Extração e Análise de Dados em Memória na Perícia Forense Computacional

Gilson Marques da Silva, Evandro Mário Lorens
Departamento de Polícia Federal - DPF

***Resumo* — Trata-se de um estudo considerando a relevância de perícia digital em computadores ligados e em funcionamento, conhecida como *live forensics*, especialmente abrangendo as perícias em memória volátil. Apresenta uma visão geral das informações armazenadas nesse tipo de memória e algumas ferramentas específicas para extrair e analisar dados a partir de *dumps* de memória; destaca procedimentos periciais importantes para uma correta análise de dados dessas memórias e por fim, apresenta um breve estudo de caso nesse contexto.**

Palavras-chave* — Perícia em computadores, Memória volátil, Vestígios digitais, Análise de memória, Crimes digitais, *Dump* de memória, *Volatility*, *Live forensics

I. Introdução

A popularização do uso dos computadores e o advento das redes de comunicação e da Internet trouxeram valiosos benefícios à sociedade em termos de comunicação, comodidade e serviços remotos. Paralelamente, introduziram a possibilidade de sua utilização para a realização de diversas atividades ilícitas que podem, inclusive, ter causadores e afetados distantes fisicamente entre si. Onde outrora era indispensável estar presente o elemento humano para a prática de determinados crimes, hoje não é mais. A disponibilidade de uma conexão à Internet e o uso de programas de computador especialmente elaborados, em muitos casos, é suficiente para permitir a qualquer indivíduo experimentar variadas condutas criminosas. Espionagem, invasões, furto e acesso indevido de informações, sabotagem, adulteração e destruição de informações, ataques contra serviços legítimos em rede, armazenamento e transmissão de fotos e vídeos de pedofilia, envio de mensagens comerciais não solicitadas, monitoração de tráfego de dados e comunicações são algumas das possibilidades.

Assim como no caso de crimes comuns, o combate aos crimes cibernéticos requer a existência de evidências e provas robustas de sua existência, que propiciem o pleno convencimento dos representantes dos poderes estabelecidos para julgar e punir condutas criminosas. A constituição de provas nos crimes cibernéticos se dá por meio da identificação, coleta e caracterização de vestígios digitais, ou seja, de dados e informações deixados pelos sistemas computacionais em periféricos, dispositivos de armazenamento e na própria memória volátil do computador.

Buscar vestígios digitais e caracterizá-los como evidências e provas de crime são atividades fundamentais da perícia forense computacional, que requerem cuidados e conhecimentos específicos. É preciso saber onde pesquisar por vestígios, que tipos de vestígios buscar, dispor de conhecimento técnico para coletar e preservar esses vestígios, e observar a minimização dos riscos de sua deterioração ou invalidação como prova.

Circunstâncias específicas apontam para a necessidade de se realizar procedimentos de coleta de vestígios digitais no local em que se encontram instalados os equipamentos computacionais, enquanto ligados e em funcionamento normal. Instalações de equipamentos de grande porte, não convencionais, ou que suscitem o risco de perda de informações significativas ou ainda, a inviabilização da perícia são exemplos dessas circunstâncias. Destaque-se também a situação cada vez mais freqüente da possibilidade de haver configurada proteção por criptografia sobre os conteúdos armazenados em dispositivos de mídia dos equipamentos computacionais, o que também justifica procedimentos de análise forense *live*.

Em análises forenses realizadas em equipamentos ligados e conectados, é possível identificar processos ativos, dados em uso pelos processos dispostos em memória, conexões de rede abertas, além de ser possível resgatar conteúdos de dados armazenados sem o efeito de uma possível e indesejada proteção criptográfica, considerado o ponto de vista do perito.

II. Dados em Memória

A. Visão geral das informações armazenadas na memória

Tudo o que está sendo executado em um computador é armazenado temporariamente na memória, seja na memória volátil, seja no arquivo de paginação relacionado com a memória virtual. Assim, por meio da extração de uma imagem da memória conhecida como *dump* de memória é possível identificar a relação dos processos em execução; é também viável estabelecer uma relação entre os processos de modo a identificar que processos tenham iniciado outros processos; da mesma forma, é possível identificar quais são os arquivos, bibliotecas, chaves de registro ou *sockets* que estavam em uso por cada processo. Em resumo, é possível mapear como o sistema estava sendo utilizado no momento da geração do *dump* da memória.

É possível também recuperar programas executáveis armazenados em memória, de modo a viabilizar ao perito a

análise desses códigos, especialmente aqueles desconhecidos e que eventualmente serão classificados como artefatos maliciosos capazes de executar ou facilitar a execução de crimes.

Da mesma forma que os programas em execução permanecem gravados em memória, seus dados de trabalho também assim se encontram. Assim, é possível que sejam recuperadas informações variadas, como, por exemplo, uma senha que porventura ainda esteja armazenada em forma textual na memória bem como outras informações específicas que podem ser recuperadas quando se conhece seu formato, posição e outras características úteis.

B. Ferramentas para gerar dumps de memória

Na maioria das vezes, para se conseguir um *dump* de memória, a opção mais simples e direta é a utilização de uma ferramenta específica para esta atividade. Tais ferramentas realizam uma leitura *bit-a-bit* da memória copiando todo o seu conteúdo para um arquivo, o próprio *dump* de memória. Como imagem espelho da memória, este arquivo terá o mesmo tamanho físico da memória do computador. Assim, em um computador com 2 *gigabytes* de memória RAM (*Random Access Memory)* o arquivo de *dump* terá os mesmos 2 *gigabytes*.

Embora existam diversas ferramentas voltadas à geração de *dumps* de memória, este artigo limita-se a apresentar duas ferramentas específicas que podem ser utilizadas para esta tarefa, de maneira simples e com resultados diretos e precisos.

Uma delas é o MDD (*ManTech Memory DD*) [5], ferramenta *freeware*, que pode ser executada nas diversas versões do Windows, desde o 2000 até o 2008, incluindo também o XP e Vista. O arquivo com o *dump* é gerado no formato binário sem formatação; é gerado por padrão também o *hash* do tipo MD5 (*Message Digest version 5*) do conteúdo do arquivo. Esse *hash* será útil na garantia da cadeia da custódia que será discutida em seção específica deste trabalho.

O MDD é um arquivo executável com menos de 100 *kilobytes* de tamanho, que não precisa ser instalado no sistema no qual se deseja fazer a coleta, podendo ser executado a partir de mídias removíveis, de maneira prática e simples. Basicamente, é preciso apenas informar o nome do arquivo que será gerado, como pode ser visto a seguir:

```
D:\>mdd_1.3.exe -o DumpMemoriaMDD.dmp
 -> mdd
 -> ManTech Physical Memory Dump Utility
    Copyright (C) 2008 ManTech Security & Mission Assurance
 -> This program comes with ABSOLUTELY NO WARRANTY; for details use option `-w'
    This is free software, and you are welcome to redistribute it under certain
conditions; use option `-c' for details.
 -> Dumping 2037.24 MB of physical memory to file 'DumpMemoriaMDD.dmp'.

 521534 map operations succeeded (100)
 0 map operations failed
 took 54 seconds to write
 MD5 is: 2f40ac650fbf88b53517a650a5256222
```

Outra opção de ferramenta é o Win32DD [15], também do tipo *freeware*, que pode ser executada na maioria das versões do Windows, não precisa ser instalado e tem menos de 100 *kilobytes*. O Win32DD também gera o *hash* do arquivo armazenado com o *dump* da memória, porém o algoritmo usado é o SHA1 (*Secure Hash Algotithm version 1*). Da mesma forma que o MDD o Win32DD gera um *dump* com toda a memória RAM no momento de sua execução, como pode ser visto a seguir:

```
D:\>win32dd.exe DumpMemoriaWin32DD.raw
 Win32dd - v1.2.2.20090608 - Kernel land physical memory acquisition
 Copyright (c) 2007 - 2009, Matthieu Suiche <http://www.msuiche.net>

  Name                  Value
  ----                  -----
  File type:            Raw dump
  Acquisition method:   \\Device\\PhysicalMemory
  Content:              Full physical address space
  Destination path:     \??\D:\DumpMemoriaWin32DD.raw
  O.S. Version:         Microsoft Windows Vista Home Premium Edition, 32-bit SP 1
  Computer name:        GILSONNTB

  Physical memory in use:          55%
  Physical memory size:       2086136 Kb (   2037 Mb)
  Physical memory available:   920168  Kb (    898 Mb)
  Paging file size:           4415272 Kb (   4311 Mb)
  Paging file available:      2943468 Kb (   2874 Mb)
  Virtual memory size:        2097024 Kb (   2047 Mb)
  Virtual memory available:   2069044 Kb (   2020 Mb)
  Extented memory available:  0 Kb             (        0 Mb)
  Physical page size:         4096 bytes
  Minimum physical address:   0x110000
  Maximum physical address:   0x7F66F000
  Address space size:         2137456640 bytes (2087360 Kb)
  Acquisition started at:     [6/7/2009 (DD/MM/YYYY) 18:8:5 (UTC)]

  Processing....Done.

  Acquisition finished at:    [6/7/2009 (DD/MM/YYYY) 18:13:44 (UTC)]
  Time elapsed:               5:38 minutes:seconds (338 secs)
  Created file size:          2137456640 bytes (   2038 Mb)
  NtStatus (troubleshooting): 0x00000103
  Total of written pages:     537135
  Total of inacessible pages: 0
  Total of accessible pages:  537135

  SHA1: B2688FE6D6404F28586AFE6B3A15209EF27C329A

  Physical memory in use:          56%
  Physical memory size:       2086136 Kb (   2037 Mb)
  Physical memory available:   914812  Kb (    893 Mb)
  Paging file size:           4415272 Kb (   4311 Mb)
  Paging file available:      2930420 Kb (   2861 Mb)
  Virtual memory size:        2097024 Kb (   2047 Mb)
  Virtual memory available:   2069044 Kb (   2020 Mb)
  Extented memory available:  0 Kb             (        0 Mb)
  Minimum physical address:   0x110000
  Maximum physical address:   0x7F66F000
  Address space size:         2137456640 bytes (2087360 Kb)
```

Independentemente da ferramenta utilizada para a geração do *dump* de memória, este pode ser processado pelo sistema *Volatility*, a ser discutido na sequência deste trabalho, desde que o *dump* tenha sido gerado no formato *raw*,

C. Outros meios para a obtenção de dumps de memória

Nem sempre será possível instalar uma ferramenta no sistema para o qual se deseja gerar o *dump* de memória. Em alguns casos, nem a simples execução será viável. Em especial, em computadores alvos de operações de busca e apreensão, nos quais o perito não pode alterar o sistema, sob pena de contaminação e nulidade da prova.

Uma primeira alternativa é buscar por arquivos de *dump* já armazenados no computador. Vale ressaltar que, neste caso, as informações do *dump* não corresponderão ao momento atual do sistema, mas a algum momento passado, quando aquele

arquivo foi gerado. O Windows, por exemplo, gera arquivos de *dump* quando ocorrem problemas severos na execução do sistema operacional, que são mais especificamente chamados de arquivos de *crash dump*. Por padrão estes arquivos são armazenados em "c:\windows\" ou "c:\winnt\" com o nome "memory.dmp". No caso do Windows XP e Vista, a opção padrão de geração do *dump* é em formato reduzido, na qual não se copia toda a memória, mas apenas a parte necessária à identificação e caracterização da falha ocorrida. No entanto, esses sistemas podem ser configurados para gerar *dumps* de memória completos, da mesma forma que fazem as ferramentas discutidas na seção anterior. O formato padrão dos sistemas Windows para servidores como o 2000, 2003 e 2008 é o formato completo. O formato de *dump* bem como a forma de armazenamento do mesmo podem ser configurados na opção "Configurações Avançadas do Sistema", guia "Avançado", item "Inicialização e Recuperação", no ícone "Sistema" do "Painel de Controle".

Uma alternativa complementar é coletar os arquivos de paginação "c:\pagefile.sys" ou o arquivo de hibernação "c:\hyberfil.sys". O primeiro contém parte da memória que tenha passado pelo processo de paginação [13]. O segundo contém uma cópia da memória RAM no momento em que o sistema foi comandado a entrar no modo de hibernação. Na maioria das vezes, os dois arquivos contêm informações privilegiadas ou confidenciais. Além disso, o arquivo de hibernação tem todas as informações da memória, tal qual um *dump* completo. Os arquivos de paginação e hibernação também podem ser explorados pela solução *Volatility*, como será apresentado em seção específica deste trabalho.

Caso a suspeita ou a necessidade de análise recaia sobre um processo específico, ou sobre um conjunto deles, é possível, no Windows Vista, a obtenção de um *dump* de memória somente daquele processo. O próprio sistema operacional fornece o mecanismo com a funcionalidade. Basta abrir o gerenciador de tarefas (CTRL+ALT+ESC), selecionar o processo desejado, clicar com o botão direito e acessar a opção "Criar despejo de memória". O arquivo será criado no diretório temporário do usuário logado, com o nome do processo seguido da extensão ".dmp". Como esta opção não existe no XP e o *Volatility* ainda não analisa *dumps* de memória do Vista, esta alternativa pode não oferecer resultados imediatos, mas poderá ser útil quando o *Volatility* passar a suportar os novos sistemas operacionais da Microsoft.

Finalmente, uma opção que pode trazer bons resultados para a atividade pericial quando o sistema está bloqueado e há suspeita de uso de sistemas de criptografia de armazenamento, é a exploração de uma vulnerabilidade de projeto do DMA (*Direct Memory Access*), que independe do sistema operacional em uso. Um ataque explorador viável é o "*Hit by bus, owned by iPod*" [1]. Como restrição, destaque-se que essa técnica só funcionará em sistemas que disponham de portas do tipo *firewire* ou *slots* do tipo *express card*. Outra ressalva a fazer é que este ataque nem sempre será bem sucedido, pois o sistema poderá reiniciar, a memória poderá não ser completamente copiada, ou poderá simplesmente não funcionar. Logo, deverá ser utilizado apenas como uma tentativa que poderá ou não gerar o resultado desejado. Sua concepção como um procedimento padrão e comprometido com resultados sempre efetivos não é adequada, porquanto uma tentativa sem sucesso poderá desgastar a credibilidade do perito. Obviamente, esse tipo de procedimento pode ser muito útil nos casos nos quais o sistema está bloqueado e utiliza solução para criptografia de armazenamento, e não se tem acesso à chave para acesso aos arquivos. Em tal cenário, as técnicas usuais provavelmente não gerariam resultados satisfatórios e a técnica de ataque referida poderia trazer algum melhor resultado. O estudo e o detalhamento desse ataque extrapolam o escopo deste trabalho, e portanto não serão discutidos.

III. PROCEDIMENTOS PERICIAIS PARA ANÁLISE DE DADOS EM MEMÓRIA

A. Atendimento a quesitos

Via de regra, os procedimentos periciais estão atrelados a investigações de crimes e objetivam responder perguntas e esclarecer aspectos circunstanciais sobre ocorrências de crimes como "o que ocorreu?", "onde?", "quando?", "qual o meio empregado?", "de que forma foi utilizado?", "quem foi o autor?". Este conjunto de seis perguntas, com a adição da questão "por que?", é conhecido no ramo da criminalística como o heptâmero das circunstâncias de Quintiliano [3] e constitui-se como fundamento nas metodologias de investigação criminal. No trabalho conjunto de investigação, cabe ao perito responder formalmente quesitos relacionados aos vestígios angariados. Os quesitos são usualmente elaborados pela autoridade policial condutora do processo investigatório, embora o Código de Processo Penal brasileiro em seu artigo 159, faculte tal prerrogativa também ao Ministério Público, ao assistente de acusação, ao ofendido, ao querelante e ao acusado [2].

No caso de perícias computacionais, especificamente as perícias em dados de memória volátil registrados em um *dump* de memória, são viabilizadas respostas para vários tipos de quesitos, a saber, dentre outros:

- Quais são a data e hora da imagem da memória?
- Que programas estavam em execução?
- Que serviços de rede o computador executava?
- Que serviços de rede o computador usava?
- A que outros computadores o computador periciado estava conectado?
- Que arquivos estavam em uso pelos programas em execução no computador?
- Que faixas de endereços de memória estavam em uso por cada programa?
- Que versões de sistemas operacionais estavam carregadas no computador?
- Que mapeamentos de endereços físicos para endereços virtuais havia no computador?

B. Cadeia de custódia

Basicamente, a cadeia de custódia pode ser entendida como um conjunto de processos de controle de integridade, disponibilidade e idoneidade do vestígio desde a sua descoberta até o seu efetivo descarte ou utilização como elemento probatório nas instâncias da Justiça.

As perícias em memória volátil também se subordinam aos aspectos da cadeia de custódia, e cuidados essenciais devem ser observados nos procedimentos de coleta, processamento e produção de resultados.

No procedimento de coleta dos dados de memória, é importante que a produção do *dump* de memória seja realizada de maneira automática, sem nenhuma intervenção ou interferência que possa afetar o conteúdo da memória enquanto ocorrer a extração. Também é essencial que o arquivo gerado para posterior análise seja uma imagem exata da memória, *bit* a *bit*, com vistas à manutenção da integridade do vestígio. Em geral, as ferramentas específicas para geração de *dumps* cuidam para que os aspectos acima sejam observados.

Imediatamente após a coleta, sugere-se que o arquivo de *dump* de memória seja submetido a uma função resumo (*hash*) que associará o conteúdo do arquivo a uma seqüência numérica única, de modo que, em termos práticos, caso ocorra qualquer alteração no conteúdo do arquivo, não seja possível validar ou reproduzir o mesmo *hash*. Outras providências interessantes são a gravação do arquivo em mídia não alterável ou a habilitação dos atributos do arquivo para "somente leitura".

Na fase de processamento dos dados de memória representados pelo arquivo de *dump*, sob a ótica da cadeia de custódia, devem ser observados os aspectos de funcionalidade, confiabilidade e eficiência como atributos de qualidade de *software* [4] do sistema empregado para extrair dados do arquivo de *dump*, considerando basicamente que o sistema realize exatamente aquilo que se propõe a realizar, de maneira funcional, confiável e eficiente, não permitindo erros de operação e especialmente da interpretação dos dados analisados.

Na formalização dos resultados, o perito deve cuidar em expor as evidências, suas interpretações e deve também apresentar a metodologia de análise e as etapas do seu trabalho, ressaltando os controles de garantia da cadeia de custódia.

C. Conclusões periciais

A perícia da memória volátil de um determinado computador por meio da análise de um *dump* de memória pode identificar inequivocamente a ocorrência de determinadas condutas delituosas realizadas com o apoio de computadores.

O modelo de arquitetura dos computadores modernos, baseado na arquitetura de Von Neumann [13] tem como característica básica o conceito de programa armazenado, que propõe que os programas em execução no computador devem estar carregados em memória, assim como os dados manuseados por eles. Em termos práticos e do ponto de vista pericial, isso significa que o uso de computador em atividades ilícitas sempre deixará vestígios na memória volátil em tempo de execução dos programas empregados.

A título ilustrativo são apresentadas a seguir algumas atividades maliciosas e alguns dos vestígios relacionados que podem ser extraídos da memória volátil em situações específicas e, consequentemente, subsidiar o trabalho pericial.

TABELA I
ATIVIDADES MALICIOSAS E VESTÍGIOS

– Furto de identidades e senhas com uso de programas espiões instalados remotamente	– Conexões com outros computadores usando serviços de transferência de arquivos suspeitos; – Códigos fontes carregados para edição com funcionalidades típicas; – Códigos executáveis componentes de soluções "cliente-servidor" típicas; – Arquivos de texto e/ou de imagens carregados para edição com mensagens forjadas em nome de instituições terceiras para ludibriar pessoas comuns; – Serviço ativo de *e-mail* recebendo mensagens automáticas com dados e senhas de contas bancárias, cartões de crédito e outros;
– Pedofilia	– Conexões com outros computadores usando serviços de transferência de arquivos suspeitos; – Arquivos de imagem ou vídeo suspeitos carregados para edição e visualização; – Histórico de navegação armazenado na *registry* contendo endereços de sítios de pedofilia; – Serviço ativo de *e-mail* e/ou mensagens instantâneas enviando ou recebendo mensagens com imagens e vídeos de pedofilia anexados;
– Ameaças e extorsões – Espionagem industrial	– Arquivos de texto, imagem, vídeos e outros contendo informações reservadas de pessoas, comerciais e/ou industriais, carregados para edição e visualização; – Serviço ativo de *e-mail* e/ou mensagens instantâneas recebendo mensagens relacionadas a ameaças, extorsões ou espionagem industrial;
– Spam	– Códigos executáveis de servidores simples de correio eletrônico; – Tabelas ou códigos carregados relacionando servidores de correio na Internet com configurações de *relay* abertas; – Arquivos de texto e/ou de imagens carregados para edição com mensagens comerciais não solicitadas – Tabelas carregadas com grande número de endereços de *e-mail*, identidades e perfis de sítios de relacionamento, e contas de serviços de mensagens instantâneas; – Serviço ativo de correio eletrônico ou mensagens instantâneas enviando mensagens com conteúdo comercial não solicitado;
– Hospedagem de arquivos ou sítios de pedofilia, pirataria, comércio de drogas, armas e outros ilícitos	– Serviço *web* ativo e servidores *web* carregados com páginas típicas; – Informações do serviço *web* e das páginas hospedadas na máquina armazenadas na *registry*; – Conexões *web* de entrada abertas com requisições para páginas hospedadas suspeitas;
– Invasões – Operação de *bots* – Uso de *rootkits*	– Códigos executáveis componentes de soluções "cliente-servidor" típicas; – Conexões com outros computadores usando serviços e portas não convencionais e suspeitos;
– Articulação criminosa – Terrorismo	– Arquivos, planilhas, mensagens instantâneas *ou e-mail* carregados com informações de alvos e comparsas, atribuições, endereços e telefones, movimentações financeiras suspeitas, planejamento e tarefas suspeitas;

IV. UMA OPÇÃO DE FERRAMENTA PARA EXTRAÇÃO DE DADOS DE *DUMPS*: *VOLATILITY*

A. Introdução à ferramenta Volatility

Volatility é uma coleção de ferramentas abertas destinada à extração de conteúdos digitais armazenados em memória volátil de sistemas operacionais *Windows XP*. Construído em linguagem *Python*, foi concebido para funcionar em qualquer plataforma de sistema operacional para a qual exista implementação de *Python*, de forma totalmente independente do sistema em investigação.

A ferramenta idealizada pela *Volatile Systems* propõe uma plataforma de apoio à coleta e análise de evidências digitais contidas em memória volátil, proporcionando a obtenção de um variado conjunto de informações ali armazenadas, a saber:

- Data e hora da imagem da memória RAM
- Processos em execução
- *Sockets* de rede abertos
- Conexões de rede abertas
- DLLs carregadas para cada processo
- Arquivos abertos para cada processo
- Chaves de registro para cada processo
- Memória endereçável de um processo
- Módulos do *kernel* do sistema operacional
- Mapeamento de endereços físicos para endereços virtuais
- Informações do *Virtual Address Descriptor*.

Como funcionalidades adicionais do conjunto *Volatility*, destacam-se a capacidade de realizar varredura no *dump* em busca de processos, *sockets*, conexões e módulos carregados, o suporte transparente a vários formatos de *dumps*, a conversão automatizada entre formatos e a possibilidade de uso de módulos de terceiros.

A arquitetura do *Volatility* sustenta-se basicamente em três pilares: espaço de endereços, objetos e perfis, e módulos de visão de dados. O espaço de endereços diz respeito à organização da memória; os objetos são uma abstração para os dados encontrados em memória; e os módulos de visão de dados referem-se à localização dos objetos na memória e a respectiva extração desses objetos [14].

B. Instalação

Nesta seção são apresentados os passos necessários para a correta instalação da ferramenta e de seus pré-requisitos em sistemas operacionais como o Windows XP ou Vista.

1. Realizar o *download* do *Python* [10] – necessário – o *Volatility* é formado por um conjunto de *scripts* desenvolvidos em *Python*, sendo essa plataforma necessária para a execução dos mesmos.
2. Instalar o *Python* em modo padrão – nenhuma alteração nas opções de instalação é necessária.
3. Realizar o *download* do *Volatility* [14].
4. Descompactar o arquivo baixado.
5. Abrir uma janela de *prompt* de comando e testar o funcionamento do *Python* e do *Volatility*, com o comando:

```
c:\Python30\python.exe c:\Volatility\volatility
    Volatile Systems Volatility Framework v1.3
    Copyright (C) 2007,2008 Volatile Systems
    Copyright (C) 2007 Komoku, Inc.
    This is free software; see the source for copying conditions.
    There is NO warranty; not even for MERCHANTABILITY or FITNESS FOR A
PARTICULAR PURPOSE.
    usage: c:\Volatility\volatility cmd [cmd_opts]
    Run command cmd with options cmd_opts
    For help on a specific command, run 'c:\Volatility\volatility cmd --help'
    Supported Internel Commands:
        connections     Print list of open connections
        connscan        Scan for connection objects
        connscan2       Scan for connection objects (New)
        datetime        Get date/time information for image
        dlllist         Print list of loaded dlls for each process
        dmp2raw         Convert a crash dump to a raw dump
        dmpchk          Dump crash dump information
        files           Print list of open files for each process
        hibinfo         Convert hibernation file to linear raw image
        ident           Identify image properties
        memdmp          Dump the addressable memory for a process
        memmap          Print the memory map
        modscan         Scan for modules
        modscan2        Scan for module objects (New)
        modules         Print list of loaded modules
        procdump        Dump a process to an executable sample
        pslist          Print list of running processes
        psscan          Scan for EPROCESS objects
        psscan2         Scan for process objects (New)
        raw2dmp         Convert a raw dump to a crash dump
        regobjkeys      Print list of open regkeys for each process
        sockets         Print list of open sockets
        sockscan        Scan for socket objects
        sockscan2       Scan for socket objects (New)
        strings         Match physical offsets to virtual addresses
        thrdscan        Scan for ETHREAD objects
        thrdscan2       Scan for thread objects (New)
        vaddump         Dump the Vad sections to files
        vadinfo         Dump the VAD info
        vadwalk         Walk the vad tree
    Supported Plugin Commands:
    Example: volatility pslist -f /path/to/my/file
```

O auxílio para cada opção pode ser obtido com o seguinte comando:

```
c:\Python30\python.exe  c:\Volatility\volatility pslist --help
            Usage: pslist [options] (see --help)
            Options:
              -h,               --help show this help message and exit
              -f FILENAME,      --file=FILENAME (required) XP SP2 Image file
              -b BASE,          --base=BASE  Physical offset of dir table base
              -t TYPE,          --type=TYPE  Identify the image type (pae, nopae, auto)
```

Desta forma, a ferramenta estará pronta para ser usada, porém é necessário que o arquivo com a imagem de memória esteja disponível para ser trabalhada.

A instalação em sistemas do tipo Linux praticamente não difere do que foi descrito nesta seção. Basta que o *Python* esteja instalado e os arquivos do *Volatility* estejam descompactados em uma pasta qualquer. Neste caso, é recomendável que as permissões dos diretórios e arquivos sejam configuradas de modo seguro, evitando a exposição desnecessária do sistema. Estas configurações extrapolam o escopo deste trabalho, contudo poderão ser consultadas em [12].

C. Funcionamento e uso

Nesta seção são apresentados os resultados de algumas opções disponibilizadas pela ferramenta Volatility para a extração de informações da memória. Uma boa alternativa

para os primeiros testes com a ferramenta é usar um arquivo com um *dump* de memória disponibilizado pelo NIST (*National Institute of Standards and Technology*), [8].

Primeiramente, a opção *ident* pode ser utilizada para identificar a imagem e mostrar a data e hora em que o *dump* foi gerado.

```
c:\Python30\python.exe c:\Volatility\volatility ident -f c:\memory-images\xp-laptop.img
        Image Name: c:\memory-images\xp-laptop.img
        Image Type: Service Pack 2
           VM Type: nopae
           DTB: 0x39000
           Datetime: Mon Jul 14 16:37:25 2009
```

Outra opção é gerar uma relação dos processos que estavam em execução. Esta relação inclui o identificador numérico do processo (*Pid*), o identificador numérico do pai do processo (*PPid*), bem como outras informações úteis para comprovar o que estava sendo executado na máquina.

c:\Python30\python.exe c:\Volatility\volatility **pslist** -f c:\memory-images\xp-laptop.img

Name	Pid	PPid	Thds	Hnds	Time
System	4	0	61	1140	Thu Jan 01 00:00:00 1970
smss.exe	448	4	3	21	Sat Jul 14 13:27:13 2009
csrss.exe	504	448	12	596	Sat Jul 14 13:27:40 2009
winlogon.exe	528	448	21	508	Sat Jul 14 13:27:41 2009
services.exe	580	528	18	401	Sat Jul 14 13:27:41 2009
lsass.exe	592	528	21	374	Sat Jul 14 13:27:42 2009
svchost.exe	740	580	17	198	Sat Jul 14 13:27:43 2009
svchost.exe	800	580	10	302	Sat Jul 14 13:27:44 2009
svchost.exe	840	580	83	1589	Sat Jul 14 13:27:44 2009
Smc.exe	876	580	22	423	Sat Jul 14 13:27:45 2009
svchost.exe	984	580	6	90	Sat Jul 14 13:27:47 2009
firefox.exe	2160	1812	6	182	Sat Jul 14 13:27:53 2009
PluckSvr.exe	944	740	9	227	Sat Jul 14 13:27:57 2009
iexplore.exe	2392	1812	9	365	Sat Jul 14 13:28:01 2009
PluckTray.exe	2740	944	3	105	Sat Jul 14 13:28:03 2009

Também de grande utilidade para os procedimentos periciais é a geração da relação de conexões e *sockets* que estavam ativos no sistema no momento do *dump* de memória. Para isso, as opções *connscan* e *sockets* podem ser utilizadas.

c:\Python30\python.exe c:\Volatility\volatility **connscan** -f c:\memory-images\xp-laptop.img

Local Address	Remote Address	Pid
192.168.2.7:1164	66.179.81.247:80	944
192.168.2.7:1082	205.161.7.134:80	2392
127.0.0.1:1055	127.0.0.1:1056	2160
192.168.2.7:1077	64.62.243.144:80	2392
192.168.2.7:1066	199.239.137.200:80	2392
127.0.0.1:1056	127.0.0.1:1055	2160

c:\Python30\python.exe c:\Volatility\volatility **sockets** -f c:\memory-images\xp-laptop.img

Pid	Port	Proto	Create Time
4	138	17	Sat Jul 14 13:27:30 2009
4	0	47	Sat Jul 14 13:27:34 2009
2160	1055	6	Sat Jul 14 13:27:56 2009
2392	1064	17	Sat Jul 14 13:28:04 2009
2392	1077	6	Sat Jul 14 13:28:07 2009
4	445	6	Sat Jul 14 13:27:38 2009
2392	1066	6	Sat Jul 14 13:28:08 2009
2392	1082	6	Sat Jul 14 13:28:11 2009
4	139	6	Sat Jul 14 13:27:48 2009
4	1028	6	Sat Jul 14 13:27:55 2009
2160	1056	6	Sat Jul 14 13:27:58 2009
4	445	17	Sat Jul 14 13:28:01 2009

D. Extração de hashs do tipo SAM e visualização das senhas

Uma informação interessante, muitas vezes disponível na memória são os *hashs* do tipo SAM (*Security Account Manager*). Cada usuário do sistema operacional tem uma senha cadastrada, armazenada na forma de um *hash*. Estes *hashs* são considerados e tratados como informação confidencial pelo sistema operacional, pois com acesso a eles, outros procedimentos de ataque podem ser executados de modo a conseguir a obtenção efetiva de senhas que permitirão acessos não autorizados no sistema. Sob o ponto de vista da perícia, caso haja respaldo judicial, este procedimento pode permitir acesso ao sistema e até mesmo aos arquivos criptografados, caso a criptografia esteja atrelada à senha de acesso ao sistema.

A tarefa de recuperação de senhas a partir dos *hashs* armazenados em memória pode ser dividida em três etapas: a obtenção do *dump* de memória, já discutida; a extração dos *hashs* do *dump* e a quebra dos *hashs* com a visualização das senhas.

Para a extração dos *hashs* é necessário que o *Volatility* esteja instalado, no entanto, outros dois módulos são necessários o *pyCrypto* [9] e o módulo "*Volatility Plugin from Moyix*" [7]. Os seguintes passos devem ser seguidos:

1. Realizar o *donwload* do "*Volatility Plugin from Moyix*", descompactar os pacotes e copiá-los sobre as pastas originais do *Volatility*, sobrescrevendo os arquivos existentes nas seguintes pastas: *forensics*, *memory_objects* e *memory_plugins*.
2. Realizar o *donwload* do *pyCrypto* e instalá-lo em suas configurações padrão.
3. Supondo que o *dump* de memória (*xp-laptop.img*) esteja disponível, executar o seguinte comando:

```
c:\Python30\python.exe c:\Volatility\volatility hivescan -f c:\memory-images\xp-
laptop.img
Offset (hex)
47367992 0×1b0e738
47513752 0×1b4a738
.....corte.....
52157104 0×1e1cd870
54816208 0×1f9604f0
```

4. Com base no primeiro *offset* apresentado, executar o seguinte comando:

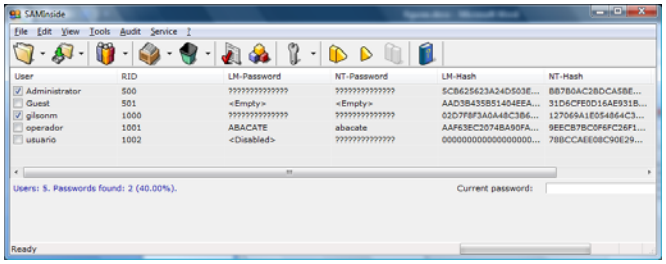
```
c:\Python30\python.exe c:\Volatility\volatility hivelist -f c:\memory-images\xp-
laptop.img -o 0×1b0e738
Address Name
0xe1bc4006 \Documents and Settings\Administrator\NTUSER.DAT
0xe1afeb68 \Documents and Settings\LocalService\Local Settings\Application
Data\Microsoft\Windows\UsrClass.dat
0xe1c4c006 \Documents and Settings\LocalService\NTUSER.DAT
0xe1c004d8 \Documents and Settings\NetworkService\NTUSER.DAT
0xe1619b68 \WINDOWS\system32\config\software
0xe161bb68 \WINDOWS\system32\config\default
0xe1751b68 \WINDOWS\system32\config\SAM
0xe1637006 \WINDOWS\system32\config\SECURITY
0xe1256756 \WINDOWS\system32\config\system
```

5. O comando seguinte é montado com a localização (*offset*) dos arquivos *system* e *SAM*, gerando os *hashs* desejados.

```
c:\Python30\python.exe c:\Volatility\volatility hashdump -f c:\memory-images\xp-
laptop.img -y 0xe1256756 -s 0xe1751b68
Extracted SAM :
Administrator:500:5CB625623A24D503EF3335D9CAEF53BF:BB7B0AC2BDCA5BECEB1B2
90EE97FA32F:::
Guest:501:AAD3B435B51404EEAAD3B435B51404EE:31D6CFE0D16AE931B73C59D7E0C
089C0:::
```

```
gilsonm:1000:02D7F8F3A0A48C3B6CF90A2DDECFFFDB:127069A1E054864C319A13B179EB2037:::
operador:1001:AAF63EC2074BA90FAAD3B435B51404EE:9EECB7BC0F6FC26F1FDCBD91E9FE4B7C:::
usuario:1002::78BCCAEE08C90E29AAD3B435B51404EE:972E8E7D5568F70AC896B2C76E1395DC:::
```

6. Gerar um arquivo texto com o resultado do comando anterior.
7. Baixar e instalar a ferramenta SamInside [11].
8. Executar a ferramenta e importar o arquivo no item "*File*", opção "*Import from PWDUMP file*". As senhas serão apresentas em tela como abaixo; um ataque de força bruta também pode ser comandado:



V. ESTUDO DE CASO APLICADO À PERÍCIA FORENSE

O objetivo deste estudo de caso é mostrar como as ferramentas apresentadas neste trabalho podem ser usadas pelos peritos e ser úteis na resposta a quesitos que podem gerar provas fortes e embasar uma investigação em curso.

Para a apresentação deste estudo de caso, foi instalado um ambiente de laboratório com um sistema Windows XP, no qual foram instalados e configurados:

1. O software *eMule*, um cliente P2P (*Peer to Peer*), que é comumente encontrado em boa parte dos computadores domésticos e que pode ser usado como ferramenta de apoio aos crimes cibernéticos no compartilhamento e transferência de imagens, pedofilia, programas maliciosos ou outras informações.
2. Um programa malicioso, criado por *crackers* para o roubo de credenciais de acesso, em especial de acessos aos serviços de *Internet Banking*. Neste caso foi utilizado o *kb1.exe*, um *banker* já obsoleto, mas adequado ao propósito didático deste trabalho.
3. Todas as ferramentas descritas neste trabalho, objetivando simplificar a extração e a análise de informações da memória.

Primeiramente, um *dump* de memória (CasoTeste1.dmp) foi gerado por meio da ferramenta MDD, já apresentada.

Uma relação de processos foi gerada com a opção *pslist* do *Volatility*; somente a parte que nos interessa é mostrada abaixo, contendo o identificador dos processos do *eMule* e do *banker*:

Name	Pid	PPid	Thds	Hnds	Time
Kb1.exe	712	5	1	11	Thu Jul 16 01:15:07 2009
emule.exe	983	337	9	231	Thu Jul 16 01:15:43 2009

De forma semelhante, utilizando as opções *connscan* e *sockets* do *Volatility*, são identificados os endereços IP (*Internet Protocol*), portas e protocolos em uso pelos dois programas:

Local Address	Remote Address	Pid
189.15.129.17:1078	194.67.36.117:80	712
189.15.129.17:51259	38.107.164.24:4661	983
127.0.0.1:51259	127.0.0.1:4661	983

Pid	Port	Proto	Create Time
712	25	6	Thu Jul 16 01:15:13 2009
983	4661	6	Thu Jul 16 01:15:52 2009

Em especial, é interessante mapear quais arquivos, chaves de registro e *DLL's* estão em uso pelo *banker*. Para isso, as opções *files*, *regobjkeys* e *dlllist* do *Volatility* podem ser usadas. Vale a pena especificar o número do processo (712) com a opção *-p*:

```
Pid: 712
File  \WINDOWS\system32
File  \net\NtControlPipe11
File  \Endpoint

Pid: 712
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Hardware
Profiles\Current\System\CurrentControlSet\SERVICES\TSWDD\DEVICE1

Command line : C:\WINDOWS\System32\kb1.exe
Base          Size      Path
0x1000000     0xa000    C:\WINDOWS\System32\kb1.exe
0x7c800000    0xf4000   C:\WINDOWS\system32\kernel32.dll
0x76f20000    0x27000   C:\WINDOWS\System32\DNSAPI.dll
```

Uma outra possível necessidade da investigação é identificar se o referido *software* realmente é malicioso, se é capaz de capturar e enviar informações confidenciais ao suposto criminoso. Para isso, o perito poderá extrair o executável do *dump* da memória, tendo acesso ao mesmo código objeto que estava em execução na máquina investigada, e monitorar seu comportamento em um ambiente de análise de artefatos, verificando suas características e eficácia. A extração pode ser feita da seguinte forma:

```
c:\Python30\python.exe c:\Volatility\volatility procdump –p 712 -f
c:\Gilson\CasoTeste1.dmp
************************************************************************
Dumping kb1.exe, pid: 712   output: executable.712.exe

C:\Gilson>dir
O volume na unidade C não tem nome.
O Número de Série do Volume é D3CB-38F9
Pasta de C:\Gilson
16/07/2009  02:14    <DIR>          .
16/07/2009  02:14    <DIR>          ..
16/07/2009  02:14             4.301 executable.712.exe
               1 arquivo(s)      4.301 bytes
               2 pasta(s)   3.467.876.864 bytes disponíveis
```

Em uma última etapa, as chaves SAM podem ser extraídas, processadas e quebradas conforme já exposto neste trabalho. Estas senhas, por comodidade do administrador ou usuário da máquina, além de permitir o acesso ao sistema, podem também coincidir com senhas de arquivos protegidos ou criptografados, senhas de contas de e-mail, dentre outras.

Logo, dentro dos limites legais de autorizações judiciais, as senhas reveladas podem ser testadas para acesso a outros sistemas do investigado.

Por meio da utilização plena das capacidades do *Volatility* e outras ferramentas de suporte, muito mais pode ser feito. O que foi mostrado neste estudo de caso pode ser considerado significativo em termos de arcabouço probatório robusto, necessário à eventual atuação da Justiça.

VI. Conclusão e Trabalhos Futuros

A necessidade crescente de análises periciais mais completas implica na adoção de novos processos que não somente a análise das mídias de armazenamento secundário. Neste sentido é necessário que procedimentos e técnicas para a extração e a análise de dados da memória volátil sejam bem definidos e adotados em benefício dos trabalhos periciais.

Uma significativa quantidade de informações pode ser obtida dos dados extraídos de memória, incluindo evidências e provas de comportamentos maliciosos e crimes digitais.

Nos processos de extração e análise de dados de memória, é fundamental que as metodologias adotadas privilegiem a qualidade e a precisão dos dados, e especialmente a cadeia de custódia dos vestígios apurados.

Já se fazem disponíveis para a comunidade forense e de usuários em geral, diversas ferramentas confiáveis, de utilização relativamente simples, voltadas à extração de dados, geração de *dumps* de memória e análise de conteúdo de *dumps* gerados. O uso combinado dessas ferramentas e outras tantas voltadas ao suporte técnico de sistemas operacionais pode agregar importante valor aos trabalhos de análise pericial em computadores investigados.

Como trabalhos futuros nesse contexto, sugerem-se análises de desempenho e comparativas entre ferramentas, conceituação e elaboração de *frameworks* integradores de ferramentas voltadas à perícia de memória volátil, levantamentos de conceitos e diretivas de organizações de memória de outros sistemas operacionais, aprimoramento de automatizações e customizações das ferramentas periciais existentes, desenvolvimento de novas ferramentas ainda mais interativas e amigáveis, e documentações mais completas de metodologias periciais relacionadas.

Referências

[1] Boileau Adam. “Hit by bus: physical access attacks with firewire”, Disponível em: http://www.ruxcon.org.au/files/2006/firewire_attacks.pdf. Acessado em 09/07/2009.

[2] Brasil. “Código de Processo Penal”. Disponível em: http://www.planalto.gov.br/ccivil_03/Decreto-Lei/Del3689.htm. Acessado em 13/07/2009.

[3] Carvalho, João. “Investigação pericial criminal e criminologia”. Campinas: Bookseller, 2005.

[4] Falbo, Ricardo. “Engenharia de software – notas de aula”. Disponível em http://www.inf.ufes.br/~falbo/download/aulas/es-g/2006-2/NotasDeAula.pdf. Acessado em 13/07/2009.

[5] MDD. Mantech Memory DD 1.3, Disponível em: http://www.mantech.com/msma/mdd.asp.Acessado em 13/07/2009.

[6] Microsoft. “Uma funcionalidade do Windows permite gerar um ficheiro de informações de estado da memória utilizando o teclado”. Disponível em: http://support.microsoft.com/kb/244139/pt. Acessado em 23/06/2009.

[7] Moyix. Volatility Plugin from Moyix , Disponível em: http://kurtz.cs.wesleyan.edu/~bdolangavitt/memory/volreg-0.2.zip. Acessado em 09/07/2009.

[8] NIST. Imagens de Memórias (NIST), Disponível em: http://www.cfreds.nist.gov/mem/memory-images.rar. Acessado em 23/06/2009.

[9] PyCrypto, PyCrypto 2.0.1, Disponível em: http://www.voidspace.org.uk/downloads/pycrypto-2.0.1.win32-py2.6.exe. Acessado em 09/07/2009.

[10] Python. Python 3.0.1, Disponível em: http://www.python.org/download/. Acessado em 23/06/2009.

[11] SamInside. Sam Inside 2.5.8.0 , Disponível em: http://saminside.en.softonic.com/. Acessado em 09/07/2009.

[12] Silva, Gilson Marques. “Segurança em Sistemas Linux”. Rio de Janeiro: Ciência Moderna, 2008.

[13] Tanenbaum, Andrew. “Organização estruturada de computadores”. 5. ed. São Paulo: Pearson Prentice Hall, 2007.

[14] Volatility. Volatility 1.3 Beta , Disponível em: https://www.volatilesystems.com/default/volatility/. Acessado em 23/06/2009.

[15] Win32DD. Win32DD 1.2.2.20090608, Disponível em: http://win32dd.msuiche.net/. Acessado em 13/07/2009.

Biografia

Gilson Marques da Silva (gilson-marques@hotmail.com) nasceu em Matutina no estado de Minas Gerais no Brasil, em 14 de junho de 1975. Ele graduou-se na Universidade Federal de Uberlândia, como bacharel em Ciência da Computação no primeiro semestre de 1998. No início de 2000 concluiu o curso de pós graduação *latu-sensu* em Telecomunicações e no segundo semestre de 2001 terminou o curso de pós graduação *latu-sensu* em Redes de Computadores. Em 2005 concluiu mestrado *strictu-sensu* em Ciência da Computação na área de Segurança em Redes sem Fios na Faculdade de Computação da Universidade Federal de Uberlândia. Sua experiência profissional inclui atividades como analista de redes na diretoria de processamento de dados da Universidade Federal de Uberlândia, como professor no nível de graduação e pós-graduação na UFU e UNIMINAS e como coordenador da área de segurança e antifraude de grande empresa operadora de telecomunicações. Atualmente é Perito Criminal Federal no Departamento de Polícia Federal, lotado na Unidade Técnico-Científica da delegacia de Marília/SP.

Evandro Mário Lorens (lorens@unb.br) é natural de Belo Horizonte/MG; graduado em Ciência da Computação pela Universidade Federal do Espírito Santo – UFES, especialista em Telecomunicações e Redes de Comunicação pela UFES e mestre em Ciência da Informação pela Universidade de Brasília – UnB. Atuou nas áreas de TI, gestão e segurança da informação no BANESTES –Banco do Estado do Espírito Santo (1989-2000), Caixa Econômica Federal (2000-2007) e Banco Central do Brasil (2007- 2009); professor nos cursos de Ciência da Computação na UnB (2000-2002), e Ciência da Computação, Sistemas de Informação, Engenharia da Computação e Engenharia Elétrica na Universidade Paulista – UNIP (2002-2009). Atualmente é Perito Criminal Federal do Departamento de Polícia Federal, lotado na Unidade Técnico-Científica da delegacia de Sinop/MT