FACULDADE DE TECNOLOGIA DE AMERICANA - FATEC

CURSO DE ANÁLISE DE SISTEMAS E TECNOLOGIA DA INFORMAÇÃO

PERÍCIA FORENSE COMPUTACIONAL

CLEVERSON RODRIGO DE OLIVEIRA

FATEC AMERICANA

2012

FACULDADE DE TECNOLOGIA DE AMERICANA - FATEC

CURSO DE ANÁLISE DE SISTEMAS E TECNOLOGIA DA INFORMAÇÃO

PERÍCIA FORENSE COMPUTACIONAL

CLEVERSON RODRIGO DE OLIVEIRA

Monografia apresentada ao Curso de Análise de Sistemas e Tecnologia da Informação, com Especialização em Segurança da Informação da Faculdade de Tecnologia de Americana, como requisito para a obtenção do título de Tecnólogo, sob orientação do Prof. Esp. Benedito Aparecido Cruz.

FATEC AMERICANA

2012

FICHA CATALOGRÁFICA elaborada pela

BIBLIOTECA – FATEC Americana – CEETPS

Oliveira, Cleverson Rodrigo de

O46p Perícia forense computacional. / Cleverson Rodrigo de Oliveira. – Americana: 2012.

48f.

Monografia (Graduação em Análise de Sistemas e Tecnologia da Informação). - - Faculdade de Tecnologia de Americana – Centro Estadual de Educação Tecnológica Paula Souza.

Orientador: Prof. Esp. Benedito Aparecido Cruz

1.Segurança em sistemas de informação I. Cruz, Benedito Aparecido II. Centro Estadual de Educação Tecnológica Paula Souza – Faculdade de Tecnologia de Americana.

CDU: 681.518.5

Bibliotecária responsável Ana Valquiria Niaradi – CRB-8 região 6203

BANCA EXAMINADORA

Carlos Henrique Rodrigues Sarro – Mestre

Benedito Aparecido Cruz – Especialista

Alexandre Garcia Aguado – Graduado

AGRADECIMENTOS

À Deus, porque dele e por ele, e para ele, são todas as coisas; glória, pois, a ele eternamente.

À meus pais, Toninho e Leonor, que com amor e dedicação sempre me apoiaram em todos os meus projetos.

À minha esposa, Eylane, que com paciência me esperou, com carinho me apoiou, por ser meu esteio em momentos difíceis, por sua paciência quando eu perdia a minha e por fazer a minha vida cada dia melhor e mais feliz.

Ao meu filho Leonardo, presente de Deus. Sei que muitas vezes me ausentei, mas é através de tudo isso que poderemos desfrutar um futuro melhor.

Ao meu Pastor, Luis Felipe, por nunca duvidar da minha capacidade.

Aos meus amigos da faculdade, em especial, minha amiga Josyane F. S. Emiliano.

Aos docentes com quem pude aprender muito. Obrigado por partilhar o conhecimento, pelos desafios e em muitas vezes pelo encorajamento.

DEDICATÓRIA

À meus pais, Toninho e Leonor...

À minha esposa Eylane...

Ao meu filho Leonardo...

...pelo amor, carinho, dedicação, confiança e por tudo que representam em minha vida.

RESUMO

Em uma sociedade, onde as pessoas e, por que não dizer, as próprias máquinas de um modo geral, estão cada vez mais dependentes da tecnologia, em um mundo onde as informações viajam em velocidades cada vez maiores por cabos, fibras óticas e satélites, novos desafios são propostos na área computacional.

É nesse emaranhado de informações, dispositivos e meios que surge a figura do Perito Forense Computacional, que é o profissional indicado para a aquisição, preservação, recuperação e análise de dados que estão em formato eletrônico e armazenados em algum tipo de mídia computacional (Guimarães et al., 2008).

Este trabalho de graduação visa descrever o trabalho do perito na realização da Perícia Forense Computacional, as técnicas para localizar ou utilizar os dados relacionados ao tempo e como os atributos de tempo de um sistema podem auxiliar na reconstrução e no entendimento de um evento.

Palavras Chave: Forense Computacional, arquivos excluídos, MACtimes

ABSTRACT

In a society where people and, why not say, the machines themselves in general, are increasingly dependent on technology, in a world where information travels at a speed increasing by cables, fiber optics and satellites, new challenges are proposed in the computational area.

Is this tangle of information, devices and media that arises the figure of Computational Forensic Expert, which is the professional indicated for acquisition, preservation, retrieval and analysis of data that are stored in electronic format and in any type of media computing (Guimarães et al. 2008).

This paper of degree aims to describe the expert's work in the realization of Computational Forensics, techniques to locate and use data related to time and as the time attributes of a system can assist in the reconstruction and understanding of an event.

Keywords*: Computational Forensic, excluded files, MACtimes*

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

INTRODUÇÃO

Pesquisadores do Departamento de Geologia da Universidade Federal do Ceará (2012) em visita ao município de Forquilha - CE alertam as autoridades para o problema da preservação dos sítios arqueológicos locais.

No local os professores observaram uma diversidade de registros rupestres gravados em paredões graníticos e alertaram sobre o problema causado pela alteração física e biológica que comprometem a preservação das pinturas arqueológicas. O Sr. Célio Cavalcante também ressaltou que a atividade mineral próximo ao sítio expõe ao risco constante, devido o uso de explosivos para a extração de granitos. As queimadas são outro motivo de preocupação disse o pesquisador da Pré-história (UFC, 2012).

Assim como os processos geológicos estão constantemente destruindo sítios arqueológicos, suas versões ciberespaciais estão constantemente destruindo informações.

Nesse sentido, a arqueologia trata dos efeitos diretos da atividade humana, como artefatos deixados para trás, e a geologia trata dos processos autônomos sobre os quais o homem não tem controle direto, como geleiras, placas tectônicas, vulcanismo e erosão. Assim, a arqueologia digital diz respeito aos efeitos diretos da atividade do usuário, como conteúdo de arquivo, registro de tempo de acesso de arquivo, informações provenientes de arquivos excluídos e registros em log do fluxo de rede e a geologia digital diz respeito aos processos autônomos sobre o quais os usuários não têm controle direto, como a alocação e reciclagem de blocos de disco, números de ID de arquivo, páginas de memória ou números de ID de processo.

O desafio da investigação no “sítio arqueológico digital” é recuperar as informações que foram parcialmente destruídas. Uma vez excluído, o conteúdo de um arquivo geralmente não muda até ser sobrescrito por um novo arquivo. Em sistemas de arquivos com boas propriedades de aglomeração de dados, arquivos excluídos são como um fóssil: um esqueleto pode não ter um osso aqui ou ali, mas o fóssil permanece imutável, até ser destruído.

Excluir um arquivo do sistema de arquivos é relativamente fácil, mas não é suficiente para destruir seu conteúdo ou atributos. As informações sobre o arquivo

excluído persistem nos blocos de disco que foram anteriormente alocados para esse arquivo.

Por ocorrer esse fenômeno da exclusão e persistência em outros níveis de abstração também é que se diz que destruir informações pode ser surpreendentemente difícil.

1 - O QUE ACONTECE QUANDO UM ARQUIVO É EXCLUÍDO?

1.1 - Introdução

Excluir um arquivo tem um efeito diretamente visível: o nome do arquivo desaparece de uma listagem de diretório. O que acontece de fato quanto um arquivo é excluído depende de aspectos internos do sistema. Alguns sistemas de arquivos como o FAT16 da Microsoft e sistemas de arquivos FAT32 marcam o arquivo como excluído ocultando o nome do arquivo de uma maneira especial. Tradicionalmente, o *Berkeley Fast File System* (FFS) interrompe todas as conexões entre a entrada de diretório, os atributos de arquivo e os blocos de dados de arquivo. Os sistemas de arquivos descendentes do FFS são comumente encontrados nos sistemas Solaris e BSD. Com os *kernels* do Linux 2.2, o sistema de arquivos Ext2fs do Linux marca a entrada de diretório como não utilizada, mas preserva as conexões entre a entrada de diretório, os atributos de arquivo e os blocos de dados de arquivo. Com os *kernels* do Linux 2.4, excluir um arquivo tornou-se mais destrutivo, assim o Ext2fs não mais preserva as conexões entre as entradas de diretório e os atributos de arquivo. Por outro lado, alguns sistemas derivados do 4.4 BSD preservam as conexões entre as entradas de diretório e os atributos de arquivo. A Tabela 1 resume quais informações são preservadas e quais informações são destruídas, demonstrando o efeito da exclusão de um arquivo sobre os nomes de arquivo, sobre os atributos de arquivo e sobre o conteúdo de um arquivo para sistemas de arquivos UNIX típicos.

Neste capítulo, os exemplos usados referem-se ao FFS (McKusick ET AL., 1984) e seus descendentes, incluindo o Solaris UFS e o Linux Ext2fs (Card ET AL., 1994) e seus descendentes. Em todos os casos, Farmer e Venema (2007) e pressupõem acesso a um sistema de arquivos local. Sistemas de arquivos remotos normalmente não fornecem acesso a informações de arquivos excluídos ou não-alocados.

Tabela 1 - Informações preservadas e destruídas na exclusão de arquivos em um sistema de arquivos UNIX típico (FARMER;VENEMA, 2007)

Propriedade de Arquivo	Localização	Efeito da exclusão de arquivo
Entrada de diretório	Blocos de dados de diretório	Marcado como não-alocado
Nome do arquivo		Preservado
Número de inode		Depende do sistema
Atributos de diretório	Bloco de inode de diretório	
Data/hora do último acesso de leitura		Data/hora de exclusão
Data/hora do último acesso de gravação		Data/hora de exclusão
Data/hora da última modificação de atributo		Data/hora de exclusão
Atributos de arquivo	Bloco de inode de arquivo	Marcado como não-alocado
Proprietário		Preservado
Posse do grupo		Preservado
Data/hora do último acesso de leitura		Preservado
Data/hora do último acesso de gravação		Depende do sistema
Data/hora da última modificação de atributo		Data/hora de exclusão
Data/hora de exclusão (se disponível)		Data/hora de exclusão
Contagem das referências de diretório		Zero
Tipo de arquivo		Depende do sistema
Permissões de acesso		Depende do sistema
Tamanho do arquivo		Depende do sistema
Endereço do bloco de dados		Depende do sistema
Conteúdo do arquivo	Bloco de dados do arquivo	Preservado, marcado como não-alocado

1.2 - Entrada do diretório pai

Quando um arquivo é excluído, a entrada de diretório com o nome do arquivo e número de inode é marcada como não utilizada. Em geral, o número de inode é configurado como zero de modo que o nome do arquivo torna-se desconectado de quaisquer informações sobre o arquivo. Esse comportamento é encontrado nos sistemas Solaris. Algumas implementações FreeBSD, UFS e LINUX Ext2fs preservam o número de inode na entrada de diretório.

Os nomes dos arquivos excluídos ainda podem ser encontrados lendo o diretório com o comando *strings*. Infelizmente, o Linux não permite que diretórios sejam lidos pelos programas dos usuários. Para contornar essa restrição, é possível utilizar o utilitário icat (copia o arquivo pelo número de inode) do *Coroner's Toolkit*. O comando a seguir lista os nomes de arquivo no diretório-raiz (inode número 2) do sistema de arquivos hdal:

 # icat /dev/hdal 2 | strings

Uma ferramenta mais sofisticada para explorar entradas excluídas de diretório é o utilitário fls (lista entradas de diretório) do pacote de *software* Sleuth Kit (Carrier, 2004a). Esse utilitário também dribla o sistema de arquivos e quaisquer restrições que ele tenta impor. O comando a seguir lista as entradas de diretório excluídas no diretório-raiz (inode 2) do sistema de arquivos hdal:

 # fls -d /dev/hdal 2

O fls também pode processar recursivamente todos os diretórios em um sistema de arquivos, incluindo os diretórios que estão ocultos sob os pontos de montagem.

1.2.1 - Atributos do diretório pai

Como um efeito colateral da atualização de entrada de diretório, os atributos de última leitura, última modificação e última alteração do status do diretório são todos os configurados como a data/hora dessa atualização. Portanto, mesmo se o próprio arquivo excluído não estiver mais disponível, a data/hora da última

modificação no diretório continuará a revelar a atividade passada dentro desse diretório.

1.3 - Blocos de inode

Nos sistemas UNIX, ainda pode haver um arquivo excluído ativo. Algum processo ainda pode ter o arquivo aberto para leitura ou gravação, ou para ambas, ou algum processo ainda pode estar executando o código a partir do arquivo. Todas as outras operações de exclusão de arquivos são postergadas até que o arquivo não mais esteja ativo. Nesse estado de exclusão suspensa, o inode continua a ser alocado, mas tem uma contagem de referência de zero. O utilitário ils (lista o arquivo pelo número de inode) do *Coroner's Toolkit* tem uma opção para localizar esses arquivos. O comando a seguir mostra todos os arquivos excluídos, mas que continuam ativos no sistema de arquivos hdal:

 # ils -o /dev/hdal

Depois de um arquivo ser realmente excluído, o bloco de inode é marcado como não utilizado no bitmap de alocação de inodes. Algumas informações sobre o atributo de arquivo são destruídas (como mostrado na tabela 1), mas uma grande quantidade das informações é preservada. Em particular, implementações do Linux 2.2Ext2fs preservam as conexões entre o bloco de inode de arquivo e seus blocos de dados de arquivo. Com implementações mais antigas e mais atuais do Linux, alguns ou todos os endereços dos blocos de dados são perdidos.

1.4 - Blocos de dados

Blocos de dados de um arquivo excluído são marcados como não utilizados no bitmap de alocação de blocos de dados, mas seu conteúdo permanece inalterado. O sistema de arquivos Ext2fs do Linux tem uma opção para apagar blocos de dados de arquivo na exclusão de um arquivo, mas esse recurso atualmente não está implementado. De modo geral, os blocos de dados de arquivo não mais estão conectados ao arquivo de nenhuma maneira, exceto no Linux 2.2 Ext2fs, em que todos os blocos de dados permanecem conectados ao bloco de

inode. Nesses sistemas Linux, o comando a seguir recupera os blocos de dados a partir de um arquivo na partição hdal com o número de inode 154881:

 # icat /dev/hdal 154881 > recovered.hdal.154881

Nesse caso, o arquivo de saída deve ser criado em um sistema de arquivos diferente do sistema de arquivos a partir do qual arquivos excluídos estão sendo recuperados.

2 - ARQUIVOS EXCLUÍDOS

2.1 A Persistência das informações de arquivos excluídos

Computadores excluem arquivos com frequência. Às vezes isso acontece devido a uma solicitação explícita de um usuário. Frequentemente, as informações são excluídas implicitamente quando um aplicativo descarta algum arquivo temporário para seu próprio uso interno. Exemplos dessa atividade implícita de exclusão de arquivos são os arquivos temporários do editor de textos, os arquivos com resultados intermediários dos compiladores de programas e os arquivos nos caches dos navegadores Web. À medida que utiliza um sistema de computador, você inconscientemente deixa para trás vestígios das informações excluídas.

Os sistemas de computador têm memória própria, deixando seus próprios vestígios da exclusão como um efeito colateral da atividade que acontece no segundo plano. Exemplos da atividade de exclusão em segundo plano são os arquivos temporários nas filas de um sistema de correio ou nas filas de uma impressora. Tais arquivos só existem por alguns segundos ou minutos. Se sua máquina fornecer serviços de rede para outros sistemas, as informações de sistemas dos quais você nem mesmo está ciente atingirão seu disco. Os arquivos de log são outro exemplo da atividade de criação e exclusão de arquivos em segundo plano.

Com muitos sistemas de computador, as informações nos arquivos excluídos permanecem intactas no disco, em blocos de dados não-alocados e em blocos de atributos de arquivo não-alocados, até serem sobrescritas por uma outra atividade. Isso pode resultar na revelação inesperada das informações quando uma máquina (ou disco) é aposentada e revendida como um equipamento de segunda mão.

Veremos agora como as informações nos arquivos excluídos podem escapar intactas à destruição por meses ou mesmo anos, e como informações nos atributos de arquivos excluídos podem dar idéias da atividade passada de um sistema. Veremos ainda como atividades passadas podem ser preservadas no espaço em disco não-alocado e por que informações nos arquivos excluídos podem ser mais persistentes do que as informações nos arquivos usuais.

2.2 - Exemplo da persistência de informações excluídas

Em 1996, Peter Gutmann apresentou um estudo sobre o problema da destruição de dados (Gutmann, 1996) e, em 2001, uma continuação (Gutmann, 2001). A preocupação de Peter é com a segurança das informações sigilosas, tais como chaves criptográficas e dados não criptografados. A melhor criptografia no mundo não é suficiente quando as chaves ou conteúdos não criptografados podem ser recuperados.

A destruição da informação acaba se revelando algo difícil de se fazer. Chips de memória podem ser lidos mesmo depois de uma máquina ser desligada. Dados em um disco magnético podem ser recuperados mesmo depois de serem sobrescrito múltiplas vezes.

Embora chips de memória e discos magnéticos serem projetados para armazenar informações digitais, a tecnologia subjacente é analógica. Com o armazenamento analógico de informaçôes digitais, o valor de um bit é uma combinação complexa dos valores armazenados no passado. Chips de memória têm modos de diagnóstico não documentados que permitem o acesso a valores menores do que um bit. Com circuitos eletrônicos modificados, os sinais a partir de cabeçotes de leitura de disco pode revelar dados mais antigos como modulações no sinal analógico.

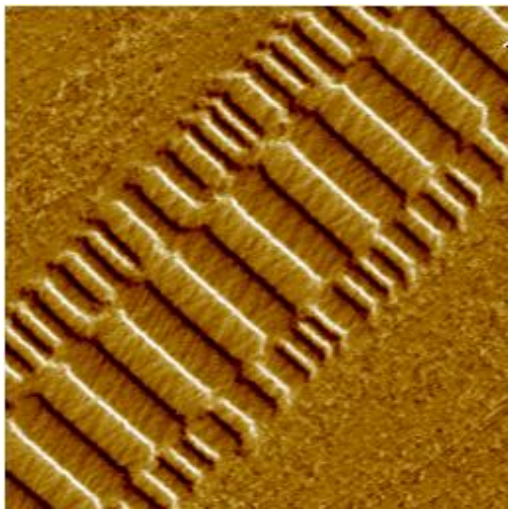

Figura 1 - Resíduos de informações sobrescritas nas trilhas do disco magnético (FARMER;VENEMA, 2007)

Outra maneira de examinar discos é por meio do exame de superfície. A Figura 1 apresenta um exemplo espectacular de antigos padrões magnéticos que persistem em uma trilha do disco.

Entretanto, uma grande quantidade de informações excluídas pode ser recuperada sem jamais examinar a superfície do disco magnético, mesmo quando essas informações foram excluídas há muito tempo. Farmer e Venema (2007) examinaram o disco de uma máquina que começou sua vida como um PC com *Windows*, teve uma segunda vida como um *firewall* Solaris, e que finalmente foi convertida em um sistema Linux. Depois que um sistema operacional foi instalado sobre outro, os arquivos excluídos do Solaris e *Windows* ainda estavam bem presentes como o conteúdo de blocos de disco não-alocados. Eles ainda localizaram cópias intactas de muitos arquivos de configuração excluídos do *firewall* Solaris. Elas poderiam permanecer na máquina por muitos anos sem nunca serem sobrescritas.

2.3 - Medindo a persistência do conteúdo de um arquivo excluído

O exemplo anterior do PC *Windows* é incomum. Foi possível estimar a idade das informações excluídas simplesmente porque os arquivos windows foram instalados antes do Solaris e porque os arquivos Solaris foram instalados antes do Linux. Embora arquivos frequentemente contenham indícios de quando as informações foram criadas, o conteúdo de um arquivo excluído raramente fornece indícios óbvios de quando esse arquivo foi excluído.

Para descobrir por quanto tempo o conteúdo de um arquivo excluído sobrevive, Farmer e Venema (2007) conduziram um experimento de 20 semanas em algumas máquinas em suas próprias redes. Foi feito um acompanhamento diário no histórico de cada bloco, do momento em que ele foi excluído ao momento em que ele foi sobrescrito. Todas as noites um script automatizado examinava cada bloco de disco de 1KB e registrava um hash do conteúdo do bloco de disco e também o status do bloco: alocado, não-alocado ou overhead como inode (atributo de arquivo) ou bloco de bitmap.

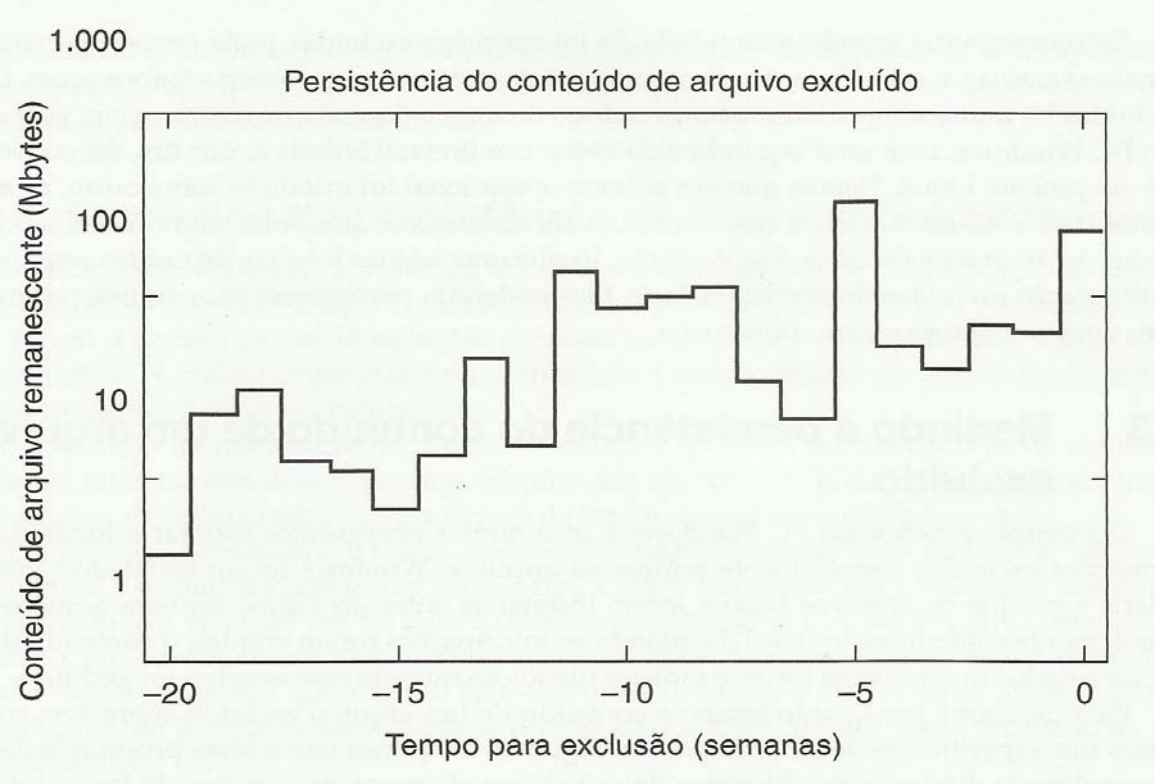

Figura 2 - Persistência do conteúdo de arquivo excluído
(FARMER; VENEMA, 2007)

A Figura 2 mostra a distribuição do conteúdo do arquivo sobrevivente versus o momento da exclusão para um pequeno sistema de arquivos de servidor. O tempo 0 corresponde ao presente e tempos negativos representam o passado. Os dados no gráfico representam um terço de todos os blocos de disco não-alocados no sistema de arquivos. A máquina, spike.porcupine.org, é um servidor FreeBSD de Venema. Apesar da flutuação significativa, a tendência é clara. Foram descobertos aproximadamente 100MB do conteúdo excluído menos de uma semana antes e cerca de 10 MB eram sobra do conteúdo excluído 20 semanas antes. Na época dessas medições, essa máquina tratava aproximadamente 1500 mensagens de correio eletrônico diariamente (cerca de 10 MB de dados) e limitava quantidades de serviços WWW, FTP e DNS. O login pelo sistema de correio foi responsável por aproximadamente 1,5 MB de dados todos os dias. O sistema de arquivos de 8 GB estava 50% cheio, e a maior parte do conteúdo do correio eletrônico e do *login* foi automaticamente excluída depois de um curto período de tempo.

Nessa máquina em particular, metade do conteúdo do arquivo excluído foi sobrescrita em 35 dias. A Tabela 2 resume os resultados para três sistemas de arquivos: FreeBSD (spike.porcupine.org), estação de trabalho e servidor Linux (flying.fish2.com) e servidor WWW e FTP (www.porcupine.org). Há alguma variação, mas as diferenças menores que um ou dois fatores não são significantes. A lição é que os dados excluídos podem permanecer por semanas ou mais.

Tabela 2 - Meia vida do conteúdo do arquivo excluído para três sistemas (FARMER;VENEMA, 2007)

Máquina	Sistema de Arquivos	Meia-vida
spike.porcupine.org	Disco Inteiro	35 dias
flying.fish2.com	/	17 dias
flying.fish2.com	/usr	19 dias
www.porcupine.org	Disco Inteiro	12 dias

O que o gráfico e a tabela não mostram é como as informações sobrevivem. Um arquivo excluído decompõe-se lentamente ou permanece intacto até ser finalmente destruído? Farmer e Venema (2007) concluem que um sistemas de arquivos que sofrem de problemas de fragmentação, um arquivo excluído seja destruído gradualmente, um fragmento por vez, já com sistemas de arquivos que impedem a fragmentação, é esperado que um arquivo excluído permaneça intacto até ser destruído, em um período de tempo relativamente curto.

3 - OS ATRIBUTOS DE TEMPO (MACTIMES) DE ARQUIVOS EXCLUÍDOS

3.1 - A persistência da força bruta de MACtimes de arquivos excluídos

Às vezes saber quando algo aconteceu é mais valioso do que saber o que aconteceu. Por isso, em uma perícia são utilizadas técnicas para localizar ou utilizar os dados relacionados ao tempo. Há duas maneiras de obter dados relacionados ao tempo: observando as atividades diretamente e observando quais efeitos as atividades secundárias têm sobre seu ambiente.

Uma das coisas mais simples de entender e usar em uma investigação são os MACtimes. MACtimes são uma maneira abreviada de se referir aos três atributos de tempo – *mtime, atime e ctime* – que são anexados a qualquer arquivo ou diretório no UNIX, *Windows* e em outros sistemas de arquivo.

O atributo *atime* refere-se à última data/hora em que o arquivo ou diretório foi acessado. O atributo *mtime*, ao contrário, muda quando o conteúdo de um arquivo é modificado. O atributo *ctime* monitora quando o conteúdo ou as meta-informações sobre o arquivo mudaram: o proprietário, o grupo, as permissões de arquivo e assim por diante. O atributo *ctime* também pode ser utilizado como uma aproximação de quando um arquivo foi excluído.

Para descobrir a solidez das informações de atributo de arquivo excluído, Farmer e Venema (2007) configuraram uma máquina Linux descartável e fizeram o *download* da versão 4 do código-fonte do *rootkit* Linux, lrk4.tgz, a partir de um dos muitos sites de *download* de *malware.* O procedimento de instalação do *softw*are *rootkit* instala um programa analisador de senha de rede e substitui uma dúzia dos programas de sistemas por versões modificadas. O procedimento de instalação do *rootkit* usa técnicas secretas para assegurar que os arquivos do programa modificado tenham os mesmos MACtimes, tamanhos de arquivo e valores de verificação de redundância cíclica de arquivo dos arquivos sendo substituídos.

Depois de compilarem o *software rootkit*, Eles executaram o procedimento que instala os utilitários modificados do sistema e então, removeram o código-fonte do *rootkit*, da mesma maneira como um invasor faria. Em seguida, realizaram a pior coisa imaginável. Fizeram o *download* da distribuição do código-fonte do *Coroner's*

Toolkit, desempacotaram o repositório de arquivos exatamente no mesmo diretório em que o “invasor” desempacotou o repositório de arquivos do *rootkit,* compilaram o conjunto de ferramentas e então executaram o *software* a fim de coletar “vestígios”.

Utilizando o *Coroner’s Toolkit* dessa maneira, conscientemente Farmer e Venema (2007) destruíram grandes quantidades de informações. Eles sobrescreveram os blocos de dados que pertenciam aos arquivos excluídos do *rootkit*, sobrescreveram os blocos de atributo de arquivo que pertenciam aos arquivos excluídos do *rootkit* e destruíram as informações sobre data/hora do último acesso para os arquivos relacionado ao compilador. Mesmo depois dessa destruição, o *Coroner’s Toolkit* ainda localizou os atributos de 476 arquivos e diretórios excluídos que existiam durante o incidente do *rootkit*.

Na Figura 3, o gráfico do *ctime* na parte superior mostra as data/horas aproximadas em que os arquivos foram excluídos. O pico mais alto à direita do gráfico mostra quando o diretório do *rootkit* foi removido, juntamente com o código-fonte e os arquivos da saída do compilador.

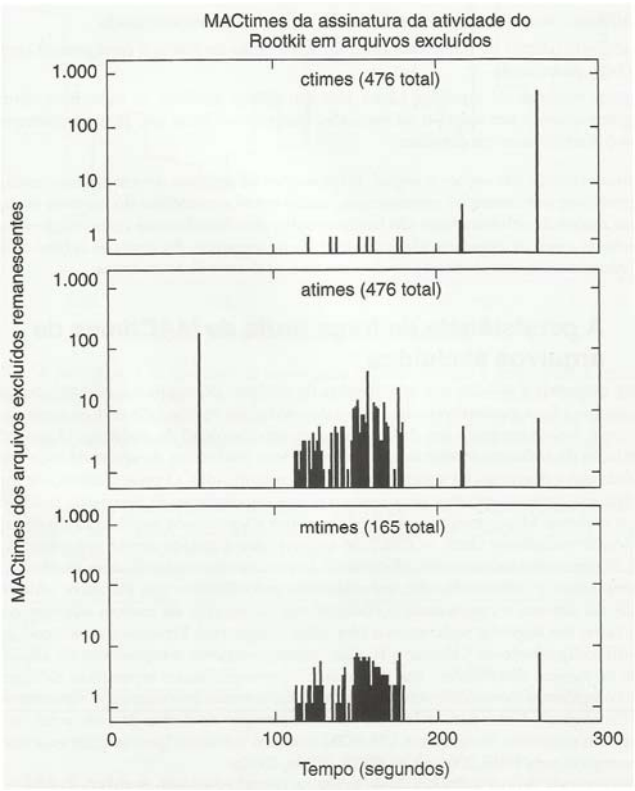

Figura 3 – MACtimes da assinatura da atividade do Rootkit em arquivos excluídos (FARMER;VENEMA, 2007)

O gráfico do *atime*, no meio, mostra quando os arquivos excluídos foram acessados a fim de compilar o código-fonte do *rootkit*. O pico mais alto a esquerda corresponde aos arquivos do *rootkit* que foram desempacotados, mas não foram utilizados. Isso é um artefato de muitos sistemas de arquivos UNIX: eles configuram o *atime* de um arquivo de acordo com a data/hora em que ele foi criado.

O gráfico de *mtime*, na parte inferior, mostra a última vez que o conteúdo do arquivo foi modificado antes de ser excluído. Somente 165 dos 476 arquivos excluídos residuais tinham *mtimes* na janela de data/hora do incidente; os pontos dos dados correspondem aos arquivos que foram produzidos ao compilar o código-fonte do *rootkit*. Os 311 arquivos excluídos residuais remanescentes tinham datas/horas da última modificação no arquivo quase idênticas no passado distante. Presumivelmente, essa era a data/hora em que o código-fonte do *rootkit* foi empacotado para distribuição em alguma outra máquina.

A indicação da sobrevivência dos MACtimes no arquivo excluído era tão forte que seria praticamente impossível que qualquer pessoa que saiba o que procurar não a percebesse, mesmo dias depois desse evento. A razão dessa forte indicação é que o *software rootkit*, como qualquer outro *software*, sofre de inchaço de código e excesso de recursos desnecessários. A versão 4 do *rootkit* Linux tem um *footprin*t total bastante grande de aproximadamente 780 arquivos e diretórios, incluindo os arquivos de saída de compilador que são produzidos quando o *software* é compilado. O *Coroner's Toolkit*, por outro lado, tem um *footprint* de "somente" 300 arquivos. Esse número não é suficientemente grande para apagar todas as informações sobre o MACtimes do arquivo excluído.

3.1.1 - Utilizando MACtimes para detecção de *malware*

Os MACtimes de arquivos excluídos ou existentes podem revelar que alguém pode ter adicionado um *malware* específico a um sistema. O *malware*, como qualquer *software*, normalmente é distribuído na forma de repositórios de arquivo que contém múltiplos arquivos. O *software* que mantém os repositórios de arquivos preserva cuidadosamente os registros de data/hora da última modificação dos arquivos originais e cuidadosamente restaura esses registros de data/hora na

extração. Mesmo depois de os arquivos serem excluídos, os registros de data/hora da última modificação do *malware* podem persistir nos blocos de atributo de arquivo não-alocados.

O incidente com o *rootkit* tem uma assinatura reveladora, como podemos ver na Figura 4. Das datas/horas da última modificação dos 311 arquivos excluídos não presentes na janela de tempo do incidente, 296 eram idênticos dentro de 15 segundos. Se a data/hora nos registros de data/hora foi ou não forjada não importa. Um pico com centenas de *mtimes* excluídos nesse intervalo de data/hora particular deve levantar suspeitas.

Uma análise de assinatura de MACtime do *malware* pode ser feita em um curto período de tempo se comparada com o tempo necessário para examinar todos os blocos em um disco. Por exemplo, o comando ils (lista inodes) do *Coroner's Toolkit* pode ler todos os 2 milhões de blocos de atributos de arquivos dentro de um sistema FreeBSD de 8GB em menos de meio minuto, bem menos tempo do que seria necessário para examinar *gigabytes* de blocos de dados.

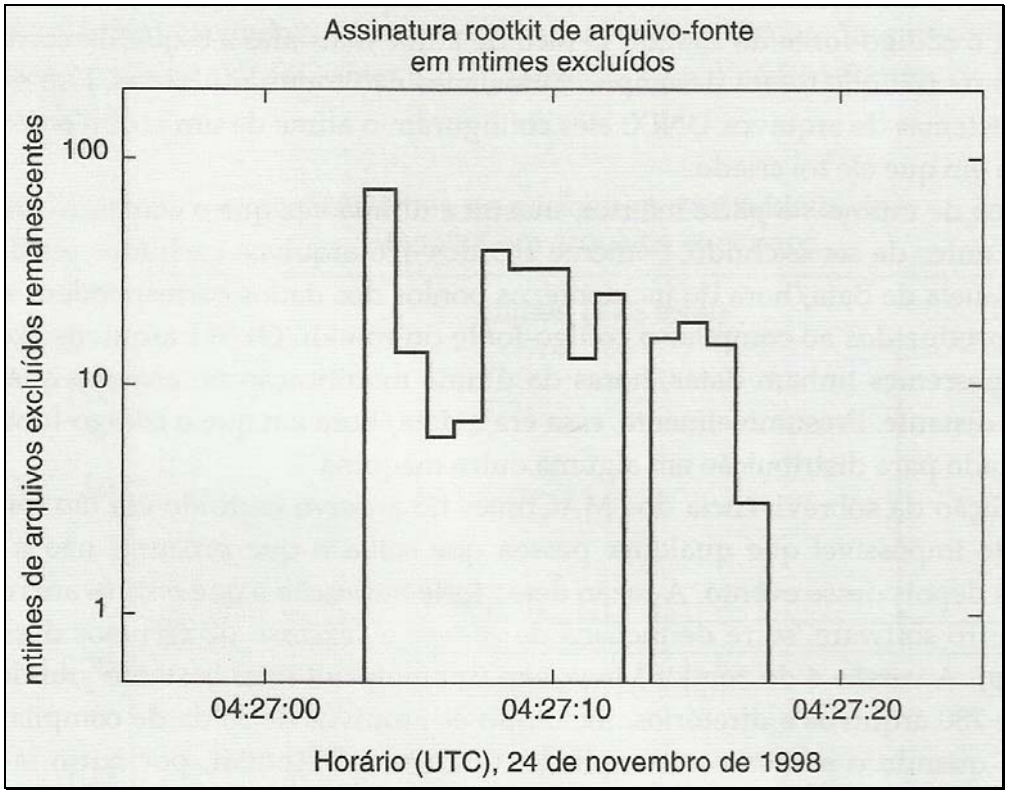

Figura 4 – A assinatura dos arquivos-fonte excluídos do rootkit (FARMER;VENEMA, 2007)

3.2 - A persistência de longo prazo de MACtimes de arquivos excluídos

A persistência de força bruta dos MACtimes de arquivos excluídos, como mostrado anteriormente, conta com atividades maciças do sistema de arquivos em um tempo relativamente curto. Isso produz uma forte indicação de que se destaca bem acima do ruído. O sinal sobrevive mesmo quando o evento é seguido de atividades significativas do sistema de arquivos.

O exemplo de força bruta não informa por quanto tempo as informações sobre o MACtime de arquivos podem sobreviver. Para explorar essa questão Farmer e Venema (2007) analisaram os arquivos excluídos de várias máquinas. Eles ficaram surpresos em descobrir que as informações sobre o MACtime dos arquivos excluídos retroagiam até um ano ou mais, em geral ao momento em que o sistema de arquivos foi criado no disco.

A Figura 5 mostra os atributos MACtime de arquivos excluídos para uma máquina de servidor FreeBSD que passa a maior parte do tempo realizando trabalho de rotina: enviar e receber *e-mails*; fornecer serviços de rede como DNS, FTP e WWW além de manter arquivos de *log*. Há uma exceção à rotina. O proprietário do sistema é o autor de um servidor de *e-mail* de código-fonte aberto e é o “primeiro usuário” de cada distribuição. O “primeiro uso” envolve desempacotar, compilar e remover o código-fonte. No momento em que essa medição foi feita, as distribuições aconteciam mais ou menos mensalmente.

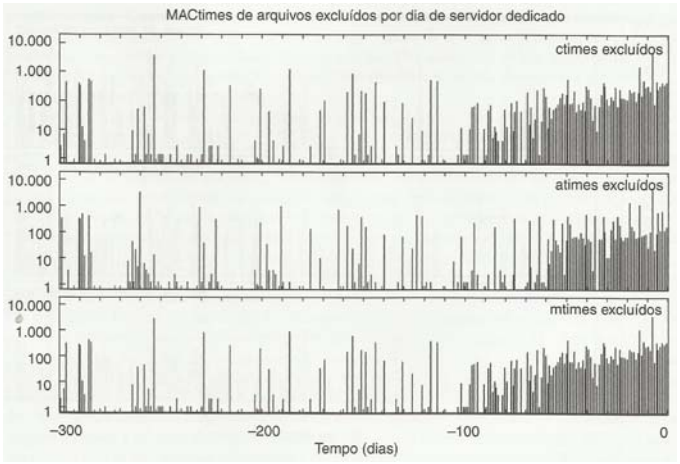

Figura 5 - A distribuição no tempo de atributos MACtime de arquivos excluídos para sistema de arquivos de um pequeno servidor (FARMER;VENEMA, 2007)

À direita da figura 5, as informações sobre o MACtimes de arquivos excluídos decaem gradualmente à medida que voltamos no tempo. Nessa máquina particular, 90 por cento das informações sobre o MACtimes de arquivos excluídos são sobrescritas em cerca de 60 dias, como resultado das atividades de rotina da máquina. Isso corresponde a uma meia-vida de aproximadamente 20 dias. Isso é menor que a meia vida de 35 dias encontrada anteriormente para conteúdo de arquivo excluído, mas essa diferença não é significativa dada à precisão das medições. À esquerda da figura, as distribuições de MACtime de arquivo excluído estão relativamente esparsas, mas os padrões retroagem até o momento em que o FreeBSD foi instalado na máquina.

O gráfico na parte superior da figura 5, com a distribuição do atributo ctime, mostra o tempo aproximado a partir do momento em que um arquivo foi excluído. Quaisquer atributos *ctime* de arquivos excluídos que sobrevivem além dos primeiros 100 dias do histórico possivelmente serão o resultado de atividade não rotineira na máquina. Para essa máquina em particular, o candidato mais provável é a compilação e instalação do novo *software* de *e-mail* na máquina e a remoção subseqüente do código-fonte.

O gráfico de *atime*, no meio da figura 5, mostra a última vez em que um arquivo foi acessado antes de ser excluído. As informações sobre o *atime* retroagem até centenas de dias assim como o gráfico de *ctime* (tempo para exclusão de arquivo). Isso de jeito nenhum é o que poderíamos encontrar com MACtimes de arquivo usuais: com arquivos usuais, os atimes são o componente MACtime mais volátil. Com as informações excluídas, as regras são diferentes: as datas/horas do último acesso de arquivos excluídos são tão persistentes quanto qualquer atributo de arquivo excluído, pois elas não mais são atualizadas.

O gráfico na parte inferior da figura 5 mostra a distribuição do atributo *mtime* (tempo para a modificação do arquivo). O sistema de arquivos FreeBSD configura o *mtime* como a data/hora da exclusão e, portanto, seu gráfico é idêntico ao gráfico *ctime*.

3.3 - O impacto da atividade dos usuários sobre MACtimes de arquivos excluídos

Assim como os MACtimes regulares, os MACtimes de arquivos excluídos são sensíveis aos padrões de uso do sistema. Os dados da seção anterior são típicos de uma máquina dedicada que passa a maior parte do tempo realizando trabalho de rotina. A análise de uma estação de trabalho pessoal é mais complexa, pois o comportamento do sistema é dominado por atividades menos previsíveis do usuário.

A Figura 6 mostra os padrões de MACtimes de arquivos excluídos para uma estação de trabalho pessoal, onde o tempo 0 corresponde ao presente e tempos negativos representam o passado. Essa máquina é o principal ambiente de trabalho do usuário para enviar e receber e-mails, navegar pela *Web* e desenvolver *softwares*. Além disso, a máquina também realiza uma quantidade limitada de serviços rotineiros de Web e DNS. Os padrões de MACtime para essa máquina são significativamente diferentes daqueles para o servidor dedicado na figura anterior.

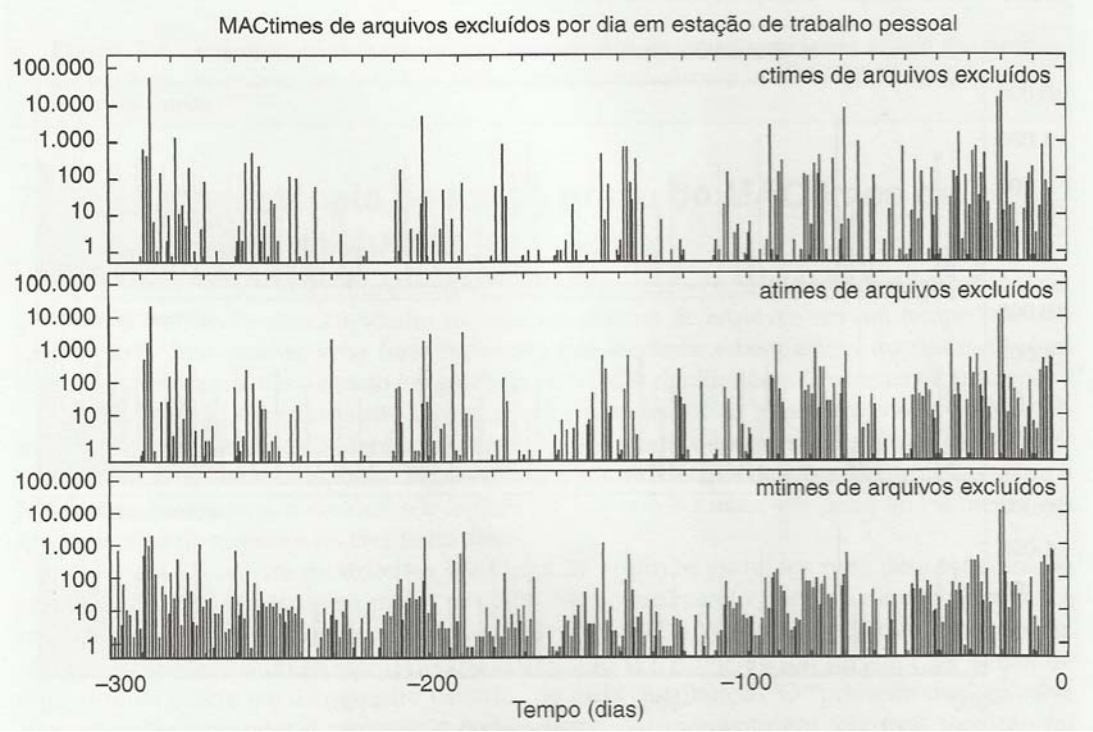

Figura 6 - A distribuição no tempo dos *MACtimes* de arquivos excluídos para o sistema de arquivos de uma estação de trabalho pessoal (FARMER;VENEMA, 2007)

À direita da figura 6, o gráfico de *ctime* (tempos para exclusão) de arquivos excluídos e o gráfico *atime* (tempos para o último acesso de leitura) mostram o decaimento do histórico recente. O decaimento não é tão suave quanto na Figura 5. À esquerda, os gráficos de *ctime* e *atime* mostram o residual das atividades significativas no passado mais distante. Como ocorre com o servidor dedicado, o residual retroage até o momento em que o sistema de arquivos foi criado.

Na figura 6, o gráfico de *mtimes* (tempos para última modificação) de arquivos excluídos da estação de trabalho é diferente de todos os outros gráficos anteriores. A distribuição da estação de trabalho na verdade é constituída de dois componentes. Um componente está correlacionado com os gráficos de ctime e atime e corresponde a arquivos com tempo de vida relativamente curto; o outro componente aparece mais ou menos como informações independentes do tempo de residuais de aproximadamente dez arquivos excluídos por dia.

A existência do componente independente de data/hora significa que alguns arquivos não tem nenhuma correlação entre a data/hora da última atualização e a data/hora da exclusão. Isso é consistente com o comportamento do principal usuário. De acordo com o usuário, os arquivos foram acumulados ao longo do tempo a uma velocidade constante. A cada poucos meses, o usuário exclui um grande número de arquivos para liberar algum espaço.

3.4 - A confiabilidade das informações de arquivos excluídos

Os MACtimes ou conteúdo de arquivos excluídos fornecem ao investigador excelentes oportunidades. Como as informações excluídas são menos visíveis do que as informações usuais, um oponente tem menos chances de estar ciente de que essas informações existem e, portanto, tem menos probabilidade de adulterá-las. Por exemplo, se um arquivo de *log* foi modificado, é possível que partes do arquivo não-modificado ainda possam ser recuperadas a partir do sistema de arquivos não-alocado.

Os MACtimes de arquivos excluídos só herdam algumas das limitações dos MACtimes de arquivos existentes. Antes da exclusão, um arquivo é relativamente fácil de acessar. As informações sobre o MACtime desse arquivo são voláteis e

facilmente forjadas. Depois da exclusão, é relativamente fácil sobrescrever os MACtimes de um arquivo excluído de maneira não-seletiva criando um grande número de pequenos arquivos. Alterar atributos excluídos específicos torna-se mais difícil, pelo menos nos sistemas que podem revogar permanentemente acesso de gravação à memória do *kernel* ou dispositivos de disco.

Um argumento semelhante pode ser feito para o conteúdo de arquivo excluído. Antes da exclusão, as informações são relativamente fáceis de acessar e, portanto, relativamente fáceis de modificar. Depois da exclusão, é relativamente fácil sobrescrever o conteúdo de arquivo excluído criando um pequeno número de arquivos grandes. Alterar blocos de dados excluídos específicos torna-se mais difícil, pelo menos nos sistemas que podem revogar permanentemente acesso de gravação à memória do *kernel* ou dispositivos de disco.

Depois da exclusão, forjar MACtimes ou conteúdo de arquivo pode ser arriscado. A abordagem simples e direta é driblar o sistema de arquivos e gravar no disco bruto. Há uma grande possibilidade do sistema de arquivos quando um oponente de má fé compete com um sistema de arquivos de boa fé quanto ao acesso ao mesmo bloco do sistema de arquivos. Uma abordagem mais confiável envolveria um módulo do *kernel* que realiza a limpeza enquanto coopera com o sistema de arquivos, em vez de competir com ele.

A inteireza é uma questão óbvia no que se refere às informações excluídas. Ao contrário dos MACtimes do conteúdo de arquivos existentes, informações excluídas podem ser sobrescritas em qualquer momento e, portanto, têm maior probabilidade de ser incompletas. A ausência de informações específicas não deve ser utilizada como uma evidência de que as informações nunca foram armazenadas. Com o armazenamento não-alocado, isso é ainda mais verdadeiro do que com informações de arquivos comuns.

3.5 - Por que informações de arquivo excluído sobrevivem intactas

Nos tópicos anteriores, foi demonstrado como as informações excluídas podem escapar da destruição por meses ou mesmo por anos. Neste tópico, veremos

como o projeto de um sistema de arquivos de alto desempenho pode influenciar a sobrevivência de longo prazo das informações de arquivos excluídos.

Sistemas de arquivos de alto desempenho evitam movimentos do cabeçote de disco, mantendo as informações relacionadas juntas. Isso não apenas reduz a fragmentação do conteúdo de um arquivo individual, como também reduz retardos ao se percorrer diretórios para acessar um arquivo. Embora os detalhes a seguir sejam específicos aos sistemas UNIX populares, é esperado que efeitos semelhantes de persistência ocorram com qualquer sistema de arquivos que tenha boas propriedades de localidade.

O típico sistema de arquivos UFS ou Ext3fs é organizado em zonas de igual tamanho, como mostrado na figura 7. Esses sistemas de arquivos descendem do *Berkeley Fast File System* (McKusick et al., 1984) e são encontrados no Solaris, FreeBSD e Linux (Card et al., 1994). Tamanhos de zona típicos são 32.768 blocos; o tamanho do bloco real depende do tipo de sistema de arquivos. Novos arquivos são criados preferencialmente na mesma zona do sistema de arquivos do seu diretório pai; isso aprimora a clusterização, ou agrupamento, das informações relacionadas. Novos diretórios são criados nas zonas que têm poucos diretórios e muito espaço não utilizado.

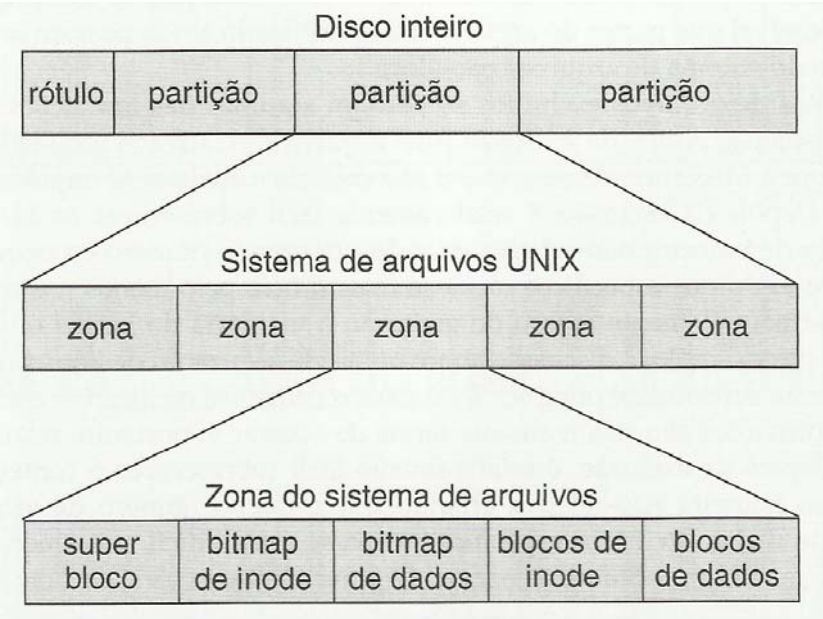

Figura 7 - Organização em disco de um típico sistema de arquivos UFS ou Ext3fs (FARMAR;VENEMA, 2007)

A Figura 7 mostra a organização em disco de um típico sistema de arquivos UFS ou Ext3fs. O espaço de armazenamento é dividido em múltiplas zonas. Cada zona contém seu próprio bitmap de alocação, blocos de dados de arquivo e blocos de atributo de arquivo (inode). Normalmente, as informações sobre um pequeno arquivo são armazenadas inteiramente dentro de uma zona.

Mantendo as informações relacionadas dentro da mesma zona do sistema de arquivos, sistemas de arquivos UFS Ext3fs típicos tendem a clusterizar, isto é, agrupar os arquivos de diferentes usuários ou aplicativos de acordo com as diferentes zonas do sistema de arquivos. Por isso, o tempo de sobrevivência das informações excluídas depende fortemente do volume das atividades de gravação em arquivo dentro da sua zona. A Figura 8 mostra a porcentagem de blocos de dados por zona do sistema de arquivos que foram sobrescritos no período de um mês para um pequeno servidor *FreeBSD* com um sistema de arquivos de 8GB que foi preenchido com 50 por cento de sua capacidade. A atividade de gravação em disco se concentra dentro de zonas específicas. Menos de 4 por cento de todos os blocos de dados foram modificados durante esse intervalo de um mês.

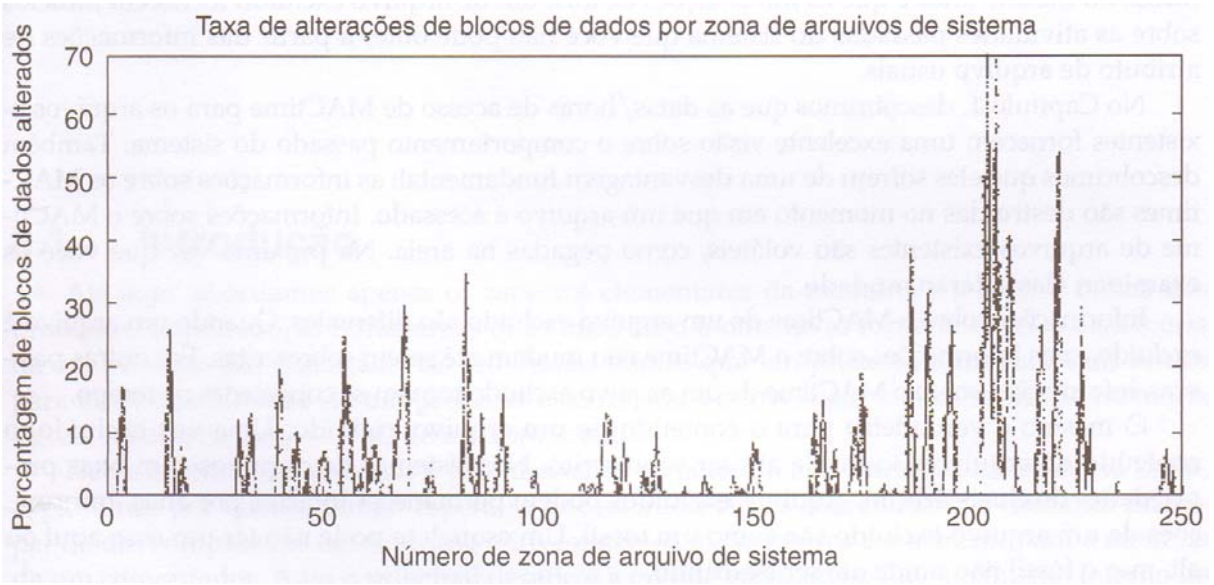

Figura 8 - Taxa de alterações de blocos de dados por zona de arquivos de sistema (FARMER;VENEMA, 2007)

Quando um arquivo é excluído em uma zona de alta atividade, as informações sobre seus blocos de dados e atributo de arquivo serão sobrescritas de maneira relativamente rápida por novos arquivos.

Por outro lado, quando um arquivo é excluído em uma zona de baixa atividade, as informações sobre seus blocos de dados e atributo de arquivo podem escapar a destruição desde que a atividade do sistema de arquivos permaneça dentro de outras zonas do sistema de arquivo. À medida que o disco se enche, a atividade de gravação inevitavelmente migrará para vizinhanças calmas das zonas de baixa atividade, transformando-as em zonas de alta atividade destrutiva. Até esse momento, as informações de arquivo excluído nas zonas de baixa atividade podem sobreviver intactas e em quantidades volumosas.

Os sistemas de computador tendem a passar a maior parte do tempo executando atividades de rotina. Em termos das zonas do sistema de arquivos, isso significa que atividades de gravação tendem a se concentrar em um número limitado de zonas em que as informações são criadas e destruídas de maneira relativamente rápida. O restante do sistema de arquivos permanece relativamente estático e há a probabilidade de qualquer arquivo excluído sobreviver por um período de tempo relativamente longo. Assim, o que é observado, é que vestígios das atividades de rotina desaparecem rapidamente, enquanto atividades incomuns se destacam porque seus vestígios sobrevivem mais tempo.

CONCLUSÃO

Pode-se observar que informações de arquivo excluído podem sobreviver intactas por meses ou mesmo anos e que as informações de atributo de arquivo excluído fornecem indícios sobre as atividades passadas do sistema que você não pode obter a partir das informações de atributo de arquivo usuais.

As datas/horas de acesso de MACtime para os arquivos existentes fornecem uma excelente visão sobre o comportamento passado do sistema, mas eles sofrem de uma desvantagem fundamental: as informações sobre os MACtimes são destruídas no momento em que um arquivo é acessado. Informações sobre o MACtime de arquivos existentes são voláteis, é muito provável que na próxima vez que forem examinadas, elas já terão mudado.

Informações sobre o MACtime de um arquivo excluído são diferentes. Quando um arquivo é excluído, suas informações sobre o MACtime não mudam até serem sobrescritas. Em outras palavras, informações sobre o MACtime de um arquivo excluído tomam-se congeladas no tempo.

O mesmo é verdadeiro para o conteúdo de um arquivo excluído. Uma vez excluído, o conteúdo de arquivo não muda até ser sobrescrito. Nos sistemas de arquivos com boas propriedades de clusterização, arquivos excluídos podem permanecer intactos por anos. Informações de um arquivo excluído são como um fóssil. Um esqueleto pode não ter um osso aqui ou ali, mas o fóssil não muda até ser destruído.

Esse fenômeno de exclusão e persistência pode acontecer em qualquer nível de abstração. No nível de abstração dos sistemas de arquivos, as informações excluídas persistem como blocos de disco não-alocados até serem sobrescritas. No nível de abstração dos cabeçotes de leitura do disco magnético, as informações sobrescritas persistem como modulações analógicas nas informações mais recentes. E, no nível de abstração dos domínios magnéticos, as informações sobrescritas persistem como padrões magnéticos nos lados das trilhas magnéticas.

Em cada camada na hierarquia das abstrações que compõe os sistemas de computador, as informações tomam-se congeladas quando excluídas. Embora informações excluídas tornem- se cada vez mais ambíguas à medida que passamos

para níveis mais baixos de abstração, também descobrimos que as informações excluídas tomam-se ainda mais persistentes. A volatilidade é um artefato das abstrações que tornam os sistemas de computador úteis.

Tudo isso tem conseqüências importantes não apenas aos invasores cuja atividade é reconstruída com a análise de invasão post-mortem, como também à privacidade dos usuários legítimos dos sistemas de computador.

REFERÊNCIAS BIBLIOGRÁFICAS

ABELL, Victor A. “**The lsof (list open files) tool**”, 2004. Disponível em: <http://ftp.cerias.purdue.edu/pub/tools/unix/sysutils/lsof/README>. Acessado em 25/10/2012

ANDERSON, Ross; NEEDHAM, Roger; SHAMIR, Adi. “**The Steganographic File System**”. In *Information Hiding, Second International Workshop, IH’98,* editado por D. Aucsmith. SpringerVerlag, 1998. Disponível em: <http://www.cl.cam.ac.uk/~rja14/Papers/stego-fs.pdf>. Acessado em: 24/09/2012

ARBAUGH, W. A.; FARBER, D. J.; SMITH, J. M. “**A Secure and Reliable Bootstrap Architecture**”. In *Proceedings of the 1997 ILEL Symposium on Security and Privacy,* mai. 1997, pp. 65-71. Disponível em: <http://www.cs.umd.edu/~waa/pubs/oakland97.pdf >. Acessado em 24/09/2012

AVALON SECURITY RESEARCH. **The amodload kernel loader for SunOS 4**, 1996. Disponível em: <http://ftp.cerias.purdue.edu/pub/lists/best-of-security/110>. Acessado em 24/09/2012

CALOYANNIDES, Michael A. ***Privacy protection and computer forensics***. 2. ed. Artech House, 2004.

CARD, Rémy, TS’O, Theodore; TWEEDIE, Stephen. “**Design and Implementation of the Second Extended Filesystem**”. In *Proceedings of the First Dutch International Symposium on Linux.* Amsterdam. 8-9 dez. 1994.Disponível em: <http://web.mit.edu/tytso/www/linux/ext2intro.html>. Acessado em: 22/09/2012

CARRIER, Brian. **The Sleuth Kit**, 2004a. Disponível em: <http://www.sleuthkit.org/>. Acessado em: 22/09/2012

CARRIER, Brian. **The Autopsy Forensic Browser**, 2004b. Disponível em: <http://www.sleuthkit.org/>. Acessado em: 22/09/2012

CHOW, Jim; PFAFF, Ben; GARFINKEL, Tal; CHRISTOPHER, Kevin; ROSENBLUM, Mendel. "**Understanding Data Lifetime via Whole System Simulation**". In *Proceedings of the 13th USENIX Security Symposium,* 2004. Disponível em: <http://suif.stanford.edu/collective/taint.pdf>. Acessado em: 15/09/2012

CIFUENTES, Cristina. **The dcc decompiler**, 1994. Disponível em: <http://www.itee.uq.edu.au/~cristina/dcc.html>. Acessado em 15/09/2012

COFFMAN, K. G.; ODLYZKO, A. M. "**Internet growth: Is there a 'Moore's Law' for data traffic?**" In *Handbook of massive data sets,* editado por J. Abello, P. M. Pardalos; M. G. C. Resende, 2002. pp. 47-93. Kluwer.

COOPERATIVE ASSOCIATION FOR INTERNET DATA ANALYSIS. **The CAIDA network telescope project**, 2003. Disponível em: <http://www.caida.org/analysis/security/telescope/>. Acessado em 16/09/2012

DASAN, Vasanthan; NOORDERGRAAF, Alex; ORDORICA, Lou. "**The Solaris Fingerprint Data-base: A Security Tool for Solaris Operating Environment Files**". Sun BluePrints OnLine, mai. 2001. Disponível em: <http://fineit.net/doc/blueprints/0501/Fingerprint.pdf>. Acessado em 18/09/2012

DRAKE, Chris; BROWN, Kimberley. ***Panic! UNIX system crash dump analysis***. Mountain View: Prentice Hall, 1995.

FARMER Dan, VENEMA Wietse, **Perícia Forense Computacional: Teoria e Prática Aplicada.** São Paulo: Pearson Prentice Hall, 2007.

FARMER, Dan; VENEMA, Wietse. **The Coroner's Toolkit**, 2004. Disponível em: <http://www.fish2.com/tct/> e <http://www.porcupine.org/forensics/tct.html>. Acessado em: 19/09/2012

GARFINKEL, Simson L.; SHELAT, Abhi. “**Remembrance of Data Passed: A Study of Disk Sanitization Practices**”, 2003. *IEEE Security; Privacy* 1 (1). Disponível em: <http://simson.net/clips/academic/2003.IEEE.DiskDriveForensics.pdf>. Acessado em: 25/09/2012

GARNER, George. **Forensic Acquisition Utilities**. Includes ‘*dd’* for Windows, 2003. Disponível em: <http://www2.opensourceforensics.org/>. Acessado em: 28/09/2012

GOLDBERG, Ian; WAGNER, David; THOMAS, Randi; BREWER Eric A. “**A Secure Environment for Untrusted Helper Applications: Confining the Wily Hacker**”. In *Proceedings of the 6th USENIX Security Symposium.* San Jose, 1996. Disponível em: <http://www.cypherpunks.ca/~iang/pubs/janus-usenix96.pdf>. Acessado em: 28/09/2012

GUIDANCE SOFTWARE. The EnCase Forensic Tool, 2004. Disponível em: <http://www.guidancesoftware.com/>. Acessado em 24/09/2012

GUIMARÃES Célio Cardoso; OLIVEIRA, Flávio de Souza; REIS, Marcelo Abdalla; GEUS Paulo Lício de, **Forense Computacional: Aspectos Legais e Padronização.** Campinas – SP. Instituto de Computação – UNICAMP, 2008.

GUTMANN, Peter. “**Secure Deletion of Data from Magnetic and Solid-State Memory**”. In *6th USENIX Security Symposium Proceedings.* San Jose, 22-25 jul. 1996. Disponível em: <http://www.cs.auckland.ac.nz/~pgut001/pubs/secure_del.html>. Acessado em: 23/09/2012

GUTMANN, Peter. “**Data Remanence in Semiconductor Devices**”. In 10*th USENIX Security Symposium.* Washington, D.C., 13-17 ago. 2001. Disponível em: <http://www.cypherpunks.to/~peter/usenix01.pdf>. Acessado em 13/09/2012

HOGLUND, Greg; MCGRAW, Gary. ***Como quebrar códigos, a arte de explorar e proteger software****.* São Paulo: Pearson Education Brasil, 2005.

INTERNET SYSTEMS CONSORTIUM. ISC **Internet Domain Survey**, 2004. Disponível em: <http://www.isc.org/>. Acessado em: 17/09/2012

KLEIMAN, 5. R. “**Vnodes: An Architecture for Multiple File System Types in Sun UNIX**”. In *Proceedings of the 1986 USENIX Summer Technical Conference,* 1986. pp. 238-247. Disponível em: <http://cs3.ist.unomaha.edu/~stanw/papers/86-vnode.pdf>. Acessado em 19/10/2012

MCDONALD, Andrew D.; KUHN, Markus G. "***StegFS: A Steganographic File System for Linux***". In *Information Hiding, Third International Workshop, IH'99,* editado por A. Pfitzmann. Dresden, Alemanha: Springer-Verlag. 29 set.-1° out. 1999. Disponível em: <http://www.cl.cam.ac.uk/~mgk25/ih99-stegfs.pdf>. Acessado em 18/10/2012

MCKUSICK, Marshall K.; JOY, William N.; LEFFLER, Samuel J.; FABRY, Robert *S.* "***A Fast File System for UNIX***". *ACM Transactions on Computer Systems,* 1984. Disponível em: <http://www.cs.berkeley.edu/~brewer/cs262/FFS.pdf>. Acessado em: 23/09/2012

MCKUSICK, Marshall Kirk; NEVILLE-NEIL, George V. ***The design and implementation of the FreeBSD operating system****.* Addison-Wesley, 2004.

MILLER, Barton P., et al. "***Fuzz Revisited: A Re-examination of the Reliability of UNIX Utilities and Services***". *Computer Sciences Department, University of Wisconsin*,2000. Disponível em: <http://pages.cs.wisc.edu/~bart/fuzz/fuzz.html>. Acessado em : 17/09/2012

MURILO, Nelson; STEDING-JESSEN, Klaus. *The* ***Chkrootkit Rootkit-Detection Tool***, 2003. Disponível em: <http://chkrootkit.org/>. Acessado em: 16/10/2012

NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY. ***The NIST National Software Reference Library***, 2004. Disponível em: <http://www.nsrl.nist.gov/>. Acessado em: 19/09/2012

N. Beebe, "**Digital Forensic Research: The Good, the Bad and the Unaddressed**," Advances in Digital Forensics V, 2009.

NEMETH, Evi, SNYDER, Garth, SEEBASS, Scott; HEIN, Trent R. ***UNIX Administration Handbook****.* 3 ed. Prentice Hall, 2000.

NEMETH, Evi; SNYDER, Garth; HEIN, Trent R. **Manual completo do Linux: guia do administrador***.* Pearson Education do Brasil/Prentice Hall, 2003.

OPENSSH. ***The OpenSSH Remote Connectivity Software***, 2004. Disponível em: <http://www.openssh.org/>. Acessado em: 08/10/2012

PTACEK, T.; NEWSHAM, T. "***Insertion, Evasion, and Denial of Service: Eluding Network Intrusion Detection***". *Secure Networks*, mc., jan. 1998. Disponível em: <http://insecure.org/stf/secnet_ids/secnet_ids.html>. Acessado em 12/10/2012

QUEIROZ Claudemir, VARGAS Raffael, **Perícia Forense Computacional: Leis Processuais e Estudos de Caso.** Rio de Janeiro: Brasport, 2010

RITCHIE, D. M.; THOMPSON, K. "***The UNIX Time-Sharing System***". *Communications of the ACM,* 1974 17 *(7):* 365-375. Disponível em: <http://cm.bell-labs.com/cm/cs/who/dmr/cacm.html>. Acessado em: 15/10/2012

ROBBINS, Daniel. "***Advanced Filesystem Implementor's Guide***". IBM *developerWorks*, jun. 2001. Disponível em: <https://www.ibm.com/developerworks/linux/library/l-fs/>. Acessado em 20/09/2012

SAFERSTEIN, Richard. ***Criminalistics: an introduction to forensic science***. Prentice HalI, 2003.

SCHNEIER, B.; KELSEY, J. "***Cryptographic Support for Secure Logs on Untrusted Machines***". *In Proceedings of the 7th USENIX Security Symposium,* jan. 1998. pp. 53-62. Disponível em: <http://www.schneier.com/paper-secure-logs.pdf>. Acessado em 15/09/2012

SIMPSON, Duncan. ***The Checkps Rootkit Detector***, 2001. Disponível em: <http://sourceforge.net/projects/checkps/>. Acessado em: 16/09/2012

UFC. **Alerta sobre os sítios arqueológicos de Forquilha-CE**. Disponível em: <http://www.geologia.ufc.br/index.php?option=com_content&task=view&id=463&Itemid=1>. Acessado em 21/10/2012

U.S. DEPARTMENT OF JUSTICE. "***Forensic Examination of Digital Evidence: A Guide for Law Enforcement***". *National Institute of Justice Special Report, Office of Justice Programs*, 2004. Disponível em: <http://www.nij.gov/pubs-sum/199408.htm>. Acessado em: 17/09/2012

WILLIAMS, Michael A. "Anti***-Trojan and Trojan Detection with In-Kernel Digital Signature testing of Executables***", 2002. Disponível em: <http://www.net-security.org/dl/articles/sigexec.pdf>. Acessado em: 13/10/2012

Anexo A - O Coroner's Toolkit e softwares relacionados

A.1 – Introdução

Durante o desenvolvimento deste trabalho, por várias vezes foi citado o uso de uma ferramenta forense denominada Coroner's Toolkit. O objetivo deste anexo é apresentar uma visão geral sobre o Coroner's Toolkit e algumas das suas extensões.

O Coroner's Toolkit é uma coleção de utilitários forenses escritos por Wietse Venema e Dan Farmer (Farmer e Venema, 2004). O *software* foi apresentado primeiro em 1999, em uma aula sobre análise forense no IBM T.J. Watson Research Center. A primeira distribuição geral aconteceu em 2000, via os websites dos autores. Esse *software* foi estendido de várias maneiras por Brian Carrier, que disponibilizou sua versão como o Sleuth Kit (Carrier, 2004a).

Sobre o nome, na Idade Média, o coroner era um funcionário do rei da Inglaterra, que nos casos morte, se dirigia ao local para arrecadar os bens do falecido e para determinar a causa da morte. Com o passar do tempo e a evolução do papel desse funcionário, a função passou a ter um caráter nitidamente investigativo. Atualmente, nos Estados Unidos, embora suas atribuições de modo geral variem de lugar para lugar, o coroner é a pessoa responsável pela gerência e eventualmente a realização da perícia nos casos de crimes contra a vida.

A.2 - Coleta de dados com grave-robber

O comando grave-robber coleta informações forenses. Essa ferramenta pode ser utilizada em uma máquina "ao vivo" da vítima ou em uma imagem do disco do sistema de arquivos da vítima. No modo de coleta "ao vivo", o grave-robber tem por objetivo respeitar a ordem de volatilidade. Ele utiliza muitos dos utilitários que são parte do Coroner's Toolkit para coletar informações nesta ordem:

- Os atributos de todos os comandos e arquivos que o Coroner's Toolkit acessa ao coletar informações. Estes são coletados primeiro para preservar seus atributos de MACtime.

- Informações de status do processo e, opcionalmente, a memória de todos os processos em execução.
- Arquivos excluídos que ainda estão ativos.
- Os arquivos executáveis de todos os processos.
- Todos os atributos dos arquivos excluídos.
- Informações sobre o status de rede.
- Hospeda as informações sobre o status, via comandos dependentes de sistema que fornecem as informações de configuração do sistema.
- Os atributos dos arquivos existentes; isso produz o arquivo body utilizado pela ferramenta mactime como descrito mais adiante.
- Opcionalmente, mantém seguras as informações sigilosas sob o controle dos usuários do sistema, como arquivos que concedem acesso remoto da conta de um usuário e trabalhos cron para execução não supervisionada de comandos em favor dos usuários.
- Cópias dos arquivos de configuração e outros arquivos importantes.

Todas essas informações são armazenadas em um "cofre", uma estrutura protegida de diretórios cujo nome provém do host e a data/hora do início da coleta de dados. Para cada arquivo armazenado nesse cofre, o grave-robber também calcula o hash MD5. Ao final, quando o cofre é fechado, o grave-robber calcula o hash MD5 de todos os arquivos de hash individuais.

Por definição, o grave-robber visualiza as informações provenientes da máquina não- confiável da vítima. Ele freqüentemente utiliza essas informações ao executar comandos do Coroner's Toolkit e comandos de sistema. Ao fazer isso, ele toma grande cuidado de nunca expor essas informações não-confiáveis a um interpretador de comandos de shell.

A.3 - Análise do tempo com mactime

O comando mactime recebe as informações sobre atributos de arquivo a partir de um arquivo body produzido pelo grave-robber e produz um relatório cronológico de todos os métodos de acesso de arquivo por nome de arquivo. Alternativamente, o mactime pode gerar um arquivo body instantaneamente

enquanto varre um sistema de arquivos. Essa ferramenta foi escrita vários anos antes de os autores começarem a trabalhar no Coroner's Toolkit e foi adaptada para se ajustar ao ambiente do grave-robber.

Como um exemplo do tipo de visão que o mactime pode fornecer, as tabelas 1 e 2 apresentam diferentes visualizações da mesma sessão de login remoto. A primeira mostra o que o usuário remoto vê e a segunda mostra o relatório MACtime correspondente. Por razões pedagógicas, esse exemplo utiliza uma máquina muito velha de tal modo que os MACtimes sejam distribuídos ao longo do tempo. Isso permite ver uma separação clara entre a inicialização do servidor telnet e do software de login, o acesso aos arquivos de sistema quando o usuário efetua o login e a inicialização do processo de shell do login do usuário.

Tabela A1 – Visualização do usuário de uma sessão de login remoto (FARMER;VENEMA, 2007)

```
$ telnet sunos.fish2.com
Trying 216.240.49.177...
Connected to sunos.fish2.com.
Escape character is ‘ ^] ’.
SunOS UNIX (sunos) login: zen
Password:
Last login: Thu Dec 25 09:30:21 from flying.fish2.com
Welcome to ancient history!
$
```

Na tabela A2 temos a visualização MACtime da sessão de login remoto mostrada na tabela A1. A coluna mac indica o método de acesso de arquivo (modify, read access ou status change). Nomes de arquivo com o mesmo registro de data/hora estão classificados alfabeticamente.

Tabela A2 - Visualização *MACtime* da sessão de login remoto (FARMER;VENEMA, 2007)

Data/Hora	Tamanho	MAC	Permissão	Proprietário	Grupo	Nome do Arquivo
19:47:04	49152	. a .	-rwxr-xr-x	root	staff	/usr/bin/login
	32768	. a .	-rwxr-xr-x	root	staff	/usr/etc/in.telnetd
19:47:08	272	. a .	-rw-r- -r- -	root	staff	/etc/group
	108	. a .	-r- -r- -r- -	root	staff	/etc/motd
	8234	. a .	-rw-r- -r- -	root	staff	/etc/ttytab
	3636	m . c	-rw-rw-rw-	root	staff	/etc/utmp
	28056	m . c	-rw-r- -r- -	root	staff	/var/adm/lastlog
	1250496	m . c	-rw-r- -r- -	root	staff	/var/adm/wtmp
19:47:09	1041	. a .	-rw-r- -r- -	root	staff	/etc/passwd
19:47:10	147456	. a .	-rwxr-xr-x	root	staff	/bin/csh

A.4 - Reconstrução de arquivo com Lazarus

Os sistemas de arquivos modernos minimizam as datas/horas de acesso de arquivo agrupando as informações relacionadas. Entre outras coisas, isso reduz a fragmentação dos arquivos individuais. O programa Coroner's Toolkit Lazarus tira proveito dessa propriedade ao tentar reconstituir a estrutura do conteúdo do arquivo excluído.

O Lazarus é um programa simples cujo objetivo é fornecer dados não-estruturados de algum tipo que seja visualizável e manipulável pelos usuários. Ele conta com alguns princípios simples e com a heurística:

- Todos os sistemas de arquivos populares dividem seu espaço de armazenamento em blocos de tamanho idêntico. Tamanhos de blocos típicos são 1.024 bytes e 4.096 bytes. Contanto que o Lazarus utilize um tamanho de bloco de entrada consistente com isso, ele nunca perderá uma oportunidade de dividir um arquivo apropriadamente.

- Os sistemas de arquivos evitam fragmentação de arquivo por razões de desempenho. Em particular, os sistemas de arquivos UNIX quase não fragmentam arquivos mesmo depois de anos de uso.
- Arquivos freqüentemente têm uma assinatura distinta no início. O utilitário file do UNIX utiliza um banco de dados com padrões para reconhecer os arquivos pela assinatura de seu conteúdo. O Lazarus utiliza esse banco de dados, além de um adaptador predefinido padrão, para reconhecer cabeçalhos de arquivos e classificar o conteúdo de outro arquivo.
- Se um bloco de disco parecer semelhante ao bloco do disco anterior, então o Lazarus irá pressupor que ambos os blocos são parte do mesmo arquivo.

Com esses princípios em mente o Lazarus implementa um tipo de dispositivo de raios-X digital primitivo. Ela cria um mapa do disco que essencialmente torna a unidade transparente: você pode examinar o disco e ver os dados pelo tipo de conteúdo, mas a abstração do sistema de arquivos altamente útil é perdida. A Figura A1 mostra um exemplo da interface e um arquivo da imagem excluída.

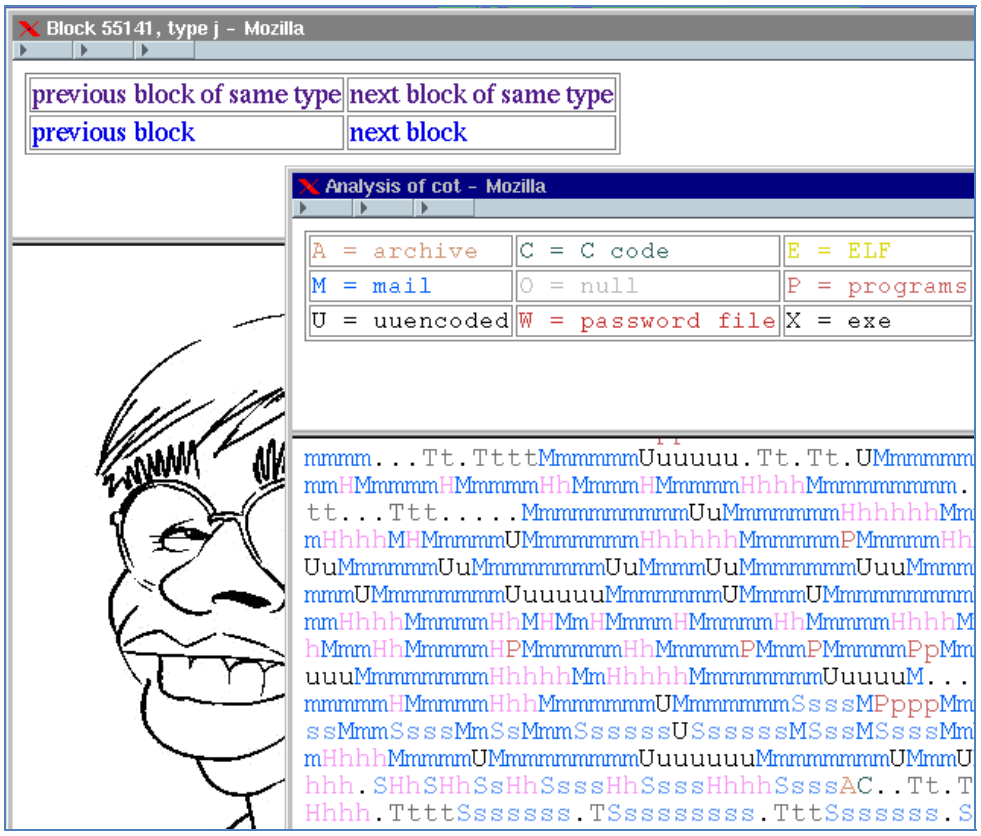

Figura A1 - O lazarus exibindo uma imagem excluída (FARMER;VENEMA, 2007)

No mapa de um disco, o Lazarus utiliza caracteres simples de texto para representar fragmentos de dados. Uma letra maiúscula é utilizada para o primeiro bloco de um fragmento e letras minúsculas são utilizadas para o restante. Por exemplo, C representa o código-fonte C, H significa hipertexto, L é um arquivo de log, M é correio, *U* é conteúdo não-codificado e um ponto (.) representa dados binários não-reconhecidos.

Para manter o mapa gerenciável, o Lazarus compacta grandes fragmentos utilizando uma escala logarítmica (base 2). Isso significa que um caractere individual é um bloco de dados, o segundo caractere são dois blocos, o terceiro são quatro blocos e assim por diante. Isso permite que grandes arquivos sejam visualmente significativos, mas sem excessos: com um tamanho de bloco de 1.024 bytes, um arquivo de um megabyte só ocuparia dez vezes o espaço de um arquivo de um único bloco.

O Lazarus demonstra que sistemas de arquivos UNIX gostam de manter as informações relacionadas dentro da mesma zona do sistema de arquivos. Por exemplo, a Figura A1 mostra que arquivos de e-mails (indicados com "Mmmm") tendem a ser clusterizados juntos. Essa figura também mostra que é provável que um e-mail com uma grande quantidade de hipertexto ou conteúdo não-codificado não seja identificado. A clusterização dos arquivos e das atividades de arquivos tem conseqüências importantes para a persistência das informações excluídas.

Um software como o Lazarus apresenta um problema de escopo não trivial. Embora o Lazarus tome cuidado em neutralizar o conteúdo ativo em hipertexto e outros formatos renderizando-o como texto simples, ele não faz nenhuma verificação de sanidade nos outros dados, como imagens. Portanto, ele pode ser levado a cometer erros por bugs em um programa de navegador Web muito grande e complexo.

O Lazarus não evoluiu desde sua distribuição inicial. Aqueles que queiram pesquisar discos devem considerar o uso da ferramenta Autopsy, de Brian Carrier (Carrier, 2004b).

A.5 - Utilitários de sistema de arquivos de baixo nível

O Coroner's Toolkit vem com alguns utilitários que driblam a camada do sistema de arquivos. Isso permite que esse software acesse informações de arquivos existentes e de arquivos excluídos. Em vez de nomes de arquivo, esses programas utilizam abstração dos números de inode de baixo nível e blocos de alocação de bitmap ou até mesmo abstração de nível mais baixo dos números do bloco de disco.

O Coroner's Toolkit suporta sistemas de arquivos UNIX populares como o UFS (BSD e Solaris), o Ext2fs e o Ext3fs (Linux). O Sleuth Kit também adiciona suporte a sistemas de arquivos não-UNIX como NTFS, FAT16 e FAT32 (Microsoft Windows).

Os utilitários que são parte da distribuição original do Coroner's Toolkit incluem os seguintes:

- ils - Acessa atributos de arquivo pelos seus números de inode. Por padrão, isso lista todos os atributos de arquivo não-alocado.
- Icat - Acessa o conteúdo de arquivo pelo seu número de inode. Isso é utilizado principalmente para pesquisar o conteúdo de arquivo excluído.
- Unrm - Acessa blocos de disco pelos seus números de bloco de disco. Por padrão, isso lê todo o conteúdo de arquivo não-alocado e gera uma saída que pode ser utilizada por programas como o Lazarus. Na distribuição do Sleuth Kit, unrm é renomeado como dls.

O Sleuth Kit adiciona alguns outros utilitários de baixo nível, como:

- ffind - Mapeia um número de inode para a entrada de diretório que referencia o inode.
- fls - Lista as entradas de diretório de lista, incluindo as excluídas. A Seção 4.14 mostra como utilizar esse utilitário.
- ifind - Mapeia um número de bloco de dados para o inode que referencia o bloco de dados.

A taxa de sucesso das ferramentas de baixo nível de sistema de arquivos com informações de arquivo excluído depende muito do tipo de sistema de arquivos e mesmo da versão do *software* do sistema operacional. No capítulo 1 foi

demonstrado como o volume de informações é perdido e o que é preservado quando um arquivo é excluído.

A.6 - Utilitários de memória de baixo nível

As ferramentas descritas aqui são projetadas mais para uso exploratório do que para uma análise sólida. Como sua saída gerada contém pouca ou nenhuma metainformação estrutural, ela só é adequada para processamento com ferramentas que não tirem proveito dessas informações.

pcat - Faz o dump de memória de um processo em execução.

memdump - Faz o dump da memória do sistema e interfere o menos possível. A saída deve ser enviada pela rede para evitar interação com o conteúdo do cache do sistema de arquivos.