de gráficos (Wikipedia, 2020). Esse mesmo ficheiro DOT é convertido, se possível, para um ficheiro com extensão `.png`. Tal como na lista de *plugins* anterior, esta lista corresponde a uma `JList`.

Na Figura 60 encontra-se a interface gráfica do módulo criado nesta dissertação, sendo que já se encontram definidos alguns parâmetros.

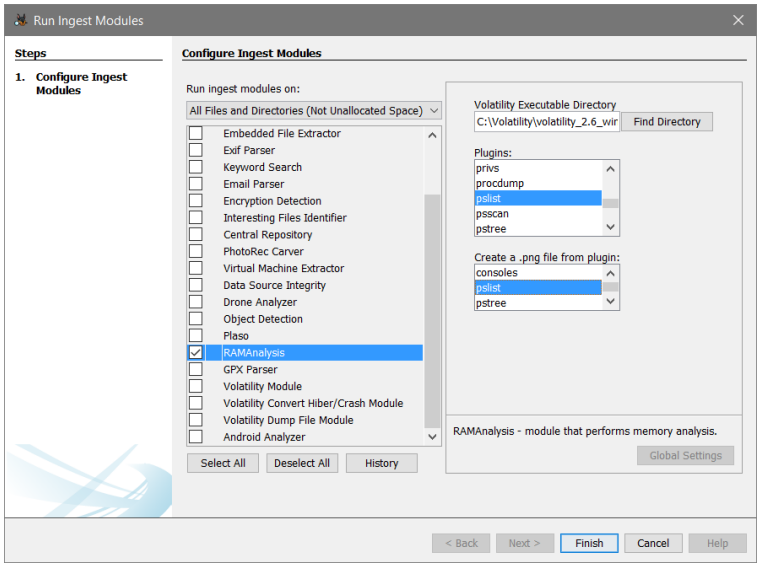


Figura 60: Interface gráfica do *data source ingest module*

Após o utilizador definir todos estes aspetos, o módulo executa, para cada imagem de memória, cada *plugin* selecionado, criando um ficheiro JSON para cada conjunto de resultados de um *plugin* acerca de uma dada imagem de memória.

Em certos casos é necessário definir parâmetros adicionais ao *plugin* do Volatility a executar. Para tal, neste módulo existe na interface gráfica o campo '*Additional Parameters*'. Contudo, só é possível definir parâmetros adicionais se o utilizador somente selecionar um *plugin* da primeira lista de *plugins* presente na interface gráfica do módulo (Figura 61). Caso contário, o campo '*Additional Parameters*' fica oculto, tal como surge na Figura 60 em que são selecionados *plugins* de ambas as listas. Na Figura 61 encontra-se a interface gráfica do módulo com um só *plugin* selecionado e o respetivo parâmetro adicional definido.

5 https://github.com/volatilityfoundation/volatility/wiki/Command-Reference

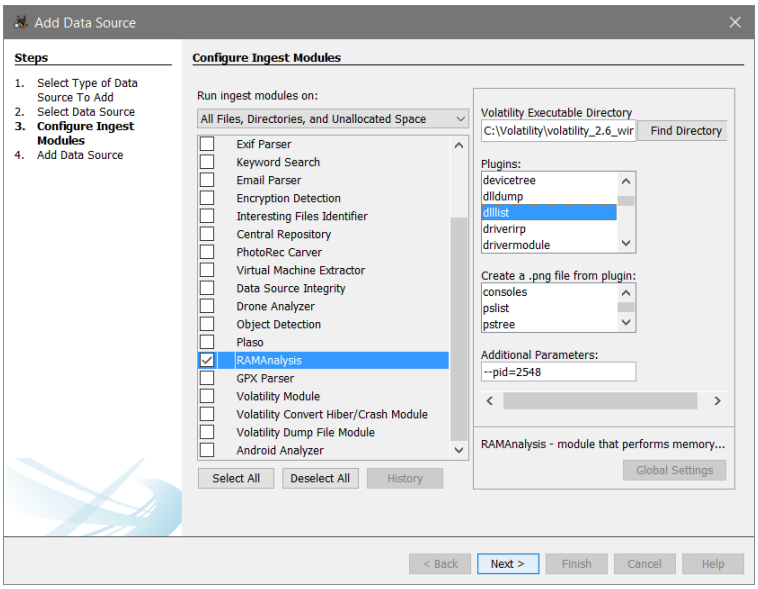


Figura 61: Interface gráfica do *data source ingest module* com um parâmetro adicional definido

6.4 ASPETOS INICIAIS DO CÓDIGO

O *data source ingest module* desenvolvido apresenta três classes, mais precisamente:

- '`RAMAnalysisDataSourceIngestModuleFactory`';
- '`RAMAnalysisDataSourceIngestModule`';
- '`RAMAnalysisDataSourceIngestModuleUISettingsPanel`'.

A classe '`RAMAnalysisDataSourceIngestModuleFactory`' é onde se encontram definidos, por exemplo, o nome do módulo e a sua descrição.

A classe '`RAMAnalysisDataSourceIngestModule`' é uma classe que possui dois métodos relevantes para o funcionamento do módulo, nomeadamente, o método '`startUp()`' que é onde o módulo é inicializado e o método '`process()`' que é onde se faz a análise em si, ou seja, onde se concentra grande parte do código do módulo. Relativamente ao método '`startUp()`', no *data source ingest module* este método é o responsável por obter cada um dos parâmetros definidos na User Interface (UI) e que, por sua vez, foram armazenados nas *settings*.

Já a classe '`RAMAnalysisDataSourceIngestModuleUISettingsPanel`' é a classe onde se encontra o código relativo à construção da UI do módulo. Os elementos que se pretendem adicionar à UI são definidos no método '`initComponents()`'. À

medida que foram adicionados alguns elementos à UI foram definidos métodos relativos aos mesmos nesta classe.

Um dos passos iniciais consistiu em se obter as imagens de memória a analisar pelo Volatility. Assim, foi necessário obter o *file manager* relativo ao caso em questão e aplicar a este o método '`findFiles()`'. Este método num dos seus parâmetros permite restringir a extensão dos ficheiros que serão obtidos. Por exemplo, colocando '`%.jpg%`' como parâmetro do método só iriam ser obtidos ficheiros com extensão `.jpg`. Dado o facto de existirem várias extensões possíveis para imagens de memória e de modo a não restringir o tipo de imagem de memória aceite pelo módulo, neste módulo o método '`findFiles()`' obtem todos os ficheiros com qualquer extensão, tendo sido definido o parâmetro '`%.%`' no método '`findFiles()`'. Este método devolve uma lista de objetos *AbstractFile* (A. D. Forensics, 2015).

Um outro passo inicial no módulo consiste na criação de uma pasta na diretoria 'ModuleOutput' que armazene os ficheiros que serão criados. Assim, a nova pasta será criada através da função '`os.mkdir()`' cujo parâmetro de entrada corresponde ao caminho da pasta 'ModuleOutput' concatenado com o nome da nova pasta (Foundation, 2020b). Contudo, tal como já foi referido, num mesmo caso do Autopsy a cada nova execução do módulo a pasta 'RAMAnalysisModule' é eliminada sendo que para isso é necessário verificar se a pasta já existe através da função '`os.path.isdir()`' (Foundation, 2020c). No caso da diretoria já existir, esta será eliminada através do método '`shutil.rmtree()`' sendo que para se utilizar este método é necessário efetuar o *import* do módulo `shutil` (Foundation, 2020d).

6.5 EXECUÇÃO DO VOLATILITY NO MÓDULO

De modo a executar o Volatility através do módulo que vai ser desenvolvido é necessário utilizar a função '`Popen`' (disponível através do módulo `os`). Esta função permite um interação com o sistema operativo tendo como objetivo chamar/executar outros programas através de um ficheiro `.py`.

Neste módulo desenvolvido, a função '`Popen`' é executada para cada *plugin* selecionado, para cada imagem de memória, sendo que permitirá a execução do Volatility e a criação dos ficheiros com os resultados obtidos. Por exemplo, para descobrir o perfil adequado à imagem de memória e até para a criação dos ficheiros `.dot` o módulo tem que recorrer à função '`Popen`'.

Como se pode verificar na Listagem 22, o comando do Volatility a executar encontra-se entre parenteses retos ([ ]) e os parâmetros pertencentes ao comando encontram-se separados por vírgula. Note-se que os parênteses retos, no contexto da linguagem *Python*, representam uma lista anónima. Na Listagem 22, '`file.getLocalAbsPath()`' devolve o caminho da imagem de memória em análise e '`selecPlugin`' corresponde ao *plugin* a ser executado. O parâmetro '`--output=json`' define que os resultados serão apresentados em formato JSON enquanto que '`--output-file`' permite especificar o ficheiro que guardará o *output*.

Listagem 22: Parte do código do módulo com a execução do Volatility a partir da função '`Popen`'

```
1      (...)
2      for file in files:
3          (...)
4          try:
5              pipe = Popen([self.volatilityEXEPathFile, "-f",
               ↪  file.getLocalAbsPath(),
6                          selecPlugin, "--output=json", "--output-file=" +
                           ↪  pathAndNameJsonFile], stdout=PIPE, stderr=PIPE)
7              (...)
8          (...)
```

6.6 RESULTADOS DA EXECUÇÃO DOS *plugins*

O Volatility permite obter os resultados da execução dos seus *plugins* nos formatos `text`, `dot`, `html`, `xlsx`, `json`, `quick` e `sqlite`. Esta característica do Volatility está disponível desde a versão 2.5 do mesmo e permite que o utilizador escolha um formato específico para os resultados de acordo com as suas necessidades (Volatility Foundation, 2016). Para verificar que formatos são suportados por cada *plugin* deve-se executar o seguinte comando (Volatility Foundation, 2016):

```
volatilityExeFile nomePlugin -h
```

No comando acima, 'volatilityExeFile' corresponde ao ficheiro executável do Volatility, 'nomePlugin' ao *plugin* acerca do qual se pretende obter mais informações e '-h' a opção do Volatility. Na Listagem 23 encontra-se a execução do comando anteriormente apresentado, sendo que no caso da Listagem 23 pretende-se obter aos opções de formato de *output* existentes para o *plugin* `pslist`, que, por sua vez, se encontram na linha que começa por '*Module Output Options*: (...)'. Verifica-se então que o *plugin* `pslist` suporta vários formatos de *output*, como, por exemplo, os formatos `.dot`, `.html` e `.json`.

Listagem 23: *Output* da execução do comando '`volatilityExeFile pslist -h`'

```
C:\Volatility\volatility_2.6_win64_standalone>volatility_2.6_win64_standalone.exe
↪  e pslist
↪  -h
Volatility Foundation Volatility Framework 2.6
Usage: Volatility - A memory forensics analysis platform.

Options:
  -h, --help            list all available options and their default values.
                        Default values may be set in the configuration file
                        (/etc/volatilityrc)
  --conf-file=.volatilityrc
                        User based configuration file
  -d, --debug           Debug volatility
  --plugins=PLUGINS     Additional plugin directories to use (semi-colon
                        separated)
  --info                Print information about all registered objects
  --cache-directory=C:\Users\santo/.cache\volatility
                        Directory where cache files are stored
  --cache               Use caching
  --tz=TZ               Sets the (Olson) timezone for displaying timestamps
                        using pytz (if installed) or tzset
  -f FILENAME, --filename=FILENAME
                        Filename to use when opening an image
  --profile=WinXPSP2x86
                        Name of the profile to load (use --info to see a list
                        of supported profiles)
  -l LOCATION, --location=LOCATION
                        A URN location from which to load an address space
  -w, --write           Enable write support
  --dtb=DTB             DTB Address
  --shift=SHIFT         Mac KASLR shift address
  --output=text         Output in this format (support is module specific, see
                        the Module Output Options below)
  --output-file=OUTPUT_FILE
                        Write output in this file
  -v, --verbose         Verbose information
  -g KDBG, --kdbg=KDBG  Specify a KDBG virtual address (Note: for 64-bit
                        Windows 8 and above this is the address of
                        KdCopyDataBlock)
  --force               Force utilization of suspect profile
  --cookie=COOKIE       Specify the address of nt!ObHeaderCookie (valid for
                        Windows 10 only)
  -k KPCR, --kpcr=KPCR  Specify a specific KPCR address
  -o OFFSET, --offset=OFFSET
                        EPROCESS offset (in hex) in the physical address space
  -p PID, --pid=PID     Operate on these Process IDs (comma-separated)
  -n NAME, --name=NAME  Operate on these process names (regex)
  -P, --physical-offset
                        Display physical offsets instead of virtual

Module Output Options: dot, greptext, html, json, sqlite, text, xlsx

----------------------------------
Module PSList
----------------------------------
Print all running processes by following the EPROCESS lists
```

Nesta dissertação foi efetuado um levantamento dos formatos suportados pelos *plugins* do Volatility, através da execução do comando anteriormente apresentado. Verificou-se que, por exemplo, no caso do *plugin* `pstree`, o *output* do comando '`volatilityExeFile pstree -h`' (Listagem 24) informa que este *plugin* suporta

as opções de formato de *output* **dot**, **greptext**, **html**, **json**, **sqlite**, **text** e **xlsx**. Contudo, nesta dissertação, ao tentar obter o *output* do **pstree** num ficheiro JSON, é apresentada a mensagem de erro '*JSON output for trees has not yet been implemented*' sendo criado um ficheiro JSON vazio (Listagem 25).

Listagem 24: *Output* da execução do comando '**volatilityExeFile pstree -h**'

```
C:\Volatility\volatility_2.6_win64_standalone>volatility_2.6_win64_standalone.exe pstree -h
Volatility Foundation Volatility Framework 2.6
Usage: Volatility - A memory forensics analysis platform.

Options:
  -h, --help            list all available options and their default values.
                        Default values may be set in the configuration file
                        (/etc/volatilityrc)
  --conf-file=.volatilityrc
                        User based configuration file
  -d, --debug           Debug volatility
  --plugins=PLUGINS     Additional plugin directories to use (semi-colon
                        separated)
  --info                Print information about all registered objects
  --cache-directory=C:\Users\santo/.cache\volatility
                        Directory where cache files are stored
  --cache               Use caching
  --tz=TZ               Sets the (Olson) timezone for displaying timestamps
                        using pytz (if installed) or tzset
  -f FILENAME, --filename=FILENAME
                        Filename to use when opening an image
  --profile=WinXPSP2x86
                        Name of the profile to load (use --info to see a list
                        of supported profiles)
  -l LOCATION, --location=LOCATION
                        A URN location from which to load an address space
  -w, --write           Enable write support
  --dtb=DTB             DTB Address
  --output=text         Output in this format (support is module specific, see
                        the Module Output Options below)
  --output-file=OUTPUT_FILE
                        Write output in this file
  -v, --verbose         Verbose information
  --shift=SHIFT         Mac KASLR shift address
  -g KDBG, --kdbg=KDBG  Specify a KDBG virtual address (Note: for 64-bit
                        Windows 8 and above this is the address of
                        KdCopyDataBlock)
  --force               Force utilization of suspect profile
  -k KPCR, --kpcr=KPCR  Specify a specific KPCR address
  --cookie=COOKIE       Specify the address of nt!ObHeaderCookie (valid for
                        Windows 10 only)

Module Output Options: dot, greptext, html, json, sqlite, text, xlsx

---------------------------------
Module PSTree
---------------------------------
Print process list as a tree
```

Listagem 25: Erro na obtenção do ficheiro JSON com o *output* do *plugin* `pstree`

```
C:\Volatility\volatility_2.6_win64_standalone>volatility_2.6_win64_standalone.exe
↪  e -f ch2.dmp --profile=Win7SP1x86_23418 pstree --output=json
↪  --output-file=fichPstree.json
Volatility Foundation Volatility Framework 2.6
Outputting to: fichPstree.json
ERROR   : volatility.debug    : JSON output for trees has not yet been
↪  implemented
```

Algo semelhante ocorre com o *plugin* `psscan` pois apesar de o *output* do comando indicar que suporta os formato de *output* `dot`, `greptext`, `html`, `json`, `sqlite`, `text` e `xlsx`, nos testes realizados não foi possível obter o *output* do *plugin* nos formatos `json` e `dot`.

No *plugin* para o Autopsy desenvolvido nesta dissertação, os resultados da execução dos *plugins* são guardados em ficheiros com o formato JSON. Assim, após o utilizador selecionar os *plugins* cujos resultados pretende visualizar, o módulo terá que processar os ficheiros JSON gerados.

O JSON é uma formatação leve de troca de dados e é completamente independente de linguagem. O ficheiro no formato JSON apresenta texto legível no formato de pares chave-valor, pares estes separados por uma vírgula. (JSONOrg, 2020) Comparativamente com o Extensible Markup Language (XML), o JSON permite um processo de análise mais rápido e é menos detalhado, permitindo uma simplificação do código (Joshi, 2017).

No caso dos ficheiros JSON criados pelo Volatility, estes apresentam duas *keys*, nomeadamente a *key* '*rows*' que agrega os resultados obtidos pela execução do *plugin* em causa e a *key* '*columns*' que contem as designações das colunas do *output* do *plugin*.

6.6.1 *Processamento dos Ficheiros JSON*

De modo a que os dados presentes nos ficheiros JSON sejam apresentados no *blackboard* é necessário que o módulo realize um conjunto de passos. Assim, primeiro, o módulo irá abrir cada ficheiro JSON gerado em modo leitura com recurso ao método '`open()`' (Foundation, 2020a). De seguida, efetua-se a leitura do conteúdo através do método '`json.load()`'.

Com a análise do *output* do *plugin* `pslist` e comparando os resultados deste na interface do Autopsy com o conteúdo presente no ficheiro JSON correspondente

verificou-se que em alguns casos a coluna '*Offset*' no ficheiro JSON tem valores, mas no Autopsy esses mesmos valores não apareciam. Deste modo, ao ser adicionado o parâmetro '`parse_int=str`' na função '`json.load()`' que converte os valores do tipo `int` para o tipo `str` tornou-se possível visualizar todos os valores de '*Offset*' (Foundation, 2019).

Para evitar erros na execução do módulo no processamento dos ficheiros JSON foram adicionadas algumas verificações, tais como, se a *key* '*rows*' do ficheiro está vazia e se o ficheiro JSON criado está vazio.

6.6.2 *Colocação dos Resultados no Blackboard*

Para cada imagem de memória, cada ficheiro JSON corresponde a um *plugin* distinto com valores cujos tipos e características não se encontram definidos no Autopsy por omissão. Assim, terá que ser criado um novo tipo de artefacto referente a cada *plugin*. A criação de um novo tipo de artefacto ocorre através do método '`addArtifactType(nomeArtifType, descricao)`', em que 'nomeArtifType' é a designação do novo tipo de artefacto e 'descricao' é a descrição do novo tipo de artefacto. Neste módulo o nome do tipo de artefacto segue o padrão 'TSK_VOLAT_nomePlugin'.

Cada *plugin* apresenta no seu *output* um conjunto de colunas que armazenam dados de diferentes tipos, sendo que no Autopsy cada coluna do *output* de um *plugin* corresponde a um atributo distinto. Assim, antes de se criar os tipos de atributos foi necessário obter o tipo de dados de cada coluna e armazená-los num *array*. Para tal, o módulo percorre todos os valores numa dada coluna até encontrar uma linha da coluna que apresente um valor definido, recorrendo-se à função '`strip()`' para verificar se o valor em questão está vazio ou com espaços em branco. A esse mesmo valor encontrado é extraído o seu tipo de dados com aplicação de '`type(value).__name __`', sendo 'value' o valor da coluna em análise. No caso de todos os valores presentes na coluna estarem vazios, o tipo de dados que o módulo define para aquela coluna é 'NOTYPE'.

Para a criação dos atributos relativos ao tipo de artefacto em causa é necessário utilizar o método '`addArtifactAttributeType(nomeAtributo, tipoAtributo, coluna)`,' em que 'nomeAtributo' corresponde ao nome do atributo, 'tipoAtributo' ao tipo do atributo e o parâmetro 'coluna' à designação da coluna em questão. O valor do parâmetro 'tipoAtributo' corresponde a um dos *public attributes* da `enum` '`BlackboardAttribute.TSK_BLACKBOARD_ATTRIBUTE_VALUE_TYPE`' do

Autopsy. Deste modo, no caso de o tipo de valor de uma dada coluna ser `LONG` ou `INT`, o valor de ‘tipoAtributo‘ será:

‘`BlackboardAttribute.TSK_BLACKBOARD_ATTRIBUTE_VALUE_TYPE.LONG`‘

Já se o tipo de valor da coluna for `STR`, `UNICODE` ou `NOTYPE` o valor do parâmetro ‘tipoAtributo‘ será:

‘`BlackboardAttribute.TSK_BLACKBOARD_ATTRIBUTE_VALUE_TYPE.STRING`‘

Com o objetivo de apresentar os resultados obtidos no *blackboard* do Autopsy é necessário para cada linha do *output* criar um novo artefacto com recurso ao método ‘`newArtifact(IdTipoArtefacto)`‘ em que o parâmetro ‘IdTipoArtefacto‘ corresponde ao ID do tipo de artefacto anteriormente criado. Com o método `addAttribute()` será, por fim, possível adicionar os valores de cada coluna da linha em questão.

6.7 DETEÇÃO DO PERFIL ATRAVÉS DO MÓDULO

Na versão 2.6 do Volatility, para se executar um *plugin* sem ser os *plugins* `help` e `imageinfo`, é necessário colocar sempre o perfil adequado à imagem de memória em causa. Assim, deve-se, primeiramente, executar o *plugin* `imageinfo` para encontrar o perfil adequado e, de seguida, executar os *plugins* pretendidos com sucesso. Já o Volatility 3.0 *Public Beta* deteta automaticamente o perfil, não existindo a necessidade de encontrar o perfil adequado antes de se iniciar a análise em si.

Deste modo, no sentido de facilitar a utilização do módulo por parte do utilizador, aplicou-se uma forma de o módulo descobrir automaticamente o perfil adequado, sem existir necessidade de ser o próprio utilizador a descobri-lo e indicá-lo.

Quando o utilizador executar o módulo, uma das primeiras tarefas deste é executar o *plugin* `imageinfo` sobre a imagem de memória através da função ‘`Popen`‘, guardando os resultados obtidos num ficheiro JSON temporário na pasta do caso do Autopsy na diretoria:

‘`ModuleOutput\RAMAnalysisModule`‘

A designação-base do ficheiro é ‘IDFichImagemMemoria _imageinfoTemp.json‘, em que ‘IDFichImagemMemoria‘ corresponde ao ID do ficheiro da imagem de memória em análise no momento. Na Listagem 26 está o conteúdo de um ficheiro JSON obtido neste mesmo processo, sendo que na primeira linha em ‘*rows*‘ encontram-se

os perfis e na primeira linha de '*columns*' a designação da coluna que no *output* apresenta os perfis adequados.

Listagem 26: Conteúdo do ficheiro JSON obtido com a execução do *plugin* `imageinfo`

```
{
  "rows": [
    [
      "Win7SP1x86_23418, Win7SP0x86, Win7SP1x86",
      "IA32PagedMemoryPae (Kernel AS)",
      "FileAddressSpace (C:\\Users\\santo\\Desktop\\imagensMem\\ch2.dmp)",
      "PAE",
      1593344,
      2190646248,
      1,
      0,
      2190650368,
      4292804608,
      "2013-01-12 16:59:18 UTC+0000",
      "2013-01-12 17:59:18 +0100"
    ]
  ],
  "columns": [
    "Suggested Profile(s)",
    "AS Layer1",
    "AS Layer2",
    "PAE type",
    "DTB",
    "KDBG",
    "Number of Processors",
    "Image Type (Service Pack)",
    "KPCR for CPU 0",
    "KUSER_SHARED_DATA",
    "Image date and time",
    "Image local date and time"
  ]
}
```

A estrutura do ficheiro JSON que é criado para o *plugin* `imageinfo` será sempre a mesma, independentemente da imagem de memória em análise, mantendo-se assim as designações das *columns* do ficheiro. Deste modo, o módulo manipula o ficheiro JSON criado, guardando numa variável o valor presente na primeira linha de '*rows*'. Esse mesmo valor armazenado na variável corresponde aos perfis que se podem aplicar, sendo que essa variável irá armazenar um valor do tipo *unicode*. No caso da Listagem 26 acima seria guardado nessa tal variável o valor 'Win7SP1x86_23418, Win7SP0x86, Win7SP1x86'.

Dado que os perfis estão no formato texto, para se obter o primeiro perfil que é sugerido, esse mesmo valor texto é transformado numa lista. Essa mesma transformação ocorre graças à utilização do método '`split()`' que divide uma *string* numa lista no separador específicado, por exemplo, a vírgula (Asipu, 2020). Deste modo, o primeiro perfil sugerido é o valor na posição zero da lista obtida anteriormente.

Após a descoberta do perfil adequado com sucesso, o ficheiro JSON criado com os resultados da execução do *plugin* `imageinfo` é eliminado.

6.8 CRIAÇÃO DOS FICHEIROS .DOT

O módulo permite ainda a criação de ficheiros com extensão `.dot` que corresponde a uma extensão de ficheiros com gráficos DOT.

Numa análise de dados presentes numa imagem de memória com recurso ao Volatility, no caso de certos *outputs* visualizar os mesmos em forma de gráfico pode ser um auxílio para o investigador.

Assim, e tal como já foi referido, na interface gráfica do módulo o utilizador seleciona os *plugins* dos quais pretende obter os resultados num ficheiro `.dot`. É através da função '`Popen`' que será executado o Volatility e no comando especificado é colocado o parâmetro '`--output=dot`'. O ficheiro `.dot` criado é armazenado na pasta '`ModuleOutput\RAMAnalysisModule`'. Nesta dissertação a visualização dos ficheiros `.dot` foi efetuada através do Graphviz[6], mais especificamente através da ferramenta 'gvedit.exe'.

Na Figura 62 encontra-se a visualização de um ficheiro `.dot` na interface da ferramenta 'gvedit.exe'. Esse mesmo ficheiro refere-se ao *output* do *plugin* `pslist`.

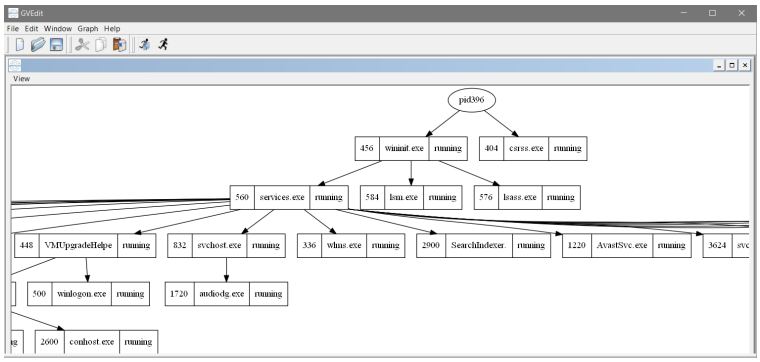


Figura 62: Visualização na ferramenta gvedit.exe do conteúdo do ficheiro `.dot` relativo ao *plugin* `pslist`

Para além do referido, este módulo, quando possível, também converte automaticamente os ficheiros no formato `.dot` para ficheiros com o formato `.png` o que facilita a visualização do conteúdo dos ficheiros `.dot`. Os ficheiros `.png` obtidos são armazenados na pasta '`ModuleOutput\RAMAnalysisModule`'.

6 http://www.graphviz.org/

6.9 FICHEIROS JSON AUXILIARES PARA O REPORT MODULE

No *data source ingest module* desenvolvido, para cada imagem de memória presente no caso do Autopsy, são criados, se possível, quatro ficheiros JSON com os resultados dos *plugins* `netscan`, `envars`, `hivelist` e `hashdump`. Estes quatro ficheiros são ficheiros auxiliares para a apresentação de informação no *report module* ao utilizador.

A execução destes quatro *plugins* do Volatility e a criação dos ficheiros JSON de cada um deles ocorre através da função '`Popen`'. Relativamente aos *plugins* `netscan`, `envars` e `hashdump`, no *data source ingest module* só é criado o ficheiro JSON com os resultados, não existindo qualquer processamento dos dados presentes no ficheiro.

Já no que toca ao *plugin* `hivelist`, o ficheiro JSON criado é processado no *data source ingest module* uma vez que existem dados presentes no *output* do *plugin* `hivelist` necessários para executar o *plugin* `hashdump`. O objetivo da execução do *plugin* `hashdump` passa pela obtenção das *cached domain credentials* guardadas na *Registry* e, para tal, é necessário especificar no comando o endereço virtual da *hive* `SYSTEM` (parâmetro '`-y`') e o endereço virtual da *hive* `SAM` (parâmetro '`-s`'), endereços estes que podem ser encontrados no *output* do *plugin* `hivelist`.

6.10 CONTEÚDO DA PASTA COM OS FICHEIROS RESULTANTES

Os ficheiros criados pelo *data source ingest module* ao longo da sua execução são guardados na seguinte pasta:

'`ModuleOutput\RAMAnalysisModule`'

Na Figura 63 encontra-se o conteúdo da pasta anteriormente referida, após a execução do *data source ingest module* na qual na interface gráfica, tanto na primeira como na segunda lista de *plugins*, foi unicamente selecionado o *plugin* `pslist`. Nesta execução foi unicamente adicionada uma imagem de memória ('ch2.dmp') à pasta adicionada como fonte de dados do caso.

Nome	Data de modificação	Tipo	Tamanho
3_envars_AuxFile.json	30/10/2020 18:54	Ficheiro JSON	79 KB
3_hashdump_AuxFile.json	30/10/2020 18:54	Ficheiro JSON	1 KB
3_hivelist_AuxFile.json	30/10/2020 18:54	Ficheiro JSON	1 KB
3_netscan_AuxFile.json	30/10/2020 18:53	Ficheiro JSON	12 KB
3_pslist.dot	30/10/2020 18:53	Modelo do Micros...	5 KB
3_pslist.json	30/10/2020 18:54	Ficheiro JSON	5 KB
3_pslist.png	30/10/2020 18:53	Ficheiro PNG	180 KB

Figura 63: Conteúdo da pasta do módulo em que os ficheiros resultantes são guardados

Como se pode ver os ficheiros iniciam-se por um número diferente, número este que corresponde ao ID da imagem de memória. Verifica-se a existência de quatro ficheiros JSON auxiliares para o *report module*, nomeadamente:

- `3_envars_AuxFile.json`;
- `3_hashdump_AuxFile.json`;
- `3_hivelist_AuxFile.json`;
- `3_netscan_AuxFile.json`.

Verifica-se ainda a presença do ficheiro JSON com os resultados obtidos com a execução do *plugin* `pslist` (ficheiro `3_pslist.json`). Relativamente à criação dos ficheiros `.dot`, foram criados dois ficheiros, nomeadamente, o ficheiro `3_pslist.dot` que possui o gráfico DOT do *plugin* `pslist` e o ficheiro `3_pslist.png` que resultou da conversão do ficheiro `.dot` para `.png`.

Na Figura 64 encontra-se ainda parte do resultado da execução do módulo referida no *blackboard* do Autopsy.

Source File	S	C	Offset(V)	Name	PID	PPID	Thds	Hnds	Sess	Wow64	Start	Exit	Data Source
ch2.dmp			2274855800	System	4	0	103	3257	-1	0	2013-01-12 16:38:09 UTC+0000		LogicalFileSet1
ch2.dmp			2294541632	smss.exe	308	4	2	29	-1	0	2013-01-12 16:38:09 UTC+0000		LogicalFileSet1
ch2.dmp			2301230400	csrss.exe	404	396	9	469	0	0	2013-01-12 16:38:14 UTC+0000		LogicalFileSet1
ch2.dmp			2301280952	wininit.exe	456	396	3	77	0	0	2013-01-12 16:38:14 UTC+0000		LogicalFileSet1
ch2.dmp			2295347712	csrss.exe	468	448	10	471	1	0	2013-01-12 16:38:14 UTC+0000		LogicalFileSet1
ch2.dmp			2301422912	winlogon.exe	500	448	3	111	1	0	2013-01-12 16:38:14 UTC+0000		LogicalFileSet1
ch2.dmp			2304939200	services.exe	560	456	6	205	0	0	2013-01-12 16:38:16 UTC+0000		LogicalFileSet1
ch2.dmp			2305042360	lsass.exe	576	456	6	566	0	0	2013-01-12 16:38:16 UTC+0000		LogicalFileSet1
ch2.dmp			2304964584	lsm.exe	584	456	10	142	0	0	2013-01-12 16:38:16 UTC+0000		LogicalFileSet1
ch2.dmp			2304962608	svchost.exe	692	560	10	353	0	0	2013-01-12 16:38:21 UTC+0000		LogicalFileSet1
ch2.dmp			2306563104	svchost.exe	764	560	7	263	0	0	2013-01-12 16:38:23 UTC+0000		LogicalFileSet1
ch2.dmp			2306888736	svchost.exe	832	560	19	435	0	0	2013-01-12 16:38:23 UTC+0000		LogicalFileSet1
ch2.dmp			2307205400	svchost.exe	904	560	17	409	0	0	2013-01-12 16:38:24 UTC+0000		LogicalFileSet1
ch2.dmp			2307305520	svchost.exe	928	560	26	869	0	0	2013-01-12 16:38:24 UTC+0000		LogicalFileSet1
ch2.dmp			2307461544	svchost.exe	1084	560	10	257	0	0	2013-01-12 16:38:26 UTC+0000		LogicalFileSet1
ch2.dmp			2307598224	svchost.exe	1172	560	15	475	0	0	2013-01-12 16:38:27 UTC+0000		LogicalFileSet1
ch2.dmp			2307553384	AvastSvc.exe	1220	560	66	1180	0	0	2013-01-12 16:38:28 UTC+0000		LogicalFileSet1

Figura 64: *Blackboard* do Autopsy com a apresentação dos resultados da execução efetuada

6.11 ANÁLISE COMPARATIVA DOS RESULTADOS APRESENTADOS

Após o desenvolvimento do *data source ingest module* foi efetuada uma análise comparativa entre os resultados obtidos por cada *plugin* na *command line*, no ficheiro JSON e os apresentados na interface do Autopsy.

Como seria de esperar, entre os resultados que surgem na *command line* e os apresentados no Autopsy existe uma ligeira diferença em termos de organização e apresentação do conteúdo apresentado.

Ao longo da análise de resultados verificou-se que existem casos como o do *plugin* `iehistory` em que apesar de na *command line* ser apresentado dados no *output*,

quando se cria um ficheiro JSON para os resultados do *plugin* o ficheiro encontra-se vazio. Algo semelhante ocorre com os *plugins* `pstree` e `psscan`.

Em alguns casos, na *command line* a execução de certos *plugins* apresenta o erro '*ERROR: volatility.debug: This command does not support the profile (...)*'. Na *command line* verificou-se que, para as imagens de memória utilizadas para teste, trocando o perfil colocado por outros sugeridos pelo *plugin* `imageinfo` o erro permanece para cada um dos diferentes perfis, não criando qualquer ficheiro JSON. Para estes casos, o *data source ingest module* desenvolvido informa o utilizador que ocorreu um erro na execução do Volatility e, consequentemente, não foi possível criar o ficheiro JSON com os resultados. Ao longo da análise dos resultados, um dos *plugins* em que se verificou este caso foi no *plugin* `connscan`.

Com esta análise verificou-se que alguns dos valores que na *command line* surgem em formato hexadecimal, no ficheiro JSON e, consequentemente, na interface do Autopsy, surgem em formato decimal. Alguns dos *plugins* em que ocorre este tipo de conversão são no *plugin* `pslist` (coluna '*Offset*'), *plugin* `hivelist` (colunas '*Virtual*' e '*Physical*') e no *plugin* `envars` (coluna '*Block*').

Constatou-se ainda nos resultados do *plugin* `pslist` na *command line* que a coluna '*Session*' têm alguns valores '`---`', contudo no ficheiro JSON e na interface do Autopsy o valor '`---`' corresponderá a '-1'.

Comparando alguns valores constatou-se uma situação particular com a coluna '*Offset*' do *plugin* `pslist`. Na *command line* os valores da coluna '*Offset*' encontram-se em formato hexadecimal enquanto que os valores de '*Offset*' presentes no ficheiro JSON e no *blackboard* do Autopsy estão em formato decimal. Todavia, constatou-se que, em alguns casos, os últimos quatro dígitos do valor de '*Offset*' apresentado no Autopsy e do valor presente no ficheiro JSON são alterados, sendo que o valor que se encontra corretamente convertido de hexadecimal para decimal é o apresentado no Autopsy. A Listagem 27 é relativa ao ficheiro JSON do *plugin* `pslist` da imagem de memória 'memdump.mem', mais precisamente, aos dados relativos ao processo com PID 4 que possui um '*Offset*' com o valor '18446675957302902000'. Contudo, o referido processo com PID 4 apresentado no *blackboard* do Autopsy tem como valor de '*Offset*' '18446675957302903416' (Figura 65). Sabendo-se que na *command line* o seu valor de '*Offset*' em hexadecimal é '0xffffc20c69c74278', efetuando a conversão deste valor para decimal verifica-se que o valor correto é o apresentado no Autopsy.

Listagem 27: Dados relativos ao processo com PID 4 da imagem de memória 'memdump.mem' presentes no ficheiro JSON do *plugin* `pslist`

```
{
  "rows": [
    [
      18446675957302902000,
      "",
      4,
      0,
      1786533608,
      0,
      -1,
      0,
      "-",
      ""
    ],
```

Source File	S	C	Offset(V)	Name	PID	PPID	Thds	Hnds	Sess	Wow64	Start	Exit
memdump.mem			18446675957302903416		4	0	1786533608	0	-1	0	-	

Figura 65: Apresentação do processo com PID 4 da imagem de memória 'memdump.mem' no Autopsy

Foi ainda efetuada uma análise aos resultados obtidos com a definição do campo 'Parâmetros Adicionais'. Para executar o *plugin* `dumpcerts` - que efetua o *dump* de chaves privadas RSA e de chaves públicas SSL - é necessário definir a diretoria em que serão guardados os ficheiros que serão obtidos. Assim, a diretoria em causa é definida no campo 'Parâmetros Adicionais'. Como se pode ver, relativamente à imagem de memória 'ch2.dmp', surge no *blackboard* os devidos artefactos (Figura 66) e na pasta que foi indicada são colocados os ficheiros `.cert` (Figura 67), resultado este similar com o obtido na *command line*.

Source File	S	C	Pid	Process	Address	Type	Length	File	Subject	Cert
ch2.dmp			468	csrss.exe	2116994	_X509_PUBLIC_CERT	964	468-204d82.crt		308203c43082032da00...
ch2.dmp			468	csrss.exe	20892242	_X509_PUBLIC_CERT	1042	468-13eca52.crt		308204123082O2faa00...
ch2.dmp			468	csrss.exe	20893288	_X509_PUBLIC_CERT	1120	468-13ece68.crt		308204603082034ca00...
ch2.dmp			468	csrss.exe	21678642	_X509_PUBLIC_CERT	1042	468-14aca32.crt		308204123082O2faa00...
ch2.dmp			468	csrss.exe	21679688	_X509_PUBLIC_CERT	1120	468-14ace48.crt		308204603082034ca00...

Figura 66: Resultados da execução do *plugin* `dumpcerts` no Autopsy

Nome	Data de modificação	Tipo	Tamanho
468-13eca52.crt	06/11/2020 15:15	Certificado de segurança	2 KB
468-13ece68.crt	06/11/2020 15:15	Certificado de segurança	2 KB
468-14aca32.crt	06/11/2020 15:15	Certificado de segurança	2 KB
468-14ace48.crt	06/11/2020 15:15	Certificado de segurança	2 KB
468-204d82.crt	06/11/2020 15:15	Certificado de segurança	1 KB
468-2237efa.crt	06/11/2020 15:15	Certificado de segurança	2 KB

Figura 67: Ficheiros obtidos com a execução do *plugin* `dumpcerts` no Autopsy

Existe ainda a possibilidade de se definirem dois parâmetros adicionais para um dado *plugin*. Por exemplo, para se obter os ficheiros DLL referentes ao processo com PID 2548 da imagem de memória 'ch2.dmp' define-se como parâmetros adicionais o PID (parâmetro '`--pid`') e a diretoria onde os ficheiros DLL serão guardados (parâmetro '`--dump-dir`').

No caso de ocorrer erros na execução de um determinado *plugin* sobre uma imagem de memória, é apresentada uma mensagem de erro ao utilizador na interface do Autopsy. Na Figura 68 encontra-se a mensagem de erro que surge no Autopsy com a falha da execução do *plugin* `hivelist` sobre a imagem de memória 'memdump.mem'. Esta indica que, apesar de o ficheiro JSON relativo ao *plugin* `hivelist` ter sido criado, este encontra-se vazio. É de referir que o módulo também apresenta as mensagens de erro no ficheiro *log* do caso do Autopsy.

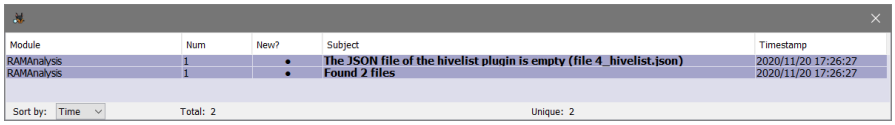


Figura 68: Mensagem de erro relativa à execução do *plugin* `hivelist` sobre a imagem de memória 'memdump.mem'

6.12 ERRO DO TIPO UNICODEENCODEERROR

No caso de alguns *plugins* - como o `handles` e `filescan` - a execução do módulo no Volatility parava quando um dos elementos do artefacto a ser criado possuía caracteres especiais e no ficheiro *log* do caso do Autopsy surgia um erro do tipo '*UnicodeEncodeError*'. Por exemplo, no *plugin* `filescan` o erro que surgia era '*UnicodeEncodeError: 'ascii' codec can't encode characters in position 23-24: ordinal not in range(128)*' e no caso do *plugin* `handles` surgia o erro '*UnicodeEncodeError: 'ascii' codec can't encode character u'ua21b' in position 23: ordinal not in range(128)*'.

Depois de várias tentativas, para resolver os erros do tipo '*UnicodeEncodeError*' foi necessário adicionar no código do *data source ingest module* o método '`setdefaultencoding()`' e a função '`reload()`' (Listagem 28) (WikiTechy, 2020). Contudo, esta solução não é recomendada nem é a mais correta de se aplicar pois com o método '`setdefaultencoding()`' será alterado todo o `defaultencoding` do código que será executado, ou seja, desde ao código em si até ao código em *third-party library* (Kuratomi, 2015).

Listagem 28: Solução para o erro '*UnicodeEncodeError*' baseada na modificação do `defaultencoding`

```
        (...)
        import sys
        reload(sys)
        sys.setdefaultencoding('utf8')
        (...)
```

Na Listagem 29 encontra-se os resultados obtidos por execução do *plugin* `filescan` na *command line* sobre a imagem de memória 'ch2.dmp' e na Figura 69 encontra-se os resultados desse mesmo *plugin* no *blackboard do Autopsy*. Comparando o segundo elemento do conjunto de resultados presente na listagem e na figura, verifica-se que no *blackboard* do Autopsy o caracter especial presente nesse elemento é devidamente convertido.

Listagem 29: Resultados da execução do *plugin* `filescan` na *command line*

```
C:\Volatility\volatility_2.6_win64_standalone>volatility_2.6_win64_stand
↪ alone.exe -f ch2.dmp --profile=Win7SP1x86_23418
↪ filescan
Volatility Foundation Volatility Framework 2.6
Offset(P)             #Ptr   #Hnd Access Name
------------------ ------ ------ ------ ----
0x0000000000653038      5      0 R--r-d
↪ \Device\HarddiskVolume1\Windows\System32\wbem\WmiPrvSE.exe
0x0000000000653cd0      8      0 R--rw- \Device\HarddiskVolume1ýéêÚÑí0
0x0000000000658820      6      0 R--r-d
↪ \Device\HarddiskVolume1\Windows\System32\clusapi.dll
0x0000000000658990      3      0 R--r-d
↪ \Device\HarddiskVolume1\Windows\System32\sscore.dll
0x0000000000658cb0      6      0 R--r-d
↪ \Device\HarddiskVolume1\Windows\System32\resutils.dll
0x0000000000658d58      2      1 ------ \Device\NamedPipe\
```

Source File	S	C	Offset(P)	Pointers	Handles	Access	Name
ch2.dmp			6631480	5	0	R--r-d	\Device\HarddiskVolume1\Windows\System32\wbem\WmiPrvSE.exe
ch2.dmp			6634704	8	0	R--rw-	\Device\HarddiskVolume1\uc088橫0
ch2.dmp			6653984	6	0	R--r-d	\Device\HarddiskVolume1\Windows\System32\clusapi.dll
ch2.dmp			6654352	3	0	R--r-d	\Device\HarddiskVolume1\Windows\System32\sscore.dll
ch2.dmp			6655152	6	0	R--r-d	\Device\HarddiskVolume1\Windows\System32\resutils.dll

Figura 69: Apresentação dos resultados do *plugin* `filescan` com o `defaultencoding` do código modificado

Assim, tentou-se solucionar o problema de outras formas, como por exemplo, com a aplicação do método '`encode('utf-8')`' ao valor, contudo sem sucesso.

Tentou-se ainda aplicar o método presente na Listagem 30 (WikiTechy, 2020). Com este método de resolução do problema, o módulo não apresenta o erro do tipo *UnicodeEncodeError* referido. Contudo, quando o módulo encontra valores com

caracteres especiais, no *blackboard* do Autopsy a célula correspondente a esse valor é apresentada sem qualquer valor.

Listagem 30: Uma das soluções encontradas para o erro do tipo '*UnicodeEncodeError*'

```
        unic = u''
        unic += valueOutput
        valueOutput = unic

        art.addAttribute(BlackboardAttribute(attTypeID,
        ↪  RAMAnalysisDataSourceIngestModuleFactory.moduleName, valueOutput))
```

Na Listagem 31 encontra-se parte do conjunto de resultados obtidos pela execução do *plugin* **handles** sobre a imagem de memória 'ch2.dmp' na *command line*, sendo que o terceiro elemento na Listagem 31 apresenta caracteres especiais no valor da coluna '*Details*'. Já na Figura 70 encontra-se parte do *output* do *plugin* **handles** no Autopsy, sendo o terceiro elemento dos resultados apresentados aquele que possui caracteres especiais, mas, como se pode verificar, a coluna '*Details*' surge vazia para esse elemento.

Listagem 31: Parte do *output* do *plugin* **handles** na *command line*

```
        C:\Volatility\volatility_2.6_win64_standalone>volatility_2.6_win64_stand
        ↪  alone.exe -f ch2.dmp --profile=Win7SP1x86_23418
        ↪  handles
        Volatility Foundation Volatility Framework 2.6
        Offset(V)     Pid     Handle     Access Type            Details
        ---------- ------ ---------- ---------- ---------------- -------

        0x8987da00      4      0x950   0x12019f File
        ↪  \Device\HarddiskVolume1\Windows\CSC\v2.0.6\temp
        0x8988ce20      4      0x954   0x12019f File
        ↪  \Device\HarddiskVolume1\Windows\CSC\v2.0.6\pq
        0x8988cb20      4      0x958   0x100081 File
        ↪  \Device\HarddiskVolume1Ûêø
        0x8988cd68      4      0x95c   0x120089 File
        ↪  \Device\HarddiskVolume1\Windows\CSC\v2.0.6\namespace
        0x8988bcc8      4      0x960   0x1fffff Thread          TID 1060 PID 4
        0x8988b9e0      4      0x964   0x1fffff Thread          TID 1064 PID 4
```

Source File	S	C	Offset(V)	Pid	Handle	Access	Type	Details
ch2.dmp			2307381760	4	2384	1180063	File	\Device\HarddiskVolume1\Windows\CSC\v2.0.6\temp
ch2.dmp			2307444256	4	2388	1180063	File	\Device\HarddiskVolume1\Windows\CSC\v2.0.6\pq
ch2.dmp			2307443488	4	2392	1048705	File	
ch2.dmp			2275249192	4	24	2031617	ALPC Port	PowerMonitorPort
ch2.dmp			2334515904	4	240	131097	Key	MACHINE\SYSTEM\CONTROLSET001\SERVICES\DISK

Figura 70: Apresentação dos resultados do *plugin* **handles** com a solução da Listagem 30

Assim, apesar dos riscos existentes em aplicar o método '`setdefaultencoding()`' e a função '`reload()`', para solucionar os erros do tipo '*UnicodeEncodeError*',

decidiu-se aplicar estas duas linhas no módulo desenvolvido uma vez que foi a solução que, até ao momento, permite apresentar melhores resultados ao utilizador.

6.13 SÍNTESE

Neste capítulo foram apresentados alguns aspetos relativos ao desenvolvimento do *data source ingest module*, módulo este que pretende executar o Volatility e apresentar os resultados no *blackboard* do Autopsy. Este módulo tem a vantagem de obter automaticamente o perfil a aplicar à imagem de memória em causa.

Inicialmente foi apresentada a arquitetura do módulo bem como alguns aspetos iniciais relativos ao código do módulo em causa. O *data source ingest module* desenvolvido apresenta uma interface gráfica, na qual o utilizador terá que definir algumas opções de modo a executar o Volatility. Ao longo da criação do módulo foram utilizados diversos métodos para que o módulo funciona-se como pretendido. Por exemplo, para que o Volatility seja executado através do módulo é necessário utilizar a função ‘`Popen`‘.

Os resultados da execução do *Volatility* são guardados em ficheiros JSON que, posteriormente, são processados pelo módulo para aceder ao seu conteúdo. A colocação dos resultados no *blackboard* ocorre através de métodos como, por exemplo, ‘`addArtifactType()`‘ e ‘`addArtifactAttributeType()`‘.

O *data source ingest module* permite ainda criar ficheiros `.dot` de alguns *plugins*, sendo que o módulo converte automaticamente esses ficheiros para formato `.png`.

Por fim, neste capítulo, foi efetuada uma análise comparativa entre os resultados obtidos na *command line*, no ficheiro JSON e os apresentados na interface do Autopsy de modo a clarificar algumas diferenças que possam existir.

7

DESENVOLVIMENTO DO REPORT MODULE PARA AUTOPSY

Os resultados da execução do *data source ingest module* são apresentados no *blackboard* do Autopsy. De modo a possibilitar uma diferente visualização gráfica dos resultados e apresentar ao utilizador uma análise adicional, nesta dissertação foi criado um *report module* que se designa por RAMAnalysis Report.

Assim, o *report module* criado apresenta não só os resultados expostos no *blackboard* do Autopsy provenientes dos ficheiros JSON mas também dados obtidos por análise e processamento dos resultados obtidos pelo *data source ingest module*.

O *report module* foi desenvolvido em *Python*, sendo que ao longo do seu desenvolvimento também foi necessário recorrer à linguagem HTML e ao Cascading Style Sheets (CSS). Para o desenvolvimento do *report module* foi utilizado o *software* IntelliJ IDEA Community Edition versão 2020.1.4 desenvolvido pela JetBrains. O desenvolvimento do *report module* ocorreu numa máquina que apresenta o sistema operativo *Microsoft Windows 10 Home* versão 10.0.17763 Compilação 17763, com 8 GB de RAM e com um processador Intel(R) Core(TM) i7-7500U CPU @ 2.70GHz.

7.1 FASE INICIAL

O Autopsy é uma ferramenta que permite que lhe sejam adicionados módulos. Nesta dissertação, para além de ser desenvolvido um *data source ingest module*, foi criado um *report module*. Este tipo de módulo permite a criação de diferentes tipos de relatórios, sendo que nesta dissertação foi criado um *report module* que cria um relatório HTML.

Para a exploração do processo de desenvolvimento de *report modules* foram realizados tutoriais[1], os quais permitiram compreender alguns aspetos relativos à construção dos *report modules*.

Ao longo da exploração do Volatility e análise dos *outputs* apresentados pelos seus *plugins* verificou-se que seria útil e proveitoso apresentar algumas informações

1 https://www.autopsy.com/python-autopsy-module-tutorial-3-the-report-module/

adicionais através da análise dos resultados pelo próprio *report module*. O objetivo dessa mesma informação adicional passa por, se possível, facilitar e otimizar a análise à memória que o utilizador está a realizar. Assim, o relatório que o *report module* cria apresenta os dados resultantes da execução dos *plugins* e, sempre que possível, dados adicionais obtidos com base na análise de alguns dos *outputs* obtidos pelo utilizador.

Nesta dissertação, o relatório que é criado pelo *report module* é armazenado numa pasta que, por sua vez, se encontra na pasta 'Reports' do caso do Autopsy. Contudo, é de realçar que na verdade o *report module* ao ser executado cria cinco ficheiros em formato HTML que, em conjunto, correspondem ao relatório pretendido. Estes cinco ficheiros HTML correspondem a cinco páginas distintas do relatório.

O utilizador para criar o relatório tem que clicar na opção '*Generate Report*' da interface do Autopsy (parte superior), não sendo necessário definir quaisquer parâmetros. Na Figura 71 encontra-se a janela que surge ao utilizador após clicar na opção referida, sendo que para executar o *report module* tem que clicar em '*Finish*'.

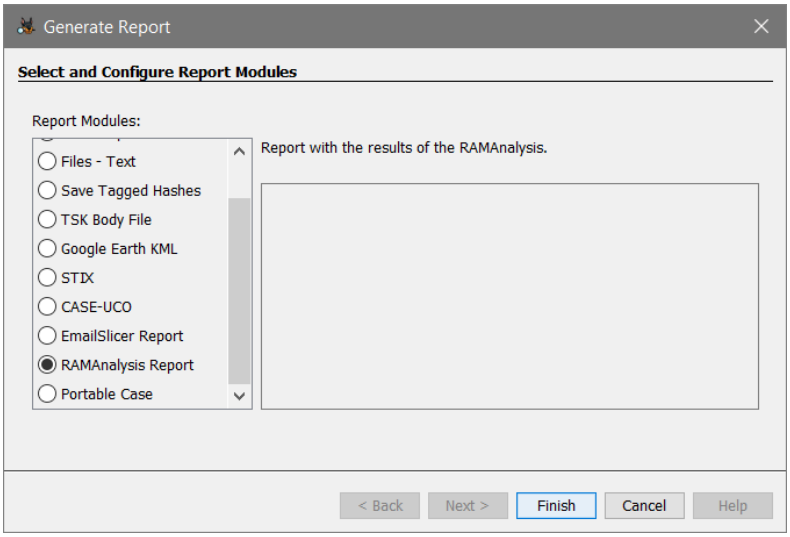


Figura 71: Seleção no Autopsy do *report module* desenvolvido

Para o desenvolvimento do *report module* utilizou-se o *template* do mesmo disponibilizado na página do GitHub do Autopsy[2] (ficheiro `reportmodule.py`).

2 https://github.com/sleuthkit/autopsy/tree/develop/pythonExamples

7.2 ASPETOS INICIAIS DO CÓDIGO

O *report module* elaborado apresenta uma só classe - a *factory class* - que se designou por '`RAMAnalysisReportModule`'.

De todos os métodos nesta classe é de realçar o método '`generateReport()`' que corresponde ao método que é chamado quando se pretende executar o módulo. Os dois parâmetros deste método correspondem à diretoria na qual os resultados serão guardados e à barra de progresso (Autopsy, 2020). É neste método que se irá encontrar grande parte do código responsável pela criação do relatório.

A classe '`RAMAnalysisReportModule`' apresenta ainda cinco métodos que devolvem o nome do ficheiro que o *report module* irá usar para o relatório que cria.

Para cada uma das cinco páginas HTML do relatório criadas pelo *report module* foi necessário definir um *template* HTML. Os *templates* têm que estar presentes na mesma pasta que o ficheiro `.py` do *report module*.

Na classe '`RAMAnalysisReportModule`' foram ainda definidos outros quatro métodos que retornam informação necessária para o funcionamento do *report module*, sendo que alguma da mesma será apresentada no relatório criado. Foi definido o método '`getAllJsonFiles()`' que devolve num *array* os nomes de todos os ficheiros JSON presentes na pasta '`ModuleOutput\RAMAnalysisModule`'.

Já os restantes três outros métodos definidos - '`getMemoryImagesIDAndName_DB()`', '`getIDAndTypeNamesArtifacts_DB()`' e '`getColumnsNames_DB`' - vão aceder a informação presente na base de dados do caso do Autopsy, mais precisamente no ficheiro '`autopsy.db`' que se encontra na pasta do caso em causa. Grande parte da informação presente no ficheiro `autopsy.db` foi adicionada pelo *data source ingest module* ao longo da sua execução. Ao se explorar o conteúdo do ficheiro `autopsy.db` verifica-se que a informação encontra-se organizada em várias tabelas cujos nomes são iguais independentemente do caso do Autopsy. De uma forma sucinta, em *Python*, para se aceder a esse mesmo ficheiro, primeiro, é necessário abrir o mesmo ('`DriverManager.getConnection()`') e efetuar a pesquisa SQL à base de dados através dos métodos '`createStatement()`' e '`executeQuery()`' aplicados às devidas variáveis. Para se aceder aos valores devolvidos da consulta SQL à base dados é necessário utilizar o método '`getString()`' que terá como parâmetro a designação da coluna cujos valores se pretende obter.

Assim, no *report module* existe o método '`getMemoryImagesIDAndName_DB()`' que, efetuando uma pesquisa SQL à tabela '`tsk_files`', devolve um *array* com

os ID das imagens de memória e outro *array* com o nome das mesmas. O método '`getIDAndTypeNamesArtifacts_DB()`' permite obter um *array* com o ID dos tipos de artefactos associados ao caso e também um *array* com a designação dos tipos de artefactos do caso, sendo que essa informação provém da tabela '`blackboard_artifact_types`'. Por fim, existe o método '`getColumnsNames_DB()`' que, efetuando uma pesquisa SQL à tabela '`blackboard_attribute_types`', devolve o nome dos tipos de atributos definidos pelo *data source ingest module*.

7.3 OBTENÇÃO DO CONTEÚDO PARA O RELATÓRIO

O conteúdo que será apresentado no relatório criado pelo *report module* corresponde:

- A alguma da informação presente na base de dados do caso do Autopsy em questão (ficheiro `autopsy.db`) (Exemplo: as designações das imagens de memória);
- Os artefactos adicionados ao *blackboard* do Autopsy pelo *data source ingest module*;
- Ao conteúdo (ou parte dele) presente nos ficheiros JSON auxiliares criados previamente pelo *data source ingest module*.

7.3.1 *Artefactos Provenientes do Blackboard*

O *data source ingest module* criado apresenta no *blackboard* do Autopsy os resultados da sua execução, mais precisamente, os resultados da execução dos *plugins* selecionados pelo utilizador.

Deste modo, dado que se pretende incluir no *report module* os resultados da análise efetuada pelo *data source ingest module*, terá que se utilizar o método '`SleuthkitCase.getBlackboardArtifacts()`', sendo que '`SleuthkitCase`' corresponde ao valor devolvido por '`Case.getCurrentCase().getSleuthkitCase()`'. O método '`getBlackboardArtifacts()`' devolve artefactos presentes no *blackboard* do Autopsy e pode apresentar diferentes argumentos. Com a designação do tipo de artefacto e o ID da imagem de memória como argumentos, o método devolve os artefactos do *blackboard* de um determinado tipo referentes a uma dada imagem de memória. É de referir que se obtém o mesmo resultado colocando-se como argumen-

tos do método ‘`getBlackboardArtifacts()`‘ o ID do tipo de artefacto e o ID da imagem de memória (TSK, 2015b).

Um outro método utilizado foi o método ‘`getBlackboardArtifactsTypeCount()`‘ que, com o ID do tipo de artefacto como argumento, permite obter o número de artefactos de cada tipo de artefacto presente no caso do Autopsy (TSK, 2015b).

Já o método ‘`getBlackboardArtifactsCount()`‘, com o ID do tipo de artefacto e o ID da imagem de memória como argumentos, devolve o número de artefactos por tipo e por imagem de memória. Contudo, no *report module* desenvolvido também se utiliza como argumentos do método ‘`getBlackboardArtifactsCount()`‘ a designação do tipo de artefacto e o ID da imagem de memória (TSK, 2015b).

7.3.2 *Resultados Provenientes dos Ficheiros JSON Auxiliares*

O *data source ingest module* cria, se possível, para cada imagem de memória um conjunto de quatro ficheiros auxiliares referentes ao *plugins* `envars`, `hashdump`, `hivelist` e `netscan`. Destes quatro ficheiros, serão processados no *report module* os ficheiros JSON referentes aos *plugins* `envars`, `hashdump` e `netscan`. É de referir que, tal como já foi explicado anteriormente, o ficheiro JSON referente ao *plugin* `hivelist` é processado e utilizado no *data source ingest module* uma vez que tem dados necessários para executar o *plugin* `hashdump`.

O processamento realizado aos ficheiros JSON dos *plugins* `envars`, `hashdump` e `netscan` no *report module* é igual ao processamento de ficheiros JSON no *data source ingest module* relatado no subcapítulo 6.6.1.

Para o caso do conteúdo do ficheiro JSON do *plugin* `envars`, será extraído unicamente o nome da máquina a que a imagem de memória se refere, informação esta que se encontrará na coluna ‘*Value*‘ numa linha que apresente o valor ‘COMPUTERNAME‘ na coluna ‘*Variable*‘.

O objetivo do ficheiro JSON do *plugin* `netscan` é obter os processos responsáveis por conexões abertas (estado ‘ESTABLISHED‘) e os processos que aguardem por um pedido de ligação (estado ‘LISTENING‘) (Institute, 1981). Assim, do conteúdo presente no ficheiro serão extraídos e guardados os elementos do *output* que tenham ‘LISTENING‘ ou ‘ESTABLISHED‘ como valor da coluna ‘*State*‘.

Dado que o ficheiro JSON referente ao *plugin* `hashdump` apresenta os utilizadores da máquina, mais especificamente, as *cached domain credentials* guardadas na

Registry, todo o conteúdo será guardado para ser apresentado no relatório criado (Volatility Foundation, 2019b).

7.4 BEAUTIFUL SOUP

Tal como já foi mencionado, o relatório que é criado pelo *report module* consiste em cinco ficheiros HTML, tendo cada um desses ficheiros um *template* correspondente. Os *templates* criados possuem a sua estrutura HTML base já definida no documento em si, bem como alguns dos seus elementos.

O conteúdo que irá surgir no relatório varia de caso para caso, sendo que certos elementos HTML só irão surgir se certas condições se verificarem nos resultados dos *plugins*. Assim, de modo a inserir o conteúdo pretendido no relatório, é necessário, a partir do ficheiro *Python* correspondente ao *report module*, manipular o código HTML de cada um dos ficheiros HTML. Para tal, no desenvolvimento do *report module* utilizou-se a biblioteca *Beautiful Soup*.

A *Beautiful Soup* é uma biblioteca *Python* para extrair dados de ficheiros HTML e XML e é compatível tanto com *Python 2.7* como com *Python 3*. Esta biblioteca, a partir de documentos HTML e XML analisados cria uma árvore de análise, permitindo navegar, pesquisar e modificar a mesma (Richardson, 2020; Tagliaferri, 2019).

Nesta dissertação, para o desenvolvimento do *report module* foi utilizada a versão 4 do *Beautiful Soup*. Para instalar e, posteriormente, utilizar o *Beautiful Soup* basta executar o comando '`pip install beautifulsoup4`' (Richardson, 2020). A *Beautiful Soup* apresenta ainda uma documentação[3], na qual se pode encontrar informação acerca desta.

É através da biblioteca *Beautiful Soup* e dos métodos da mesma que, por exemplo, vários elementos (por exemplo, tabelas e *cards*) das páginas HTML que compõem o relatório são criados e também é adicionado conteúdo a certos elementos já definidos nos *templates*.

7.4.1 *Processo de Utilização*

Para manipular um ficheiro HTML através do *Beautiful Soup* é necessário abrir o ficheiro HTML correspondente ao *template* em questão (método '`open()`') e ler o seu

3 https://www.crummy.com/software/BeautifulSoup/bs4/doc/

conteúdo com o método '`read()`'. Utilizando o construtor `BeautifulSoup` e tendo como argumento o documento HTML, obtem-se um objeto do tipo `BeautifulSoup`. Este processo foi executado para cada um dos cinco *templates*.

Ao longo do desenvolvimento do *report module*, ao objeto do tipo `BeautifulSoup` foram várias vezes aplicados dois métodos, nomeadamente, os métodos '`find()`' e '`new_tag()`'. O método '`find()`' procura por um determinado elemento no ficheiro HTML relativo ao objeto `BeautifulSoup` criado. O elemento a ser procurado é definido como argumento do método, existindo várias formas de identificar o elemento (por exemplo, colocando como argumento o ID do elemento) (Richardson, 2020).

Através do método '`new_tag()`' é possível criar novas *tags* HTML, sendo que para isso basta colocar como argumento a designação da *tag* (por exemplo, `new_tag('p')` cria uma nova *tag* `<p>`). Ao objeto devolvido pelo método '`new_tag()`' é possível definir certos atributos, como por exemplo, os atributos '*class*' e '*style*' (Richardson, 2020).

Com uma nova *tag* criada é necessário inserir a mesma no ficheiro HTML. No *report module* criado utilizou-se os métodos '`append()`' e '`insert()`' que são iguais, sendo a única diferença o facto de com o método `insert()` o elemento será inserido na posição numérica indicada como argumento (Richardson, 2020). É de referir que com ambos os métodos o resultado no ficheiro HTML é o mesmo.

Por fim, após a criação e adição de elementos e conteúdo a cada uma das páginas HTML, com o método '`open()`' é aberto em modo escrita o ficheiro HTML correspondente, sendo que este ficheiro irá guardar o conteúdo presente no objeto do tipo `BeautifulSoup`.

7.4.2 *Exemplos da Utilização da Beautiful Soup*

Nesta subcapítulo encontram-se exemplos com excertos da utilização da *Beautiful Soup* no *report module* criado nesta dissertação.

Na Listagem 32 encontra-se parte do código relativo à colocação de dados num dado elemento HTML através da *Beautiful Soup*. Esse mesmo elemento já se encontra criado no *template* relacionado, mas sem informação acerca do módulo. Na Listagem 32, procura-se pelo elemento HTML com o ID 'totalArtefactsPlugins' e, de seguida, define-se o valor que este deve apresentar (variável 'countTotalArtifacts') através de '`smallTagTotalNumberArtfs.string`'.

Listagem 32: Exemplo de adição de conteúdo a um elemento HTML já existente no *template*

```
        (...)
        smallTagTotalNumberArtfs = soup_Dashb.find(id="totalArtefactsPlugins")
        smallTagTotalNumberArtfs.string = "Total de Artefactos: " +
        ↪  str(countTotalArtifacts)
        (...)
```

Já na Listagem 33 encontra-se um extrato do código do *report module* relativo à contrução de um novo elemento HTML com recurso à *Beautiful Soup* e a inserção deste no devido ficheiro HTML. Como se pode constatar na Listagem 33 é criado uma nova *tag* `<p>` que tem a classe '`card-text`' e com uma *string* a apresentar definida. Por fim, a nova *tag* é adicionada ao elemento 'divCardMb3Body', que, por sua vez, já se encontra presente no ficheiro HTML.

Listagem 33: Exemplo da criação de um novo elemento HTML com recurso da *BeautifulSoup*

```
        (...)
        pTagMessage = soup_Detecoes.new_tag("p")
        pTagMessage["class"] = "card-text"
        pTagMessage.string = "Imagem de memoria referente: %s" % nameMemoryImage
        divCardMb3Body.append(pTagMessage)
        (...)
```

7.5 ANÁLISE DOS FICHEIROS HTML

O relatório criado pelo *report module* é composto por cinco ficheiros HTML, que, por sua vez, apresentam informação distinta entre eles. Cada um desses ficheiros HTML que são criados pelo *report module* correspondem às seguintes páginas HTML do relatório:

- *Dashboard*
- Resultados por *Plugin*
- Resultados por *Plugin*/Imagem de Memória
- Imagens dos Gráficos DOT
- Deteções

Para a criação dos ficheiros HTML foi utilizada a linguagem HTML e o CSS. Foi ainda utilizada o *Bootstrap* (versão 4.1.3)[4] que é uma *front-end framework* gratuita que permite um desenvolvimento *web* rápido e fácil, sendo que esta *framework*

4 https://getbootstrap.com/

Listagem 34: Elementos necessários ao funcionamento do *Bootstrap*

```
1  <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/css/bootstrap.min.css"
   ↪ integrity="sha384-MCw98/SFnGE8fJT3GXwEOngsV7Zt27NXFoaoApmYm81iuXoPkFOJwJ8ERdknLPMO"
   ↪ crossorigin="anonymous">
2  <script src="https://code.jquery.com/jquery-3.3.1.slim.min.js"
   ↪ integrity="sha384-q8i/X+965DzO0rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abtTE1Pi6jizo"
   ↪ crossorigin="anonymous"></script>
3  <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.3/umd/popper.min.js"
   ↪ integrity="sha384-ZMP7rVo3mIykV+2+9J3UJ46jBk0WLaUAdn689aCwoqbBJiSnjAK/l8WvCWPIPm49"
   ↪ crossorigin="anonymous"></script>
4  <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/js/bootstrap.min.js"
   ↪ integrity="sha384-ChfqqxuZUCnJSK3+MXmPNIyE6ZbWh2IMqE241rYiqJxyMiZ6OW/JmZQ5stwEULTy"
   ↪ crossorigin="anonymous"></script>
```

apresenta vários *templates* em HTML e CSS de elementos como, por exemplo, tabelas. É de referir que várias das classes utilizadas nos elementos das páginas HTML do *report module* são as presentes nos exemplos de *templates* disponibilizados na documentação. Para que o *Bootstrap* funcione nos ficheiros HTML criados é necessário que sejam adicionado o `<link>` e os três `<script>` presentes na Listagem 34 (Bootstrap, 2019).

Para além do referido, para adicionar alguns ícones ao relatório foi utilizada a *Font Awesome*[5] (versão 4.7.0), sendo que, para tal, foi necessário adicionar a seguinte linha em todos os ficheiros HTML do relatório (w3schools, 2019):

```
<link rel="stylesheet"href="https://cdnjs.cloudflare.com/ajax
/libs/font-awesome/4.7.0/css/font-awesome.min.css»
```

Todas estas páginas HTML apresentam uma barra superior com o nome do módulo e uma barra lateral simples no lado esquerdo da páginas que permite, a partir de uma das páginas HTML do relatório aceder a outras páginas (Figura 72). Assim, por exemplo, estando com o ficheiro HTML da página '*Dashboard*' aberto e se pretender-se aceder à página que apresente as imagens referentes aos gráficos `.dot` basta clicar na opção 'Imagens dos Gráficos DOT' da barra lateral.

5 https://fontawesome.com/

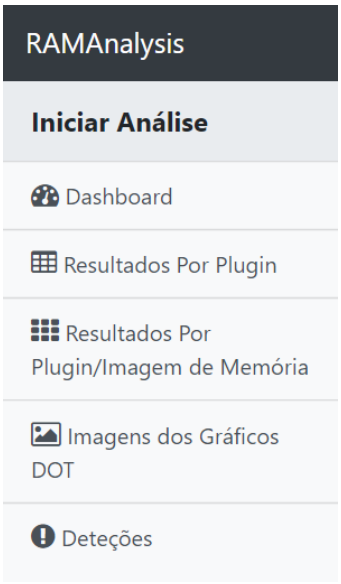


Figura 72: Barra lateral das páginas HTML do relatório

De seguida, será efetuada uma análise e exploração a cada uma das páginas HTML pertencentes ao relatório de modo a se perceber que informação é disponibilizada no relatório ao utilizador e a origem da mesma. As imagens que são apresentadas nas seguintes subsecções apresentam resultados do relatório referentes à execução dos *plugins* `imageinfo` e `pslist` sobre as imagens de memória 'ch2.dmp' e 'memdump.mem'.

7.5.1 *Dashboard*

O *Dashboard* é a página principal e inicial do relatório, sendo nesta página que é apresentado um resumo com a informação essencial acerca do caso e da análise efetuada.

No topo da página existem três *cards* que apresentam informação distinta (Figura 73). O *card* 'Imagens de Memória' apresenta a designação das imagens de memória presentes no caso, sendo que essa informação provém da base de dados do caso (ficheiro `autopsy.db`). O *footer* deste *card* apresenta o total de imagens de memória. O segundo *card* - 'Plugins Executados' - expõe os *plugins* que foram executados com sucesso sobre todas as imagens de memória; esta informação tem origem da base de dados do caso. Tal como se pode ver na Figura 73, o total de *plugins* executados no caso é apresentado no *footer* deste *card*.

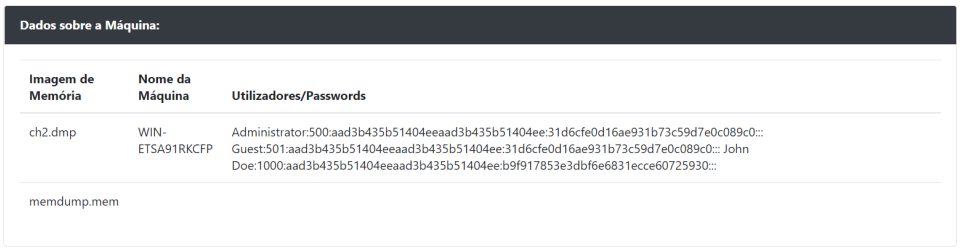
Já o *card* com o título 'Total de Artefactos por *Plugin*' indica o total de artefactos para cada um dos *plugins* executados, sendo que esses dados são obtidos através do método '`getBlackboardArtifactsTypeCount()`' com o ID do tipo de artefacto como argumento. O total de artefactos presentes é indicado no seu *footer*.

Figura 73: Primeiros elementos da página *Dashboard* do relatório

Após estes três *cards*, é apresentada uma tabela com o número de artefactos obtidos com a execução de cada *plugin* sobre cada imagem de memória do caso (Figura 74). Os dados para as colunas 'Imagem de memoria', 'ID Imagem de Memoria', 'Plugin' e 'ID Plugin' provêm da informação obtida da base de dados do caso. Já o total de artefactos é obtido através do método '`getBlackboardArtifactsCount()`' que tem como argumentos o ID do tipo de artefacto e o ID da imagem de memória.

Figura 74: Total de artefactos obtidos por *plugin* sobre cada imagem de memória no *Dashboard*

Ainda no '*Dashboard*' são apresentados, se possível, alguns dados acerca da máquina a que se refere cada imagem de memória (Figura 75). O nome da máquina a que cada imagem de memória se refere provém do ficheiro JSON auxiliar do *plugin* `envars`, enquanto que os *users/passwords* são originários do ficheiro JSON auxiliar do *plugin* `hashdump`. Como se pode constatar na Figura 75, na tabela não existem valores referentes à imagem de memória 'memdump.mem' pois não foi possível encontrar nenhum dos dados.

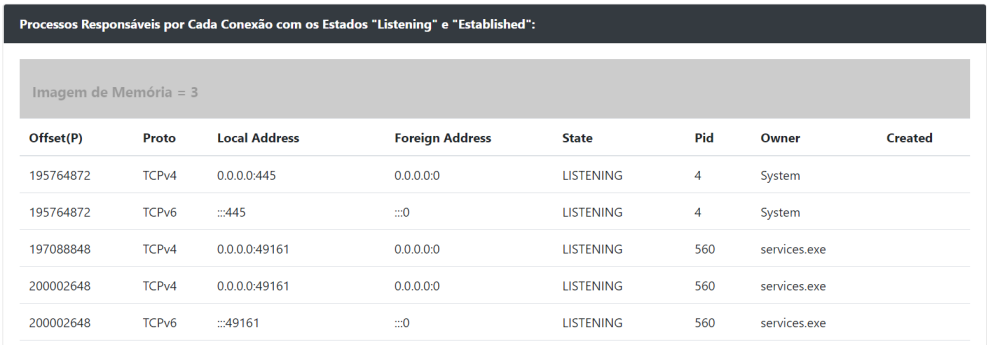


Dados sobre a Máquina:

Imagem de Memória	Nome da Máquina	Utilizadores/Passwords
ch2.dmp	WIN-ETSA91RKCFP	Administrator:500:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0::: Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0::: John Doe:1000:aad3b435b51404eeaad3b435b51404ee:b9f917853e3dbf6e6831ecce60725930:::
memdump.mem		

Figura 75: Apresentação no '*Dashboard*' de dados sobre as máquinas a que as imagens de memória se referem

Por último, na página '*Dashboard*' do relatório são apresentadas, se possível, as conexões com estado 'ESTABLISHED' (ligações abertas) e as ligações com estado 'LISTENING' de cada imagem de memória (Figura 76). Esta informação é extraída do ficheiro JSON auxiliar do *plugin* `netscan`. Ter conhecimento de conexões com estes estados pode ser relevante para descobrir o que aconteceu ao sistema. Por exemplo, no caso das conexões 'ESTABLISHED' é importante verificar se existe alguma com um '*Foreign Address*' não reconhecido. Um outro aspeto a realçar é o facto de quando um sistema é alvo de um ataque, em alguns casos, já existe um *backdoor* à escuta para que ocorram ligações mais facilmente (Sanders, 2011).



Processos Responsáveis por Cada Conexão com os Estados "Listening" e "Established":

Imagem de Memória = 3

Offset(P)	Proto	Local Address	Foreign Address	State	Pid	Owner	Created
195764872	TCPv4	0.0.0.0:445	0.0.0.0:0	LISTENING	4	System	
195764872	TCPv6	:::445	:::0	LISTENING	4	System	
197088848	TCPv4	0.0.0.0:49161	0.0.0.0:0	LISTENING	560	services.exe	
200002648	TCPv4	0.0.0.0:49161	0.0.0.0:0	LISTENING	560	services.exe	
200002648	TCPv6	:::49161	:::0	LISTENING	560	services.exe	

Figura 76: Parte da tabela do '*Dashboard*' com as conexões 'LISTENING' e 'ESTABLISHED' referentes à imagem de memória com ID 3

Na Figura 77 encontram-se as conexões referentes à imagem de memória 'memdump.mem' (ID=4). Como se pode verificar, em alguns casos, o PID dos processos pode apresentar o valor '-1'. Um modo existente de descobrir o processo responsável por essa mesma ligação passa pela utilização do *plugin* `yarascan`, indicando no parâmetro `-Y` o endereço Internet Protocol (IP) presente na coluna *Foreign Address* (soji256, 2019). Contudo, para o caso presente na Figura 77, após executar na linha de comandos o *plugin* `yarascan` para alguns dos endereços IP, não foi devolvida qualquer informação.

Imagem de Memória = 4

Offset(P)	Proto	Local Address	Foreign Address	State	Pid	Owner	Created
213358578697040	TCPv4	10.0.2.143:51431	13.107.255.168:443	ESTABLISHED	-1		
213358582478784	TCPv4	10.0.2.143:51452	13.107.4.50:80	ESTABLISHED	-1		
213358627962048	TCPv4	10.0.2.143:51446	72.21.91.29:80	ESTABLISHED	-1		
213358629212176	TCPv4	10.0.2.143:51451	52.160.42.250:443	ESTABLISHED	-1		

Figura 77: Parte da tabela do '*Dashboard*' com as conexões 'LISTENING' e 'ESTABLISHED' referentes à imagem de memória com ID 4

7.5.2 *Resultados Por Plugin*

O relatório, na página 'Resultados Por Plugin', apresenta os resultados da execução dos *plugins* organizados por *plugin*, e independentemente da imagem de memória, sendo que os resultados encontram-se em tabelas.

A informação que se pode visualizar nesta página do relatório corresponde ao artefactos presentes no *blackboard* do Autopsy. Assim, utilizando-se o método '`getBlackboardArtifacts()`' com o ID do tipo de artefacto como argumento obtem-se todos os artefactos que pertençam a um determinado tipo de artefacto.

Na Figura 78 encontra-se uma tabela com os resultados da execução do *plugin* `imageinfo` sobre todas as imagens de memória. Como se pode verificar, na tabela não é especificada a imagem de memória. Por vezes, como as tabelas têm muitas colunas, uma parte dos resultados não fica visível de imediato, logo, tem que se utilizar o *scroll* na tabela para visualizar o restante conteúdo.

Resultados do Plugin imageinfo

Suggested Profile(s)	AS Layer1	AS Layer2	PAE type	DTB	KDBG	N o P
Win7SP1x86_23418, Win7SP0x86, Win7SP1x86	IA32PagedMemoryPae (Kernel AS)	FileAddressSpace (C:\Users\santo\Desktop\imagensMem_Cenario1\ch2.dmp)	PAE	1593344	2190646248	1
Win10x64_10586, Win10x64_14393, Win10x64, Win2016x64_14393	Win10AMD64PagedMemory (Kernel AS)	FileAddressSpace (C:\Users\santo\Desktop\imagensMem_Cenario1\memdump.mem)	No PAE	1757186	272682280801568	1

Figura 78: Tabela com os resultados do *plugin* `imageinfo` de todas as imagens de memória

7.5.3 *Resultados Por Plugin/Imagem de Memória*

A página 'Resultados Por Plugin/Imagem de Memória' do relatório contém os resultados obtidos pela execução do Volatility organizados por *plugin* e por imagem de memória, ou seja, existe uma tabela para cada *plugin* e respetiva imagem de memória.

Os dados necessários para as tabelas desta página do relatório são obtidos através do método '`getBlackboardArtifacts()`' com o ID do tipo de artefacto e o ID da imagem de memória como argumentos.

Na Figura 79 encontra-se uma tabela com os resultados obtidos pela execução do *plugin* `imageinfo` e uma outra tabela com alguns dos resultados obtidos pelo *plugin* `pslist`, sendo que ambas as tabelas referem-se à imagem de memória 'ch2.dmp'.

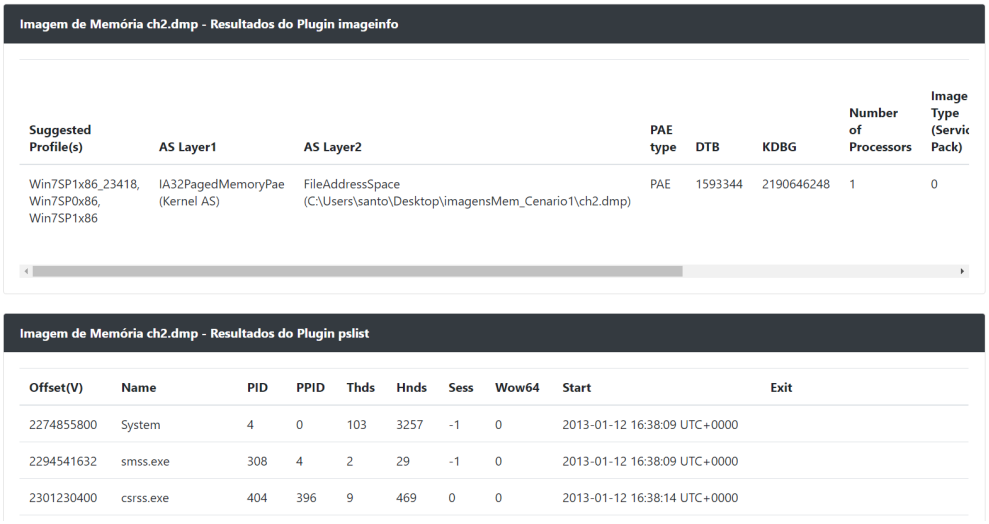


Imagem de Memória ch2.dmp - Resultados do Plugin imageinfo

Suggested Profile(s)	AS Layer1	AS Layer2	PAE type	DTB	KDBG	Number of Processors	Image Type (Servic Pack)
Win7SP1x86_23418, Win7SP0x86, Win7SP1x86	IA32PagedMemoryPae (Kernel AS)	FileAddressSpace (C:\Users\santo\Desktop\imagensMem_Cenario1\ch2.dmp)	PAE	1593344	2190646248	1	0

Imagem de Memória ch2.dmp - Resultados do Plugin pslist

Offset(V)	Name	PID	PPID	Thds	Hnds	Sess	Wow64	Start	Exit
2274855800	System	4	0	103	3257	-1	0	2013-01-12 16:38:09 UTC+0000	
2294541632	smss.exe	308	4	2	29	-1	0	2013-01-12 16:38:09 UTC+0000	
2301230400	csrss.exe	404	396	9	469	0	0	2013-01-12 16:38:14 UTC+0000	

Figura 79: Visualização no relatório de alguns resultados por *plugin* e por imagem de memória

7.5.4 *Imagens dos Gráficos DOT*

O *data source ingest module* quando cria gráficos DOT referentes a um ou mais *plugins*, se possível, converte automaticamente esses mesmos ficheiros para ficheiros com o formato `.png`.

Assim, se existirem imagens na diretoria '`ModuleOutput\RAMAnalysisModule`', essas serão apresentadas no relatório na página 'Imagens dos Gráficos DOT' (Figura 80). O nome das imagens `.png` indica o ID da imagem de memória e a designação do *plugin* a que se refere ('idImagemMemoria_designacaoPlugin.png').

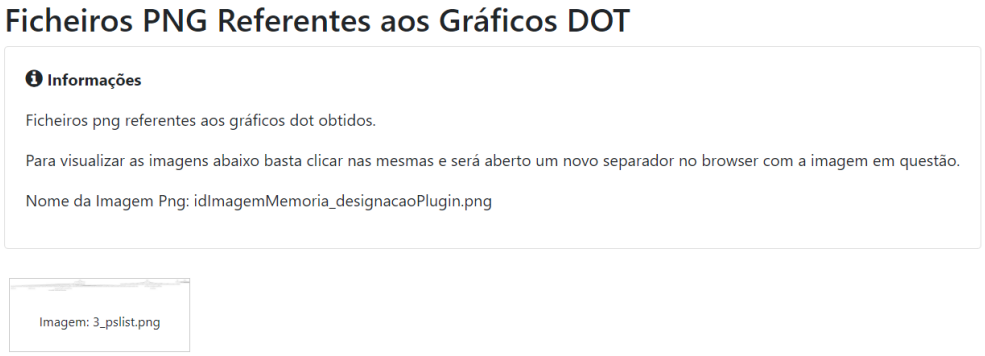


Figura 80: Apresentação no relatório da página com as imagens existentes referentes aos gráficos DOT

Para visualizar as imagens basta clicar nas mesmas e será aberto no *browser* um novo separador com a imagem selecionada (Figura 81), sendo que assim o utilizador pode fazer uso do *zoom* do *browser*.

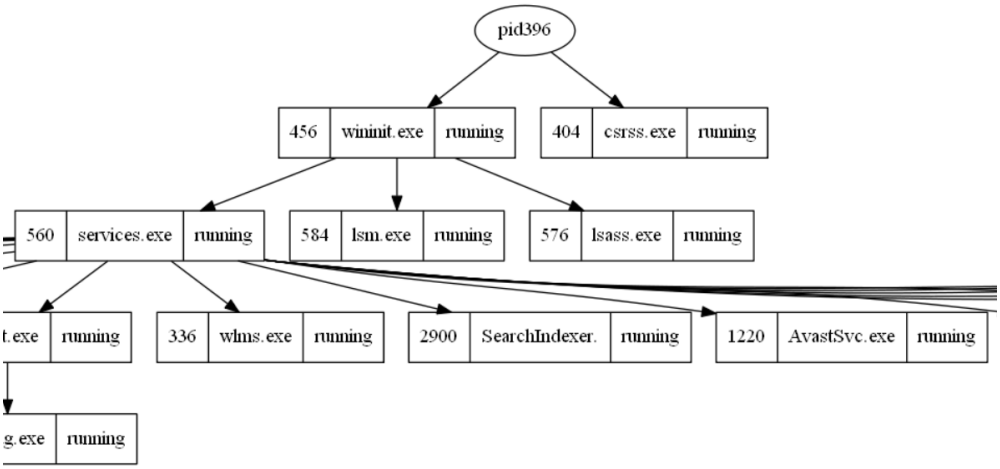


Figura 81: Visualização no relatório de parte da imagem '`3_pslist.png`' apresentada na Figura 80

7.5.5 *Deteções*

A última página do relatório criado pelo *report module* intitula-se por 'Deteções'. O objetivo desta página é apresentar ao utilizador alguma informação suspeita obtida por análise dos dados obtidos da execução do Volatility.

Até ao momento, esta página só apresenta irregularidades que o módulo detete automaticamente nos resultados do *plugin* `pslist`. É de notar que tal só acontece se

se verificar que o utilizador executou o *plugin* `pslist`. Assim, é apresentado nesta página o nome dos processos que tenham um PPID inexistente na lista de processos e o nome dos processos que tenham nomes iguais mas apresentem PPID diferentes, sendo para cada um dos casos indicada a imagem de memória em causa (A, 2016).

Os dados necessários para a página 'Deteções' do relatório que é criado provêm do método '`getBlackboardArtifacts()`', tendo como argumentos a designação do tipo de artefacto do *plugin* `pslist` ('TSK_VOLAT_PSLIST') e o ID da imagem de memória em causa.

Na Figura 82 encontram-se duas deteções acerca da imagem de memória 'ch2.dmp'. A primeira deteção apresentada indica que na imagem de memória 'ch2.dmp', o processo '*csrss.exe*' tem um PPID (396) que não consta na lista de PID dos processos. Já a segunda deteção presente na Figura 82 indica que, tal como na deteção anterior, o PPID (396) do processo '*wininit.exe*' não está na lista de PID dos processos.

Figura 82: Dois exemplos de duas deteções do tipo 'PPID inexistente na lista dos PID dos processos'

Já na Figura 83 encontram-se outras duas deteções relativas à imagem de memória 'ch2.dmp'. A primeira deteção apresentada revela que há mais do que um processo com o nome '*iexplore.exe*', mas com PPID diferentes. A segunda deteção indica que há mais do que um processo com o nome '*soffice.bin*', mas os mesmos apresentam PPID diferentes.

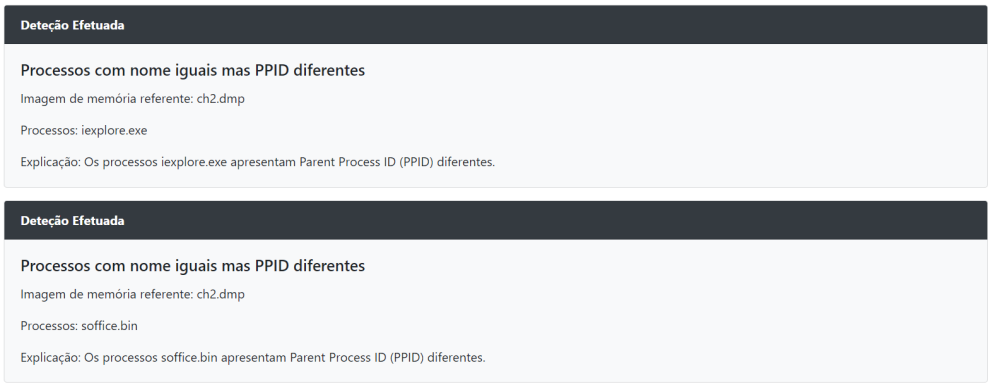


Figura 83: Dois exemplos de duas deteções do tipo 'Processos com nome iguais mas PPID diferentes'

No caso de o utilizador não executar o *plugin* `pslist`, a página 'Deteções' do relatório informa-o que não existem deteções para apresentar, uma vez que o referido *plugin* não foi executado (Figura 84).

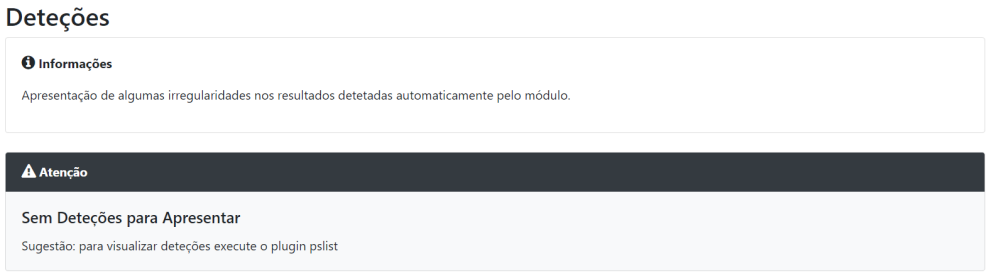


Figura 84: Apresentação do aviso apresentado na página 'Deteções' quando não existem deteções

7.6 SÍNTESE

Este capítulo é relativo ao *report module* desenvolvido nesta dissertação que tem o objetivo de mostrar os resultados presentes no *blackboard* do Autopsy e ainda apresentar, se possível, informação adicional, criando, para tal, um relatório HTML. Assim, neste capítulo são abordados alguns aspetos e conceitos relacionados com o desenvolvimento do *report module*. É ainda clarificado neste capítulo que os dados que irão constituir o relatório provêm de três fontes que são a base de dados do caso do Autopsy, o *blackboard* do Autopsy e os ficheiros JSON auxiliares.

A *BeautifulSoup* é uma biblioteca *Python* que foi utilizada de modo a se criar as páginas HTML que constituem o relatório, tendo como base para cada uma das mesmas o devido *template*. Neste capítulo são apresentados alguns aspetos desta

biblioteca, bem como exemplos da sua utilização provenientes do código do *report module* criado.

Por fim, é apresentada e descrita cada página HTML que constitui o relatório referindo-se, por exemplo, a origem dos dados presentes em cada uma das páginas.

8

CONCLUSÃO

Neste capítulo são apresentadas as conclusões acerca do trabalho desenvolvido e ainda algumas propostas de trabalho futuro.

8.1 CONCLUSÕES

A dissertação desenvolvida focou-se nos processos de aquisição e análise de memória e nas tarefas inerentes a estes. Para além disso, contemplou ainda a implementação de um módulo relativo à fase de análise. Inicialmente, foram definidos um conjunto de objetivos, os quais foram cumpridos.

O estudo e análise de conceitos e procedimentos relacionados com a aquisição e análise de memória é um dos objetivos cumpridos e foi algo que permitiu clarificar alguns aspetos acerca destes dois processos. Para além disso, verifica-se que a realização de uma investigação digital forense exige a consideração de certas variáveis e existem diversos conceitos relacionados com a mesma.

Um outro objetivo delineado e concretizado foi a realização da exploração de ferramentas de aquisição e análise, que permitiu perceber o funcionamento das mesmas.

Foram exploradas quatro ferramentas de aquisição total da memória e uma ferramenta de aquisição de processos específicos, nomeadamente a MAGNET Process Capture. Relativamente às ferramentas de aquisição total exploradas, todas permitem obter um ficheiro com o conteúdo da memória de um sistema. Verificou-se que o modo de funcionamento destas é muito semelhante e simples. Com os testes realizados, constatou-se que a ferramenta mais rápida a efetuar uma aquisição é a AcessData FTK. É de referir que esta ferramenta permite obter os ficheiros `hiberfil.sys`, `swapfile.sys` e ainda o `pagefile.sys`.

Já no que toca às ferramentas de análise da memória, foi explorado o Volatility (versões 2.6 e 3.0 *Public Beta*) e o Rekall. Verificou-se que estas ferramentas são muito semelhantes no que toca ao seu funcionamento. Duas das diferenças existentes é o facto de a versão 3.0 *Public Beta* do Volatility conseguir detetar o perfil

automaticamente e o facto do Rekall permitir filtrar os resultados através da linguagem EFilter.

Ao longo da exploração das ferramentas de aquisição, aquando o teste das mesmas, foram realizadas aquisições à máquina utilizada nesta dissertação, as quais permitiram obter com sucesso as imagens de memória. Contudo, não foi possível analisar os ficheiros obtidos nas aquisições realizadas com algumas das ferramentas de análise em estudo nesta dissertação. Todavia, o processo de análise de imagens de memória disponíveis na *web* (por exemplo, a 'ch2.dmp') nas mesmas ferramentas de análise é bem sucedido.

O estudo de ferramentas de análise do *clipboard* foi realizado com sucesso e permitiu concluir que existem poucas ferramentas de análise do *clipboard*, sendo esta uma área ainda não muito explorada. Todas as ferramentas exploradas só permitem visualizar o elemento mais recente no *clipboard*.

Nesta dissertação foram desenvolvidos dois módulos para Autopsy - um *data source ingest module* designado de RAMAnalysis e um *report module* designado de RAMAnalysis Report - no sentido de auxiliar os investigadores na tarefa de análise dos dados obtidos na fase de aquisição.

O objetivo do *data source ingest module* consiste em efetuar a análise de imagens de memória, apresentando os dados presentes nas mesmas ao utilizador. Este módulo faz uso do Volatility. Assim, após o utilizador selecionar quais o *plugins* que pretende executar, os resultados destes são armazenados em formato JSON e apresentados no *blackboard* do Autopsy, tal como se pretendia. Foi possível obter os dados do Volatility já no formato JSON, o que tornou desnecessária qualquer conversão dos mesmos, pois estando os dados nesse formato o módulo já os consegue processar sem problema.

Em relação ao *report module* desenvolvido, este cria um relatório HTML que terá os resultados obtidos da execução do módulo anterior. Um dos objetivos desta dissertação consistia em que fosse possível a ferramenta detetar situações suspeitas e apresentar dados que auxiliassem o investigador a encontrar evidências relevantes. Deste modo, a execução do *report module* poderá ser útil ao investigador uma vez que o relatório HTML apresenta automaticamente alguma informação obtida por análise dos resultados obtidos com o Volatility e que poderá ser um auxílio no processo de investigação. Possivelmente esta informação adicional também poderá ser um ponto de partida para a análise à memória.

Ao longo do desenvolvimento da dissertação foram delineados novos objetivos, como, por exemplo, a criação de gráficos `.dot` com os resultados. Com a exploração

do Volatility verificou-se que é possível obter os resultados de alguns *plugins* em gráficos DOT o que, em certos casos, poderá auxiliar na compreensão dos resultados obtidos. Para facilitar a visualização dos gráficos `.dot`, estes são convertidos pelo módulo para o formato `.png`.

8.2 TRABALHO FUTURO

Os dois módulos desenvolvidos nesta dissertação já permitem realizar uma boa análise a imagens de memória através do Volatility e até mesmo, se possível, efetuar certas deteções automaticamente. Contudo, existem melhorias que podem ser efetuadas no sentido de permitir uma experiência de utilização ainda melhor. Assim, nesta secção serão apresentadas algumas propostas de trabalho futuro a ser desenvolvido nos módulos criados.

Relativamente ao *data source ingest module*, a visualização dos ficheiros `.png` criados por conversão dos ficheiros `.dot` na interface do Autopsy permitiria que o utilizador visualizasse as imagens geradas sem sair da interface do Autopsy.

No *data source ingest module*, até ao momento, para se executar um *plugin* com parâmetros adicionais, na interface só se pode selecionar esse mesmo *plugin* pois caso contrário não é possível adicionar parâmetros ao comando. Assim, uma proposta de trabalho futuro para o *data source ingest module* passa por tornar possível definir diferentes parâmetros adicionais selecionando vários *plugins* em simultâneo, de modo a otimizar a utilização do módulo pelo utilizador.

Verificou-se que, em alguns casos, não é possível obter o *output* de alguns *plugins* do Volatility em formato JSON. Encontrar uma solução que permita apresentar no Autopsy os resultados que o módulo RAMAnalysis não consiga obter em formato JSON é algo a realizar.

No que toca ao *report module* existem também algumas sugestões de trabalho futuro. Uma funcionalidade interessante a integrar no *report module* consiste no envio dos ficheiros `.exe` relativos aos processos para o *website* VirusTotal. Deste modo, torna-se possível verificar de uma forma mais rápida e automatizada se um dado ficheiro contém conteúdo malicioso pois o processo de extração e envio do ficheiro é realizado pelo módulo e não pelo utilizador. Dois aspetos a mencionar são o facto de o VirusTotal possuir uma API[1] e o facto de existirem *scripts* já

1 https://developers.virustotal.com/v3.0/reference

elaborados em diferentes linguagens de programação que permitem interagir com a API do VirusTotal.

No *report module* a apresentação de gráficos sobre os processos auxilia a análise dos processos encontrados. Um tipo de gráfico que seria útil é o gráfico *timeline* pois este permite visualizar e analisar o comportamento dos processos ao longo do tempo, com base no começo e término de cada processo.

Ainda no que diz respeito ao *report module*, as deteções automáticas pelo módulo permitem que o utilizador tenha acesso direto a alguns dados suspeitos detetados da análise efetuada. Uma proposta de trabalho futuro passa por possibilitar a apresentação de mais deteções com base na informação recolhida na análise com o Volatility. Um exemplo de uma nova deteção a apresentar seria casos em que um ficheiro com um nome igual ao nome de um dos ficheiros legítimos do sistema (por exemplo, o ficheiro `iexplore.exe`) se encontre fora da sua localização legítima. A importância desta deteção está relacionada com o facto de, por vezes os ficheiros que contêm *malware* apresentam o mesmo nome que os ficheiros legítimos do sistema de modo a passar despercebidos.

A existência de técnicas e processos automatizados de procura de código malicioso na imagem de memória em análise é algo interessante a ser explorado e desenvolvido futuramente.

BIBLIOGRAFIA

A, Monnappa K (2016). *Detecting Malicious Processes Using Psinfo Volatility Plugin*. URL: https://cysinfo.com/detecting-malicious-processes-psinfo-volatility-plugin/ (acedido em 18/02/2020).

AcessData (2020). *FTK® Imager 3.4.2*. URL: http://marketing.accessdata.com/ftkimager3.1.1 (acedido em 10/11/2019).

Agarwal, Sandeep (2012). *What is the Difference Between NTFS and FAT 32 File Systems*. URL: https://www.guidingtech.com/11205/difference-between-ntfs-and-fat-32-file-systems/ (acedido em 27/09/2019).

Ajinkya (2020). *What Is Swapfile.sys? Can I delete Swapfile.sys?* URL: https://devsjournal.com/what-is-swapfile-sys.html (acedido em 10/08/2020).

Asipu, Pawan (2020). *Python String | split()*. URL: https://www.geeksforgeeks.org/python-string-split/ (acedido em 13/09/2020).

Autopsy (2020). *Python Autopsy Module Tutorial 3: The Report Module*. URL: https://www.autopsy.com/python-autopsy-module-tutorial-3-the-report-module/ (acedido em 10/05/2020).

Balapure, Aditya (2018). *Memory Forensics and Analysis Using Volatility*. URL: https://resources.infosecinstitute.com/memory-forensics-and-analysis-using-volatility/ (acedido em 11/10/2019).

Bootstrap (2019). *Introduction*. URL: https://getbootstrap.com/docs/4.1/getting-started/introduction/ (acedido em 09/09/2020).

Brinkmann, Martin (2009). *Forensic Windows Registry Software Registry Report*. URL: https://www.ghacks.net/2009/08/14/forensic-windows-registry-software-registry-report/ (acedido em 10/11/2019).

Carpenter, Tom (2011). *Microsoft Windows Server Administration Essentials*. Ed. por Inc. John Wiley & Sons. Sybex. (Acedido em 20/02/2020).

Case, Andrew e Golden G Richard III (2017). «Memory forensics: The path forward». Em: *Digital Investigation* 20, pp. 23–33. URL: https://doi.org/10.1016/j.diin.2016.12.004.

Casey, Eoghan (2009). *Handbook of digital forensics and investigation*. Academic Press.

Chandel, Raj (set. de 2015a). *4 ways Capture Memory for Analysis (Memory Forensics)*. URL: https://www.hackingarticles.in/4-ways-capture-memory-for-analysis-memory-forensics/ (acedido em 22/12/2019).

— (out. de 2015b). *Forensics Analysis of Pagefile and hibersys File in Physical Memory*. URL: https://www.hackingarticles.in/forensics-analysis-of-pagefile-and-hibersys-file-in-physical-memory/ (acedido em 22/12/2019).

Cohen, Michael (2017). *Rekall Agent - OSDFCon 2017*. URL: http://www.osdfcon.org/presentations/2017/Cohen_Rekall-Agent.pdf (acedido em 10/10/2019).

Cohen, Mike (2019). *TypeError: Set() missing 1 required positional argument: 'value'*. Github - Rekall Issues - solução segundo problema execução rekall. URL: https://github.com/google/rekall/issues/493 (acedido em 24/10/2019).

Darknet (2016). *Volatility Framework – Advanced Memory Forensics Framework*. URL: https://www.darknet.org.uk/2016/09/volatility-framework-advanced-memory-forensics-framework/ (acedido em 06/10/2019).

Defense, Forward (2017). *Windows Memory Analysis with Volatility*. Rel. téc. Forward Defense. URL: https://www.forwarddefense.com/pdfs/Memory-Analysis-with-Volatility.pdf (acedido em 09/10/2019).

Diffen (2019). *FAT32 vs. NTFS*. URL: https://www.diffen.com/difference/FAT32_vs_NTFS (acedido em 28/09/2019).

ExeFiles (mar. de 2020). *Resolver erros de Blue Screen do Swapfile.sys (Download gratuito)*. URL: https://www.exefiles.com/pt/sys/swapfile-sys/ (acedido em 09/02/2020).

Explorer, UFS (2019). *Understanding file systems*. URL: https://www.ufsexplorer.com/articles/file-systems-basics.php (acedido em 26/09/2019).

fabricmagic72 (2019). *A collection of malware samples caught by several honeypots i manage*. URL: https://github.com/fabrimagic72/malware-samples (acedido em 12/12/2019).

Fisher, Tim (2019). *What Exactly is a File System?* URL: https://www.lifewire.com/what-is-a-file-system-2625880 (acedido em 10/12/2019).

Forensics, Autopsy Digital (2013). *Autopsy Feature: Graphical Timeline Analysis for Cyber Forensics*. URL: https://www.autopsy.com/autopsy-feature-graphical-timeline-analysis-for-cyber-forensics/ (acedido em 09/10/2019).

— (2015). *Python Autopsy Module Tutorial 2: The Data Source Ingest Module*. URL: https://www.autopsy.com/python-autopsy-module-tutorial-2-the-data-source-ingest-module/ (acedido em 18/06/2020).

Forensics, Blog Rekall (abr. de 2015). *The Pmem Memory acquisition suite*. URL: http://blog.rekall-forensic.com/2015/04/the-pmem-memory-acquisition-suite.html (acedido em 23/12/2019).

— (2016). *Searching memory with Rekall*. URL: http://blog.rekall-forensic.com/2016/07/searching-memory-with-rekall.html (acedido em 15/11/2019).

Forensics, MAGNET (2015). *Acquiring Memory with Magnet RAM Capture*. URL: https://www.magnetforensics.com/blog/acquiring-memory-with-magnet-ram-capture/ (acedido em 20/10/2019).

— (2017). *Memory Analysis with Magnet RAM Capture and Magnet AXIOM*. URL: https://www.magnetforensics.com/resources/memory-analysis-magnet-ram-capture-magnet-axiom-recorded-webinar/?submission=https://go.magnetforensics.com/l/52162/2017-02-15/h2pqpv (acedido em 11/09/2020).

— (2018). *Magnet Process Capture – Saving Your Memory, One Process at a Time*. URL: http://www.magnetforensics.com/blog/magnet-process-capture/ (acedido em 13/11/2019).

— (2019a). *MAGNET RAM Capture*. URL: https://www.magnetforensics.com/resources/magnet-ram-capture/ (acedido em 20/10/2019).

— (2019b). *MAGNET Process Capture*. URL: https://www.magnetforensics.com/resources/magnet-process-capture/ (acedido em 16/10/2019).

Forensics, Rekall (2019a). *EFilter - A query language for Rekall*. URL: https://rekall.readthedocs.io/en/latest/efilter.html (acedido em 14/11/2019).

— (2019b). *Rekall At a Glance*. URL: http://www.rekall-forensic.com/documentation-1/rekall-documentation/rekall-at-a-glance (acedido em 12/10/2019).

— (2019c). *Rekall Tutorial*. URL: http://www.rekall-forensic.com/documentation-1/rekall-documentation/tutorial (acedido em 28/10/2019).

— (2019d). *Windows Plugins - Support for Windows memory analysis*. URL: http://web.rekall-innovations.com/docs/Manual/Plugins/Windows/index.html?v=13 (acedido em 21/12/2019).

Forensics, SANS Digital e Incident Response Blog (2009). *Forensics 101: Acquiring an Image with FTK Imager*. URL: https://digital-forensics.sans.org/blog/2009/06/18/forensics-101-acquiring-an-image-with-ftk-imager/ (acedido em 10/11/2019).

Forensics, SANS Digital e Incident Response (2019). *Community: Downloads*. URL: https://digital-forensics.sans.org/community/downloads (acedido em 22/12/2019).

Foundation, Python Software (2019). *json — JSON encoder and decoder*. URL: https://docs.python.org/3.4/library/json.html (acedido em 12/09/2020).

— (2020a). *Built-in Functions*. URL: https://docs.python.org/3/library/functions.html (acedido em 18/06/2020).

— (2020b). *os — Miscellaneous operating system interfaces*. URL: https://docs.python.org/3/library/os.html (acedido em 20/06/2020).

— (2020c). *os.path — Common pathname manipulations*. URL: https://docs.python.org/3/library/os.path.html (acedido em 16/06/2020).

— (2020d). *shutil — High-level file operations*. URL: https://docs.python.org/3/library/shutil.html (acedido em 18/06/2020).

Frade, Miguel e Baltazar Rodrigues (2019). «RAM Analysis». (Acedido em 26/09/2019).

Gaijin (2019). *RegistryReport*. URL: https://www.gaijin.at/en/software/registryreport (acedido em 10/11/2019).

GitHub, Inc. (2019). *Visualizing additions and deletions to content in a repository*. URL: https://help.github.com/en/github/visualizing-repository-data-with-graphs/visualizing-additions-and-deletions-to-content-in-a-repository (acedido em 20/12/2019).

Gomes, Carlos et al. (2019). «Análise Forense de Memória com Rekall». (Acedido em 11/11/2019).

Hoffman, Chris (jul. de 2017a). *Beginner Geek: Hard Disk Partitions Explained*. URL: https://www.howtogeek.com/184659/beginner-geek-hard-disk-partitions-explained/ (acedido em 04/08/2020).

— (2017b). *What Is Memory Compression in Windows 10?* URL: https://www.howtogeek.com/319933/what-is-memory-compression-in-windows-10/ (acedido em 05/11/2019).

— (set. de 2018). *Using Windows 10's New Clipboard: History and Cloud Sync*. URL: https://www.howtogeek.com/351978/using-windows-10s-new-clipboard-history-and-cloud-sync/ (acedido em 04/01/2020).

Hub, Windows O. S. (2018). *Memory Compression Process: High Memory and CPU Usage in Windows 10*. URL: http://woshub.com/memory-compression-process-high-usage-windows-10/ (acedido em 06/11/2019).

Hunt, Cale (jul. de 2019). *How to disable Windows 10 fast startup (and why you'd want to)*. URL: https://www.windowscentral.com/how-disable-windows-10-fast-startup (acedido em 14/02/2020).

Infosec (2019). *Autopsy: a platform overview*. URL: https://resources.infosecinstitute.com/category/computerforensics/introduction/free-open-source-tools/autopsy-forensics-platform-overview/ (acedido em 19/10/2019).

Institute, Information Sciences (set. de 1981). *Transmission Control Protocol*. RFC 793. DOI: 10.17487/RFC0793. URL: https://rfc-editor.org/rfc/rfc793.txt.

IPOG, blog (2018). *Sistema IPED: Conheça as principais funcionalidades do software utilizado na investigação da Operação Lava Jato*. URL: https://blog.ipog.edu.br/tecnologia/sistema-iped-software-usado-pela-policia-federal/ (acedido em 10/08/2020).

Joshi, Vineet (2017). *The Rekall Forensic and Incident Response Framework*. URL: https://blog.cloud-elements.com/using-json-over-xml (acedido em 01/10/2020).

JSONOrg (2020). *Introducing JSON*. URL: https://www.json.org/json-en.html (acedido em 15/09/2020).

Kuratomi, Toshio (2015). *Why sys.setdefaultencoding() will break code*. URL: https://anonbadger.wordpress.com/2015/06/16/why-sys-setdefaultencoding-will-break-code/ (acedido em 02/11/2020).

Ligh Hale, Michael et al. (2014). *The Art of Memory Forensics: detecting malware and threats in Windows, Linux, and Mac memory*. John Wiley & Sons. (Acedido em 01/09/2019).

martijnveken (2012). *Automating Volatility*. URL: https://forensec.wordpress.com/2012/11/13/automating-volatility-2/ (acedido em 25/11/2019).

McKinnon, Mark (2017). *Volatility Autopsy Plugin Module*. URL: https://medium.com/@markmckinnon_80619/volatility-autopsy-plugin-module-8beecea6396 (acedido em 13/10/2019).

— (2019). *markmckinnon/Autopsy-Plugins*. Repositório módulo Volatility para o Autopsy. URL: https://github.com/markmckinnon/Autopsy-Plugins/tree/master/Volatility (acedido em 13/10/2019).

Microsoft (2018). *Window Station and Desktop Creation*. URL: https://docs.microsoft.com/en-us/windows/win32/winstation/window-station-and-desktop-creation (acedido em 12/12/2019).

— (2020). *Get help with clipboard*. URL: https://support.microsoft.com/en-in/help/4464215/windows-10-get-help-with-clipboard (acedido em 21/02/2020).

Mike Auty, Andrew Case (2019). *Volatility 3 Public Beta: Insider's Preview*. Inglês. OSDFCon. URL: https://www.osdfcon.org/presentations/2019/Volatility-3-Public-Beta.pdf (acedido em 03/12/2019).

networkinghowtos (abr. de 2013). *Windows Virtual Store Location*. URL: https://www.networkinghowtos.com/howto/windows-virtual-store-location/ (acedido em 03/08/2020).

NirSoft (2020). *InsideClipboard v1.15 Copyright (c) 2007-2017 Nir Sofer*. URL: https://www.nirsoft.net/utils/inside_clipboard.html (acedido em 15/02/2020).

Okolica, James e Gilbert Peterson L. (2011). «Extracting the Windows clipboard from physical memory». Em: *Digital Investigation* 8, S118–S124. URL: https://www.sciencedirect.com/science/article/pii/S1742287611000387 (acedido em 06/01/2020).

Omar Sardar, Dimiter Andonov (2019a). *Extracting Compressed Pages from the Windows 10 Virtual Store*. Rel. téc. Fireeye. URL: https://i.blackhat.com/USA-19/Thursday/us-19-Sardar-Paging-All-Windows-Geeks-Finding-Evil-In-Windows-10-Compressed-Memory-wp.pdf (acedido em 29/10/2019).

— (2019b). *Finding Evil In Windows 10 Compressed Memory*. URL: https://i.blackhat.com/USA-19/Thursday/us-19-Sardar-Paging-All-Windows-Geeks-Finding-Evil-In-Windows-10-Compressed-Memory.pdf (acedido em 03/11/2019).

Phillips, Gavin (2019). *6 Free Hash Checkers to Check the Integrity of Any File*. URL: https://www.makeuseof.com/tag/free-hash-checkers-file-integrity/ (acedido em 11/12/2019).

Reichl, Dominik (2020a). *KeePass Features*. URL: https://keepass.info/features.html (acedido em 29/08/2020).

— (2020b). *KeePass Password Safe*. URL: https://keepass.info/ (acedido em 31/08/2020).

Rekall (mai. de 2016). *Release 1.5.1 Furka*. URL: https://github.com/google/rekall/releases (acedido em 26/12/2019).

— (2019a). *Insights - Code Frequency*. URL: https://github.com/google/rekall/graphs/code-frequency (acedido em 20/12/2019).

— (2019b). *Rekall Memory Forensic Framework*. URL: https://github.com/google/rekall (acedido em 20/12/2019).

— (2019c). *rekall/LICENSE.txt*. URL: https://github.com/google/rekall/blob/master/LICENSE.txt (acedido em 12/12/2019).

Ribeiro, Arlindo et al. (2019). «RAM with Rekall». (Acedido em 19/10/2019).

Richardson, Leonard (2020). *Beautiful Soup Documentation*. URL: https://www.crummy.com/software/BeautifulSoup/bs4/doc/ (acedido em 19/09/2020).

Ruff, Nicolas (2008). «Windows memory forensics». Em: *Journal in Computer Virology* 4.2, pp. 83–100.

Sanders, Chris (2011). *Determining If You are Actively Being Compromised*. URL: http://techgenix.com/determining-you-actively-being-compromised/ (acedido em 05/09/2020).

Scott, Jeremy (mai. de 2016). *Hibernation and Page File Analysis*. URL: https://technical.nttsecurity.com/post/102dwiw/hibernation-and-page-file-analysis (acedido em 22/12/2019).

— (2017). *Hunting malware with memory analysis*. URL: https://technical.nttsecurity.com/post/102egyy/hunting-malware-with-memory-analysis (acedido em 09/10/2019).

SecTechno (dez. de 2018). *WinPmem – Memory Acquisition Tool*. URL: https://sectechno.com/winpmem-memory-acquisition-tool/ (acedido em 26/12/2019).

Sieber, Tina (2019). *How to Format a Large Hard Drive With FAT or FAT32*. URL: https://www.makeuseof.com/tag/format-large-hard-drive-fat-fat32/ (acedido em 05/08/2020).

Singh, Azad, Pankaj Sharma e RajenderNath (dez. de 2016). «Role of Hibernation File in Memory Forensics of windows 10». Em: *International Journal of Scientific & Engineering Research* 7, pp. 42, 43, 44, 45, 46, 47. ISSN: 2229-5518. URL: https://pdfs.semanticscholar.org/5426/d4e0a2fa22bab95cb7b2a355f0cd24023a90.pdf (acedido em 21/12/2019).

Singh, B. e U. Singh (2017). «Program Execution Analysis using UserAssist Key in Modern Windows». Em: *SECRYPT* 4, pp. 420–429. URL: https://www.scitepress.org/Papers/2017/64167/64167.pdf (acedido em 09/10/2019).

soji256 (2019). *How to Find Missing Process ID When Using netscan - Volatility*. URL: https://medium.com/@soji256/how-to-search-for-unknown-process-id-in-volatilitys-netscan-39e16fcdaa9a (acedido em 28/08/2020).

Stancill, Blaine, Sebastian Vogl e Omar Sardar (2019). *Finding Evil in Windows 10 Compressed Memory, Part One: Volatility and Rekall Tools*. URL: https://www.fireeye.com/blog/threat-research/2019/07/finding-evil-in-windows-ten-compressed-memory-part-one.html (acedido em 03/11/2019).

Tagliaferri, Lisa (2019). *How To Scrape Web Pages with Beautiful Soup and Python 3*. URL: https://github.com/google/rekallhttps://www.digitalocean.com/community/tutorials/how-to-scrape-web-pages-with-beautiful-soup-and-python-3 (acedido em 01/10/2020).

The Sleuth Kit - TSK (2018). *UI Layout*. URL: https://sleuthkit.org/autopsy/docs/user-docs/4.5.0/uilayout_page.html (acedido em 07/08/2020).

Tróia, Pedro (2017). *Para que servem o FAT32, NTFS e exFAT*. URL: https://www.pcguia.pt/2017/06/as-diferencas-fat-32-ntfs-exfat/ (acedido em 28/09/2019).

TSK, The Sleuth Kit - (2015a). *Installing 3rd-Party Modules*. URL: https://sleuthkit.org/autopsy/docs/user-docs/3.1/module_install_page.html (acedido em 13/10/2019).

— (2015b). *SleuthkitCase Class Reference*. URL: http://sleuthkit.org/sleuthkit/docs/jni-docs/4.8.0/classorg_1_1sleuthkit_1_1datamodel_1_1_sleuthkit_case.html (acedido em 02/09/2020).
— (2018a). *Autopsy - Module Development Overview*. URL: http://www.sleuthkit.org/autopsy/docs/api-docs/4.4/platform_page.html (acedido em 14/10/2019).
— (2018b). *Autopsy User Documentation - Ingest Modules*. URL: http://sleuthkit.org/autopsy/docs/user-docs/4.8.0/ingest_page.html (acedido em 08/10/2019).
— (2018c). *Autopsy User Documentation - Live Triage*. URL: http://sleuthkit.org/autopsy/docs/user-docs/4.8.0/live_triage_page.html (acedido em 09/10/2019).
— (2018d). *Autopsy User Documentation - Quick Start Guide*. URL: http://sleuthkit.org/autopsy/docs/user-docs/4.8.0/quick_start_guide.html (acedido em 07/10/2019).
— (2018e). *Autopsy User Documentation - Setting Up Multi-user Environment*. URL: http://sleuthkit.org/autopsy/docs/user-docs/4.8.0/install_multiuser_page.html (acedido em 07/10/2019).
— (2018f). *Developing Report Modules*. URL: http://www.sleuthkit.org/autopsy/docs/api-docs/4.4/mod_report_page.html (acedido em 12/12/2019).
— (2020a). *Autopsy User Documentation - Experimental Module*. URL: http://sleuthkit.org/autopsy/docs/user-docs/4.15.0/experimental_page.html (acedido em 18/05/2020).
— (2020b). *Autopsy User Documentation - Volatility Data Source Processor*. URL: http://sleuthkit.org/autopsy/docs/user-docs/4.15.0/volatility_dsp_page.html (acedido em 17/05/2020).
— (2019a). *Autopsy*. URL: https://www.sleuthkit.org/autopsy/ (acedido em 09/10/2019).
— (2019b). *Open Source Digital Forensics*. URL: https://www.sleuthkit.org/ (acedido em 08/10/2019).
— (2019c). *Timeline Analysis*. URL: https://www.sleuthkit.org/autopsy/timeline.php (acedido em 09/10/2019).
vmware (2019). *Enable a Shared Folder for a Virtual Machine*. URL: https://pubs.vmware.com/workstation-11/index.jsp?topic=%2Fcom.vmware.ws.using.doc%2FGUID-D6D9A5FD-7F5F-4C95-AFAB-EDE9335F5562.html (acedido em 21/12/2019).

Volatility Foundation (2016). *Unified Output*. URL: https://github.com/volatilityfoundation/volatility/wiki/Unified-Output (acedido em 22/08/2020).

— (2017). *Command Reference Mal*. URL: https://github.com/volatilityfoundation/volatility/wiki/Command-Reference-Mal (acedido em 20/12/2019).

— (2019a). *About The Volatility Foundation*. URL: https://www.volatilityfoundation.org/about (acedido em 07/10/2019).

— (2019b). *Command Reference*. URL: https://github.com/volatilityfoundation/volatility/wiki/Command-Reference (acedido em 09/10/2019).

— (2019c). *Insights - Code Frequency*. URL: https://github.com/volatilityfoundation/volatility/graphs/code-frequency (acedido em 20/12/2019).

— (2019d). *Volatility Framework - Volatile memory extraction utility framework*. URL: https://github.com/volatilityfoundation/volatility (acedido em 08/10/2019).

w3schools (2019). *Font Awesome Introduction*. URL: https://www.w3schools.com/icons/fontawesome_icons_intro.asp (acedido em 18/09/2020).

Wikipedia (2020). *DOT (graph description language)*. URL: https://en.wikipedia.org/wiki/DOT_(graph_description_language) (acedido em 18/10/2020).

WikiTechy (2020). *Solved-5 Solutions - UnicodeEncodeError*. URL: https://www.wikitechy.com/errors-and-fixes/python/unicodeencodeerror-ascii-codec-cant-encode-character-u-xa0 (acedido em 02/11/2020).

Zimmerman, Eric (2019). *Introducing KAPE – Kroll Artifact Parser and Extractor*. URL: https://www.kroll.com/en/insights/publications/cyber/kroll-artifact-parser-extractor-kape (acedido em 09/08/2020).

Zollner, Stephan, Kim-Kwang Raymond Choo e Nhien-An Le-Khac (out. de 2019). «An Automated Live Forensic and Postmortem Analysis Tool for Bitcoin on Windows Systems». Em: *IEEE Access* 7, pp. 158250–158263. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=8878085 (acedido em 26/12/2019).