Desafios da informática forense no cenário de Cloud Computing (Julho 2009)

Carlos Eduardo Marins, *Analista, Banco Central do Brasil*

***Abstract*—This paper presents some of the challenges traditional Computer Forensic methodology is facing with the growing adoption of Cloud Computing services and models by organizations.**

It will explain how the results of recent researches in computer physical memory analysis, new methodologies and tolls, can help address these arising challenges. It demonstrates the use of these techniques by analyzing how a worm that employs Cloud characteristics works. The recent, and still in the wild, Conficker worm.

***Index Terms*—Cloud Computing, Computer Forensics, RAM Analysis.**

I. Introdução

Cloud Computing, computação em nuvem, é uma expressão genérica que descreve a evolução de tecnologias e processos, compostos de serviços, aplicações, informações e infraestrutura distribuídos, de modo que estes possam ser arranjados dinâmica, elástica e rapidamente na medida em que forem consumidos. A computação em nuvem implica na separação das informações e aplicações da infraestrutura que os suporta. Várias descrições para os serviços baseados na *Cloud* (nuvem) existem, bem como vários modelos para representá-la. Neste estudo, será adotado o descrito pela *Cloud Security Alliance*, CSA, que foca no modelo e nas metodologias da perspectiva da área de segurança da informação [1].

Neste cenário, os métodos, as metodologias e as técnicas tradicionais de análise forense podem se mostrar pouco eficientes. Pesquisadores e investigadores da área de informática forense precisam conhecer este novo cenário e os desafios que ele pode trazer para suas atividades.

Serão apresentados neste trabalho os novos estudos na área de análise forense da RAM que podem ajudar pesquisadores e investigadores a vencer as dificuldades trazidas pela computação em nuvem.

II. Computação em Nuvem. Uma visão geral.

A. Arquitetura

A computação em nuvem prevê um forte isolamento das aplicações e informações, da infraestrutura e dos mecanismos usados para suportá-las, de modo que os recursos ofertados podem ser dinamicamente alocados de acordo com a demanda.

Na maioria das vezes, computação em nuvem está diretamente ligada às tecnologias de virtualização, notadamente pela facilidade de prover integração, provisionamento, escalabilidade, mobilidade e alocação de forma dinâmica dos recursos consumidos.

Para a informática forense é relevante entender o impacto que a arquitetura distribuída e dispersa da computação em nuvem traz para a coleta de evidências, já que, por exemplo, os dados podem estar distribuídos em vários países.

B. Principais Características

A CSA enumera as principais características da computação em nuvem.

1. Abstração da infraestrutura

A infraestrutura de computação, rede e armazenamento (*storage*) é isolada dos recursos de informação e aplicação. Do ponto de vista da aplicação e dos serviços fornecidos, não importa onde e por quais meios os dados são processados, transmitidos ou armazenados.

2. Democratização dos recursos

Conseqüência natural da abstração da infraestrutura, a democratização dos recursos, sejam eles infraestrutura, aplicações ou informações, permite que os mesmos sejam disponibilizados na forma de *pool* para aqueles que tenham autorização para acessá-los.

3. Serviços Orientados a Arquitetura

A abstração da infraestrutura e a democratização dos recursos levam aos serviços orientados a arquitetura, onde os recursos podem ser acessados e usados de forma padronizada. Assim, o foco está na entrega do serviço e não no gerenciamento da infraestrutura.

4. Elasticidade/Dinamismo

Os altos níveis de automação e de virtualização, juntamente com conectividade mais rápida e confiável, permitem que a alocação de recursos seja expandida ou retraída, de forma a atender a capacidade desejada. Assim, os recursos podem ser melhor aproveitados e os níveis de serviços mais facilmente alcançados.

5. Modelo utilidade de consumo e alocação

As quatro características listadas anteriormente combinadas possibilitam uma visibilidade no nível atômico dos recursos, por parte dos ofertantes dos serviços. Esta visibilidade permite que modelos de custo-utilidade e de uso (o contratante de um serviço pode consumi-lo à vontade, mas pagará por tudo o que for provido e consumido) sejam implementados.

Isto leva à melhora do gerenciamento do ambiente, com aumento de escala e de custos previsíveis.

C. Modelos de Entrega dos Serviços

Os serviços de computação em nuvem podem ser entregues de acordo com três modelos:

1. Software as a Service (SaaS)

O que é oferecido neste modelo são aplicações que rodam na infraestrutura da nuvem e podem ser acessadas por clientes leves (*thin clients*), como os navegadores. O usuário consumidor não gerencia nem controla a infraestrutura usada pelas aplicações (rede, dispositivos de armazenamento, sistemas operacionais etc.), nem mesmo configurações das aplicações (salvo algumas poucas configurações específicas).

2. Plataform as a Service (PaaS)

Aqui, o consumidor usa a infraestrutura da nuvem para disponibilizar suas aplicações que devem ser desenvolvidas em linguagens suportadas pelo provedor do serviço. O usuário consumidor não gerencia nem controla a infraestrutura usada pelas aplicações (rede, dispositivos de armazenamento, sistemas operacionais etc.), mas tem total controle e responsabilidade sobre as aplicações disponibilizadas.

3. Infrastructure as a Service (IaaS)

O que é oferecido ao consumidor neste modelo é o aluguel de processamento, de dispositivos de armazenamento, de redes e outros recursos considerados básicos. O consumidor pode rodar qualquer software (como sistemas operacionais e aplicações diversas), mas é o responsável por ele.

D. Modalidades de Consumo

Independentemente do modelo de entrega, existem ao menos três maneiras de implementá-los:

1. Privado (*Private*)

Conhecida também por nuvem interna (*Internal Cloud*). Sua característica principal é o fato ser de oferecida pela própria organização, ou pelo provedor de serviços que a organização indique, que vai consumir os recursos. Um ambiente operacional dedicado e que considera as funcionalidades e características da computação em nuvem é disponibilizado. A infraestrutura física é da própria organização, e pode estar localizada em seu *datacenter* ou no *datacenter* do provedor de serviços determinado.

Desta forma, os usuários dos serviços são considerados confiáveis (funcionários/empregados, terceiros e outros que tenham alguma relação contratual com a organização). Usuários não confiáveis são aqueles que não são extensões lógicas da organização, mesmo que de alguma forma consumam os serviços da organização.

2. Pública (*Public*)

São oferecidas por provedores de serviços. O ambiente operacional oferecido, contemplando todas as funcionalidades da computação em nuvem, pode ser dedicado ou compartilhado. Os consumidores dos serviços são considerados não confiáveis.

3. Híbrida (*Hybrid*)

É uma combinação da Pública com a Privada. Os consumidores dos serviços podem ser considerados confiáveis ou não confiáveis.

Christofer Hoff, no guia da CSA, propõe uma quarta possibilidade de implementação, chamada Gerenciada (*Managed*). Nela, os provedores de serviço oferecem um ambiente operacional dedicado ou compartilhado com todas as funcionalidades do modelo de computação em nuvem. Ele entende que a infraestrutura física desta implementação é da organização ou está em seu *datacenter*. Entretanto, a gerência e a segurança desta implementação são controladas pelo provedor dos serviços.

E. Modelo de Referência

A CSA descreve um modelo de referência para a computação em nuvem que ilustra como os modelos de entrega dos serviços estão dispostos em camadas. A camada mais alta (*SaaS*) depende da camada imediatamente inferior, que a suporta. A camada mais baixa, *IaaS*, serve como base para as demais. Cabe lembrar que este é um modelo de referência, que cumpre seu papel de esclarecer os relacionamentos dos modelos de entrega dos serviços de computação em nuvem.

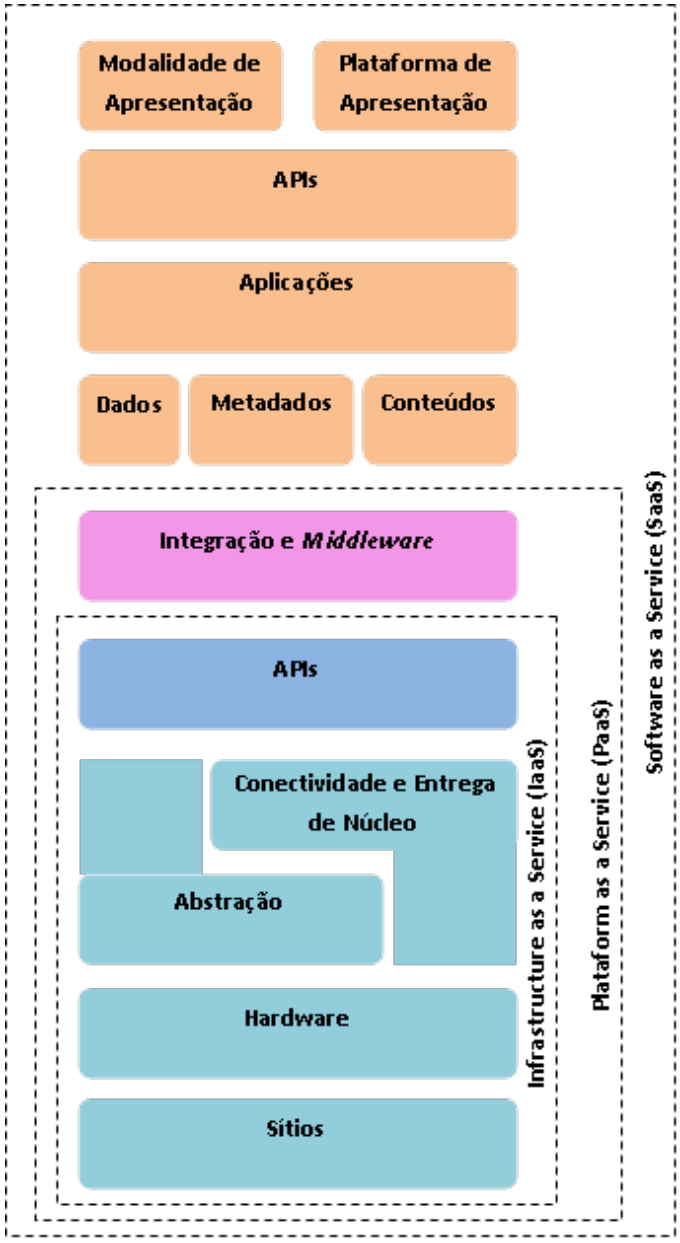


Fig. 1. Modelo de referência ilustrando a sobreposição dos modelos de entrega dos serviços em ambientes de Cloud Computing.

F. Arquitetura de Segurança em Ambientes de Computação em Nuvem

A arquitetura de segurança vai depender da combinação do modelo de entrega e da modalidade de consumo contratados.

O modelo de entrega vai definir quais controles devem ser implementados e como deve ocorrer esta implementação, já que existem controles específicos para cada camada do modelo de referências. Exemplo disto é que no modelo *IaaS* devem existir controles em nível físico (alarmes, guardas etc.), enquanto no *SaaS* controles como *firewall* para aplicações web.

A modalidade de consumo aponta quem será o responsável pela gerência dos controles. Assim, por exemplo, no caso da modalidade privada o contratante será o responsável pela gerência da arquitetura de segurança. Na modalidade pública, o provedor do serviço de computação em nuvem é o responsável por gerenciar a arquitetura de segurança.

III. INFORMÁTICA FORENSE E COMPUTAÇÃO EM NUVEM

A. Informática forense. Desafios nos cenários de Computação em Nuvem

Em muitos casos a metodologia tradicional empregada na informática forense, desligar o equipamento e fazer uma imagem *bit-a-bit* dos discos rígidos encontrados, não se mostra uma opção viável. Por exemplo, discos rígidos criptografados, tornam difícil, se não impossível, o acesso aos dados neles armazenados. Outro problema, é que muitas vezes desligar um equipamento por várias horas para fazer imagem de seus discos pode trazer prejuízos financeiros para a empresa, além de comprometer os acordos de níveis de serviços contratados.

A computação em nuvem contribui para tornar mais complexa a análise usando a metodologia tradicional de informática forense, pois introduz variáveis inerentes as suas características, modelos e arquitetura.

1. Localização dos dados

Tradicionalmente, os dados e informações estavam sob a custódia e responsabilidade das empresas e, na maioria das vezes, em suas instalações físicas (no máximo no *datacenter* de empresas terceiras contratadas para tal fim).

A arquitetura distribuída inerente à computação em nuvem suporta dados geograficamente distribuídos. Desta forma, uma imagem dos discos rígidos locais pode não se mostrar eficiente, já que estes discos podem não conter os mesmos dados que estão na nuvem. Dependendo ainda do modelo de entrega contratado, a empresa contratante pode optar por usar clientes leves (*thin clients*), sem que nem mesmo exista um disco rígido, com os dados inteiramente na nuvem.

A distribuição geográfica dos dados traz outro fator de preocupação para os analistas forense computacionais, pois estes dados podem estar em diferentes estados ou países e, por isso, sujeitos a diferentes regulações e legislações. O acesso a estes dados torna-se mais difícil.

2. Sistemas de arquivos (*filesystems*) não conhecidos

Para atender o dinamismo e elasticidade inerentes à arquitetura distribuída da computação em nuvem, os sistemas de arquivos usados nesta arquitetura podem ter sido redesenhados, personalizados ou até mesmo criados especificamente para atender determinadas demandas ou guardar determinados tipos de dados [2].

As pessoas que empregam a metodologia tradicional da informática forense podem enfrentar dificuldades para

recuperar dados gravados nestes sistemas de arquivos, já que suas estruturas podem não ser do conhecimento de quem esteja fazendo a análise. Além disso, o sistema de arquivos pode ser proprietário e tornar a tarefa de conhecê-lo ainda mais complexa.

3. Volume dos dados a serem analisados

A capacidade das mídias de armazenamento de massa cresce rapidamente [3] e cada vez mais estão acessíveis ao público. Para a informática forense, o uso da metodologia tradicional pode ser um problema, pois o tempo gasto para gerar a imagem *bit-a-bit* tende a ser maior, mesmo considerando o avanço dos dispositivos de *hardware* que fazem este tipo de cópia. Além disso, há de se considerar o espaço necessário para armazenar as imagens geradas e suas respectivas cópias.

A crescente oferta de serviços na computação em nuvem torna fundamental uma maior área de armazenamento (vários provedores de e-mails oferecem mais de 5 GB de armazenamento por usuário, por exemplo), aumentando o volume de dados que podem vir a ser analisados. Para agravar, esta quantidade imensa está distribuída ao longo da nuvem.

4. Criptografia dos sistemas de armazenamento na nuvem (*Cloud-based storages*)

A CSA recomenda em seu guia que o provedor de serviços de computação em nuvem use criptografia forte nos sistemas de armazenamento para prevenir que os dados não possam ser acessados por quem não tenha direito de fazê-lo, mesmo depois que os dispositivos de armazenamento (*storages*) tenham sido descartados.

A criptografia dos sistemas de armazenamento dificulta, caso não torne até impossível, a análise das imagens feitas destes dispositivos seguindo a metodologia tradicional da informática forense.

Diante destes desafios, que a arquitetura e modelo da computação em nuvem tornam mais complexos, é preciso encontrar metodologias alternativas para a informática forense de modo que estas questões possam ser melhor atendidas.

B. *Informática forense. "Live Response" para endereçar os novos desafios*

Previamente foram descritos os desafios que a metodologia tradicional usada pela informática forense enfrenta e como os cenários de computação em nuvem tornam estes desafios ainda mais complexos. Outro fator que não pode ser desconsiderado é a existência de *malwares* que existem apenas em memória, enquanto são executados.

A proposta aqui apresentada para endereçar estes desafios tem como base a análise dos equipamentos enquanto estiverem ligados. Esta técnica, conhecida como *live response*, tem como objetivo coletar os dados voláteis de um sistema enquanto em funcionamento. O escopo é Além disso, destaca a coleta de evidências considerando sua Ordem de Volatilidade, em particular o conteúdo da RAM. O escopo desta pesquisa é *live response* em equipamentos que rodem sistemas operacionais da família Windows.

A RFC 3227 [4] traz recomendações sobre o processo de coleta de evidências. Ela recomenda que esta coleta siga a Ordem de Volatilidade das evidências, das mais voláteis para as menos voláteis. Assim, uma das primeiras evidências que devem ser coletadas é a memória física (RAM). A RAM contém todos os programas que estão em execução num determinado período. As novas técnicas de análise da RAM, que serão descritas *a posteriori*, permitem descobrir processos e *threads* que estavam sem execução e os arquivos e *handles* usados por eles. Isto, sem dúvidas, permite que analistas e investigadores tenham uma visão mais detalhada e precisa do que acontecia em determinados sistemas, e pode ajudar a endereçar melhor os desafios apresentados.

IV. INFORMÁTICA FORENSE. ANÁLISE DA RAM

A. *Análise da RAM. Coleta*

Seguindo a ordem de volatilidade recomendada pela RFC 3227, a memória RAM deve ser coletada e analisada primeiro. Desta mesma forma entendem Dan Farmer e Wietse Venema [5]. Antes de serem discutidas as técnicas de análise da RAM, serão apresentadas algumas formas de coleta da RAM.

O objetivo de coletar as informações da RAM é ter uma "fotografia" do que estava em execução em determinado tempo gravada num arquivo para posterior análise. Existem várias técnicas que podem ser usadas para extrair o conteúdo da RAM para um arquivo, mas talvez a mais conhecida e usada seja a ferramenta dd (http://en.wikipedia.org/wiki/dd_(Unix)), ou algumas de suas variações, como a Win32dd (http://win32dd.msuiche.net) ou a MDD (http://www.mantech.com/msma/MDD.asp).

B. *Análise da RAM. Técnicas tradicionais*

Tradicionalmente, a análise do conteúdo extraído da memória física consistia em procurar no arquivo com a imagem da RAM senhas, endereços de e-mails, endereços IP ou quaisquer outras cadeias de caracteres que pudessem dar pistas durante a análise.

O problema desta abordagem é que era difícil correlacionar a informação encontrada com um processo específico, ou seja, ela não tinha contexto.

A contextualização da informação encontrada na RAM passou a ser o alvo de estudos mais aprofundados. A fim de promover o debate, a pesquisa e o desenvolvimento de ferramentas e técnicas nesta área o *Digital Forensic Research*

Workshop (DFRWS – www.dfrws.org), em 2005, lançou um desafio de análise do conteúdo da RAM. O objetivo foi alcançado. Houve um grande avanço no entendimento da estrutura da RAM pela comunidade e várias ferramentas foram lançadas.

C. *Análise da RAM. Novas abordagens*

Antes de serem apresentadas as novas abordagens para análise da RAM, alguns pontos precisam ser tratados.

1. Versão do sistema operacional

De posse de um arquivo de despejo de memória RAM, a primeira coisa a ser feita é determinar o sistema operacional deste arquivo de despejo, pois em se tratando de Windows a versão importa. Isto porque as estruturas usadas para a definição das *threads* e dos processos em memória variam de acordo com as versões do Windows, sofrendo variações inclusive entre diferentes *service packs*, conforme destacado por Harlan Carvey [6].

Existem várias ferramentas que cumprem bem este papel. De modo simplificado, o que muitas delas fazem é tentar localizar no arquivo de imagem da RAM o *kernel* do Windows, que é um arquivo executável, e, dentro de sua estrutura, procurar pela seção VS_VERSION_INFO. Esta seção contém o nome do produto e sua versão. Talvez esta seja, até o momento, a maneira mais confiável de apurar a versão do Windows de uma imagem da RAM.

2. Funcionamento básico dos Processos em plataformas Windows

Muitas das pesquisas têm os processos como fontes de informações relativas à análise forense da RAM, por isso, entender as estruturas dos processos e como eles são criados é essencial. Harlan Carvey comenta ainda em seu livro que a maioria dos conceitos relacionados a processos são válidos para as diferentes versões do Windows, e que a diferença estaria na própria estrutura do processo. Existem duas estruturas principais dos processos que são relevantes para a informática forense: EProcess e a PEB (*Process Environment Block*).

Cada processo no Windows é representado por um processo executivo, EProcess (*Executive Process*), que é uma estrutura de dados que armazena os atributos do processo e ponteiros para outras estruturas contidas também no processo. É importante notar que estas estruturas são completamente dependentes da versão do Windows de modo que seus tamanhos e até os valores das estruturas mudam ao longo das diferentes versões e *service packs*. Felizmente é possível ver como é esta estrutura, e as demais estruturas de um processo, com a ferramenta Microsoft Debugging Tools e os símbolos corretos para o sistema operacional e *service pack*, que podem ser baixados gratuitamente do site da própria Microsoft.

Para ver a estrutura EProcess deve-se, depois de instalar a ferramenta e os símbolos corretos, abrir o prompt de comando e digitar livekd -w. Na janela que será aberta, a interface gráfica do debugger, basta digitar dt –a –b –v _EPROCESS. Toda a estrutura EProcess será listada. Abaixo seguem dois trechos desta estrutura como exemplos das variações ela pode ter em versões diferentes de sistemas operacionais da família Windows:

- Windows XP SP3.

```
kd> dt -a -b -v _EPROCESS
ntdll!_EPROCESS
   struct _EPROCESS, 107 elements, 0x260
bytes
      +0x000 Pcb               : struct _KPROCESS,
29 elements, 0x6c bytes
         +0x000 Header           : struct
_DISPATCHER_HEADER, 6 elements, 0x10 bytes
```

- Windows 7 RC.

```
kd> dt -a -b -v _EPROCESS
nt!_EPROCESS
   struct _EPROCESS, 133 elements, 0x2c0
bytes
      +0x000 Pcb               : struct _KPROCESS,
34 elements, 0x98 bytes
     +0x000  Header                     :  struct
_DISPATCHER_HEADER, 30 elements, 0x10 bytes
```

Propositadamente foi destacada a linha da estrutura que informa seus elementos e seu tamanho nos dois trechos. Percebe-se que a estrutura EProcess do Windows 7 tem mais elementos e é maior. A saída completa deste comando é maior (0x2c0 – 704 – bytes) e, por isso, não foi completamente reproduzida.

Existe outro importante elemento que é referenciado na estrutura EProcess, o PEB (*Process Environment Block*). Para a informática forense se destacam os seguintes elementos do PEB, conforme destaca Harlan Carvey em sua obra:

- Ponteiro para a estrutura do *loader data* (PPEB_LDR_DATA), que possui ponteiros ou referências para as DLLs usadas pelo processo;
- Ponteiro para o endereço da imagem-base, onde espera-se encontrar o começo do arquivo executável; e
- Ponteiro para a estrutura de parâmetros do processo, que contém o caminho para a DLL, o caminho para o arquivo executável e linha de comando usada para iniciar o processo.

Agora que já se tem uma visão geral da estrutura de um processo, é preciso entender como eles são criados nos sistemas operacionais Windows. Mark Russinovich e David Solomon fazem um excelente trabalho ao descrever o fluxo

de criação de um processo, que pode ser assim resumido [7]:

1. Validação dos parâmetros; conversão das *flags* do subsistema Windows e das opções para forma nativa; fazer o *parsing*, a validação e a conversão do atributo lista para sua forma nativa.
2. Abrir o arquivo de imagem (.exe) a ser executado dentro do processo.
3. Criar o objeto do processo *executive* do Windows.
4. Criar a *thread* inicial.
5. Realizar tarefas pós-criação do processo, e inicializar o subsistema Windows.
6. Iniciar a execução da *thead* incial.
7. No contexto do novo processo, completar a inicialização do *address space* (carregar as DLLs necessárias, por exemplo) e começar a execução do programa.

A partir deste momento o processo recém-criado usa a memória de acordo com a estrutura EProcess (e demais estruturas) e pode começar a usar mais a medida que está em execução.

D. *Análise da RAM. Conficker em ação*

O *Conficker* é um *malware* que emprega as características da computação em nuvem para se instalar e se atualizar. A versão B, que será usada para análise, possui algoritmo que gera diariamente a lista de domínios que podem ser usados para baixar suas atualizações. Este mecanismo provê um serviço de atualização eficiente e altamente móvel – as localizações são recalculadas todos os dias pelas máquinas infectadas [8]. Trata-se de um *malware* que opera de forma distribuída ao longo da Internet, com vários clientes (micros infectados) e servidores (máquinas que hospedam o programa malicioso e suas novas versões) distribuídos pela Internet. Pode-se entender como uma nova modalidade de entrega de serviços: *Virus as a Service* (*VaaS*) ou *Malware as a Service* (*MaaS*), onde as máquinas contaminadas são ofertadas como redes que podem ser usadas por *spammers*, *phishers*, pedófilos e grupos que realizam atividades maliciosas pela Internet.

Assim, a fim de demonstrar como a análise da RAM, em casos de *live response*, pode ajudar pesquisadores e analistas na área de informática forense, o *Conficker* B foi instalado num laboratório "virtual" com as seguintes especificações:

- Virtualização: Virtual Box (www.virtualbox.org);
- Sistema operacional: Windows XP SP3 com todas as correções até 20/07/2009 aplicadas e 256 MB configurados para a RAM;
- Coleta da imagem da RAM: Helix Pro (www.e-fense.com);
- Análise do arquivo de memória: Memorize e Audit Viewer (www.mandiant.com)[1].

Não se trata de uma análise profunda do Conficker, mas como usar as novas metodologias de análise da RAM para identificar um processo responsável por atividades suspeitas.

Depois de configurado o ambiente do laboratório descrito, foi necessário infectar a máquina com um programa malicioso, neste caso o Conficker. Com a certeza que a máquina de testes estava contaminada, foi obtida uma imagem de sua RAM usando o Helix Pro.

O passo seguinte foi analisar a imagem com o Memorize e o Audit Viewer. O Audit Viewer foi usado para interpretar os resultados e exibi-los numa interface gráfica de fácil navegação.

A figura 2 mostra parte da tela do Audit Viewer depois de processar o resultado da análise feita pelo Memorize. Nesta tela estão listados todos os processos encontrados.

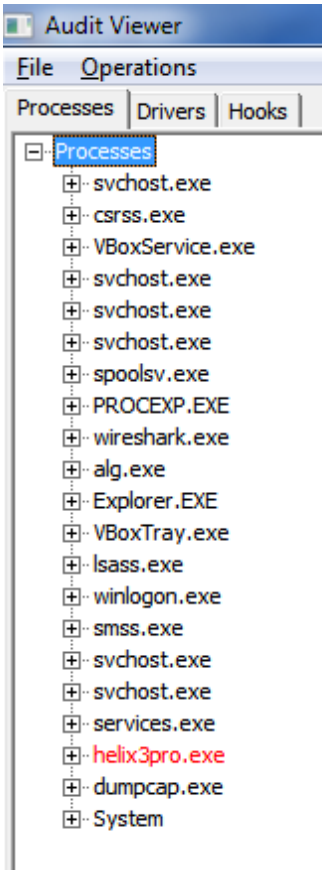
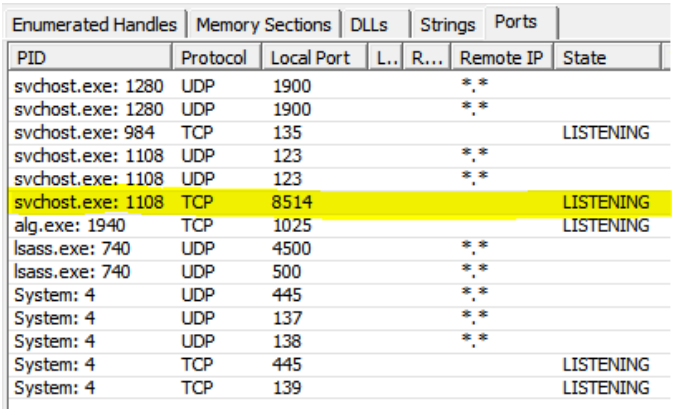


Fig. 2. Parte da tela do Audit Viewer. Lista de todos os processos encontrados no arquivo da imagem da RAM.

[1] O Memorize é uma ferramenta poderosa que permite, entre outras coisas, fazer a imagem de toda a RAM de uma máquina (sem usar as APIs do Windows), listar todos os processos ativos (incluídos aqueles escondidos por *rootkits*), gerar uma imagem em disco do *address space* de um processo, exibir todas as portas que estavam em uso e qual processoas usava. Os resultados gerados pelo Memorize são no formato XML e o Audit Viewer pode ser usado para interpretar e navegar pelos resultados.

A figura 3 exibe as portas que estavam em uso e os processos que as usavam. Com esta visualização foi possível perceber que o processo de ID 1108, svchost.exe, estava escutando na porta 8514.



Enumerated Handles	Memory Sections	DLLs	Strings	Ports		
PID	Protocol	Local Port	L..	R...	Remote IP	State
svchost.exe: 1280	UDP	1900			*.*	
svchost.exe: 1280	UDP	1900			*.*	
svchost.exe: 984	TCP	135				LISTENING
svchost.exe: 1108	UDP	123			*.*	
svchost.exe: 1108	UDP	123			*.*	
svchost.exe: 1108	TCP	8514				LISTENING
alg.exe: 1940	TCP	1025				LISTENING
lsass.exe: 740	UDP	4500			*.*	
lsass.exe: 740	UDP	500			*.*	
System: 4	UDP	445			*.*	
System: 4	UDP	137			*.*	
System: 4	UDP	138			*.*	
System: 4	TCP	445				LISTENING
System: 4	TCP	139				LISTENING

Fig. 3. Parte da tela do Audit Viewer. Processo escutando em porta suspeita.

O processo svchost.exe suspeito tem nome de um processo legítimo dos sistemas operacionais Windows. O processo legítimo tem como função verificar numa chave do registro o grupo de serviços que ele deve carregar e está localizado em %SYSTEMROOT%\system32 [9].

O Audit Viewer mostra o caminho de onde o processo foi chamado, conforme mostra a figura 4. Pode-se então concluir que se trata de um processo legítimo, mas que poderia ter sido usado para carregar um código malicioso.

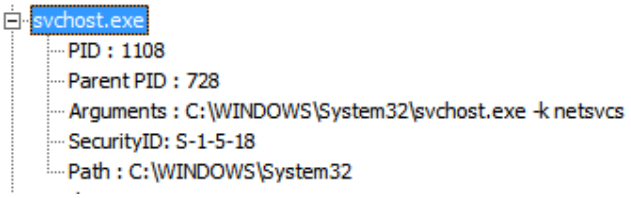


Fig. 4. Parte da tela do Audit Viewer. Caminho de onde o processo foi executado.

Com base nesta informação pode ser interessante, em determinadas situações, capturar a imagem do processo para analisá-la mais detalhadamente. É possível fazer isto de dentro do próprio Audit Viewer, bastando clicar com o botão direito do mouse sobre o processo desejado e depois clicar em *Acquire Process*.

Navegando pela aba *Handles* do Audit Viewer para o processo em questão, é possível notar que existe um *handle* que aponta para a dll c:\windows\system32\pqquewv.dll, como ilustra a figura 5.

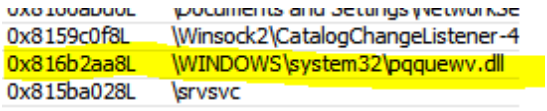


Fig. 5. Parte da tela do Audit Viewer. Handles referentes ao processo de ID 1108.

É interessante notar que apesar de ser uma dll, ela não aparece na lista de dlls que o programa carrega por si só, ou seja, é muito provável que esta dll não conste na *import table*[2] do executável svchost.exe. Este fato também é muito suspeito. Por que uma dll precisaria ser carregada se ela não constar na *import table* de um executável?

Conforme mencionado, o executável svchost.exe lê de uma chave de registro as informações sobre determinado grupo de serviços que deve ser carregado em memória. Basta então ver na chave de registro apropriada o grupo de serviços que é chamado e quais serviços fazem parte deste grupo. A chave é *HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\SvcHost*. Deve-se então clicar com o botão direito do mouse sobre a entrada *netsvcs* e em seguida clicar em *Modificar*. Na lista que se abre, ir até seu final e procurar por um nome de serviço aleatório [10]. No caso em estudo, o nome do serviço desconhecido é aebri.

O próximo passo é ir até HKLM\SYSTEM\CurrentControlSet\Services\'nome do serviço malicioso', aebri neste caso.

O problema é que o Conficker é um código malicioso bem esperto e que procura ocultar sua presença ao máximo. Por isso, esta chave de registro teve suas permissões alteradas de forma que não fosse visível num primeiro momento. As permissões devem ser restauradas para que a chave seja vista. Ela contém os parâmetros que são usados para chamar o Conficker. O valor encontrado no registro da máquina do laboratório foi %SystemRoot%\system32\pqquewv.dll, como mostra a figura 6.

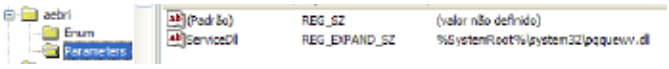


Fig. 6. Trecho do editor de registro do Windows. Caminho de onde o Conficker é carregado.

O Conficker é um código malicioso que, depois que se instala, busca por atualizações de seu código em diversos servidores na Internet e também fica escutando em determinadas portas de modo que possa servir de repositório para que outros Confickers se atualizem [11].

V. CONCLUSÃO

O uso dos serviços da computação em nuvem pelas empresas coloca novos desafios para a Informática forense, além daqueles oriundos da evolução natural das tecnologias.

A tendência é que cada vez mais redes de máquinas infectadas sejam oferecidas como serviços para toda a sorte de mal-intencionados e que os *malwares* diminuam seus rastros nestas máquinas, podendo existir apenas na RAM, por exemplo.

[2] A *import table* contém informações sobre as dlls e funções que um executável precisa quando estiver em funcionamento. Quando o executável é carregado, o sistema operacional vai ler a *import table* para saber quais dlls e quais funções ele precisa carregar para seu correto funcionamento.

A metodologia tradicional empregada por analistas e investigadores cumpre bem seu papel, mas as novas técnicas de análise da RAM podem ajudar na superação destas dificuldades à medida que agregam mais informações e contexto ao processo de análise. Estas técnicas, que conseguem contextualizar as informações encontradas, ligando-as a processos específicos, mapeando as atividades e *threads* do processo, permitem que programas maliciosos sejam analisados quando em execução, já que muitos usam técnicas avançadas para se ocultarem, o que torna difícil sua análise antes de serem executados.

O Conficker foi usado como exemplo de como estas novas técnicas podem ser aplicadas, mas suas aplicações são várias. Por exemplo, a informação apresentada à comunidade depois do DFRW de 2005 permitiu que Jeff Bryner desenvolvesse ferramentas capazes de buscar em um arquivo-imagem da RAM artefatos, como contatos, últimos registros acessados etc., do GMail e do Yahoo!, pdgmail e pdymail, respectivamente [11].

Ainda sobre análise do conteúdo da RAM, Andreas Schuster afirma em seu blog [12] que existe uma área que merece ser estudada mais profundamente, qual seja, a persistência de dados em memória depois de uma reinicialização ou depois de um *crash dump*. Ele diz ainda que vê na análise da memória física uma próspera área para a informática forense.

REFERENCES

[1] Cloud Security Alliance " Security Guidance for Critical Areas of Focus in Cloud Computing" 2009.

[2] Ghemawat , S., Gobioff , H., Leung, S-T, Chen,W.-K, *The Google File System.200*3.

[3] Kingston. http://www.kingston.com/flash/dt300.asp?id=1.

[4] RFC 3227. http://www.faqs.org/rfcs/rfc3227.html.

[5] Farmer, D.,Venema, W. "Perícia Informática forense. Teoria e Prática Aplicada" Pearson Prentice Hall, 2007.

[6] Carvey, H. "Windows Forensics Analysis DVD toolkit", 2nd Ed., Syngress, 2009.

[7] Russinovich, M., Solomon, David A. "Microsoft Windows Internals", 5th Ed., Microsoft Press, 2004.

[8] SRI International. http://mtc.sri.com/Conficker/.

[9] Microsoft. "A description of Svchost.exe in Windows XP Professional Edition". http://support.microsoft.com/kb/314056.

[10] Microsoft."Virus alert about the Win32/Conficker.B worm".http://support.microsoft.com/kb/962007

[11] Jeff Bryner. Pdgmail – http://www.jeffbryner.com/code/pdgmail. Pdymail - http://www.jeffbryner.com/code/pdymail.

[12] Andreas Schuster. "Data Lifetime". http://computer.forensikblog.de/en/2006/04/data_lifetime.html