Universidade de Brasília
Faculdade de Tecnologia
Departamento de Engenharia Elétrica

Uso de Raciocínio Baseado em Casos na Simulação de Análise *Live*

Frederico Imbroisi Mesquita

Orientador: Célia Ghedini Ralha
Co-Orientador: Bruno Werneck Pinto Hoelz

Dissertação de Mestrado em Engenharia Elétrica
Área de Concentração Informática Forense e Segurança da Informação

Publicação: PPGENE.DM - 79 A/11

Brasília / DF: Agosto/2011

Universidade de Brasília
Faculdade de Tecnologia
Departamento de Engenharia Elétrica

Uso de Raciocínio Baseado em Casos na Simulação de Análise *Live*

Frederico Imbroisi Mesquita

Dissertação de mestrado submetida ao Departamento de Engenharia Elétrica da Faculdade de Tecnologia da Universidade de Brasília, como parte dos requisitos necessários para a obtenção do grau de mestre profissional em Informática Forense e Segurança da Informação.

Aprovado por:

Célia Ghedini Ralha, PhD, UnB
(Orientadora)

Alexandre Ricardo Soares Romariz, PhD, UnB
(Examinador Interno)

Paulo Quintiliano da Silva, Doutor, DPF
(Examinador Externo)

Flávio Elias Gomes de Deus, Doutor, UnB
(Suplente)

Data: Brasília/DF, 19 de Agosto de 2011.

Ficha Catalográfica

MESQUITA, FREDERICO IMBROISI
Uso de Raciocínio Baseado em Casos na Simulação de Análise *Live*. [Distrito Federal] 2011.
ixx, 86p., 297 mm (ENE/FT/UnB, Mestre, Engenharia Elétrica, 2011).

Dissertação de Mestrado – Universidade de Brasília, Faculdade de Tecnologia. Departamento de Engenharia Elétrica.

1. análise *live* 2. raciocínio baseado em casos 3. perícia digital

I. ENE/FT/UnB. II. Título (Série)

Referência Bibliográfica

MESQUITA, FREDERICO IMBROISI (2011). Uso de Raciocínio Baseado em Casos na Simulação de Análise *Live*. Dissertação de Mestrado, Publicação PPGENE.DM - 79 A/11, Departamento de Engenharia Elétrica, Universidade de Brasília, Brasília, DF, 86p.

Cessão de Direitos

Nome do autor: Frederico Imbroisi Mesquita
Título da dissertação: Uso de Raciocínio Baseado em Casos na Simulação de Análise *Live*.
Grau/Ano: Mestre/2011.

Frederico Imbroisi Mesquita
SHIG/S 706 bloco L casa 54
CEP 70.350-762 – Brasília – DF – Brasil

Dedico este trabalho à minha família

pelo amor incondicional recíproco.

Agradecimentos

A minha orientadora Prof. Dr. Célia Ghedini Ralha e co-orientador Bruno Werneck Hoelz pelo constante apoio, sugestões e revisões essenciais para o desenvolvimento deste trabalho.

O presente trabalho foi realizado com o apoio do Departamento Polícia Federal – DPF, com recursos do Programa Nacional de Segurança Pública com Cidadania – PRONASCI, do Ministério da Justiça.

Resumo

Uso de Raciocínio Baseado em Casos na Simulação de Análise *Live*

Autor: Frederico Imbroisi Mesquita

Orientador: Célia Ghedini Ralha

Programa de Pós-graduação em Engenharia Elétrica

Brasília, agosto de 2011

A presença de conteúdo criptografado e o grande volume de informações a serem periciados são os novos desafios para a perícia digital convencional. A análise *live* pode ser utilizada para garantir o acesso ao conteúdo do disco rígido e realizar a triagem de dados e equipamentos ainda em execução. Porém, trata-se de uma perícia digital complexa devido a grande quantidade e variedade de informações a serem analisadas em curto período de tempo. Este trabalho apresenta uma arquitetura com foco na abordagem de Raciocínio Baseado em Casos, a qual utiliza conhecimentos adquiridos em casos similares anteriores para solução de casos atuais. A partir da definição arquitetural foi implementado e testado um protótipo com a utilização de casos reais do Departamento de Polícia Federal, simulados em máquinas virtuais. Durante os experimentos realizados foi identificada repetição de 84% nos processos em execução e de 95% nas DLLs carregadas dos casos cadastrados. Além disso, o protótipo apresentou bons resultados no reconhecimento de ameaças e oportunidades durante a análise de equipamentos computacionais ligados, sugerindo procedimentos periciais adicionais em cerca de 76% dos casos simulados. Uma outra contribuição importante deste trabalho está relacionada ao uso contínuo da solução proposta, a qual possibilitará uma padronização dos procedimentos periciais realizados em casos de sucesso, bem como a difusão do conhecimento relacionado aos casos periciais de sucesso aos peritos responsáveis pela realização da análise *live*.

Abstract

Using Case Based Reasoning in Live Analysis Simulation

Author: Frederico Imbroisi Mesquita

Supervisor: Célia Ghedini Ralha

Programa de Pós-graduação em Engenharia Elétrica

Brasília, August de 2011

Conventional digital analysis is being challenged by the presence of cyphered contents and the large volume of data to be processed. Live analysis can be applied to secure access to hard disk information and to perform data and equipment triage. However, this kind of analysis is considered complex due to the large number and variety of information to be processed in a short period of time. This work presents a framework built upon case-based reasoning approach by reusing knowledge from previous cases to solve new cases. A prototype was implemented and tested against Brazilian Federal Police real cases, using virtual machine simulation. The studied cases showed repetition rates of around 84% of running process and approximately 95% of loaded DLLs. Furthermore, the prototype was successful on recognizing threats and opportunities during the turned-on machine analysis, suggesting additional forensics procedures in 76% of the simulated cases. We consider that the continuous use of the proposed solution would allow the standardization of the successful pericial cases' adopted procedures and also the diffusion of acquired knowledge among law enforcement agents in charge of live analyses processes.

Sumário

Listas de Quadros e Tabelas xviii

Lista de Figuras xix

Lista de Símbolos, Nomenclatura e Abreviações xx

1. Introdução 1

1.1 Definição do Problema 2

1.2 Objetivos do Trabalho 3

1.3 Metodologia Adotada 3

2. Análise *Live* 5

2.1 Diferenças entre Perícia Convencional e Análise *Live* 5

2.2 Vantagens da Análise *Live* 7

2.2.1 Extração de dados voláteis 7

2.2.2 Triagem de equipamentos 7

2.2.3 Triagem de dados 8

2.2.4 Preservação de dados criptografados 9

2.2.5 Possibilidade de estabelecer flagrante 9

2.3 Desvantagens da Análise *Live* 9

2.3.1 Aspectos legais e impossibilidade de reprodução do exame 10

2.3.2 Tempo gasto 11

2.3.3 Complexidade e variedade de cenários 11

2.3.4 Mudança de paradigma na investigação 11

2.3.5 Rootkits 12

2.4 Análise da Memória RAM 12

2.4.1 Características 13

2.4.2 Análise por *live response* 13
2.4.3 Análise do *dump* da memória 15
2.4.4 Mecanismos para cópia da memória RAM 16
3. Raciocínio Baseado em Casos 18
3.1 Classes de RBC 19
3.2 Princípios do RBC 19
3.3 Características do modelo RBC 19
3.4 Ciclo RBC 20
3.5 Similaridade e distância entre os casos 22
3.5.1 Simple Matching 22
3.5.2 Coeficiente de Jaccard 23
3.5.3 Distância de Hamming 25
3.5.4 Distância Euclidiana 26
4. Solução Proposta 27
4.1 Sistema de Raciocínio Baseado em Casos (SRBC) 27
4.2 Modelo RBC aplicado ao SRBC 28
4.3 Requisitos fundamentais do Sistema 30
4.4 Cálculo de similaridade 31
4.4.1 Representação do caso 32
4.4.2 Fórmula de Similaridade do SRBC 33
4.5 Interação do Perito com o SRBC 36
5. Protótipo Implementado 37
5.1 Aplicativos 39
5.1.1 Cópia integral da memória RAM 39
5.1.2 Análise da memoria RAM 41
5.2 Servidor de aplicação 44
5.3 Bancos de dados 47

5.3.1 Repositório de casos 47
5.3.2 Repositório de Procedimentos 49
6. Experimentação e Análise dos Resultados 51
6.1 Ambiente de testes 51
6.2 Inserção de aplicativos conhecidos 52
6.3 Inserção de casos reais 54
6.4 Verificação da viabilidade técnica da proposta 54
6.5 Identificação dos aplicativos conhecidos 63
6.6 Coeficiente de similaridade entre os casos 65
7. Conclusões e Trabalhos Futuros 71
A. Lista completa de parâmetros do Volatility 1.3a 74
B. Matrizes de Similaridades 76
Referências Bibliográficas 82

Listas de Quadros e Tabelas

Tabela 2-1: Posição das estruturas de dados Windows (adaptado de Vidas, 2006)..... 15

Tabela 4-1: Representação da presença de aplicativos em casos específicos. 32

Tabela 4-2: Matrizes de distâncias com diferentes fórmulas de cálculo 34

Tabela 6-1: Exemplos de aplicativos conhecidos inseridos na Base de Casos 53

Tabela 6-2: Propriedade dos casos 55

Tabela 6-3: Crescimento da Base de Casos 58

Tabela 6-4: Taxa de crescimento da Base de Casos por intervalo de casos..... 61

Tabela 6-5: Sequência binária representativa dos 15 primeiros casos para 16 aplicativos..... 66

Tabela 6-6: Matriz de similaridades entre os 15 primeiros casos para 16 aplicativos. 67

Tabela 6-7: Sequência binária representativa dos 15 primeiros casos para 29 aplicativos..... 67

Tabela 6-8: Matriz de similaridades entre os 15 primeiros casos para 29 aplicativos. 68

Lista de Figuras

Figura 3-1: Ciclo RBC –(adaptado de Aamodt & Plaza, 1994). 21

Figura 4-1: Arquitetura SRBC 27

Figura 4-2: Modelo RBC aplicado ao SRBC 29

Figura 5-1: Funcionamento do protótipo 38

Figura 5-2: Interface gráfica do SRBC mostrando os casos cadastrados no Sistema. 44

Figura 5-3: Interface gráfica do SRBC mostrando dados de um caso cadastrado. 45

Figura 5-4: Diagrama de entidade-relacionamento do SRBC implementado. 48

Figura 5-5: Página na Wiki contendo os procedimentos periciais referentes ao Emule. 49

Figura 6-1: Gráfico representativo da quantidade de processos por caso 56

Figura 6-2: Gráfico representativo da quantidade de DLLs por casos 57

Figura 6-3: Curva de crescimento de processos na Base de Casos 59

Figura 6-4: Curva de crescimento de DLLs na Base de Casos 59

Figura 6-5: Gráfico de novas inserções de processos na Base de Casos 60

Figura 6-6: Gráfico de novas inserções de DLLs na Base de Casos 60

Figura 6-7: Gráfico de inclusões de processos no Sistema por intervalo de casos 62

Figura 6-8: Gráfico de inclusões de DLLs no Sistema por intervalo de casos 62

Figura 6-9: Exemplo de relatório contendo as similaridades entre os casos. 70

Lista de Símbolos, Nomenclatura e Abreviações

API – *Application Programming Interface*

BD – Banco de Dados

DLL – *Dynamic-link Library*

DMA – *Direct Memory Access*

DPF – Departamento de Polícia Federal

DTB – *Directory Table Base*

HTML – *HyperText Markup Language*

IA – Inteligência Artificial

IEEE - *Institute of Electrical and Electronics Engineers*

IM – *Instant Messaging*

KFF – *Known File Filter*

MD5 – *Message-Digest 5*

P2P – *Peer-to-Peer*

PCI – *Peripheral Component Interconnect*

PHP – *Hypertext Preprocessor*

PID – *Process ID*

PPID – *Parent Process ID*

RAID – *Redundant Array of Independent Disks*

RAM – *Random-Access Memory*

RBC – Raciocínio Baseado em Casos

SEPINF – Serviço de Perícias em Informática

SO – Sistema Operacional

SP – *Service Pack*

SRBC – Sistema de Raciocínio Baseado em Casos

USB – *Universal Serial Bus*

1. Introdução

A popularização do uso de aplicativos de criptografia forte e a crescente demanda por análises periciais em equipamentos computacionais tornaram-se um grande desafio para o campo da perícia digital. A perícia convencional, realizada em um laboratório de informática forense, encontra cada vez mais dificuldade em atingir seus objetivos. O grande volume de informações e conteúdos criptografados inviabilizam a elaboração tempestiva de laudos à Justiça, criando morosidade judicial e frustração nos profissionais envolvidos. Para contribuir na minimização desses problemas, a perícia digital conhecida como *live forensics* ou análise *live* pode ser utilizada. A técnica de análise *live* consiste na execução de procedimento periciais na máquina investigada ainda ligada, fornecendo um recurso antes inacessível à perícia convencional: o conteúdo da memória RAM. Segundo Anson & Bunting (2007), a análise *live* permite a extração de informações relevantes na memória RAM (*Random-Access Memory*) que seriam perdidas quando o sistema fosse desligado.

Ao contrário da perícia convencional estática que possui apenas acesso ao sistema de arquivo da máquina investigada, a perícia com computador ligado permite o acesso a diversas informações contidas na memória volátil, tais como processos em execução, portas de comunicação abertas, aplicativos instalados, *strings* em memória, arquivos abertos, conexões estabelecidas, entre outras. O uso da perícia digital convencional pode ser totalmente inviabilizado se, por exemplo, forem utilizados no computador investigado discos rígidos protegidos com esquemas criptográficos do tipo *full disk encryption*. Tais situações exigem a extração dos dados ainda com a máquina ligada. Segundo Waits et al. (2008), não é mais possível ignorar o conteúdo da memória RAM durante a análise pericial; uma vez que esse tipo de análise também pode ser usada para facilitar ou abreviar o tempo gasto pela perícia convencional, através da triagem de dados ou equipamentos encaminhados à perícia.

As informações extraídas durante a análise de um computador ligado oferecem muitas vantagens em comparação com a perícia convencional, porém introduz novos problemas. O cenário de uma análise *live* é de alta complexidade devido à grande quantidade de variáveis que podem ser encontradas, tais como a diversidade de sistemas operacionais, aplicativos instalados, dispositivos de E/S, *hardwares,* entre outros. A boa

prática de perícia forense digital exige que qualquer exame de um dispositivo computacional seja feito de forma a minimizar a possibilidade de contaminação digital (Marshall, 2008).

Apesar de avanços significativos alcançados em diversos trabalhos ao abordarem modelos para o processo de investigação digital (Carrier & Spafford, 2003) e (Reith, Carr, & Gunsch, 2002), a análise *live* permanece sendo um campo controverso da perícia digital. Um dos debates clássicos no meio de computação forense é a abordagem correta na manipulação de sistemas computacionais em execução (Solomon, Barrett, & Broom, 2005). Alguns trabalhos limitam-se a citar as boas práticas forenses durante a *live forensics* (Kruse II & Heiser, 2002) ou somente apresentam ferramentas específicas para análise da memória RAM (Walter e Petroni, 2007; Petroni et al,. 2006; Waits et al., 2008).

Técnicas de Inteligência Artificial (IA) também podem ser utilizadas para auxiliar o perito durante a realização de uma análise de um equipamento computacional em execução. Em um curto período de tempo, um sistema seria capaz de analisar de forma automatizada as diversas variáveis encontradas durante a análise *live* e, a partir do conhecimento já estabelecido, sugerir ao perito procedimentos periciais bem-sucedidos em situações passadas.

1.1 Definição do Problema

A grande quantidade de variáveis a serem analisadas em um curto espaço de tempo dificulta a preservação adequada dos vestígios digitais em uma análise *live*. Esse tipo de análise exige um conhecimento amplo em Ciência da Computação, nas suas diversas subáreas, tais como sistemas operacionais, sistemas de arquivos, redes de computadores, arquitetura de computadores, segurança de dados, entre outras. A expectativa de que um perito domine todas essas áreas e esteja preparado para atuar em todas as combinações possíveis não é um cenário plausível. Envolver mais profissionais para trabalharem em conjunto na realização de uma análise *live* também não é viável, devido a atual carência de peritos para atuarem durante a operação de busca e apreensão.

Como consequência desse problema, existe o risco de perder informações importantes ou de inviabilizar a perícia posterior, devido principalmente à presença de criptografia nos computadores apreendidos. Sem a realização bem-sucedida da análise

live também não é possível estabelecer um situação de flagrante ou mesmo realizar uma triagem de dados e equipamentos que sejam realmente importantes à investigação. Sendo assim, é possível identificar ameaças ou oportunidades durante a análise *live* e com isso facilitar o trabalho do perito durante a perícia digital convencional, seja pela garantia de acesso aos dados - conteúdo sem criptografia; seja pela redução da quantidade de dados a serem encaminhados à pericia - triagem de dados e equipamentos computacionais.

1.2 Objetivos do Trabalho

O objetivo principal desse trabalho é verificar a hipótese que o uso de Raciocínio Baseado em Casos (RBC) pode orientar peritos a procederem de maneira mais adequada durante a execução de uma análise *live,* de acordo com conhecimento já adquirido em casos anteriores. O protótipo desenvolvido utiliza as informações encontradas em um equipamento computacional ligado para auxiliar o perito a identificar ameaças e oportunidades, bem como sugerir procedimentos periciais já realizados com sucesso em casos anteriores, o que, consequentemente, reduz problemas encontrados durante a perícia digital convencional. Como objetivos secundários, a abordagem adotada resolve as dificuldades relacionadas com a apreensão de conteúdo criptografado e o grande volume de dados encaminhados à análise pericial. Vale ressaltar que as técnicas e procedimentos periciais utilizados durante a perícia convencional, as quais buscam resolver esses problemas através do uso de computação distribuída, KFF (*Known File Filter*), *Rainbow Tables* e dicionários biográficos estão fora do escopo desse trabalho.

1.3 Metodologia Adotada

Para realização deste trabalho foi feita a revisão da literatura relacionada a procedimentos de análise *live* e de técnicas de IA com ênfase no RBC. Adicionalmente, foi proposto e desenvolvido um protótipo de sistema que utiliza RBC para auxiliar peritos durante a perícia de um computador ligado.

Dando continuidade ao trabalho, observou-se, por amostragem, quais informações poderiam ser coletadas durante uma análise *live* e a similaridade destas informações entre os casos processados pelo Departamento de Polícia Federal (DPF). Os dados colhidos de um equipamento computacional ligado podem ser utilizados para associar automaticamente um caso a uma modalidade de crime, sendo que tais

experimentos e resultados foram apresentados em 2011 na *Seventh Annual IFIP WG 11.9 International Conference on Digital Forensics* (Hoelz et al., 2011a) e será publicado no volume VII do livro *Advances in Digital Forensics*, da editora Springer (Hoelz et al., 2011b).

Novos experimentos foram realizados para verificar a viabilidade do uso da mesma metodologia RBC para identificar ameaças ou oportunidades na investigação de um computador ligado e sugerir ao perito procedimentos periciais adotados com sucesso em casos passados. Um protótipo foi desenvolvido e cinquenta casos reais do DPF foram adaptados para simular análises *live* em busca de similaridades, já que o modelo RBC utiliza a experiência de casos anteriores para resolver novos problemas. Em seguida, foram realizados os experimentos para calcular o coeficiente de similaridade entre os casos e identificar ameaças e oportunidades durante a simulação de uma análise *live*.

O restante desta dissertação está organizada da seguinte forma: nos Capítulos 2 e 3 é apresentada a fundamentação teórica do trabalho, abordando tópicos relativos à análise *live* e RBC; No Capítulo 4 é apresentada a solução proposta, com a arquitetura definida e os requisitos necessários para o desenvolvimento do Sistema; No Capítulo 5 é apresentado o protótipo baseado na solução arquitetural proposta; no Capítulo 6 são apresentados os experimentos e a análise dos resultados; e finamente, no Capítulo 7 são apresentadas as considerações finais e sugestões de trabalhos futuros.

2. Análise *Live*

A análise *live,* também denominada de *live forensics*, consiste em uma análise digital realizada através de procedimentos periciais e conduzida no equipamento computacional ainda em execução. Segundo Carrier (2006), a análise *live* ocorre quando o sistema é mantido em execução e os investigadores usam o próprio sistema operacional da máquina para acessar os seus dados. O procedimento pode ser relativamente simples e rápido, tal como listar as portas de conexões abertas. Porém, pode também ser complexo e demorado, como por exemplo, realizar buscas por palavras-chave no disco rígido utilizando expressões regulares e copiar integralmente o conteúdo deste disco através da porta USB (*Universal Serial Bus*). Em se tratando de operações policiais, esse tipo de exame é normalmente realizado no local da busca e apreensão.

Segundo Anson & Bunting (2007), os ingredientes principais para realizar uma análise *live* bem-sucedida são:

a) Interagir o mínimo possível com o sistema em análise.
b) Utilizar ferramentas confiáveis.
c) Pensar e repensar; pois uma vez feito o procedimento em um sistema em execução, o sistema modificará o estado atual, sendo impossível retornar ao estado inicial.
d) Documentar todo o procedimento.

2.1 Diferenças entre Perícia Convencional e Análise *Live*

A perícia convencional, também chamada de estática, ocorre com o sistema investigado desligado. Para evitar novas escritas no disco, remoção de arquivos temporários ou qualquer modificação no sistema, aconselha-se desligar o computador utilizando o procedimento *pull the plug* na busca e apreensão. Esse procedimento consiste na interrupção do fornecimento de energia ao equipamento pela retirada do cabo de energia da tomada. Após a busca e apreensão do equipamento computacional, uma imagem (cópia forense digital) do disco rígido do sistema é realizada e, a partir daí, essa imagem é analisada em um laboratório de informática, utilizando-se de um sistema operacional e aplicações forenses confiáveis (Carrier, 2006).

Diferentemente da perícia estática convencional que fornece apenas uma visão limitada das informações do sistema, ferramentas para análise *live* podem informar ao investigador um cenário mais completo do estado do computador (Hay et al., 2009). Segundo Adelstein (2006), enquanto a perícia convencional tenta preservar os discos rígidos em um estado inalterado, as técnicas de análises em equipamentos computacionais ligados têm como objetivo tirar *snapshots* do estado da máquina, similares às fotografias de uma cena de crime.

Ferramentas periciais, na maioria das vezes, são bem-sucedidas na extração de dados dessas mídias, inclusive na recuperação de arquivos apagados não sobrescritos e buscas por palavras-chave. A perícia convencional, apesar de amplamente usada atualmente na persecução penal, apresenta limitações nos seguintes casos:

a) Impossibilidade de apreender o equipamento computacional;
b) Necessidade de estabelecer o flagrante do suspeito; e
c) Uso de criptografia forte.

No item a), existem casos em que não há permissão legal para apreender o equipamento computacional devido à importância do mesmo para a organização. Equipamentos de grande porte, como *mainframes,* também inviabilizam sua apreensão pela dificuldade de transporte e armazenamento do *hardware*.

No item b), o desligamento sumário do equipamento computacional inviabilizará o flagrante, já que não haverá a constatação dos requisitos necessários para a sua configuração. Sendo assim, a prova deve ser extraída e documentada antes do procedimento de desligamento e apreensão do equipamento.

O item c) refere-se ao mais recente desafio da análise digital forense: o uso, cada vez mais difundido, de esquemas criptográficos robustos nos computadores pessoais, dificulta consideravelmente a extração de dados pela perícia convencional, podendo até mesmo inviabilizá-la por completo.

A análise *live* possui vantagens e desvantagens se comparada com a perícia estática convencional. Segundo Anson & Buting (2007), o investigador deve determinar qual opção representa uma ameaça maior à investigação: a perda dos dados da memória RAM; ou a modificação dos dados no disco rígido.

2.2 Vantagens da Análise *Live*

Devido à análise em um computador ligado fornecer uma visão mais completa do sistema investigado, com acesso a informações na memória RAM, ela pode ser usada para resolver algumas das limitações encontradas pela perícia estática convencional. Entre as vantagens obtidas no uso desse tipo análise, é possível elencar como as principais: (i) extração de dados voláteis; (ii) triagem de equipamentos; (iii) triagem de dados; (iv) preservação de dados criptografados; (v) a possiblidade de estabelecer flagrante. Passaremos a detalhar cada uma delas nas seções que se seguem.

2.2.1 Extração de dados voláteis

De acordo com Adelstein (2006), a análise *live* pode resguardar tanto as informações voláteis, quanto as informações estáticas sobre o sistema de arquivos. De acordo com Carrier & Spafford (2005), o pré-processamento de dados na cena de crime é apenas uma das fases na investigação digital. A perícia em um computador ainda ligado permite ao investigador analisar um elemento indisponível durante a perícia convencional: a memória RAM. Sendo assim, o investigador terá acesso a informações não tipicamente escritas em disco, tais como: portas abertas, conexões de rede ativas, programas em execução, dados temporários, interação com usuário, chaves criptográficas e conteúdos não criptografados (Hay, Nance, & Bishop, 2009).

Com o uso apenas da perícia estática convencional, essas informações eram simplesmente ignoradas, o que pode ser prejudicial à investigação. Quanto maior a probabilidade de alteração nas informações do dispositivo computacional, maior é a prioridade de extração e preservação desde dados (Farmer & Venema, 2006). Como a memória RAM é mais suscetível a mudanças, (Adelstein, 2006) alerta que a extração deve seguir a ordem de volatilidade. Portanto, na maioria das vezes, é necessário que as informações sejam extraídas da memória antes da extração dos dados do disco rígido.

2.2.2 Triagem de equipamentos

A presença crescente de computadores e mídias de armazenamento computacional na vida cotidiana refletiu-se também nas cenas de crime, onde comumente são encontrados computadores que apresentam relação com o fato sob investigação (Hoelz, 2009). Segundo Adelstein (2006), discos rígidos com mais

capacidade de armazenamento aumentam o tempo necessário para análise, dificultando e encarecendo-a quando há a coleta de todos os discos rígidos.

A *live forensic* permite ao investigador filtrar os equipamentos computacionais que são realmente de interesse à investigação. A busca por palavras-chave no disco rígido ou por aplicativos instalados na máquina, por exemplo, pode evitar a apreensão desnecessária de computadores. Em casos onde os resultados da perícia devem ser disponibilizados em um curto espaço de tempo, um modelo para triagem de equipamentos pode ser utilizado durante a análise *live* (Rogers et al., 2006). Esse modelo envolve consultas no computador investigado em busca de informações contidas nos arquivos de registro, histórico de Internet, mensagens eletrônicas entre outros.

A triagem de equipamentos ajuda o perito a dedicar-se a perícia dos equipamentos computacionais relevantes, pois reduz a quantidade total de equipamentos apreendidos. Sendo assim, as análises resultam em um relatório com mais qualidade e tempestividade.

2.2.3 Triagem de dados

Devido ao atual aumento da quantidade de evidências digitais disponíveis, em breve será impossível obter todos os dados referentes ao caso e o paradigma da análise *live* será considerado o procedimento padrão (Adelstein, 2006). Técnicas como *data mining* (Beebe & Clark, 2005) e KFF (*Known File Filters*) (Mead, 2006) estão sendo utilizadas para processar casos contendo grande volume de dados, porém não resolvem o problema por completo.

A extração seletiva de dados no computador investigado em execução pode facilitar a perícia posterior, principalmente em casos onde os dados estão em servidores corporativos (banco de dados, servidores de e-mail), *mainframes* ou em máquinas que contenham *hardwares* que dificultem a perícia convencional, como RAID (*Redundant Array of Independent Disks*), por exemplo. Segundo Aquilina et al., (2008), nem sempre é possível extrair todos os dados de todas as máquinas envolvidas no incidente, sendo mais eficiente a extração de alguns dados de cada máquina para determinar quais sistemas realmente foram afetados.

2.2.4 Preservação de dados criptografados

Segundo Lowman (2010), a criptografia é um dos melhores métodos para ocultar informação e tem sido amplamente utilizada por criminosos para esconder o conteúdo de seus arquivos. O uso de volumes criptografados complica significativamente à perícia estática. Supondo o uso de algoritmos fortes, métodos convencionais de investigação normalmente possuem um baixo retorno em função do investimento (Walters & Petroni, 2007). Uma vez que o sistema é desligado, a chave criptográfica necessária para acessar a mídia de armazenagem normalmente não está mais disponível (Hay et al., 2009). Houve avanços no processamento dados criptografados pela perícia, como o uso de *Rainbow Tables* e ataques por dicionário, porém técnicas anti-forenses também evoluíram para dificultar a decifração destes conteúdos criptografados (Pinkas & Sander, 2002). Caso o sistema não seja desligado, a análise *live* permite ao investigador acessar os dados de forma transparente, como um usuário do sistema, e realizar uma cópia para analisá-los posteriormente.

2.2.5 Possibilidade de estabelecer flagrante

Outra grande vantagem do uso de técnicas de análise em computadores ligados é a possibilidade de constatação de uma situação de flagrante. A perícia convencional estática simplesmente ignorava tal possibilidade e desligava a máquina investigada, perdendo-se a oportunidade de realizar uma prisão em flagrante em casos como o de compartilhamento de material de pedofilia. Ferramentas periciais como o *NuDetective,* que identificam imagens de pedofilia, podem ser utilizadas durante a análise *live*. Esse aplicativo vasculha o conteúdo do disco rígido atingindo taxas de detecção de aproximadamente 95% para imagens de nudez (Polastro & Eleuterio, 2010).

Considerando que as leis penais brasileiras relativas ao crime de pedofilia atribuem uma punição maior ao compartilhamento de material contendo pedofilia em relação a apenas a posse deste material, o infrator seria beneficiado pela ineficiência do investigador.

2.3 Desvantagens da Análise *Live*

Apesar de resolver alguns problemas encontrados na perícia convencional, a perícia em equipamentos computacionais em execução também introduz novos desafios e possui suas próprias limitações. Além de aumentar quantidade de informações que o

perito deve analisar, é possível citar as seguintes desvantagens no uso desse tipo de análise: (i) aspectos legais e impossibilidade de reprodução do exame; (ii) tempo gasto; (iii) complexidade e variedade de cenários; (iv) mudança de paradigma na investigação; (v) *rootkits*. Passaremos a detalhar cada uma delas nas seções seguintes.

2.3.1 Aspectos legais e impossibilidade de reprodução do exame

É necessário considerar as indagações sobre aspectos legais como uma parte do esforço de pesquisa nas análises de sistemas em execução (Hay et al., 2009). Caso a prova digital não seja válida legalmente, pouco adianta os resultados da análise *live*.

Durante a realização da perícia em um computador ligado é muito fácil contaminar a prova no sistema, exigindo que os procedimentos sejam feitos por um profissional qualificado (Adelstein, 2006). Durante a realização dos exames, o sistema se modifica continuamente, impossibilitando obter exatamente os mesmos resultados ao se repetir a perícia. A boa prática exige que o investigador, quando realizando um procedimento em uma máquina investigada, minimize o impacto e compreenda o efeito desse procedimento no sistema analisado (Walters & Petroni, 2007).

Uma vez que a análise *live* extrai dados da memória volátil, esse exame não poderá ser repetido posteriormente produzindo exatamente os mesmos resultados. Em Walter & Petroni (2007), demonstram-se mudanças consideráveis no conteúdo da memória física em relação ao tempo gasto entre as consultas. Essa impossibilidade de reprodução exige uma documentação precisa dos procedimentos realizados e uma atenção ainda maior na preservação da integridade dos dados extraídos durante a análise dos equipamentos computacionais ligados.

Os dados extraídos durante os exames devem seguir o mesmo tratamento dispensado a perícia convencional. Deve-se utilizar uma função *hash* para garantir integridade destes dados. Além disso, a análise *live* deve se manter em harmonia com os preceitos da cadeia de custódia. A cadeia de custódia realiza o rastreamento de uma prova pericial desde sua origem até sua apresentação à justiça, demonstrando que se trata de uma prova autêntica (Schweitzer, 2003). Segundo Kruse II & Heiser (2002), o objetivo de manter cuidadosamente a cadeia de custódia não consiste apenas em proteger a integridade da evidência, mas também de tornar difícil ao advogado de defesa arguir que a evidência foi mal manipulada enquanto esteve na posse do investigador.

2.3.2 Tempo gasto

Conforme descrito em Waits et al. (2008), faz-se necessário o discernimento entre quais atividades de análise devem ser feitas no ambiente em execução e quais podem ser realizadas posteriormente sem prejuízo às investigações.

Alguns procedimentos utilizados quando os computadores estão ligados, tais como a cópia integral do disco rígido, podem ser demorados. O local da análise *live*, no caso policial, não é o local mais adequado para realização de exames periciais. Sendo assim, espera-se que esses procedimentos sejam os mais breves possíveis. Para Adelstein (2006), o investigador pode realizar uma triagem e coletar dados essenciais, examiná-los, e usar o resultado dessa análise para decidir o que é mais necessário.

2.3.3 Complexidade e variedade de cenários

A grande quantidade de aplicativos, sistemas operacionais e dispositivos computacionais encontrados durante a perícia em um equipamento ligado frustram a preparação do perito na realização desse tipo de análise. A preparação é requisito fundamental para o sucesso da análise, sendo inadmissível testar ferramentas e procedimentos durante a realização da análise *live* (Mandia et al., 2003).

O sucesso nesse tipo de análise depende de um treinamento constante do perito na área de Computação, assim como o estudo e as evoluções contínuas dos procedimentos periciais realizados em um computador ligado. Como visto, o investigador deve ter um conhecimento amplo em diversas áreas computacionais, porém é humanamente impossível exigir o domínio em todas as particularidades encontradas durante a análise *live*, sendo necessária a utilização de uma ferramenta para auxiliá-lo nesse processo.

2.3.4 Mudança de paradigma na investigação

Para a realização de uma análise *live,* é indispensável que a máquina esteja ligada. Caso o investigador chegue ao local da busca e apreensão e se depare com a máquina desligada, não existe nada mais a fazer além de embalar o equipamento para realização de perícia convencional, torcendo para que a máquina não esteja utilizando algum algoritmo de criptografia forte. Caso seja necessário realizar a análise de uma máquina ligada, a investigação deve se adaptar à necessidade imposta por esse tipo de

análise, ou seja, realizar a busca e apreensão apenas quando o alvo esteja com o equipamento computacional ligado.

Em investigação sobre crimes relacionados a *hacker* ou terrorismo, o uso da análise *live* pode ser indispensável dependendo do conhecimento de informática do investigado. Sendo assim, a investigação deverá traçar o perfil do alvo na Internet, estabelecendo o horário em que o computador esteja ligado através de monitoramento telemático. Portanto, o procedimento de busca e apreensão ocorrerá apenas no momento em que o computador estiver ligado, para que um perito tenha acesso ao mesmo e realize a análise pericial.

2.3.5 Rootkits

Uma ameaça encontrada durante a realização da perícia em uma máquina ligada é a presença de *rootkits*. Esse tipo de aplicativo é capaz de esconder informações relevantes à investigação através do uso de filtros no fluxo de dados do computador, produzindo potencialmente falsos negativos aos resultados da análise *live*. *Rootkits* podem ser classificados em dois níveis: de aplicativo e de *kernel*. O primeiro tipo é encontrado em aplicativos nativos do próprio sistema operacional, sendo capaz de omitir resultados de uma consulta deste aplicativo modificado através de um filtro pré-determinado. Já o segundo é incorporado ao sistema operacional e, por isso, utiliza um filtro para omitir resultados independentemente do aplicativo que está sendo executado. Segundo Carrrier (2006), os *rootkits* de nível de aplicativo podem ser contornados utilizando executáveis conhecidos e trazidos pelo próprio investigador; mas os *rootkits* de nível de *kernel* necessitam da utilização de uma abordagem mais complexa, sendo necessário buscar inconsistências ao correlacionar diversas estruturas em memória.

2.4 Análise da Memória RAM

A grande vantagem do uso da perícia *live* em relação à perícia convencional é o acesso aos dados residentes na memória do computador investigado. Apesar de ser substancialmente menor que a capacidade de armazenamento dos discos rígidos, a memória RAM é uma rica fonte de informações para investigação. É possível extrair da memória diversas informações, tais como: processos em execução, arquivos abertos, conexões estabelecidas, portas abertas e possíveis senhas. Em Sistemas Windows pode-se extrair a lista de DLLs (*Dynamic Link Library*) e os arquivos de registro (*hive files*)

que permanecem residentes na memória (Carvey, 2011). Essas informações, com exceção dos arquivos de registro que também são acessíveis pelo disco rígido, eram simplesmente ignoradas pela perícia convencional. Com o uso da perícia em equipamentos em execução, essas informações ganham papel de destaque, tornando esse tipo de análise uma importante fase da perícia digital.

2.4.1 Características

A RAM é uma memória volátil e bastante dinâmica que exige cuidado durante sua análise. Devido à volatilidade da memória, a análise em equipamentos computacionais ligados oferece ao perito uma breve janela de oportunidade para extração de dados desta, antes que seja efetuado o desligamento e apreensão do equipamento computacional. A memória RAM de um computador ligado permanece continuamente alterando seu conteúdo. Sendo assim, torna-se necessária uma documentação criteriosa das ações realizadas pelo perito para verificar quais dados foram alterados por esses procedimentos periciais e evitar questionamentos judiciais posteriores.

Dados podem ser extraídos *diretamente* da memória RAM da máquina analisada através de aplicativos do próprio sistema operacional ou programas específicos. A extração também pode ser feita *indiretamente*, através da análise do arquivo de *dump* da memória RAM do sistema investigado.

2.4.2 Análise por *live response*

A análise por *live response* é a forma direta de extração de informações da memória RAM e assemelha-se aos procedimentos utilizados na resposta a incidentes (*Incident Response*). O perito executa aplicativos no próprio equipamento computacional investigado e coleta as informações de interesse ao caso. Esses aplicativos podem ser nativos do próprio sistema operacional ou ferramentas específicas para realizar a extração das informações. Em ambos os casos, são utilizadas APIs (*Application Programming Interface*) e recursos do próprio sistema operacional da máquina em análise. Sendo assim, essas informações são fornecidas pelo sistema operacional para os aplicativos de forma transparente, sem a necessidade de conhecer a organização interna da memória RAM.

A vantagem da *live response* é a simplicidade dos procedimentos e resultados imediatos. Porém, depois do desligamento da máquina, não existe a possibilidade de reproduzir os exames, nem de realizar novos exames caso surjam novas perguntas (Waits et al., 2008). Como visto na Seção 2.3.5, deve-se atentar que a presença de *rootkits* pode gerar falsos negativos nesse tipo de análise. A extração de dados por *live response* pode ser realizada de duas formas distintas:

a) Utilizando o próprio sistema operacional

Ferramentas disponíveis pelo próprio sistema operacional da máquina em investigação podem ser utilizadas durante a análise *live reponse*. Aplicativos como *netstat*, oferecidos por diversos sistemas operacionais, podem listar conexões estabelecidas e portas abertas. Esse método de análise, apesar de cômodo, deve ser evitado durante a perícia forense por se tratar de uma análise em um ambiente "hostil", ou seja, um ambiente não controlado e sem nenhuma garantia que os aplicativos são legítimos do sistema operacional.

O exame pode gerar falsos negativos caso o aplicativo do sistema operacional tenha sido alterado para omitir informações de interesse da investigação (*rootkits* de nível de aplicativo), sendo aconselhável realizar a verificação de integridade nessas ferramentas através de uso de funções *hash*.

b) Utilizando ferramentas específicas

Além de aplicativos próprios do sistema operacional, existem outras ferramentas digitais forenses, aplicativos de administração de rede e utilitários de diagnósticos que podem ser usados durante a *live response*. Não se pode subestimar a importância do processo monótono de criação de um kit de ferramentas para a *live response*. O tempo gasto nesse processo será compensado pelos resultados mais rápidos, profissionais e bem-sucedidos na resposta a incidentes (Mandia et al., 2003).

Ferramentas da *Sysinternals*, por exemplo, disponibilizam gratuitamente ao usuário informações sobre os recursos e atividades do sistema operacional em ambiente *Windows* (Russinovich et al., 2009).

Mesmo utilizando ferramentas específicas, é possível a ocorrência de falsos negativos caso um *rootkit* de nível de *kernel* esteja instalado no sistema operacional. Sendo assim, o sistema operacional pode omitir informações de interesse a investigação independente da ferramenta sendo utilizada na análise *live response*.

2.4.3 Análise do *dump* da memória

Nesse tipo de análise, o conteúdo integral da memória RAM é copiado para um arquivo denominado *dump*. Esse arquivo é então processado em um ambiente preparado e controlado através do uso de *scripts* de *parsers*. Os *parsers* são analisadores sintáticos responsáveis por percorrer o conteúdo do arquivo em busca de padrões. Na análise do arquivo de *dump* da memória é feita a busca por assinaturas das estruturas de dados em memória (Schuster, 2006).

Tabela 2-1: Posição das estruturas de dados Windows (adaptado de Vidas, 2006).

	2000	**XP**	**XP SP2**	**2003**	**Vista**
EP_PageDirBase	18	18	18	18	18
EP_processors	34	34	34	34	34
EP_T_Forward	50	50	50	50	50
EP_T_Back	54	54	54	54	54
EP_priority	62	62	62	62	64
EP_T_Quantum	63	63	6f	63	*
EP_T_Qant_dis	69	69	69	69	60*
EP_exitStatus	6c	24c	1d0	24c	234
EP_createTime	88	70	70	70	88
EP_exitTime	90	78	78	78	90
EP_PID (client Unique)	9c	84	84	84	9c
EP_WorkSetSize	e4	20c	20c	214	208
EP_WorkSetMin	e8	210	210	1f8	1ec
EP_WorkSetMax	Ec	214	214	1fc	1f0
EP_AccessToken	12c	c8	c8	c8	e0
EP_PPID	1c8	14c	14c	128	124
EP_name	1fc	174	174	154	154
EP_size	290	258	260	278	268
TH_size	248	258	258	260	278
TH_createTime	1b0	1c0	1c0	1c8	1d0
TH_exitTime	1b8	1c8	1c8	1d0	1d8
TH_exitStatus	1c0	1d0	1d0	1d8	1e0
TH_PID (client Unique)	1e0	1ec	1ec	1f4	1fc
TH_TID (client Unique)	1e4	1f0	1f0	1f8	200
TH_isTerminated	224	248	248	250	250
TH_startAddr	230	224	224	22c	234

Em Russinovich et al. (2009), pode-se perceber a complexidade dessa tarefa no ambiente Windows. A Tabela 2-1 mostra a diferença entre o tamanho do deslocamento

em hexadecimal das estruturas dos processos para os vários sistemas operacionais Windows.

Por não utilizar as APIs do próprio sistema operacional, essa abordagem de análise apresenta uma desvantagem: incompatibilidade com os diversos sistemas operacionais e versões. As estruturas de dados usadas e a organização dos elementos carregados em memória dependem do sistema operacional do computador. É necessário conhecer profundamente essa organização para realizar a extração destes elementos do arquivo da memória *dump*.

Percebe-se que há alteração na posição das estruturas de dados dependendo da versão e até mesmo do *service pack*. O aplicativo de análise do arquivo de memória RAM (*dump* da memória) deve ser capaz de reconhecer as estruturas de dados e extrair as informações de vários sistemas operacionais.

Apesar de não ser possível extrair novamente o conteúdo da memória após o desligamento da máquina investigada, é possível reanalisar o arquivo *dump*. Essa propriedade é desejável para a filosofia da perícia digital, já que permite uma reprodução parcial do procedimento, além de permitir a extração de outros dados não extraídos durante a primeira análise.

2.4.4 Mecanismos para cópia da memória RAM

Diferentemente da duplicação de discos rígidos pela perícia convencional, a memória RAM não pode ser desconectada do sistema para realização de cópia devido à volatilidade de seus dados. Conforme à arquitetura física, uma vez que a memória não é mais alimentada de energia, o estado dos dados na RAM é desconhecido (Vidas, 2006).

Existem duas formas para realizar a cópia total dos dados da memória RAM para um arquivo de *dump* com suporte: de *software* ou de *hardware*.

a) Cópia integral da memória RAM com suporte de *software*

Consiste na técnica mais utilizada para cópia do conteúdo da memória física. Existem diversos aplicativos (gratuitos ou pagos) que realizam essa cópia, porém como tais aplicativos são executados no próprio sistema operacional investigado, há uma alteração do conteúdo na memória. Sendo assim, o examinador deve dar preferência às ferramentas menos invasivas, ou seja, que alterem o mínimo possível do conteúdo da memória (Vidas, 2006). Um *framework* contendo um bloqueador de escrita para

extração de memória volátil de um computador investigado integrado pode ser utilizado para diminuir ainda mais a interação com o sistema (Chan et al., 2010).

b) Cópia integral da memória RAM com suporte de *hardware*

Devido à volatilidade dos dados, não existe um *hardware* especifico para realizar uma cópia do conteúdo da memória RAM, porém é possível utilizar a propriedade DMA (*Direct Memory Access*) de alguns dispositivos de hardware para realizar essa cópia. O DMA fornece transferência de dados entre o barramento PCI e memória sem que seja necessário usar recursos do processador (Carrier, 2004). A cópia física do conteúdo da memória RAM é possível pelo acesso direto à memória na porta *Firewire* (IEEE 1394) (Boileau, 2006). Apesar de menos invasiva em relação à cópia física por *software*, essa abordagem mostrou-se inconsistente e instável para ser utilizada como padrão pela perícia digital, porém recomendada quando o computador encontra-se em modo bloqueado pelo usuário. Essa técnica permite o acesso à memória do computador investigado bloqueado e, através da alteração de processos em execução, desbloqueá-lo (Woodward & Hannay, 2008).

Nesse capítulo foi apresentada a fundamentação teórica relativa à análise *live,* a qual serviu de base para familiarização com as características inerentes a esse tipo de perícia digital. No capítulo seguinte é apresentada uma breve revisão dos conceitos de uma técnica de IA denominada Raciocínio Baseado em Casos.

3. Raciocínio Baseado em Casos

Segundo Slade (1991), RBC fornece um modelo cognitivo científico que utiliza a experiência na resolução de problemas. RBC é uma abordagem de apoio à decisão semelhante ao modelo usado por humanos na resolução de problemas. RBC utiliza a experiência de casos anteriores para resolver novos problemas. RBC tem sido utilizado na área da IA no intuito de desenvolver sistemas IA mais eficientes. Diferentemente da maioria das técnicas em IA, RBC é baseado em memória, refletindo o uso humano da recordação de problemas e soluções na resolução de um novo problema (Lópes et al, 2005). Humanos são hábeis solucionadores de problemas e seu desempenho melhora com a aquisição contínua de experiência, mesmo considerando as incertezas e limitações de conhecimento. Naturalmente, essa característica humana é altamente desejável para desenvolver sistemas que necessitam de tecnologias de IA.

De acordo com a termologia do RBC, um caso é considerado uma situação de um problema. Um caso passado, armazenado anteriormente ou retido corresponde a uma situação já experimentada que foi devidamente aprendida e pode ser usada para resolver um novo problema. Um novo caso é a descrição de um novo problema a ser resolvido. Depois que uma solução é gerada, o passo final é utilizar essa solução, repará-la se necessário e aprender a experiência (Leake, 1996). Segundo Aamodt & Plaza (1994), o RBC utiliza uma metodologia incremental, baseado em uma aprendizagem sustentável, já que casos recém-aprendidos podem ser usados imediatamente para solucionar problemas atuais.

Um sistema que utilize a metodologia de RBC pode aprender não apenas a partir de casos de sucesso, mas também quando um caso malsucedido é reparado, uma nova solução é então armazenada. Segundo Kolodner (1992), um sistema que relembre as falhas impede que seja sugerido um procedimento incorreto a um caso semelhante. Sendo assim, o sistema continua aprendendo independentemente de a solução proposta não se ter mostrada satisfatória. De acordo com Leake (1996), o aprendizado no RBC é orientado pelo sucesso e pelo fracasso, e ambos aumentam a velocidade para adquirir conhecimento. Tanto casos inéditos, quanto rotineiros também são importantes nessa abordagem de aprendizado. A maior fonte de conhecimento em sistemas RBC é o banco de conhecimento contendo os problemas solucionados e as respectivas soluções (Craw, 2003).

3.1 Classes de RBC

Segundo Marling et al. (2002), os sistemas RBC podem ser divididos em duas classes: interpretativo ou orientado a solução de problemas.

Na classe de RBC interpretativo, novas situações são avaliadas no contexto das situações passadas (Kolodner, 1992). Um novo problema é comparado e contrastado com modelos classificados e armazenados, ou seja, os casos armazenados são usados como justificativa para utilizar uma solução.

Já na classe de RBC orientado a solução de problemas, existe uma adaptação do modelo armazenado para solucionar o novo problema. Sendo assim, essa classe de RBC depende fortemente do uso do processo de adaptação de casos (Kolodner, 1992).

3.2 Princípios do RBC

Segundo Leake (1996), RBC baseia-se em dois princípios: problemas similares possuem soluções similares; e os tipos de problemas encontrados tendem a se repetir.

De acordo com Watson (1999), a implementação de um sistema RBC deve ser guiada pelas seguintes propriedades:

a) o desejo inicial em resolver o problema pela explícita tentativa de reusar uma solução de um problema passado. Existe então a *recuperação* de casos em uma base e um cálculo de similaridade entre os casos antigos e o novo problema.
b) o sistema RBC deve tentar reusar a solução sugerida pelo caso recuperado com ou sem uma *revisão*.
c) o sistema RBC deve procurar aumentar seu conhecimento através da retenção dos novos casos.

3.3 Características do modelo RBC

Sistemas RBC foram empregados com sucesso em diferentes áreas, desde identificação de problemas em automóveis (Lancaster & Kolodner, 1987), resolução de problemas matemáticos (Faries & Schlossberg, 1994), ou para diagnósticos médicos (Schmidt et al., 1990), entre outras aéreas de aplicação.

Kolodner (1992) cita algumas das vantagens no uso de RBC:

a) RBC permite solucionar problemas rapidamente, evitando o tempo necessário para revolvê-los se fosse começar do zero.
b) RBC permite solucionar problemas em que pessoas não tenham o domínio completo do assunto, como aprendizes.
c) RBC sugere possíveis soluções onde não exista um algoritmo disponível para fornecer tais soluções.
d) Relembrar experiências passadas é particularmente útil para avisar quais problemas já aconteceram e alertar quais ações devem ser tomadas para evitar erros passados.
e) Casos podem ajudar a dar foco em partes do problema, ressaltando as que são realmente importantes.

Essa abordagem pode ser usada até mesmo em situações que casos são descritos de forma incompleta (Bogaerts & Leake, 2004). Porém, segundo (Kolodner, 1992), RBC pode levar ao uso indiscriminado de casos antigos, baseando-se em uma experiência passada, porém sem validá-la de acordo com a nova situação.

3.4 Ciclo RBC

Aamodt & Plaza (1994) definem o ciclo RBC tal como descrito na Figura 3-1, como uma série de quatro processos consecutivos, conhecidos como os quatro REs:

1. REcuperar: dado um problema atual, um ou mais casos bem-sucedidos são recuperados do repositório de casos. A recuperação de casos é feita de acordo com a similaridade entre os mesmos, tornando o cálculo de similaridade crucial nessa fase.
2. REusar: é sugerido o uso ou adaptação de um caso recuperado na fase anterior para resolver o problema atual.
3. REvisar: os resultados são avaliados, revisados e ajustados por um especialista.
4. REter: a solução revisada pode ser retida como um novo caso, expandindo, portanto, o repositório de casos.

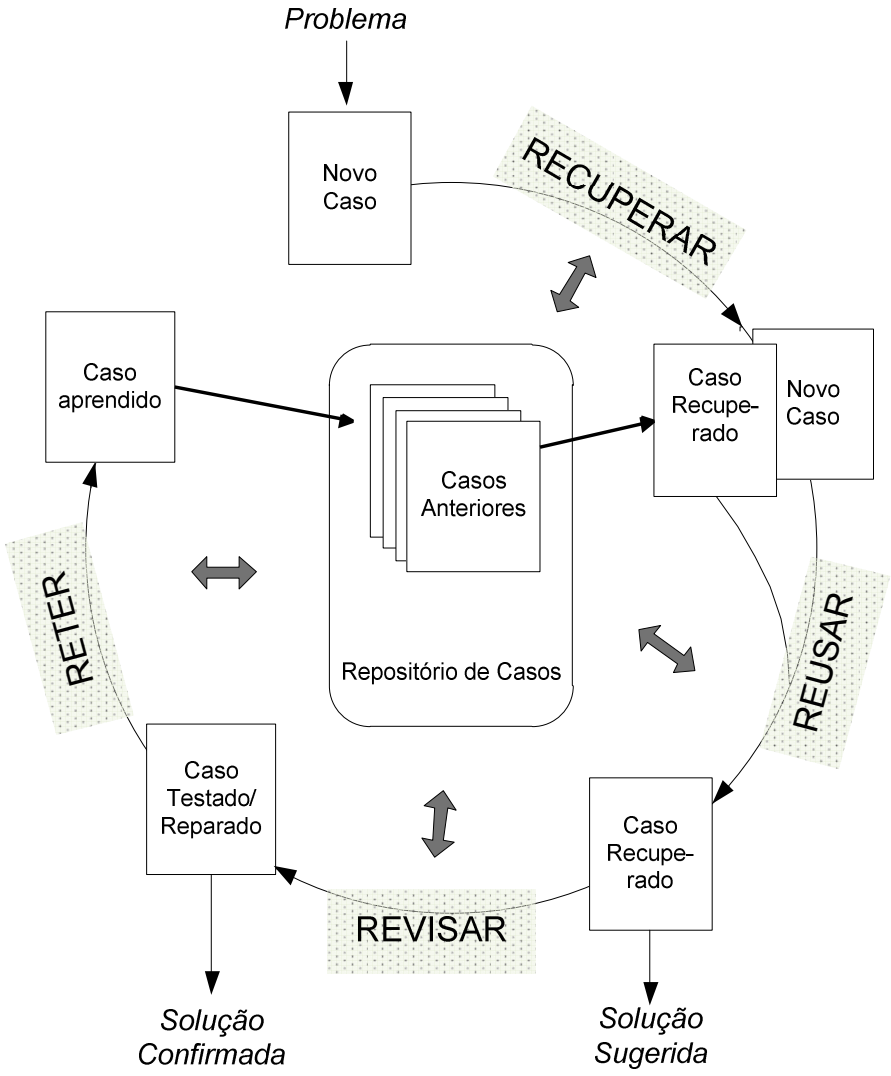

Figura 3-1: Ciclo RBC (adaptado de Aamodt & Plaza, 1994).

Seguindo o ciclo da Figura 3-1, tem-se que: quando um novo caso é inserido no Sistema, um ou mais casos anteriores são *recuperados* da base de casos utilizando algum mecanismo de cálculo de similaridade. A solução usada nesses casos similares é diretamente *reusada*, ou adaptada para ser *reusada* no novo problema. Os resultados são então analisados por um especialista que realiza a *revisão* do caso e a consequente reparação caso necessária. Finalmente, o caso é *retido* na base de casos, aumentando assim o domínio do conhecimento do sistema.

Resumidamente, um novo problema é solucionado através da recuperação de um ou mais casos armazenados, reusando-os de alguma forma, revisando a solução baseada

neste reuso e retendo a nova experiência através do armazenamento na base de conhecimento (Aamodt & Plaza, 1994).

3.5 Similaridade e distância entre os casos

Em sistemas que utilizam conhecimento adquirido previamente faz-se necessário que haja primeiramente a obtenção desse conhecimento, para que o mesmo seja utilizado no problema atual (Russel & Norvig, 2010). Em sistemas RBC, um ou mais casos são recuperados de uma base de casos de acordo com as similaridades com o problema atual. Esse procedimento é realizado na fase *Recuperar* do ciclo RBC (Figura 3-1), sendo fundamental usar um mecanismo de cálculo de similaridades adequado ao tipo de problema. Fórmulas matemáticas são capazes de calcular a distância e/ou a similaridade entre os casos em um sistema RBC. Para realizar esse cálculo faz-se necessário que um caso e seus atributos sejam representados na forma de um conjunto, sequência, vetor ou coordenada de valores. Serão descritos alguns dos mecanismos para o cálculo de similaridade e distância.

3.5.1 Simple Matching

Segundo Rijsbergen (1979), o *Simple Matching* é o mais simples de todos os cálculos de similaridade. Essa similaridade (S_M) pode ser medida pela Equação 1.

$$S_M(A, B) = n(A \cap B) \tag{1}$$

Onde A e B são os objetos em comparação e $n(X)$é o número de elementos em um conjunto X. Para sequências de valores binários a Equação 2 é utilizada para o cálculo do coeficiente de similaridade S_{ij}.

$$S_{ij} = \frac{p+s}{t} \tag{2}$$

Onde,

p = número de variáveis contendo dígito 1 para as sequências em comparação.
s = número de variáveis contendo dígito 0 para as sequências em comparação.

q = número de variáveis contendo dígito 1 para i e 0 para j.

r = número de variáveis contendo dígito 0 para i e 1 para j.

$t = p + q + r + s$, ou seja, o número total de variáveis.

Para calcular a distância utilizando *Simple Matching* têm-se a Equação 3, onde S_{ij} é o coeficiente de similaridade.

$$d_{ij} = 1 - S_{ij} \tag{3}$$

Ou seja,

$$d_{ij} = 1 - \frac{p+s}{t} = \frac{p+q+r+s}{t} - \frac{p+s}{t} = \frac{q+r}{t} \tag{4}$$

Considerando, como exemplo, quatro objetos A, B, C, D, e suas respectivas coordenadas, têm-se os cálculos de similaridade e distância, conforme Equações 2 e 4.

A={0,1,1,1}

B={0,0,1,1}

C={0,0,0,1}

D={1,0,0,0}

$S_{AB} = \frac{2+1}{4} = 0.75$ e $d_{AB} = \frac{1+0}{4} = 0{,}25$

$S_{AC} = \frac{1+1}{4} = 0.50$ e $d_{AC} = \frac{2+0}{4} = 0{,}50$

$S_{AD} = \frac{0+0}{4} = 0$ e $d_{AD} = \frac{3+1}{4} = 1$

$S_{BC} = \frac{1+2}{4} = 0.75$ e $d_{BC} = \frac{1+0}{4} = 0{,}25$

$S_{BD} = \frac{0+1}{4} = 0.25$ e $d_{BD} = \frac{2+1}{4} = 0{,}75$

$S_{CD} = \frac{0+2}{4} = 0.50$ e $d_{CD} = \frac{1+1}{4} = 0{,}50$

3.5.2 Coeficiente de Jaccard

Segundo Real & Vargas (1996), o índice ou coeficiente de Jaccard destaca-se como um dos mais usados índices de similaridade para dados binários, a qual pode ser calculada conforme Equação 5.

$$S_J(A,B) = \frac{n(A \cap B)}{n(A \cup B)} = \frac{n(A \cap B)}{n(A)+n(B)-n(A \cap B)} \tag{5}$$

Para sequências de valores binários, a Equação 5 pode ser reescrita como a Equação 6.

$$S_{ij} = \frac{p}{p+q+r} \tag{6}$$

Onde,

p = número de variáveis contendo dígito 1 para as sequências em comparação.
q = número de variáveis contendo dígito 1 para i e 0 para j.
r = número de variáveis contendo dígito 0 para i e 1 para j.
$t = p + q + r + s$, ou seja, o número total de variáveis.

Para calcular a distância utilizando coeficiente de Jaccard pode-se utilizar a Equação 7, onde S_{ij} é o coeficiente de similaridade.

$$d_{ij} = 1 - S_{ij} \tag{7}$$

ou seja,

$$d_{ij} = 1 - \frac{p}{p+q+r} = \frac{p+q+r-p}{p+q+r} = \frac{q+r}{p+q+r} \tag{8}$$

Considerando, como exemplo, quatro objetos A, B, C, D, e suas respectivas coordenadas, têm-se os cálculos de similaridade e distância, conforme as Equações 6 e 8.

A={0,1,1,1}
B={0,0,1,1}
C={0,0,0,1}
D={1,0,0,0}

$S_{AB} = \frac{2}{2+1+0} = 0.67$ e $d_{AB} = \frac{1+0}{2+1+0} = 0{,}33$

$S_{AC} = \frac{1}{1+2+0} = 0.33$ e $d_{AC} = \frac{2+0}{1+2+0} = 0{,}67$

$S_{AD} = \frac{0}{0+3+1} = 0$ e $d_{AD} = \frac{3+1}{0+3+1} = 1$

$S_{BC} = \frac{1}{1+1+0} = 0.50$ e $d_{BC} = \frac{1+0}{1+1+0} = 0{,}50$

$S_{BD} = \frac{0}{0+2+1} = 0$ e $d_{BD} = \frac{2+1}{0+2+1} = 1$

$S_{CD} = \frac{0}{0+1+1} = 0$ e $d_{CD} = \frac{1+1}{0+1+1} = 1$

3.5.3 Distância de Hamming

De acordo com Lourenço et al. (2004), a distância de Hamming foi originalmente definida para codificação binária, mas pode ser aplicada para qualquer sequência de mesmo tamanho, conforme apresentado na Equação 9.

$$d_H(x,y) = \sum_{i=1}^{n} \delta(x_i, y_i) \quad (9)$$

Onde,

$$\delta(x_i, y_i) = \begin{cases} 0\ se\ (x_i = y_i) \\ 1\ se\ (x_i \neq y_i) \end{cases}$$

Para sequências de valores binários, a distância de Hamming pode ser calculada através da Equação 10.

$$d_{ij} = q + r \quad (10)$$

Onde,

q = número de variáveis contendo dígito 1 para i e 0 para j.
r = número de variáveis contendo dígito 0 para i e 1 para j.

Considerando, como exemplo, quatro objetos A, B, C, D, e suas respectivas coordenadas, têm-se os seguintes cálculos de distância, conforme Equação 10.

A={0,1,1,1}
B={0,0,1,1}
C={0,0,0,1}
D={1,0,0,0}

$$d_{AB} = 1 + 0 = 1$$
$$d_{AC} = 2 + 0 = 2$$
$$d_{AD} = 3 + 1 = 4$$
$$d_{BC} = 1 + 0 = 1$$
$$d_{BD} = 2 + 0 = 1$$
$$d_{CD} = 1 + 1 = 2$$

3.5.4 Distância Euclidiana

A distância Euclidiana consiste na medida mais comum quando se trata de distância. Usualmente, essa distância é utilizada para mensurar a menor distância entre dois pontos e pode ser definida conforme a Equação 11.

$$d_{xy} = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \tag{11}$$

Onde,

n = número de elementos no vetor.

O quadrado da distância Euclidiana para vetores com valores binários produz o mesmo resultado da distância de Hamming (Lourenço, et al., 2004)

Considerando, como exemplo, dois objetos A, B, e suas respectivas coordenadas, têm-se o cálculo de distância Euclidiana, conforme a Equação 11.

A={0,3,5,7}

B={1,-1,4,3}

$$d_{AB} = \sqrt{(0-1)^2 + (3-(-1))^2 + (5-4)^2 + (7-3)^2}$$
$$= \sqrt{1 + 16 + 1 + 16}$$
$$= 5{,}83$$

Nesse capítulo foram apresentadas a fundamentação teórica relativa ao RBC e algumas equações para cálculo de distância e similaridade entre os casos. No capítulo seguinte é apresentada a solução proposta por este trabalho.

4. Solução Proposta

Para auxiliar o perito durante a realização da análise *live*, propomos a criação de sistema capaz de processar as informações colhidas da máquina em execução e automaticamente sugerir procedimentos periciais realizados em casos anteriores de sucesso. Esse capítulo descreve a arquitetura da solução proposta com os requisitos fundamentais, o cálculo de similaridade adotado para recuperar os casos, bem como a interação do perito com o sistema proposto

4.1 Sistema de Raciocínio Baseado em Casos (SRBC)

A análise *live* exige do profissional de informática um conhecimento muito amplo em diversas subáreas da Ciência da Computação, tais como: Redes, Segurança, Sistemas Operacionais e Criptografia. Diferentes arquiteturas e a diversidades de aplicativos torna-se ainda maior o desafio durante esse tipo de análise. Em suma, dificilmente um único perito domina todas essas variáveis que compõe um cenário tão dinâmico e complexo.

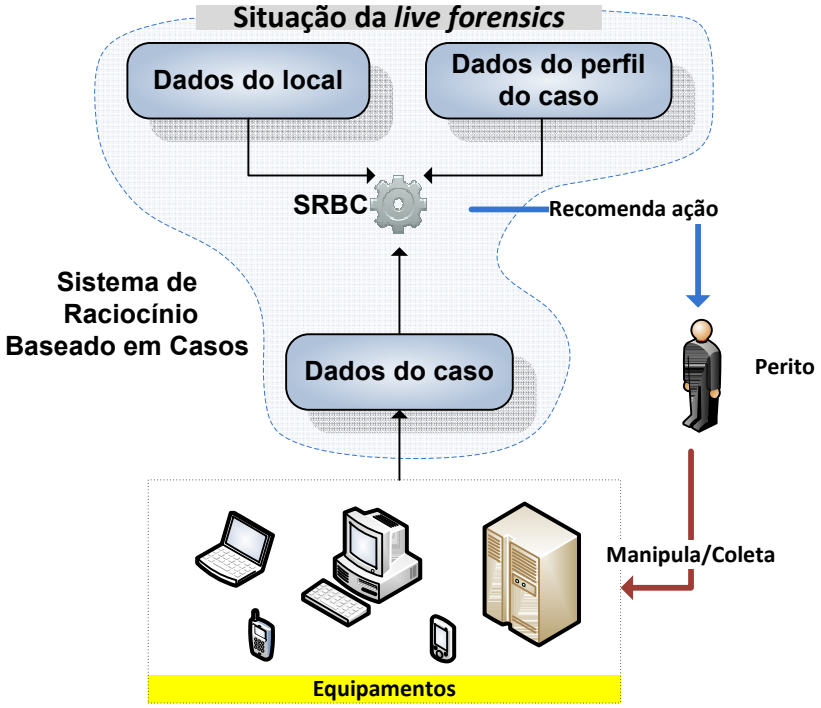

Figura 4-1: Arquitetura SRBC

Conforme apresentado no Capítulo 3, RBC pode ser aproveitado para auxiliar profissionais de informática durante a análise de um computador ligado. A Figura 4-1 apresenta a arquitetura do SRBC proposto, o qual utiliza RBC como modelo para auxiliar o perito na tomada de decisões em uma análise pericial *live*.

Antes de qualquer informação ser obtida de um sistema computacional ligado, a maioria das investigações coletará informações básicas sobre o sistema e com o registro apropriado dos procedimentos realizados (Steel, 2006). Dados do caso são informações de sistema referentes ao equipamento ligado, tais como: processos em execução, DLL carregadas, arquivos abertos, conexões estabelecidas e portas abertas. Essas informações podem ser extraídas diretamente da máquina utilizando aplicativos específicos ou, também, indiretamente através da análise automatizada do *dump* da memória RAM da máquina investigada.

Dados do local são informações sobre a topologia da rede, configuração física da máquina, outros equipamentos computacionais encontrados no local da busca e apreensão e que sejam de interesse à investigação. Já dados do perfil do caso são dados de inteligência sobre a investigação e indicam a modalidade do crime sendo investigada e o nível de conhecimento em informática do alvo.

O perito designado, ao chegar ao local da busca e apreensão, caso o equipamento computacional esteja ligado, realiza coleta automatizada dos dados do sistema operacional. Esses dados, juntamente com os dados já conhecidos da operação (perfil do caso) e de outros possíveis dados colhidos no local, são inseridos no SRBC e o Sistema sugere ao perito que ações devem ser tomadas para o sucesso desse tipo de perícia.

4.2 Modelo RBC aplicado ao SRBC

Os dados inseridos no Sistema (dados do caso, dados do local e dados do perfil do caso) são utilizados no cálculo de similaridade para identificar um ou mais casos semelhantes ao caso em análise. O modelo RBC é a parte do Sistema responsável por encontrar um caso no repositório semelhante ao caso atual. A partir daí, tem-se o uso ou a adaptação deste caso recuperado para recomendar ao perito adotar os procedimentos adequados à situação em questão.

A Figura 4-2 apresenta o Sistema utilizando a metodologia RBC. O repositório de casos é responsável pelo armazenamento de informações referentes aos casos,

incluindo resultados e quais procedimentos periciais foram adotados para solucioná-los. Já o repositório de procedimentos mantém atualizada a descrição dos procedimentos e de ferramentas periciais de forma explicativa e detalhada. As duas bases são centralizadas e colaborativas, necessitando de alimentação constante para permanecerem atualizadas.

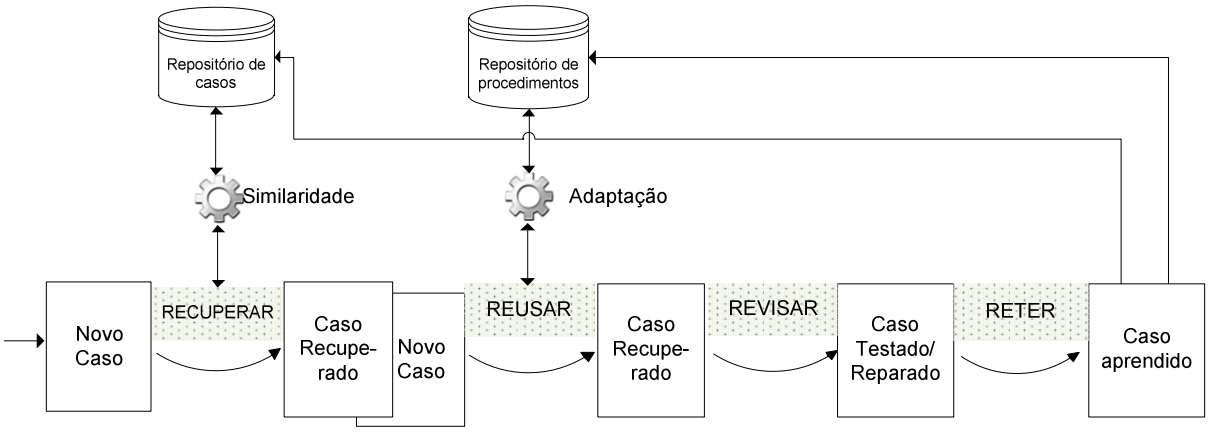

Figura 4-2: Modelo RBC aplicado ao SRBC

O repositório de casos é alimentado automaticamente a cada caso encaminhado ao SRBC para análise. O repositório de procedimentos, por sua vez, cresce a partir da experimentação e validação dos procedimentos periciais necessários para solucionar problemas e dificuldades encontrados durante a perícia dos casos armazenados no repositório. Normalmente esse trabalho de validação é feito em um laboratório de informática onde são simuladas condições semelhantes às encontradas no caso problemático.

Somente após a inserção de um novo caso para análise no SRBC, iniciam-se as fases inerentes à metodologia RBC, descritas como:

a) Recuperar: quando um novo caso é apresentado ao Sistema, o repositório de casos é consultado em busca de casos contendo características semelhantes. Uma fórmula é utilizada para o cálculo de similaridade entre os casos. Um ou mais casos podem ser recuperados e apresentados ao usuário do Sistema, juntamente com os procedimentos periciais adotados nesses casos.

b) Reusar: Se o usuário decide por reusar ou adaptar o caso recuperado, é feita uma consulta na base de procedimentos em busca da descrição detalhada dos procedimentos periciais previamente utilizados no caso sugerido.
c) Revisar: após o final da perícia, o usuário apresenta ao Sistema o resultado obtido naquele caso. Caso a sugestão do Sistema não tenha sido satisfatória, é possível reproduzir em laboratório situações semelhantes, em busca de um procedimento pericial mais adequado àquele caso.
d) Reter: resultados obtidos no caso, independentemente de bem-sucedidos ou malsucedidos, são enfim incorporados à base de casos, podendo ser usados em casos futuros para auxiliar em soluções de novos problemas.

4.3 Requisitos fundamentais do Sistema

Além de focar nos objetivos da análise *live,* a arquitetura proposta deve atender mais um requisito fundamental: os aspectos temporais. Por se tratar de uma análise realizada no próprio local da apreensão, o tempo também é um fator relevante na execução de uma perícia em equipamentos computacionais ligados. Não existe legalmente um impeditivo que limite a quantidade de tempo máximo para conduzir um mandado de busca e apreensão. Porém, como se trata de um ambiente não controlado, espera-se que esse procedimento seja o mais breve possível por questões de logística, segurança da equipe, constrangimento no local, entre outros aspectos.

Sendo assim, os requisitos fundamentais para o SRBC de Análise *Live* são:

1) Auxiliar o perito a resguardar os dados para análise (preservação da prova);
2) Auxiliar o perito na filtragem dos dados e dispositivos (otimização da pericia); e
3) Auxiliar o perito a realizar as tarefas 1 e 2 no menor tempo possível, sem prejuízo a qualidade dos dados apreendidos.

O item 1 consiste no objetivo primário do Sistema, sendo que o SRBC deve alertar o perito sobre aplicativos em execução considerados críticos. Segundo Walter & Petroni (2007), a análise da memória no processo de investigação digital permite identificar aplicativos criptográficos sendo executados no Sistema. Esses aplicativos inviabilizam a perícia posterior caso o equipamento seja desligado sem a devida preservação dos dados. Em sua grande maioria, trata-se de aplicativos que utilizam criptografia forte para proteger as informações de usuários ou do Sistema como um

todo. Casos como o de combate a pedofilia, que permitem consolidar a situação de flagrante delito durante o exame, também se enquadram nesse item. Nesses casos, é necessária a constatação, a preservação dos dados e a documentação de todo procedimento para subsidiar a prisão em flagrante do alvo, pois informações voláteis serão perdidas após o desligamento da máquina investigada.

Já o item 2 corresponde ao objetivo secundário, uma vez que o SRBC deve facilitar a perícia convencional posterior, feita em um laboratório de informática. Para isso, deve-se avisar o perito sobre a presença ou não de aplicativos relevantes ao caso que estejam em execução no computador analisado. O Sistema, portanto, pode filtrar os equipamentos computacionais realmente de interesse à investigação, evitando apreensão de todos os equipamentos no localo. Além disso, o SRBC pode sugerir a realização de alguns exames periciais no próprio local, tais como a exportação de registros de aplicativos pré-selecionados. Esses exames têm como objetivo facilitar e abreviar a perícia posterior.

Uma vez alcançados os objetivos propostos nos itens 1 e 2, o SRBC deve fornecer a solução visando minimizar o tempo gasto durante a realização da análise de equipamentos computacionais ligados. A extração de dados no local da apreensão é normalmente demorada, não se admitindo retrabalho e procedimentos desnecessários. Esses procedimentos devem ser feitos celeremente, porém assegurando-se a qualidade da prova e assim resguardar a cadeia de custódia. Caso o SRBC não identifique nenhuma ameaça ou oportunidades durante a análise no local da busca e apreensão, é sugerido ao perito desligar o equipamento computacional e realizar a perícia convencional estática posteriormente em um laboratório de informática adequadamente equipado.

4.4 Cálculo de similaridade

A recuperação de um caso similar é um passo fundamental para o sucesso de um SRBC. O cálculo de similaridade é usado para medir o grau de semelhança entre os dois casos. Para realizar esse cálculo, os casos armazenados devem ser representados por um objeto contendo os atributos do caso como um vetor, conjunto ou coordenada. Essa proposta utiliza um vetor de atributos e a recuperação de casos similares ao problema atual é feita através da comparação do resultado do cálculo de similaridade entre os vetores que representam casos anteriores.

No caso específico do SRBC, o objetivo do Sistema é utilizar soluções realizadas com sucesso em casos passados, os quais possam ser úteis nos novos casos de análise *live*, a partir da identificação de processos em execução que sejam relevantes à perícia. Esses processos conhecidos são de interesse à análise *live* por inviabilizar ou dificultar a análise pericial convencional posterior.

4.4.1 Representação do caso

Cada caso armazenado na Base de Casos do SRBC pode ser representado simbolicamente em função da presença ou ausência de processos em execução. A partir de uma listagem de processos conhecidos contidos na Base de Casos é possível representar um caso através de uma sequência binária utilizando a Equação 11.

$$\text{Caso[número]} = \{(0|1)^n \mid n > 0\} \tag{11}$$

Onde,

n = número de processos de interesse da perícia cadastrados na Base de Casos.

O dígito 0 (zero) corresponde a ausência do processo em execução e o dígito 1 (um) representa a presença deste processo. A posição do dígito na sequência binária corresponde a um aplicativo específico cadastrado no banco de dados. Por exemplo, considerando os aplicativos TrueCrypt, Emule, Skype e PostgreSQL como aplicativos conhecidos cadastrados no SRBS e de interesse à análise *live*. Os casos hipotéticos A, B, C e D poderiam ser representados pelas seguintes sequências binárias, conforme apresentado na Tabela 4-1:

Caso[A]=1110 Caso[B]=1101 Caso[C]=1000 Caso[D]=0000

Tabela 4-1: Representação da presença de aplicativos em casos específicos.

	TrueCrypt	Emule	Skype	PostgreSQL
A	Presente	Presente	Presente	Ausente
B	Presente	Presente	Ausente	Presente
C	Presente	Ausente	Ausente	Ausente
D	Ausente	Ausente	Ausente	Ausente

No exemplo dado, o caso B teria em execução os aplicativos TrueCrypt, Emule e PostgreSQL. Enquanto o caso D não teria nenhum destes aplicativos em execução.

Após a geração da sequência binária que representa o caso atual, o cálculo de similaridade é feito pela comparação desta sequência com as demais sequências binárias dos casos já armazenados na Base de Casos. Para isso, é utilizada uma fórmula para cálculo de distância ou coeficiente de similaridade.

4.4.2 Fórmula de Similaridade do SRBC

Segundo Liao et al. (1998), um sistema RBC normalmente usa a Distância Euclidiana ou Distância Hamming como medida de similaridade, sendo que o SRBC necessita de uma fórmula de cálculo de similaridade para comparar casos contendo os aplicativos em execução de interesse à análise *live*. Casos que compartilham tais aplicativos em execução podem ser tratados utilizando os mesmos procedimentos periciais em equipamentos computacionais ligados. Sendo assim, no SRBC a similaridade na presença do aplicativo em execução é importante no cálculo, enquanto a ausência deste aplicativo tem pouca ou nenhuma influência na similaridade dos casos.

Entre as fórmulas de cálculo de similaridade de distância apresentadas na Seção 3.5, as fórmulas de *Simple Matching*, a Distância Hamming e a Distância Euclidiana são consideradas simétricas para medição de informações binárias. Sendo assim, as similaridades nos dígitos 0s têm o mesmo peso no cálculo que as similaridades nos dígitos 1s. Já a fórmula de Jaccard desconsidera as similaridades em 0s para o cálculo do coeficiente, tratando-se de uma medição assimétrica para informações binárias. Desta forma, consideramos que o SRBC proposto se adequa melhor ao modelo assimétrico. Para exemplificar, considerando A, B, C, D e E como casos hipotéticos do Sistema e suas respectivas sequências binárias representando os aplicativos de interesse à perícia em execução, tem-se:

A={1111} B={1110} C={0011} D={1000} E={0000}

As matrizes de distância entre os casos de acordo com a fórmula de cálculo são apresentadas na Tabela 4-2 de [a] a [d].

Tabela 4-2: Matrizes de distâncias com diferentes fórmulas de cálculo.

[a] *Simple Matching*

	A	B	C	D	E
A	-	0,25	0,5	0,75	1
B	0,25	-	0,75	0,5	0,75
C	0,5	0,75	-	0,75	0,5
D	0,75	0,5	0,75	-	0,25
E	1	0,75	0,5	0,25	-

[b] *Jaccard*

	A	B	C	D	E
A	-	0,25	0,5	0,75	1
B	0,25	-	0,75	0,67	1
C	0,5	0,75	-	1	1
D	0,75	0,67	1	-	1
E	1	1	1	1	-

[c] *Hamming*

	A	B	C	D	E
A	-	1	2	3	4
B	1	-	3	2	3
C	2	3	-	3	2
D	3	2	3	-	1
E	4	3	2	1	-

[d] *Euclidiana*

	A	B	C	D	E
A	-	1	1,41	1,73	2
B	1	-	1,73	1,41	1,73
C	1,41	1,73	-	1,73	1,41
D	1,73	1,41	1,73	-	1
E	2	1,73	1,41	1	-

Percebe-se, pela comparação das matrizes de distância apresentadas na Tabela 4-2, que as distâncias calculadas pelas fórmulas *Simple Matching,* Hamming e Euclidiana utilizando apenas valores binários possuem resultados relacionados pela Equação 13.

$$DM * n = DH = DE^2 \tag{13}$$

Onde,

$DM = distância\ Simple\ Match$
$DH = distância\ de\ Hamming$
$DE = distância\ Euclidiana$
$n = número\ de\ elementos\ no\ vetor$

Para o SRBC utilizar uma fórmula que considere a similaridade em dígitos 0s (zeros), significa que a ausência de um aplicativo em dois casos aumenta a similaridade entre os dois. Isso pode gerar algumas situações indesejadas como, por exemplo, no cálculo de similaridade entre os casos do exemplo apresentado na Tabela 4-2:

a) A e B possuem distância 0,25 para o cálculo de distância *Simple Matching* e Jaccard. Ou seja, ambas as fórmulas calcularam coeficiente de similaridade 0,75.

b) D e E possuem distância *Simple Matching* de 0,25 e distância Jaccard de 1. Sendo assim os coeficientes de similaridades são respectivamente 0,75 e 0.

c) A e C possuem distância 0,5 para o cálculo de distância *Simple Matching* e Jaccard. Ou seja, ambas as fórmulas calcularam coeficiente de similaridade 0,5.

d) C e E possuem distância *Simple Matching* de 0,5 e distância Jaccard de 1. Sendo assim os coeficientes de similaridades são respectivamente 0,5 e 0.

As similaridades entre os casos A e B são mais relevantes ao SRBC do que as similaridades entre os casos D e E (respectivamente itens *a* e *b*). Os casos A e B compartilham 3 processos de interesse à perícia em execução, enquanto os casos D e E não compartilham nenhum. O mesmo efeito ocorre na comparação entre os casos A e C com os casos C e E (respectivamente itens *c* e *d*). O SRBC tem como objetivo utilizar os procedimentos realizados com sucesso em casos passados para solucionar um novo caso. Para que isso aconteça, o SRBC necessita que os casos tenham processos em execução em comum, não sendo relevantes as similaridades entre as ausências na execução de processos.

Em situações em que há similaridade na ausência de processos em execução entre os casos (itens *b* e *d*), a fórmula de similaridade de Jaccard mostrou-se mais adequada as necessidades impostas pelo SRBC, sendo essa fórmula escolhida para o cálculo de similaridade nesse Sistema. Caso não existam processos em execução em comum entre o caso em análise e o caso armazenado, consequentemente não haverá procedimentos periciais que possam ser aproveitados para resolver o problema atual. Se o SRBC não conseguir encontrar nenhum caso com coeficiente de similaridade maior que zero, significa a ausência de ameaças e oportunidades na realização da perícia em um computador ligado de acordo com o conhecimento atual do Sistema. Nesse caso, recomenda-se o desligamento do equipamento computacional e a realização da perícia convencional estática.

4.5 Interação do Perito com o SRBC

O perito deve interagir com o SRBC através de três categorias de mensagens: a) *alerta*, b) *aviso*, c) *ignore*. A mensagem de *alerta* exige que o perito realize um procedimento de análise *live* para resguardar a prova. Caso os dados não sejam extraídos e preservados antes do desligamento da máquina em análise, é provável que a perícia convencional seja inviabilizada ou prejudicada de forma irreparável. Já a mensagem de *aviso* indica ao perito que caso seja realizado um procedimento com a máquina ligada, a perícia convencional será facilitada. A mensagem de *ignore* é usada pelo Sistema para evitar falsos positivos. Ou seja, mesmo encontrando alguma evidência aparentemente prejudicial à perícia, essa estará relacionada a um evento que não prejudicará a perícia convencional, podendo assim ser ignorada.

Após a emissão da mensagem pelo Sistema ao perito, o SRBC utiliza seu repositório de casos para procurar e recuperar um caso semelhante ao atual utilizando o cálculo de similaridade. A partir daí, os procedimentos periciais já realizados com sucesso no caso recuperado são sugeridos ao perito. A descrição detalhada de como realizar tais procedimentos encontra-se no repositório de procedimentos.

Nesse capítulo foram apresentadas as características da solução proposta para auxiliar o perito durante a realização de uma análise *live*. No capítulo seguinte é apresentado um protótipo implementado a partir desta solução proposta.

5. Protótipo Implementado

Como apresentado no Capítulo 4, o SRBC proposto tem como objetivo auxiliar o perito durante a realização de uma perícia em um equipamento computacional ligado. A partir das informações inseridas, o Sistema sugere procedimentos adotados com sucesso em casos já armazenados na base de casos. Para validar a hipótese levantada foi implementado um protótipo seguindo as especificações descritas na Seção 4.3.

A análise *live* deve processar as diversas informações extraídas da máquina em execução, além de outras informações de interesse ao caso e inseridas pelo perito. O protótipo implementado restringiu-se à análise de dois elementos de informação cruciais à esse tipo de análise: a) processos em execução e b) DLLs carregadas em memória. Esses dois elementos são capazes de identificar indícios que possam ameaçar ou dificultar a perícia convencional posterior, sugerindo realizar os procedimentos já validados na análise *live* em casos semelhantes anteriores. O processo consiste em uma abstração de um programa em execução, conforme apresentado em Tanenbaum (2003). As DLLs são arquivos contendo códigos executáveis de funções que podem ser compartilhadas entre diversos processos (Hart, 2004). Em algumas circunstancias a listagem de DLLs carregadas por processos em execução pode fornecer uma indicação de atividades suspeitas (Walters, 2006). Os processos em execução e DLLs carregadas em memória são os dois elementos utilizados pelo protótipo para calcular a similaridade entres os casos. Apesar de o SRBC ter sido idealizado para atender qualquer sistema operacional, o protótipo foi otimizado para receber informações de sistemas operacionais Windows devido à casuística encontrada pela perícia no DPF.

O protótipo utiliza ferramentas gratuitas e de fácil reprodução do ambiente. A arquitetura cliente-servidor foi escolhida para evitar instalação e atualização de aplicativos no equipamento pericial utilizado para realizar a análise *live*. Apesar de não ser mandatório, o protótipo utiliza preferencialmente a forma indireta (arquivo *dump*) de extração dos dados da memória RAM. Essa forma foi escolhida em detrimento à forma direta (*live reponse*), já que resulta em uma menor interferência e alteração no computador investigado (Walters & Petroni, 2007). Esse fator, juntamente com a possibilidade de se repetir parcialmente o exame, torna a extração indireta mais

adequada à perícia forense. A Figura 5-1 descreve o funcionamento do protótipo implementado.

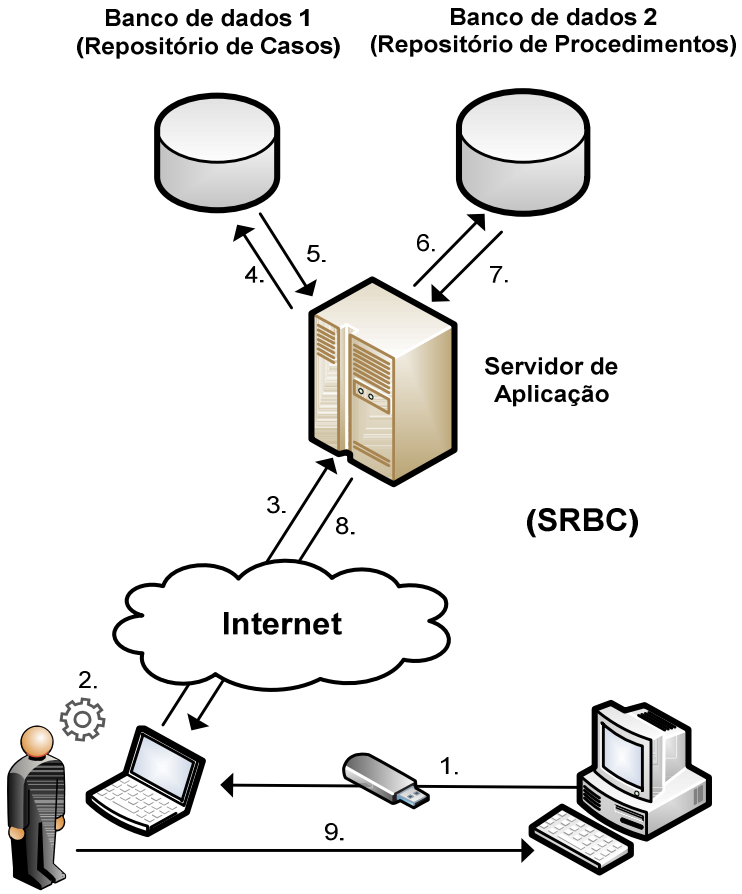

Figura 5-1: Funcionamento do protótipo

O passo 1 consiste no início da análise *live*. Caso o perito constate que o computador a ser analisado encontra-se ligado, a memória RAM deste é inteiramente copiada para um arquivo no *pen drive*. No passo 2, esse arquivo contendo a memória extraída é processado por *parsers* na estação pericial móvel. O resultado do processamento é enviado pela Internet para a interface do Sistema no passo 3. O arquivo do caso é processado pelo Servidor de Aplicação e inserido no banco de dados no passo 4. No passo 5, o Sistema busca por processos conhecidos ou DLL que possam ameaçar a perícia convencional e identifica os casos semelhantes ao caso atual utilizando o cálculo de similaridade apresentado na Seção 4.4.2. Ainda no passo 5, a lista de procedimentos a serem adotados é enviada ao servidor de aplicação. O passo 6 consiste na solicitação dos procedimentos em um modo detalhado. O passo 7 refere-se ao retorno da lista de procedimentos detalhados armazenados no Repositório de Procedimentos. O

SRBC, então, emite um relatório no passo 8 com os procedimentos já adotados em casos semelhantes, orientando o perito na realização da análise do equipamento computacional ligado e na preservação correta de dados para uma perícia convencional posterior. O passo 9 consiste na execução pelo perito dos procedimentos periciais sugeridos pelo Sistema.

5.1 Aplicativos

Para implementar o SRBC proposto, o protótipo deve conter alguns elementos básicos, tais como:

a) Um aplicativo para realizar a cópia para um arquivo do conteúdo integral da memória RAM (*dump* da memória RAM).
b) Aplicativos capazes de analisar o arquivo de *dump* de memória RAM, através do reconhecimento das estruturas de dados da memória e gerar um arquivo de resultado para ser inserido no Sistema.
c) Um Sistema desenvolvido capaz de processar o resultado da análise do arquivo de memória RAM, realizar cálculos de similaridades entre os casos e fazer interface com o usuário do Sistema.
d) Bancos de dados para armazenar os dados dos casos e a descrição dos procedimentos periciais de forma detalhada.

Passaremos a detalhar nas próximas seções cada etapa da análise *live* com uso do protótipo do SRBC desenvolvido.

5.1.1 Cópia integral da memória RAM

Conforme descrito no início deste Capítulo, foi selecionada preferencialmente a extração de informações indireta da memória RAM. Sendo assim, as análises são realizadas a partir do arquivo de *dump* da memória. É necessário um aplicativo específico para gerar o arquivo de *dump* contendo a cópia física integral da memória RAM. Existem diversos programas, comerciais e gratuitos, que realizam esse tipo de cópia. A ferramenta utilizada pelo protótipo foi a *Mantech Physical Memory Dump Utility*, conhecida também por MDD (ManTech Memory DD, 2011), sendo que foi utilizada a versão 1.3 durante os experimentos. Esse executável tem algumas características desejáveis para o uso forense, tais como:

a) Tamanho: o executável é pequeno e não conta com uma GUI (*Graphical User Interface*). Sendo assim, quando executando, ocupa pouco espaço em memória e consequentemente altera menos o sistema investigado;
b) *Hash*: o MDD calcula o código *hash* MD5 (*Message-Digest 5*) do arquivo de *dump* da memória. Esse procedimento é fundamental para garantir a integridade do arquivo após a sua coleta, sendo que o código gerado pode ser inserido no auto de busca e apreensão e permite afirmar que não houve alterações no arquivo durante as análises posteriores.
c) Código aberto: Além de gratuito, o aplicativo ainda possui código-fonte aberto, sendo possível validá-lo e alterar seu código-fonte para incluir novas funcionalidades, conforme a necessidade da perícia.

A cópia integral da memória deve ser feita de forma padronizada e documentada detalhadamente. Esse procedimento não pode ser reproduzido para gerar os mesmos resultados depois do desligamento da máquina investigada.

O comando utilizado para gerar o arquivo *dump* da memória, considerando o E:\ como o *drive* do *pendrive,* é:

```
E:\mdd.exe –o  memorydump_[numero do caso].dd > memorydump_[numero do caso].log
```

```
-> mdd
 -> ManTech Physical Memory Dump Utility
   Copyright (C) 2008 ManTech Security & Mission Assurance

 -> This program comes with ABSOLUTELY NO WARRANTY; for details use option `-w'
   This is free software, and you are welcome to redistribute it
   under certain conditions; use option `-c' for details.

 -> Dumping 511.48 MB of physical memory to file 'memorydump_1455.dd'.


130940 map operations succeeded (1.00)
0 map operations failed

took 478 seconds to write
MD5 is: dc03a0ac18b1c3fe347fe08f0eacd37a
```

Quadro 5-1: Conteúdo do arquivo de log do aplicativo MDD.

O resultado do comando é a cópia de dois arquivos para o *pendrive*: um arquivo de extensão .dd com o conteúdo integral da memória RAM do computador em análise; e

um arquivo de *log* contendo o código *hash* MD5, entre outras informações conforme conteúdo do Quadro 5-1.

É possível verificar no arquivo de *log* gerado o tempo gasto durante a cópia e possíveis erros no procedimento. No exemplo em questão, verifica-se que o tempo gasto para realização da cópia integral do conteúdo da RAM foi 478 segundos e não houve erros durante o procedimento.

A memória RAM é copiada *bit* a *bit,* sendo assim, o arquivo resultante contém exatamente o mesmo tamanho da capacidade de memória física do computador em análise. Após a geração desse arquivo de *dump* é feita a análise para extração dos dados a serem inseridos e processados pelo protótipo do SRBC.

5.1.2 Análise da memoria RAM

Para realizar a extração das informações do arquivo *dump* da memória RAM, foi utilizado um conjunto de *scripts* na linguagem de programação *Python*. Esses *scripts* fazem parte do *framework* chamado Volatility (Volatile Systems, 2011) e funcionam para análise de arquivos *dumps* gerados em Sistemas operacionais Linux, Cygwin, Windows e OSX. Trata-se de analisadores sintáticos que percorrem o conteúdo do arquivo de *dump* em busca de assinaturas dos tipos de dados e estruturas conhecidas. O *framework,* na versão 1.3 beta, suporta a extração de informações através de parâmetros de entrada. O protótipo utilizou três desses parâmetros:

a) *ident:* exibe propriedades do arquivo de *dump* da memória, tais como: nome do arquivo, o tipo do sistema operacional e hora da criação do arquivo.
b) *pslist:* exibe a lista de processos em execução.
c) dlllist: lista as *DLLs* na memória referenciadas para cada processo.

A lista completa dos parâmetros que podem ser utilizados no Volatility versão 1.3 beta encontra-se no Apêndice A. Além das funcionalidades nativas do *framework* Volatility, é possível adicionar novos *plug-ins* desenvolvidos por terceiros, como por exemplo, para extração de arquivos de registro (*hive files*) e de LM e NT *hashes* do registro de Sistemas Windows.

O SRBC implementado utiliza os processos em execução e DLLs carregadas para identificar ameaças ou oportunidades durante a análise do computador ligado. Sendo assim, foram utilizadas as opções *ident*, *pslist* e *dlllist* do Volatility para extração

dos dados do arquivo *dump* de memória RAM das máquinas analisadas. Os comandos utilizados para gerar um arquivo contendo tais informações foram respectivamente:

1) C:\Volatility-1.3_Beta>python volatility ident -f memorydump.dd > memorydump.txt
2) C:\Volatility-1.3_Beta>python volatility pslist -f memorydump.dd >> memorydump.txt
3) C:\Volatility-1.3_Beta>python volatility dlllist -f memorydump.dd >> memorydump.txt

```
Image Name: memorydump_1455.dd
Image Type: Service Pack 3
VM Type: pae
DTB: 0x336000
Datetime: Thu Apr 14 13:37:00 2011
```

Quadro 5-2: Exemplo de informações extraídas com a opção *ident* do Volatility.

O Comando 1 utilizando a opção *ident* extrai informações básicas do arquivo *dump*. A opção permite a extração de dados, tais como: o nome do arquivo *dump*, o tipo de sistema operacional, a data/hora da criação do arquivo *dump* e o endereço do DTB (*Directory Table Base*). O Quadro 5-2 exibe um exemplo dessa extração utilizando a opção *ident*.

Name	Pid	PPid	Thds	Hnds	Time
System	4	0	65	914	Thu Jan 01 00:00:00 1970
smss.exe	504	4	3	20	Thu Apr 14 16:33:13 2011
csrss.exe	592	504	11	505	Thu Apr 14 16:33:16 2011
winlogon.exe	616	504	24	522	Thu Apr 14 16:33:16 2011
services.exe	660	616	17	388	Thu Apr 14 16:33:16 2011
lsass.exe	672	616	21	338	Thu Apr 14 16:33:16 2011
svchost.exe	836	660	24	203	Thu Apr 14 16:33:17 2011
svchost.exe	944	660	10	275	Thu Apr 14 16:33:17 2011
svchost.exe	984	660	96	1709	Thu Apr 14 16:33:17 2011
svchost.exe	1032	660	5	58	Thu Apr 14 16:33:17 2011
svchost.exe	1136	660	14	191	Thu Apr 14 16:33:19 2011
aawservice.exe	1196	660	7	67	Thu Apr 14 16:33:19 2011
spoolsv.exe	1264	660	15	146	Thu Apr 14 16:33:22 2011
netdde.exe	1360	660	9	65	Thu Apr 14 16:33:22 2011
avp.exe	1404	660	32	698	Thu Apr 14 16:33:22 2011
clipsrv.exe	1424	660	2	41	Thu Apr 14 16:33:22 2011
jqs.exe	1480	660	9	235	Thu Apr 14 16:33:22 2011
svchost.exe	1648	660	6	119	Thu Apr 14 16:33:23 2011
tlntsvr.exe	1684	660	4	105	Thu Apr 14 16:33:23 2011
wdfmgr.exe	1720	660	6	64	Thu Apr 14 16:33:23 2011
explorer.exe	304	196	12	324	Thu Apr 14 16:33:26 2011
VTTimer.exe	364	304	1	38	Thu Apr 14 16:33:27 2011
SMax4.exe	392	304	1	30	Thu Apr 14 16:33:27 2011
avp.exe	404	304	12	211	Thu Apr 14 16:33:27 2011
jusched.exe	412	304	1	29	Thu Apr 14 16:33:28 2011
jucheck.exe	420	412	4	101	Thu Apr 14 16:33:28 2011
ctfmon.exe	436	304	1	69	Thu Apr 14 16:33:28 2011

Quadro 5-3: Exemplo de informações extraídas com a opção *pslist* do Volatility.

O Comando 2 utilizando a opção *pslist* extrai dados sobre os processos que estavam em execução durante a cópia da memória RAM. Essa opção lista os processos em execução juntamente com seu Pid (*Process Id*), PPid (*Parent Process Id*), números de *threads*, número de *handles* e o hora de início da execução. O Quadro 5-3 exibe um trecho de um exemplo dessa extração utilizando a opção *pslist*.

O Comando 3 utilizando a opção *dlllist* extrai as DLLs referenciadas por cada processo em execução. O resultado exibe o nome do processo, o Pid (*Process Id*), a linha de comando que iniciou o processo e as DLLs. Para cada DLL referenciada, é possível verificar o endereço, o tamanho em hexadecimal, e o diretório no disco rígido. O Quadro 5-4 exibe um trecho de um exemplo dessa extração utilizando a opção *dlllist*.

```
 ************************************************************************
smss.exe pid: 504
Command line : \SystemRoot\System32\smss.exe


Base         Size         Path
0x48580000   0xf000       \SystemRoot\System32\smss.exe
0x7c900000   0xb3000      C:\WINDOWS\system32\ntdll.dll

************************************************************************
csrss.exe pid: 592
Command line : C:\WINDOWS\system32\csrss.exe ObjectDirectory=\Windows
SharedSection=1024,3072,512       Windows=On       SubSystemType=Windows
ServerDll=basesrv,1       ServerDll=winsrv:UserServerDllInitialization,3
ServerDll=winsrv:ConServerDllInitialization,2          ProfileControl=Off
MaxRequestThreads=16
Service Pack 3

Base         Size         Path
0x4a680000   0x5000       \??\C:\WINDOWS\system32\csrss.exe
0x7c900000   0xb3000      C:\WINDOWS\system32\ntdll.dll
0x75b10000   0xb000       C:\WINDOWS\system32\CSRSRV.dll
0x75b20000   0x10000      C:\WINDOWS\system32\basesrv.dll
0x75b30000   0x4b000      C:\WINDOWS\system32\winsrv.dll
0x77e50000   0x49000      C:\WINDOWS\system32\GDI32.dll
0x7c800000   0x100000     C:\WINDOWS\system32\KERNEL32.dll
0x7e360000   0x91000      C:\WINDOWS\system32\USER32.dll
0x7e690000   0xb0000      C:\WINDOWS\system32\sxs.dll
0x77f50000   0xab000      C:\WINDOWS\system32\ADVAPI32.dll
0x77db0000   0x92000      C:\WINDOWS\system32\RPCRT4.dll
0x77f20000   0x11000      C:\WINDOWS\system32\Secur32.dll
```

Quadro 5-4: Exemplo de informações extraídas com a opção *dlllist* do Volatility

5.2 Servidor de aplicação

O servidor de aplicação é responsável por receber o *upload* do arquivo gerado na análise do *dump* da memória RAM e inserir tais informações no repositório de casos. Além disso, ele também suporta o cadastro dos aplicativos de interesse à análise *live*. Após a inserção do caso no Sistema, o servidor de aplicação realiza o cálculo de similaridade entre o caso recém-inserido com os demais casos já cadastrados. Os casos mais similares são então recuperados e os procedimentos periciais realizados nesses casos podem ser sugeridos ao usuário do Sistema.

Número do Caso	Nome do arquivo processado	Data de inclusão
50	memorydump_735.dd	05/05/2011 03:09:09
49	memorydump_1614.dd	05/05/2011 03:09:07
48	memorydump_1455.dd	05/05/2011 03:09:05
47	memorydump_1453.dd	05/05/2011 03:09:03
46	memorydump_1221.dd	05/05/2011 03:09:01
45	memorydump_1771.dd	05/05/2011 03:08:59
44	memorydump_1745_10.dd	05/05/2011 03:08:56
43	memorydump_1619.dd	05/05/2011 03:08:52
42	memorydump_1609_10.dd	05/05/2011 03:08:51
41	memorydump_1606_10.dd	05/05/2011 03:08:49
40	memorydump_1605_10.dd	05/05/2011 03:08:47
39	memorydump_1566_10.dd	05/05/2011 03:08:45
38	memorydump_1564_10.dd	05/05/2011 03:08:42
37	memorydump_774.dd	05/05/2011 03:08:39
36	memorydump_1562_10.dd	05/05/2011 03:08:37
35	memorydump_238.dd	05/05/2011 03:08:36
34	memorydump_1548_10.dd	05/05/2011 03:08:34
33	memorydump_1497.dd	05/05/2011 03:08:32

Figura 5-2: Interface gráfica do SRBC mostrando os casos cadastrados no Sistema.

A interface gráfica do SRBC foi desenvolvida utilizando linguagem HTML (*HyperText Markup Language*) e com suporte de JavaScript. A Figura 5-2 exibe uma

página do protótipo implementado contendo os casos cadastrados no Sistema, onde é possível verificar o nome dos arquivos de *dump* de memória RAM analisados e as datas de inclusão no Sistema.

Na Figura 5-2 é possível observar alguns registros da amostra de cinquenta casos do DPF simulados em análise *live* e incluídos automaticamente no Sistema. Cada entrada nessa página refere-se a um caso analisado e cada caso contém uma lista de processos em execução e as DLLs referenciadas por cada processo.

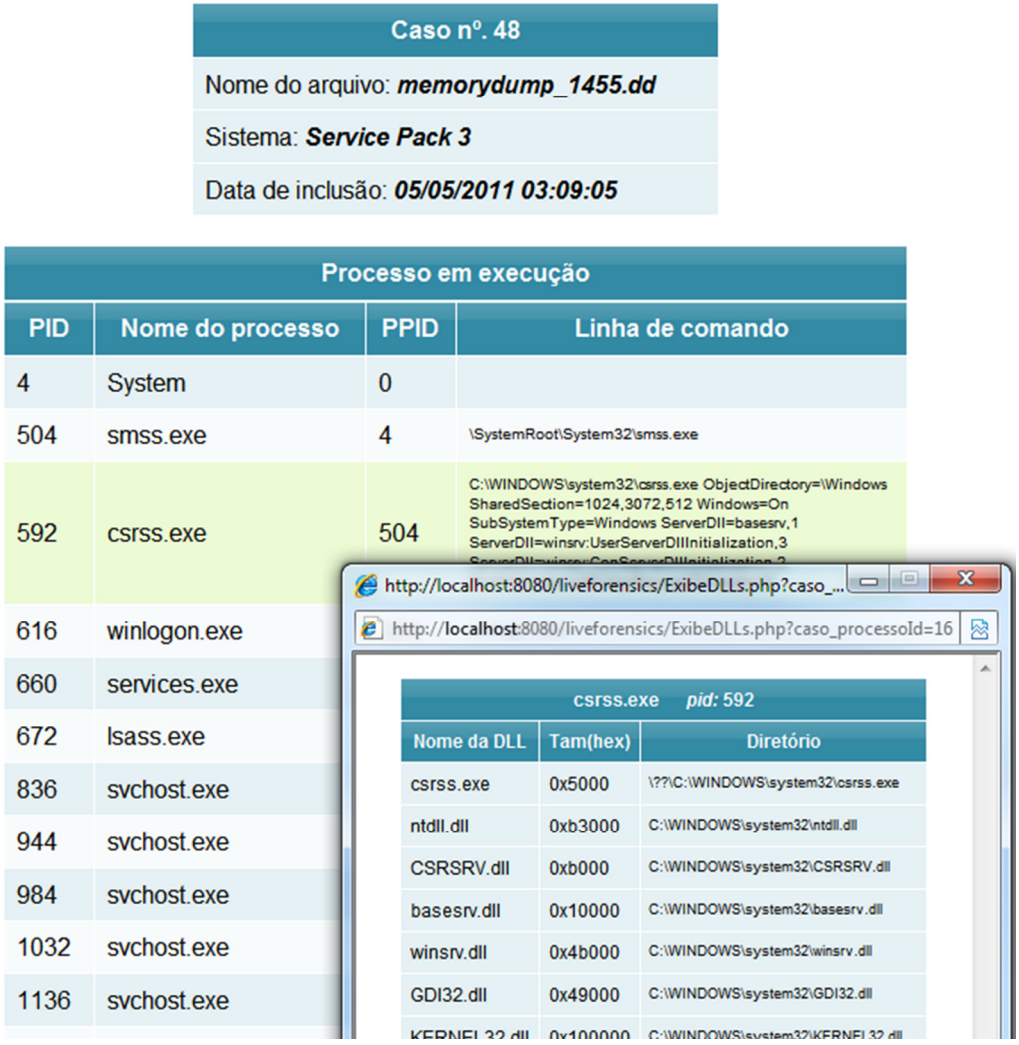

Figura 5-3: Interface gráfica do SRBC mostrando dados de um caso cadastrado.

A Figura 5-3 exemplifica a página contendo os dados referentes ao Caso nº 48. Além dos dados básicos relativos ao Caso, é possível verificar a lista de processos em execução, juntamente com o Pid, PPid e a linha de comando que iniciou o processo. A

Figura 5-3 mostra ainda, na janela de *popup*, as DLLs referenciadas pelo processo *csrss.exe* após o clique no mesmo.

O servidor de página de código aberto Apache na versão 2.2.16 (Apache Project, 2011) foi utilizado e toda a programação foi desenvolvida na linguagem PHP na versão 5.3.3 (PHP, 2011) . Todo desenvolvimento foi realizado para garantir um Sistema escalável e atender a requisitos futuros e novas funcionalidades.

O servidor de aplicação também é responsável pelo cálculo de similaridades entre os casos. Esse procedimento é fundamental na fase de recuperação do ciclo RBC, onde os casos mais similares ao problema atual são selecionados. A fórmula escolhida para realizar esse cálculo foi o coeficiente de Jaccard conforme descrito na Seção 4.4.2.

```
function resultadoUniao($array_A, $array_B){
	$tam=count($array_A);
	$resultado=0;
	for ($i=1; $i<=$tam; $i++){
		if (($array_A[$i])==1 || ($array_B[$i]==1)){
			$resultado++;
		}

	}
	return $resultado;
}

function resultadoInterseccao($array_A, $array_B){
	$tam=count($array_A);
	$resultado=0;
	for ($i=1; $i<=$tam; $i++){
		if (($array_A[$i])==1 && ($array_B[$i]==1)){
			$resultado++;
		}
	}
	return $resultado;
}

function calculaCoeficiente($array_A, $array_B){
	$inter=resultadoInterseccao($array_A,$array_B);
	$uniao=resultadoUniao($array_A, $array_B);
	$coeficiente=intval($inter)/intval($uniao);
	$coeficiente= round($coeficiente,2, PHP_ROUND_HALF_UP);
	return $coeficiente;
}
```

Quadro 5-5: Funções em PHP para cálculo da fórmula de similaridade de Jaccard.

Considerando A e B os casos em comparação, o coeficiente de similaridade de Jaccard pode ser calculado conforme Equação 5. A codificação dessa fórmula foi implementada de forma extremamente simples utilizando a linguagem de programação PHP e três funções conforme apresentado no Quadro 5-5.

A primeira função contida no Quadro 5-5 é responsável por calcular o denominador da Equação 5, enquanto a segunda função calcula o numerador. A terceira função divide os resultados das funções anteriores que resulta no valor do coeficiente de Jaccard entre os casos A e B. Os parâmetros de entrada para as funções, denominados $array_A e $array_B, são vetores contendo uma sequência binária. Cada posição $i desse vetor representa um aplicativo de interesse à análise *live*. O dígito 0 representa a ausência e o dígito 1 representa a presença desse aplicativo em execução no caso representado pelo vetor.

O tamanho do vetor que representa um caso depende da quantidade de aplicativos de interesse à análise *live* cadastrados no Sistema. A cada fase *Revisar* do ciclo RBC, novos aplicativos são inseridos no Sistema se forem identificados ameaças ou oportunidades não cadastradas.

O protótipo foi idealizado utilizando a arquitetura cliente-servidor. Sendo assim, não há a necessidade de atualização constante das estações periciais móveis (*notebooks*) a cada nova versão desenvolvida. Essa abordagem também possibilita a centralização dos bancos de dados, tornado novas adições de casos e atualizações procedimentos periciais imediatamente disponíveis a todos os usuários do Sistema.

5.3 Bancos de dados

A persistência de dados do protótipo SRBC implementado é composta de dois bancos de dados: (i) referente a uma base de dados dos casos e aplicativos denominada Base de Casos ou Repositório de Casos; e (ii) contendo a descrição detalhada dos procedimentos periciais, denominado Repositório de Procedimentos, conforme apresentado na Figura 5-1 pelas entidades em azul claro.

5.3.1 Repositório de casos

O repositório de dados é responsável pelo armazenamento das seguintes informações: dados dos casos, aplicativos conhecidos e procedimentos periciais adotados nos casos.

O banco de dados relacional de código aberto PostgreSQL na versão 9.0 (PostgresSQL, 2011) foi utilizado seguindo o diagrama de entidade-relacionamento descrito na Figura 5-4. É possível verificar nesse diagrama as tabelas correspondentes às principais entidades do SRBC: Caso_info; Aplicativo; Procedimento; Processo e DLL.

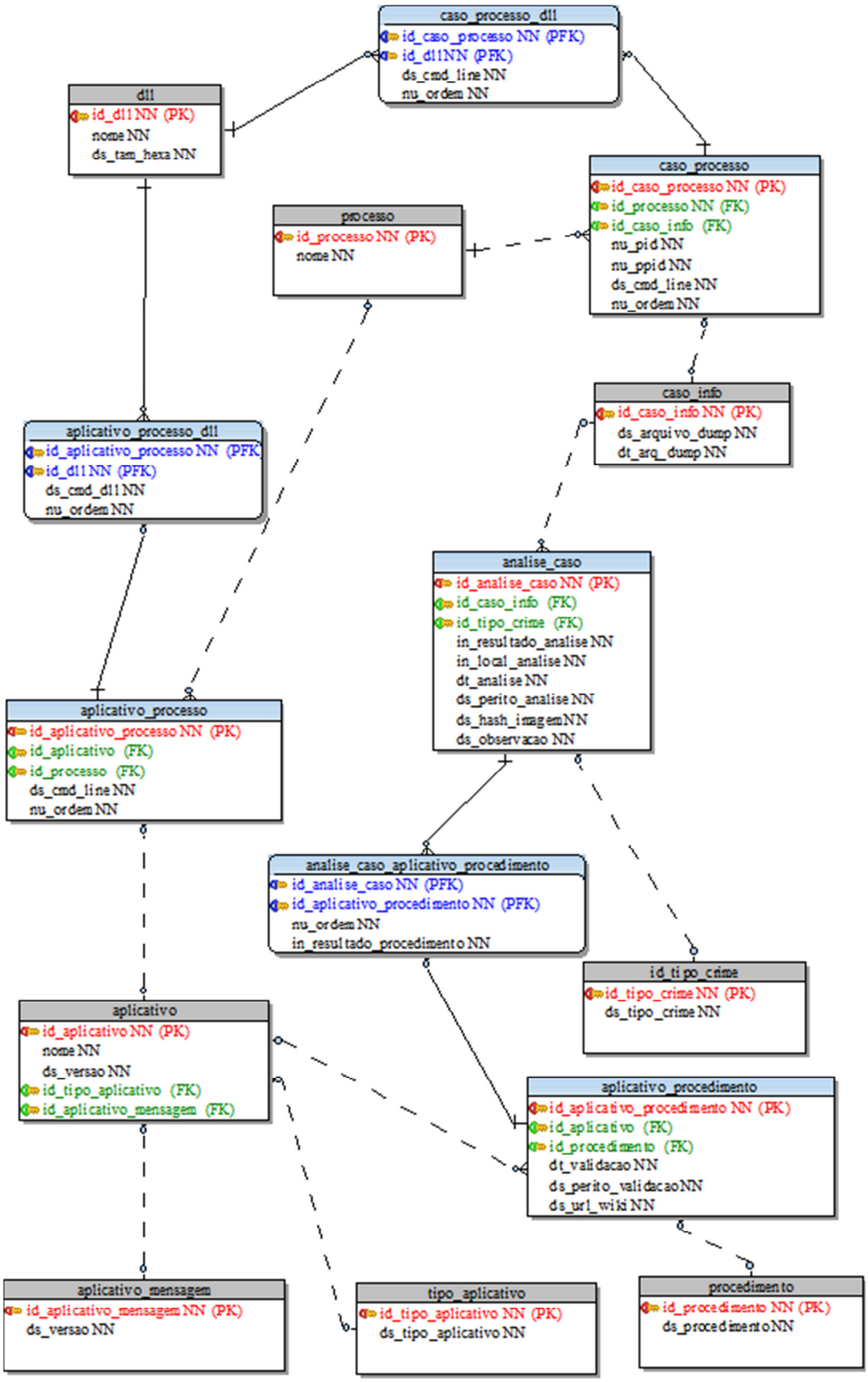

Figura 5-4: Diagrama de entidade-relacionamento do SRBC implementado.

5.3.2 Repositório de Procedimentos

As descrições dos procedimentos periciais encontram-se organizadas na Wiki mantida pelo Serviço de Perícia em Informática (SEPINF). Essa abordagem foi escolhida por já ser um canal de consulta dos peritos criminais federais, por conter diversos procedimentos já cadastrados e pelo caráter colaborativo em sua elaboração. Todos os peritos criminais do DPF possuem acesso a essa base, e foi criada uma cultura organizacional de mantê-la sempre atualizadas com as boas práticas na perícia em informática.

Figura 5-5: Página na Wiki contendo os procedimentos periciais referentes ao Emule.

A Figura 5-5 exibe um exemplo de registro na Wiki utilizado pelo SRBC para realizar um procedimento pericial no equipamento computacional contendo o aplicativo Emule em execução. Sendo assim, o perito pode identificar quais são os arquivos importantes para a análise e quais informações devem ser coletadas durante a análise do computador ligado.

A cada aplicativo de interesse à análise *live* inserido no SRBC, uma nova descrição dos procedimentos periciais referentes a esse aplicativo deve ser cadastrada na Wiki. Os procedimentos periciais cadastrados nesse repositório podem ser atualizados, ou até mesmo alterados, caso seja encontrado algum problema ou falha durante a análise *live*.

A sincronia entre a base de casos e o repositório de procedimentos é importante para garantir a funcionalidade do SRBC. Ou seja, a partir da identificação de um aplicativo durante a realização da perícia em um equipamento computacional ligado, o Sistema possui a capacidade de sugerir o procedimento pericial bem-sucedido relativo àquele aplicativo.

Vale ressaltar que o uso contínuo desta proposta de solução, com o funcionamento do SRBC, será possível a padronização dos procedimentos periciais realizados em casos de sucesso e permitirá a difusão do conhecimento entre os peritos responsáveis pela análise *live* vinculados ao DPF. Julgamos esta uma importante contribuição deste trabalho de mestrado.

6. Experimentação e Análise dos Resultados

Para validar a hipótese que a metodologia de Raciocínio Baseado em Casos pode ser usada para auxiliar o perito durante a realização da análise em um computador ligado foram feitos alguns experimentos utilizando o protótipo SRBC desenvolvido e adotando seis passos metodológicos, a saber:

1) Criação de um ambiente para ser usado na simulação dos casos e identificação de aplicativos de interesse à análise *live*.
2) Cadastro inicial de alguns aplicativos de interesse à análise *live* no SRBC para servirem de padrão na busca por ameaças ou oportunidades durante a análise.
3) Inclusão de alguns casos simulados de análise *live* no SRBC e análise da taxa de repetição nos processos em execução e DLLs referenciadas dos casos para verificar a viabilidade da proposta.
4) Análise de resultados referentes aos casos inseridos, com a identificação dos aplicativos de interesse à análise *live* e cálculo de similaridades entre os casos.
5) Revisar os casos inseridos em busca de novos aplicativos de interesse à análise *live* para cadastrá-los no SRBC.
6) Reanálise dos resultados e do cálculo de similaridade após o cadastro dos novos aplicativos no SRBC.

O resultado esperado pelo uso do SRBC é a identificação dos aplicativos de interesse à análise *live,* que estejam em execução na máquina investigada, bem como a recuperação dos casos mais similares armazenados no repositório de casos do Sistema.

Serão detalhados os seis passos da metodologia adotada na experimentação e análise dos resultados deste trabalho.

6.1 Ambiente de testes

Para testar o protótipo foi necessário configurar um ambiente capaz de simular uma máquina ligada investigada durante uma análise. Máquinas virtuais podem ser usadas tanto para simular casos de análise *live*, quanto para validar procedimentos a serem adotados na presença de aplicativos relevantes à perícia.

Os testes foram feitos utilizando o aplicativo de máquina virtual VMware (VMware, 2011) para evitar a complexidade de configurar várias máquinas reais. O uso de *snapshots* permite a rápida restauração do ambiente a um estado anterior da máquina virtual, facilitando a constatação das alterações ocorridas no Sistema após a instalação de um aplicativo.

6.2 Inserção de aplicativos conhecidos

A presença de alguns aplicativos na máquina investigada pode atrasar ou mesmo inviabilizar a realização da perícia convencional, sendo importante identificar tais aplicativos na máquina ligada durante a análise *live*. O protótipo do SRBC utiliza informações sobre processos em execução e DLLs carregadas em memória para identificar ameaças ou oportunidades durante a análise *live* conforme apresentado no Capítulo 5. Sendo assim, é necessário que o Sistema tenha cadastrados os processos e DLLs dos aplicativos relevantes à perícia.

Para associar um aplicativo a seus respectivos processos e DLLs, basta realizar uma análise comparativa em um computador depois da instalação deste aplicativo. O protótipo usa uma máquina virtual para realizar tal análise. Sendo assim, os processos e DLLs constantes em um aplicativo relevante à perícia são identificados e inseridos no SRBC. Quando o Sistema é atualizado para identificar um novo aplicativo conhecido, o Repositório de Procedimentos também é atualizado com os procedimentos periciais associados a esse novo aplicativo. A máquina virtual também pode ser usada para testar e validar os procedimentos periciais a serem adotados na presença destes aplicativos.

Se o resultado final da perícia convencional, após a realização de uma análise em equipamento computacional ligado, não é considerado satisfatório, o ciclo RBC utiliza-se da fase *Revisar* para identificar possíveis falhas. Novos aplicativos de interesse à perícia e revisão de procedimentos periciais são incorporados às bases de dados do SRBC. Esses aplicativos são incorporados ao Sistema, sendo considerados como aplicativos de interesse à análise *live*. Quando presentes em novos casos inseridos, esses novos aplicativos são identificados e tratados de acordo com o conhecimento previamente adquirido.

Inicialmente foram inseridos dezesseis aplicativos conhecidos, divididos em quatro categorias: criptografia, IM (*Instant Messaging*), P2P (*Peer-to-Peer*), BD (Banco de Dados).

a) *Criptografia:* TrueCrypt, BitLocker, PGPDisk, Symantec Endpoint Encryption.
b) *IM*: Skype, MSN Messenger, Yahoo! Messenger, Windows Live Messenger.
c) *P2P*: eMule, LimeWire, Kazaa, BitTorrent.
d) *BD*: Mysql, PostgreSQL, Firebird e MS SQLServer.

Ferramentas periciais normalmente encontram dificuldades em extrair e disponibilizar dados de usuários destes aplicativos durante a perícia convencional. Os processos e algumas DLLs pré-selecionadas destes aplicativos foram inseridos no protótipo e são utilizados como amostras para realização de buscas destes aplicativos nos novos casos inseridos no Sistema. A Tabela 6-1 mostra exemplos de aplicativos conhecidos, os respectivos processos, e algumas DLLs que foram inseridos na Base de Casos para serem utilizados na identificação de ameaças e oportunidades durante a análise *live*.

Tabela 6-1: Exemplos de aplicativos conhecidos inseridos na Base de Casos

Categoria	Aplicativo	Processo	DLL
Criptografia	TrueCrypt	TrueCrypt.exe	Secur32.dll
			Crypt32.dll
			Cryptui.dll
			Wininet.dll
			Wintrust.dll
	PGPDisk	PGPserv.exe	PGPsdk.dll
			Secur32.dll
			PGPmapih.dll
			Crypt32.dll
P2P	Emule	emule.exe	Wininet.dll
			Winspoll.drv
			COMRes.dll
			Hnetcfg.dll
			DNSAPI.dll
			ICMP.DLL
			Mswsock.dll
IM	Skype	Skype.exe	ezPMUtils.dll
			Rasman.dll
			IPHLPAPI.DLL
			Winspool.drv
			Cryptnet.dll
			Wsock32.dll
			Schannel.dll
	MSN Messenger	Msnmsgr.exe	Msidcrl.dll
			Msgslang.dll
			Custsat.dll
			Msv1_0.dll
			Netapi32.dll
			IMM32.dll

6.3 Inserção de casos reais

Para utilizar uma base de testes reais, foram selecionados cinquenta casos periciados pelo DPF, os quais foram inseridos no protótipo. Os casos escolhidos referem-se a operações recentes deflagradas nos anos de 2009 e 2010. Como é impossível reproduzir a máquina investigada no momento da busca e apreensão, foram utilizadas máquinas virtuais configuradas a partir da cópia forense (imagem) dos discos rígidos desses casos selecionados.

Todos os cinquenta casos possuíam sistema operacional Windows, sendo que cerca de 60% destes casos apresentavam tela de *login* e senha na inicialização. Nesses casos foi necessário utilizar a ferramenta *Locksmith* do ERD Commander para reconfigurar as senhas e continuar a análise.

Com a máquina virtual devidamente configurada com os dados do disco rígido do caso real, foi realizada a cópia integral da memória RAM para um *pendrive*. Após a análise desse arquivo de *dump* da memória, um relatório automatizado foi inserido no Sistema. Apesar de não reproduzir fielmente o ambiente original, foi possível extrair da máquina virtual os mesmos processos e DLLs executados na inicialização do sistema original, sendo possível identificar a execução de aplicativos P2P, IM, BD, antivírus, entre outros.

6.4 Verificação da viabilidade técnica da proposta

O SRBC desenvolvido visa solucionar o seguinte problema: quais procedimentos serão realizados a partir das diversas variáveis apresentadas durante a análise *live*. O protótipo utiliza informações de quais processos estão em execução e quais DLLs estão carregadas na máquina investigada para sugerir procedimentos periciais a serem seguidos. Para verificar a hipótese de que o SRBC pode ser usado para auxiliar o perito durante uma análise de um computador ligado, é necessário verificar se os dois princípios RBC foram de fato observados em uma situação de análise *live*, a saber:

a) Problemas similares possuem soluções similares: os procedimentos periciais são determinísticos, apresentando resultados iguais quando aplicados repetidamente no mesmo caso. A partir da retenção do conhecimento, a última fase do RBC, os

procedimentos periciais aprendidos podem ser utilizados em casos similares e devem produzir resultados similares.

b) Os tipos de problemas encontrados tendem a se repetir: para que esse princípio seja verdadeiro nesse tipo de análise, é necessário demonstrar que as informações extraídas durante esse tipo de análise tendem a se repetir. Especificamente para o protótipo implementado, é preciso que haja repetição em casos distintos de processos em execução e DLLs carregadas na memória RAM.

A Tabela 6-2 exibe os resultados obtidos com inclusão dos cinquenta casos reais selecionados. Nessa tabela é possível verificar as seguintes informações para cada caso analisado: o número de processos em execução, o número de processos distintos em execução, o número de DLLs referenciadas em memória e o número de DLLs distintas carregadas. O mesmo processo pode estar sendo executado por mais de uma vez no mesmo caso, assim como a mesma DLL também pode ser referenciada por mais de um processo.

Tabela 6-2: Propriedade dos casos

Propriedade dos Casos									
Caso	**Processos**		**DLLs**		**Caso**	**Processos**		**DLLs**	
	Executando	**Distintos**	**Referenciadas**	**Distintas**		**Executando**	**Distintos**	**Referenciadas**	**Distintas**
1	26	22	1.061	282	**26**	38	33	1.398	365
2	29	21	1.284	468	**27**	39	33	1.586	410
3	34	27	1.324	351	**28**	30	25	1.022	293
4	35	29	1.628	389	**29**	53	44	2.072	513
5	30	25	1.068	290	**30**	28	21	986	288
6	66	59	2.763	528	**31**	38	30	1.598	415
7	39	32	1.731	424	**32**	55	47	1.678	370
8	26	21	1.269	375	**33**	25	21	1.103	334
9	26	22	1.139	317	**34**	35	29	1.229	344
10	36	28	1.389	357	**35**	24	19	914	283
11	34	27	1.083	313	**36**	31	25	1.209	337
12	39	32	1.522	429	**37**	41	33	1.654	444
13	65	52	2.975	596	**38**	40	34	1.512	389
14	32	28	1.116	315	**39**	31	24	1.293	364
15	33	27	1.450	462	**40**	28	24	933	267
16	41	33	1.666	413	**41**	32	25	1.382	375
17	33	27	1.314	349	**42**	25	19	892	267
18	30	25	1.340	332	**43**	64	44	2.324	527
19	23	19	947	318	**44**	40	31	1.556	385
20	18	14	672	221	**45**	44	36	1.313	358
21	32	26	1.202	304	**46**	29	22	1.161	389
22	36	29	1.606	361	**47**	24	19	1.106	394
23	32	27	1.234	335	**48**	34	27	1.354	448
24	33	28	1.304	438	**49**	27	19	1.123	314
25	26	22	938	283	**50**	27	22	975	274

A Figura 6-1 é a representação gráfica dos dados referentes a processos contidos na Tabela 6-2. O gráfico exibe a quantidade de processos encontrados por caso. É possível verificar a quantidade de processos em execução distintos e a quantidade de processos repetidos em execução no mesmo caso.

Já a Figura 6-2 é a representação gráfica dos dados referentes às DLLs contidas na Tabela 6-2. O gráfico exibe a quantidade de DLLs referenciadas por caso. É possível verificar a quantidade de DLLs carregadas em memória e a quantidade referências repetidas às DLLs já carregadas no mesmo caso.

Em relação aos dados constantes na Tabela 6-2 referentes aos cinquentas casos reais inseridos, tem-se a seguinte compilação:

a) Média de 34,72 processos por caso.

b) Média de 28,16 processos distintos por caso.

c) Média de 1.368 DLLs referenciadas por casos.

d) Média de 367,94 DLLs carregadas por casos.

Na amostragem de casos verificou-se uma porcentagem de aproximadamente 19% em média no número de processos repetidos no mesmo caso (*intra-caso*). Para DLLs, o número delas referenciadas repetidamente no mesmo caso alcançou aproximadamente 73% em média (*intra-caso*).

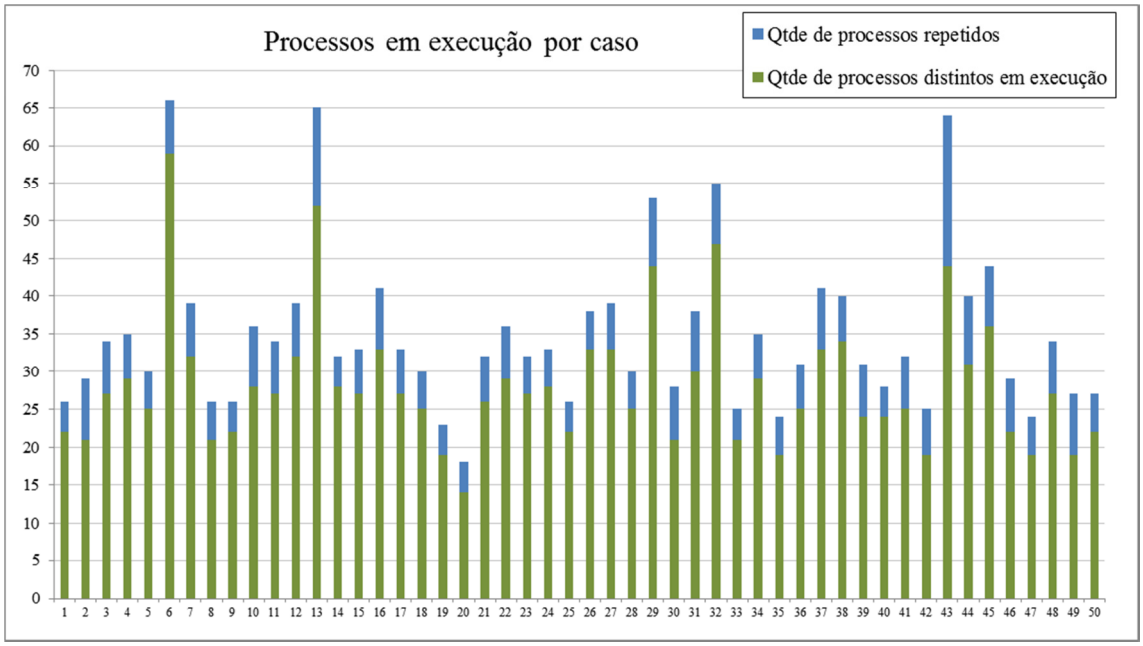

Figura 6-1: Gráfico representativo da quantidade de processos por caso

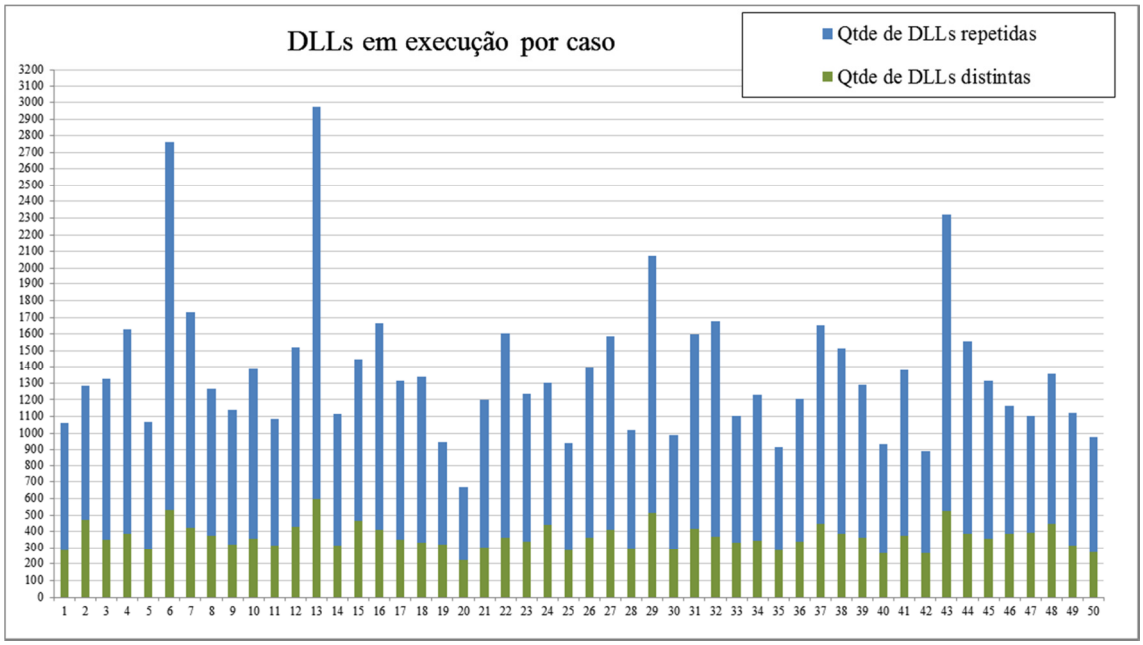

Figura 6-2: Gráfico representativo da quantidade de DLLs por casos

As DLLs foram concebidas pela Microsoft com o objetivo de serem compartilhadas entre os vários processos. Esse compartilhamento otimiza o uso da memória já que uma DLL pode ser referenciada por diversos processos, porém ocupa apenas o espaço endereçado a ela (Hart, 2004). A Figura 6-2 mostra como essa estratégia foi bem-sucedida. Apesar do grande número de DLLs referenciadas por cada processo, o número de DLLs distintas de fato carregadas em memória é significativamente menor.

A Tabela 6-2 e as Figuras 6-1 e 6-2 demonstram que existe a repetição de processos em execução e DLLs referenciadas em um mesmo caso (*intra-caso*). Porém, para justificar o reuso de procedimentos realizados em casos passados em novos casos é necessário que haja essas repetições também entre os casos (*inter-casos*). Isso pode ser feito pela análise do crescimento da Base de Casos, uma vez que processos e DLLs somente são incluídos nos casos quando já não existam registro deles na base. Tratando-se de processos e DLLs já incluídos, é apenas verificada a relação deles com o novo caso inserido.

A Tabela 6-3 exibe o somatório parcial de processos e DLLs após a inclusão de um novo caso. O primeiro caso adiciona todos os processos distintos em execução e todas as DLLs carregadas em memória daquele caso. A partir daí, somente serão inseridos novos processos e DLLs se os mesmos já não pertençam a Base de Casos.

Percebe-se que enquanto os primeiros casos inserem bastantes processos e DLLs na Base de Casos, o número dessas inclusões é visivelmente menor nos últimos casos da amostra.

A Figura 6-3 representa a curva de crescimento da quantidade de processos na Base de Casos. O gráfico demonstra uma tendência de diminuição progressiva nas inserções de processos a cada novo caso. A Figura 6-4 demonstra o mesmo comportamento com relação a inserções de DLLs na Base de Casos.

É possível afirmar pela análise das Figuras 6-3 e 6-4 que existe de fato repetição nos processos em execução e DLLs carregadas entre os casos da amostra de cinquenta casos selecionados. Uma vez constatada essa repetição na ocorrência de diversos processos e DLLs, é perfeitamente viável utilizar conhecimento adquiridos em casos anteriores para resolução de problema em novos casos seguindo o princípio do RBC.

Tabela 6-3: Crescimento da Base de Casos

Crescimento da Base de Casos									
Caso	**Processos**		**DLLs**		**Caso**	**Processos**		**DLLs**	
	Inclusões	**∑ parcial**	**Inclusões**	**∑ parcial**		**Inclusões**	**∑ parcial**	**Inclusões**	**∑ parcial**
1	22	22	282	282	**26**	1	211	60	2.268
2	8	30	245	527	**27**	5	216	55	2.323
3	13	43	171	698	**28**	0	216	4	2.327
4	12	55	107	805	**29**	11	227	91	2.418
5	11	66	42	847	**30**	2	229	8	2.426
6	40	106	241	1.088	**31**	2	231	57	2.483
7	6	112	89	1.177	**32**	8	239	39	2.522
8	2	114	60	1.237	**33**	0	239	4	2.526
9	2	116	27	1.264	**34**	2	241	49	2.575
10	8	124	61	1.325	**35**	0	241	1	2.576
11	5	129	36	1.361	**36**	4	245	31	2.607
12	7	136	87	1.448	**37**	3	248	56	2.663
13	17	153	194	1.642	**38**	6	254	56	2.719
14	6	159	49	1.691	**39**	1	255	39	2.758
15	5	164	35	1.726	**40**	1	256	3	2.761
16	2	166	36	1.762	**41**	4	260	25	2.786
17	4	170	44	1.806	**42**	0	260	8	2.794
18	11	181	77	1.883	**43**	6	266	77	2.871
19	0	181	6	1.889	**44**	6	272	40	2.911
20	0	181	2	1.891	**45**	0	272	13	2.924
21	5	186	34	1.925	**46**	0	272	1	2.925
22	5	191	38	1.963	**47**	0	272	0	2.925
23	8	199	60	2.023	**48**	3	275	14	2.939
24	8	207	170	2.193	**49**	0	275	4	2.943
25	3	210	15	2.208	**50**	0	275	1	2.944

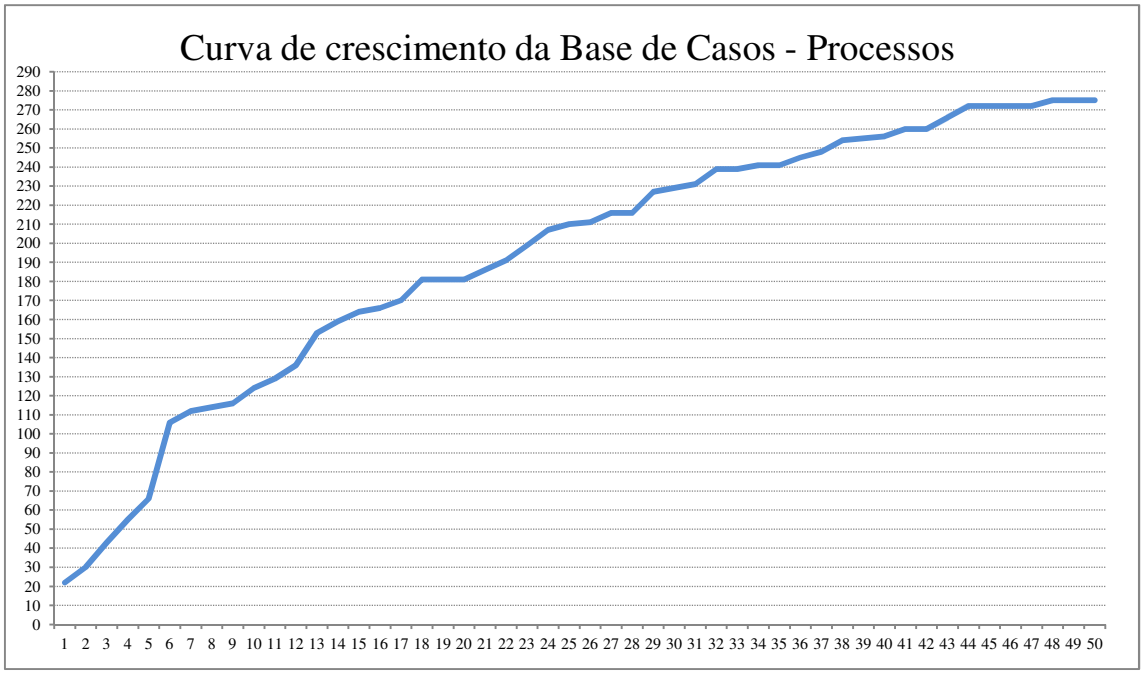

Figura 6-3: Curva de crescimento de processos na Base de Casos

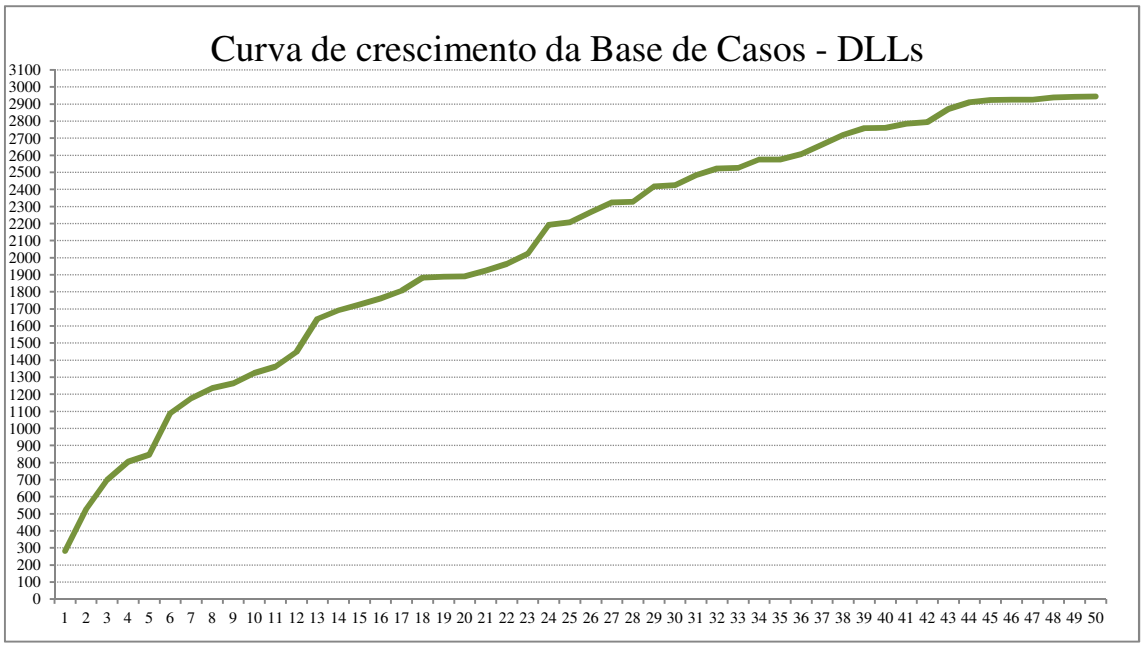

Figura 6-4: Curva de crescimento de DLLs na Base de Casos

As Figuras 6-5 e 6-6 ilustram respectivamente a quantidade de inserções de novos processos e DLLs na Base de Casos. Os picos referem-se aos casos que mais incluem processos e DLLs. Percebe-se, comparando as duas figuras, que normalmente os casos que inserem mais processos são os mesmos que inserem uma grande quantidade de DLLs. Esses picos indicam os casos em que o Sistema está

"apreendendo" novos processos e novas DLLs, o que explica a predominância dos mesmos nos casos iniciais onde a Base de Casos ainda não possui muitos registros.

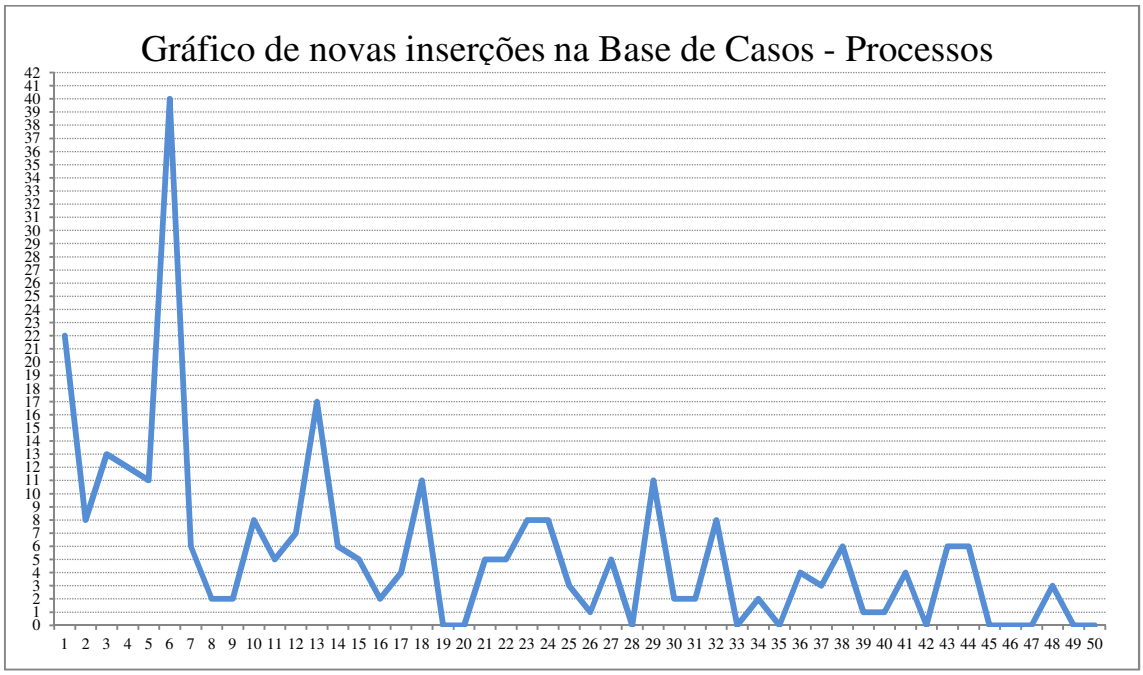

Figura 6-5: Gráfico de novas inserções de processos na Base de Casos

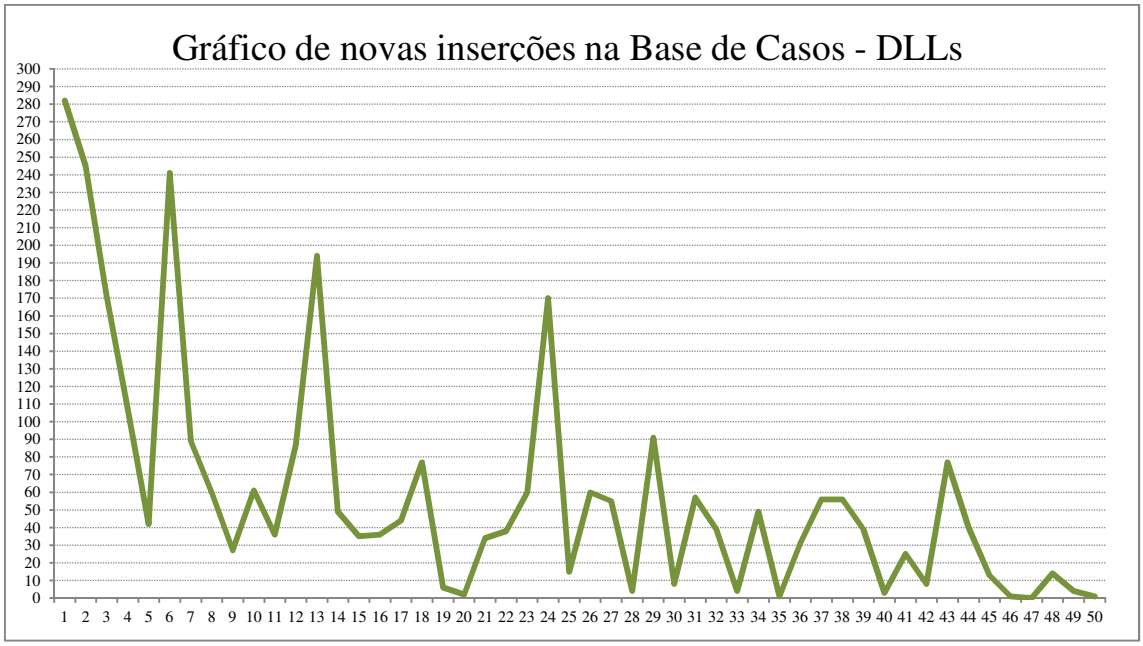

Figura 6-6: Gráfico de novas inserções de DLLs na Base de Casos

Assim como nas Figuras 6-3 e 6-4, que indicam uma redução sistemática de inclusões na Base de Casos, os gráficos de novas inserções, representados nas Figuras

6-5 e 6-6, demonstram uma tendência de diminuição nos tamanhos e nas frequências de picos de inclusões. Essa diminuição significa que a cada novo caso apresentado ao Sistema, uma menor quantidade de novos processos e DLLs são identificados e inseridos na Base de Casos.

Conforme os dados constantes na Tabela 6-3, após a inserção de cinquentas casos, a Base de Casos acumulou a quantidade de 275 processos e 2.944 DLLs cadastrados.

Os cinquenta casos da amostra selecionada possuem um somatório de 1.736 processos em execução, sendo que 1.408 processos distintos no próprio caso (*intra-casos*). Considerando que foram inseridos apenas 275 processos distintos (*inter-casos*) na Base de Casos, tem-se a repetição de aproximadamente de 84,16% de processos em execução na amostra de casos selecionada.

A amostra apresentou o somatório de 68.398 DLLs referenciadas, sendo que 18.397 DLLs foram carregadas pelos casos (*intra-casos*). Considerando que foram inseridas apenas 2.944 DLLs na Base de Casos (*inter-casos*), tem-se a repetição de aproximadamente de 95,70% de DLLs carregadas na amostra de casos selecionada.

Para verificar a tendência de crescimento na quantidade de processos e DLLs inseridos no banco de dados, os casos da amostra foram divididos em cinco grupos de dez casos cada. Foram contabilizados o somatório da quantidade de processos e DLLs inseridos por cada grupo na Base de Casos conforme dados na Tabela 6-4.

Tabela 6-4: Taxa de crescimento da Base de Casos por intervalo de casos

Intervalo de casos	**Processos inseridos**	**%**	**DLLs inseridas**	**%**
0-10	124	45,09%	1325	45,01%
11-20	57	20,73%	566	19,23%
21-30	48	17,45%	535	18,17%
31-40	27	9,82%	335	11,38%
41-50	19	6,91%	183	6,22%

Percebe-se que, enquanto os dez primeiros casos são responsáveis por cerca de 45% dos processos e DLLs inseridas no Banco de Casos, os dez últimos casos são responsáveis por menos de 7% da inclusões.

As Figuras 6-7 e 6-8 são as representações gráficas da Tabela 6-4 e conseguem mostrar a tendência de redução nas inclusões de novos processos e DLLs na Base de Casos, sem as interferências causadas pelos picos de inclusões de casos individuais.

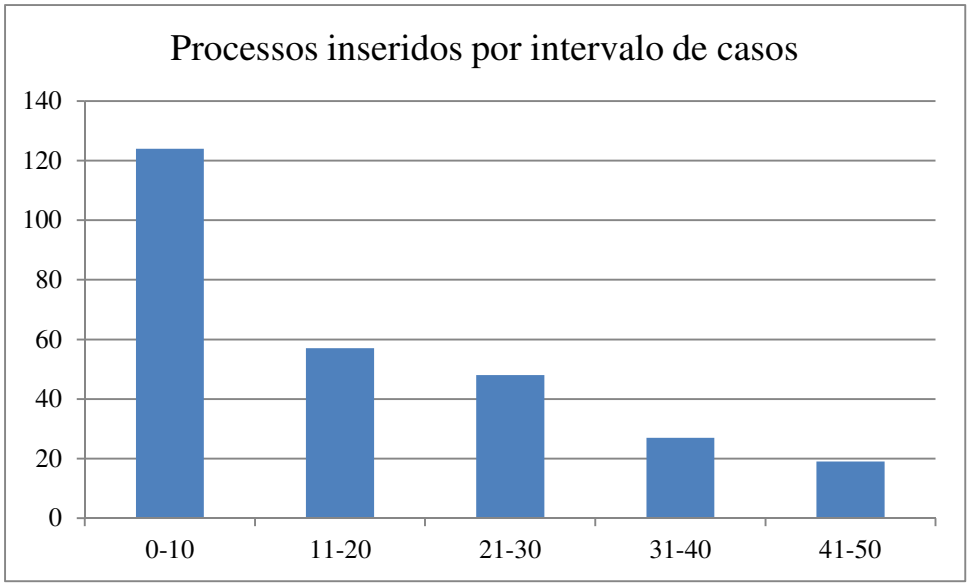

Figura 6-7: Gráfico de inclusões de processos no Sistema por intervalo de casos

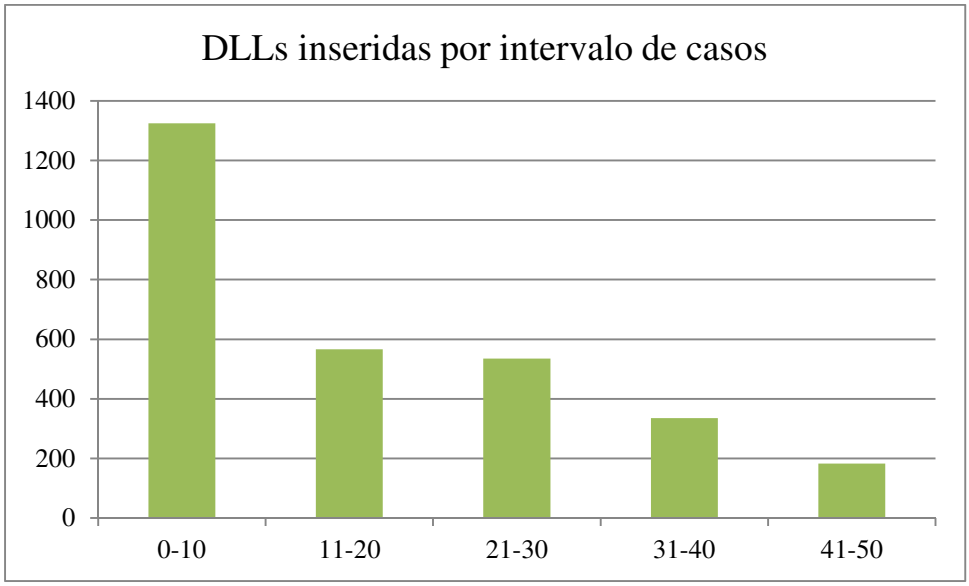

Figura 6-8: Gráfico de inclusões de DLLs no Sistema por intervalo de casos

6.5 Identificação dos aplicativos conhecidos

Para testar o protótipo, foram inseridos no SRBC os dados extraídos dos cinquenta casos reais. Utilizando-se de máquinas virtuais, foram feitas simulações de análises *live* destes casos e os dados foram extraídos dos arquivos de *dump* da memória RAM, conforme descrito na Seção 6.3. O Sistema, então, processou tais dados em busca de elementos (processos e DLLs) que possam identificar os aplicativos conhecidos. A Seção 6.2 contém a lista de dezesseis aplicativos conhecidos incialmente inseridos na Base de Caso.

Com a amostra de cinquenta casos foram feitas as seguintes identificações positivas a partir dos aplicativos conhecidos e constantes da Base de Casos:

a) 19 casos contendo o aplicativo Windows Live Messenger, da categoria IM, em execução.
b) 19 casos contendo o aplicativo MSN Messenger, da categoria IM, em execução.
c) 10 casos contendo o aplicativo Firebird SQL Server, da categoria BD, em execução.
d) 3 casos contendo o aplicativo Skype, da categoria IM, em execução.
e) 1 caso contendo o aplicativo Emule, da categoria P2P, em execução.
f) 1 caso contendo o aplicativo Microsoft SQL Server, da categoria BD, em execução.

Nesses casos o Sistema enviou uma mensagem de *aviso* ao perito. Sendo assim, o perito pode seguir um roteiro, sugerido pelo SRBC, para extração de alguns dados que facilitarão a perícia convencional realizada após a busca e apreensão em um laboratório de informática.

O SRBC identificou 19 casos com mensagens de *alerta*. Isso significa uma ameaça de inviabilizar a perícia convencional se o computador fosse desligado sem realizar a devida extração dos dados do disco rígidos destes casos. Era notório que a amostra de casos não poderia ter tais ameaças, pois inviabilizaria a própria simulação da análise *live*. O processo que gerou tais mensagens de falso-positivo de ameaças foi o *gbpsv.exe*. Trata-se de um processo do aplicativo G-Buster Browser Defense comumente instalado na máquina para permitir a realização de transações bancárias pela Internet. Apesar do processo *gbpsv.exe* não estar presente na Base de Caso associado a

um aplicativo conhecido, esse processo referencia a biblioteca *Secur32.dll.* apresentado DLL está cadastrada e associada a diversos aplicativos de criptografia conforme visto na Tabela 6-1. A DLL *Secur32.dll* consiste em uma biblioteca contendo funções de segurança do Windows e pode ser referenciada por diversos processos inofensivos a perícia convencional, como acontece com o processo *gbpsv.exe.* Para evitar falsos-positivo e consequentemente trabalho desnecessário durante a análise de um equipamento computacional ligado, tais aplicativos devem ser inseridos na Base de Caso com a mensagem *ignore* associada a eles.

Após a identificação dos aplicativos conhecidos foi realizada a fase de *revisão* do RBC conforme apresentado na Figura 4-2. Nessa fase, os dados extraídos de uma máquina investigada em execução (processos e DLLs) dos cinquenta casos que foram inseridos na Base de Casos, foram reanalisados em busca de novos aplicativos de interesse a perícia. Esse processo resultou no cadastro de novos aplicativos conhecidos pelo SRBC e aumentando a capacidade de identificação de ameaças e oportunidades durante a análise *live*. Essa fase cadastrou os seguintes novos aplicativos conhecidos no Sistema:

a) Aplicativo de servidor Telnet, processo *tlntsver.exe*, encontrado em 4 casos.
b) Aplicativo Sybase Central, processo *scjview.exe*, encontrado em 3 casos.
c) Aplicativo SQL Anywhere 10, processo *dbsqlg.exe*, encontrado em 3 casos.
d) Aplicativo Interbase Server, processo *ibserver.exe*, encontrado em 3 casos.
e) Aplicativo Remote Control Tool, processo *r_server.exe*, encontrado em 2 casos.
f) Aplicativos LogMeIn, processo *LMIGuardian.exe*, encontrado em 2 casos.
g) Aplicativo IncrediMail, processo *ImApp.exe*, encontrado em 2 casos.
h) Aplicativo Secure Client Service, processo *SR_service.exe*, encontrado em 1 caso.
i) Aplicativo TeamViewer Remote Control, processo *TeamViewer.exe*, encontrado em 1 caso.
j) Aplicativo Via RAID Tool, processo *raid_tool.exe*, encontrado em 1 caso.
k) Aplicativo Orbit P2P Downloader, processo *orbitnet.exe*, encontrado em 1 caso.
l) Aplicativo Messenger Plus!, processo *MsgPlus.exe*, encontrado em 1 caso.
m) Aplicativo UTSCSI Application, processo *utscisi.exe*, encontrado em 1 caso.

Após a inserção desses aplicativos, é necessário também inserir os procedimentos periciais referentes a esses aplicativos no Repositório de Procedimentos (Wiki). A partir daí, novos casos contendo tais aplicativos já serão identificados e o Sistema terá condições de sugerir os procedimentos recém-cadastrados testados e validados no Repositório de Procedimentos.

6.6 Coeficiente de similaridade entre os casos

Duas análises de coeficiente de similaridade entre os cinquenta casos cadastrados foram feitas. A primeira com apenas os 16 aplicativos cadastrados incialmente no SRBC. Já a segunda análise totalizava o número de 29 aplicativos cadastrados após a fase de revisão, onde foram identificados mais 13 aplicativos de interesse à análise *live*.

Os resultados dos cálculos de similaridades foram expostos em matriz 50 por 50 onde cada posição refere-se ao coeficiente de similaridade entre dois casos. Como dito anteriormente, a fórmula de similaridade foi a de Jaccard e o coeficiente deve ser considerado de acordo com as seguintes propriedades:

a) O coeficiente de similaridade de Jaccard consiste em um valor que varia de 0 (zero) a 1 (um), onde 0 significa a similaridade nenhuma e 1 a similaridade total.
b) A similaridade total, valor 1, significa que ambos os casos sob análise possuem exatamente os mesmos aplicativos de interesse à análise *live* em execução.
c) O valor 0 (zero) significa que nenhum dos dois casos sob análise compartilham sequer um aplicativo de interesse à análise *live* em comum.
d) Cada aplicativo de interesse à análise *live* compartilhado entre os casos resulta em um aumento no coeficiente de similaridade.
e) A execução de aplicativos de interesse à análise *live* em apenas um dos dois casos resulta em uma redução no coeficiente de similaridade.
f) Se um aplicativo de interesse à análise *live* não é encontrado em execução nos dois casos em análise, não existe alteração no coeficiente de similaridade.

Para o cálculo de similaridade, os casos são representados como uma sequência binária. Cada posição da sequência corresponde a um aplicativo de interesse à análise

live. O dígito 1 representa a presença do aplicativo em execução, enquanto o dígito 0 representa a ausência.

As matrizes 50 por 50 contendo o coeficiente de similaridade entre os casos cadastrados no Sistema encontra-se no Apêndice B desse trabalho. Um trecho dessa matriz correspondente aos 15 primeiros casos é apresentado na Tabela 6-6. Essa matriz contém o coeficiente de similaridade considerando os 16 aplicativos de interesse à análise *live* incialmente cadastrados no SRBC.

Já a Tabela 6-8 exibe o coeficiente de similaridade dos mesmos 15 casos, porém considerando o total 29 aplicativos. As sequências binárias que representam os 15 casos encontram-se na Tabela 6-5 para 16 aplicativos e na Tabela 6-7 para os 29 aplicativos.

Quando o SRBC encontra um caso contendo apenas sequências binárias com zeros, é possível afirmar que não foram encontrados pelo SRBC ameaças ou oportunidades para realização de exames periciais durante a análise em uma máquina ligada. Sendo assim o coeficiente de similaridade deste caso como os demais cadastrados também terá o valor 0 (zero) e o SRBC recomendará o desligamento do equipamento computacional e a análise pericial estática posterior. Essa análise é altamente depende da quantidade de aplicativos de interesse à analise *live* cadastrada no Sistema.

Quanto maior o número de aplicativos conhecidos pelo SRBC, mais fina será a granularidade do cálculo de similaridade entre os casos e quanto maior o número de casos cadastrados, maior a probabilidade de encontrar casos similares.

a) Matriz de similaridades 15 por 15, com 16 aplicativos:

Tabela 6-5: Sequência binária representativa dos 15 primeiros casos para 16 aplicativos.

Caso 01: 0000000000000000	Caso 06: 0000100010000000	Caso 11: 0000000000000000
Caso 02: 0000010000000000	Caso 07: 0000110000000000	Caso 12: 0000010000000000
Caso 03: 0000000100000010	Caso 08: 0000000100000000	Caso 13: 0000010100000010
Caso 04: 0000010100000010	Caso 09: 0000000100000000	Caso 14: 0000000100000010
Caso 05: 0000000000000000	Caso 10: 0000000100000000	Caso 15: 0000010000000000

,

Tabela 6-6: Matriz de similaridades entre os 15 primeiros casos para 16 aplicativos.

Caso	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	-	0	0.33	0	0	0.5	0	0	0	0	1	0.33	0	1
3	0	0	-	0.67	0	0	0	0.5	0.5	0.5	0	0	0.67	1	0
4	0	0.33	0.67	-	0	0	0.25	0.33	0.33	0.33	0	0.33	1	0.67	0.33
5	0	0	0	0	-	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	-	0.33	0	0	0	0	0	0	0	0
7	0	0.5	0	0.25	0	0.33	-	0	0	0	0	0.5	0.25	0	0.5
8	0	0	0.5	0.33	0	0	0	-	1	1	0	0	0.33	0.5	0
9	0	0	0.5	0.33	0	0	0	1	-	1	0	0	0.33	0.5	0
10	0	0	0.5	0.33	0	0	0	1	1	-	0	0	0.33	0.5	0
11	0	0	0	0	0	0	0	0	0	0	-	0	0	0	0
12	0	1	0	0.33	0	0	0.5	0	0	0	0	-	0.33	0	1
13	0	0.33	0.67	1	0	0	0.25	0.33	0.33	0.33	0	0.33	-	0.67	0.33
14	0	0	1	0.67	0	0	0	0.5	0.5	0.5	0	0	0.67	-	0
15	0	1	0	0.33	0	0	0.5	0	0	0	0	1	0.33	0	-

Pela análise da matriz e das sequências binárias percebe-se que os casos 1, 5 e 11 não apresentaram em execução nenhum dos 16 aplicativos de interesse a análise *live*. Já a matriz completa 50 x 50 (considerando todos os 50 casos cadastrados), apresentada no Apêndice B, os casos 1, 5, 11, 18, 19, 20, 24, 25, 26, 35, 26, 39, 42, 46, 47 e 49, ou seja, 16 casos não apresentaram nenhum dos 16 aplicativos em execução.

b) Matriz de similaridades 15 por 15, com 29 aplicativos:

Tabela 6-7: Sequência binária representativa dos 15 primeiros casos para 29 aplicativos.

Caso 01: 00000000000000001000000000	Caso 06: 00001000100000000000010000000	Caso 11: 00000000000000000000000000
Caso 02: 00000100000000000000000000	Caso 07: 00001100000000000000000000000	Caso 12: 00000100000000000000000001
Caso 03: 00000001000000100000000000	Caso 08: 00000001000000000000000000	Caso 13: 00000101000000101000000000
Caso 04: 00000101000000100000000001010	Caso 09: 00000001000000000000100000000	Caso 14: 00000001000000100000000000
Caso 05: 00000000000000000000000000	Caso 10: 00000001000000000000000000	Caso 15: 00000100000000000110001000000

Tabela 6-8: Matriz de similaridades entre os 15 primeiros casos para 29 aplicativos.

Caso	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	-	0	0	0	0	0	0	0	0	0	0	0	0.25	0	0
2	0	-	0	0.2	0	0	0.5	0	0	0	0	0.5	0.25	0	0.25
3	0	0	-	0.4	0	0	0	0.5	0.33	0.5	0	0	0.5	1	0
4	0	0.2	0.4	-	0	0	0.17	0.2	0.17	0.2	0	0.17	0.5	0.4	0.13
5	0	0	0	0	-	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	-	0.25	0	0	0	0	0	0	0	0
7	0	0.5	0	0.17	0	0.25	-	0	0	0	0	0.33	0.2	0	0.2
8	0	0	0.5	0.2	0	0	0	-	0.5	1	0	0	0.25	0.5	0
9	0	0	0.33	0.17	0	0	0	0.5	-	0.5	0	0	0.2	0.33	0
10	0	0	0.5	0.2	0	0	0	1	0.5	-	0	0	0.25	0.5	0
11	0	0	0	0	0	0	0	0	0	0	-	0	0	0	0
12	0	0.5	0	0.17	0	0	0.33	0	0	0	0	-	0.2	0	0.2
13	0.25	0.25	0.5	0.5	0	0	0.2	0.25	0.2	0.25	0	0.2	-	0.5	0.14
14	0	0	1	0.4	0	0	0	0.5	0.33	0.5	0	0	0.5	-	0
15	0	0.25	0	0.13	0	0	0.2	0	0	0	0	0.2	0.14	0	-

Pela análise da matriz e das sequências binárias percebe-se que os casos 1, 5 e 11 não apresentaram em execução nenhum dos 29 aplicativos de interesse a análise *live*. Já a matriz completa 50 x 50 (considerando todos os 50 casos cadastrados), exibida no Apêndice B, um total de 12 casos não apresentaram nenhum dos 29 aplicativos em execução.

Sendo assim, utilizando a busca por 16 aplicativos, o Sistema sugeriu procedimentos periciais para a análise em uma máquina ligada em 68% dos 50 casos cadastrados. Já utilizando os 29 aplicativos, o SRBC foi capaz de identificar similaridades e sugerir procedimentos periciais em 76% dos casos.

c) Análise detalhada do cálculo de similaridade de um caso

Para mostrar o funcionamento do SRBC no cálculo de similaridade, pode-se utilizar como exemplo o Caso 14. Considerando os 29 aplicativos de interesse à análise *live*, tem-se o Caso 14 representado pela seguinte sequência binária conforme a Tabela 6-7:

Caso 14: 00000001000000100000000000000

É possível verificar na sequência binária acima que existem 2 aplicativos de interesse à análise *live* em execução nesse caso. Esses aplicativos são representados

pelos dígitos 1s da sequência binária. O aplicativo de IM MSN Messenger e o banco de dados Firebird são os aplicativos de interesse à investigação identificados em execução no Caso 14.

O número de aplicativos de interesse à análise *live* cadastrados no SRBC define o tamanho de dígitos da sequência binária que representa um caso, nesse caso são 29 dígitos.

O próximo passo é o cálculo de similaridade entre os casos já cadastrados no Sistema. Esse resultado pode ser observado na linha 14 da matriz de similaridade contida na Tabela 6-8, transcrita a seguir:

Casos	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
14	0	0	1	0.4	0	0	0	0.5	0.33	0.5	0	0	0.5	-	0

Além de identificar os 2 aplicativos em execução no Caso 14, foram encontrados 6 casos com algum nível de similaridade entre os 15 primeiros casos cadastrados no SRBC. O Sistema, então, exibe ao usuário a lista de casos dos 6 casos similares ordenada decrescentemente a partir do coeficiente de similaridades entre os casos.

Sequências binárias idênticas significam coeficiente de similaridade igual a 1, sendo que se o SRBC deve sugerir o mesmo procedimento pericial adotado no caso armazenado. Sendo assim, os procedimentos adotados no Caso 3 poderiam ser aplicativos de forma integral durante a realização de uma perícia em um computador ligado no Caso 14.6

O resultado final do protótipo do SRBC é um relatório contendo a identificação de todos os aplicativos de interesse à análise *live* cadastrados no SRBC, os quais estejam em execução no caso investigado. Além disso, o Sistema retorna uma lista de tamanho configurável de casos similares ordenados de acordo com coeficiente de similaridades, sugerindo os procedimentos adotados nos casos mais similares ao problema atual.

A Figura 6-9 exibe um exemplo de relatório de similaridade referente ao Caso 14. Na coluna *Aplicativos Similares* existem 2 números: i) quantidade de processos de interesse à perícia em comum nos dois casos (cor verde); e ii) quantidade de processos existentes em apenas um dos casos (cor vermelha). Entre os 29 aplicativos de interesse à

perícia cadastrados no SRBC, o Caso 14 apresentou 2 destes aplicativos em execução. O Caso 3 apresentou coeficiente de similaridade 1, ou seja, havia os mesmos 2 aplicativos de interesse à perícia em execução e os vetores representativos dos Casos 3 e 14 são idênticos. O Caso 13 apesar de apresentar os mesmos 2 aplicativos de interesse à perícia em execução, também continha outros 2 aplicativos não presentes no Caso 14, o que reduziu o coeficiente de similaridade para 0,5. Já os demais casos presentes na Figura 6-9 continham apenas 1 aplicativos de interesse à perícia em comum com o Caso 14 resultando em coeficiente de similaridade 0,5.

Relatório do Caso nº. 14
Nome do arquivo: ***memorydump_680.dd***
Nº. de aplicativos identificados: ***2 de 29***

Casos Similares		
Nº. do caso	**Aplicativos Similares**	**Coeficiente Jaccard**
Caso nº. 3	+2 -0	**1** (100%)
Caso nº. 13	+2 -2	**0.5** (50%)
Caso nº. 8	+1 -0	**0.5** (50%)
Caso nº. 10	+1 -0	**0.5** (50%)
Caso nº. 21	+1 -0	**0.5** (50%)
Caso nº. 30	+1 -0	**0.5** (50%)
Caso nº. 38	+1 -0	**0.5** (50%)
Caso nº. 40	+1 -0	**0.5** (50%)
Caso nº. 44	+1 -0	**0.5** (50%)
Caso nº. 50	+1 -0	**0.5** (50%)

Figura 6-9: Exemplo de relatório contendo as similaridades entre os casos.

Nesse capítulo foram apresentadas a experimentação realizada e a análise dos resultados, com a realização de testes de viabilidade, identificação de aplicativos de interesse à perícia e recuperação de casos a partir do cálculo de similaridade. No capítulo seguinte é apresentada a conclusão e possíveis trabalhos futuros.

7. Conclusões e Trabalhos Futuros

Esse trabalho apresentou uma proposta de sistema automatizado para auxiliar o perito durante uma análise *live*. O Sistema utiliza RBC para identificar similaridades entre o problema atual e casos anteriores já conhecidos e armazenados nos bancos de dados (Repositórios de Casos e Procedimentos). Os casos mais similares são recuperados pelo SRBC, que sugere ao perito os procedimentos periciais adotados nos casos anteriores de sucesso.

O protótipo implementado apresentou bons resultados na identificação de oportunidades durante a perícia em equipamentos computacionais ligados pela análise dos processos e DLLs em execução referentes aos casos inseridos no Sistema. Essas informações podem ser extraídas dos arquivos de *dump* da memória RAM das máquinas investigadas ligadas. Cinquenta casos reais do DPF foram utilizados para simular análises *live* utilizando máquinas virtuais e o conteúdo integral da memória RAM desses casos foram copiados e analisados por *parsers*.

Tendo em vista que RBC utiliza a experiência de casos anteriores para resolver novos casos, é necessário que haja similaridade entre os casos para que o SRBC apresente bons resultados. Durante os experimentos realizados, observou-se que apesar de uma média de aproximadamente 35 processos por casos, resultando em um número total de 1.736 processos, foram cadastrados apenas 275 processos distintos em execução conforme apresentados na Tabela 6-3 e na Figura 6-3. A inserção dos cinquenta casos resultaram em uma média de aproximadamente 1.368 DLLs referenciadas por caso, resultando em número total de 68.398 DLLs. Porém, novamente apenas 2.944 DLLs distintas foram cadastradas no SRBC conforme apresentados na Tabela 6-3 e na Figura 6-4.

Verificou-se uma porcentagem de repetição de 84,16% em relação a processos em execução, e de 95,70% para DLLs referenciadas na amostra de 50 casos cadastrados no Sistema. Esse alto índice de repetição nos processos em execução e DLLs referenciadas reafirma a viabilidade da proposta com o uso do RBC no modelo do Sistema. Observou-se um crescimento maior na base de processos e DLLs nos primeiros casos inseridos no SRBC, enquanto os últimos casos acrescentaram poucos cadastros. Isso pôde ser constatado na curva de crescimento de processos e DLLs e nos

gráficos de picos de inserções no Sistema conforme apresentado nas Figuras 6-5, 6-6 e na Tabela 6-4.

Para testar a identificação de ameaças ou oportunidades durante a análise em computadores ligados, foram inseridos inicialmente 16 aplicativos de interesse à perícia. Esse aplicativos podem inviabilizar ou, pelo menos, dificultar a perícia convencional. Sendo assim, ao identificar tais aplicativos em execução em um computador ligado, o Sistema pode sugerir procedimentos periciais bem-sucedidos adotados realizados em casos similares.

Considerando apenas a busca pelos 16 aplicativos de interesse à perícia, o SRBC sugeriu procedimentos periciais durante a análise *live* em 68% dos 50 casos cadastrados. Nessa amostra de 50 casos, somente 16 casos não apresentaram pelo menos 1 dos aplicativos de interesse à perícia. Na fase *Revisar* do cliclo RBC foram encontrados mais 13 aplicativos considerados de interesse. Esses aplicativos foram cadastrados no SRBC e uma nova rodada de análise foi feita. Já utilizando o total de 29 aplicativos, o Sistema foi capaz de identificar similaridades e sugerir procedimentos periciais em 76% dos casos. Para essa nova quantidade de aplicativos somente 12 casos não apresentaram pelo menos 1 aplicativo de interesse à perícia em execução na amostra de 50 casos.

Para o Sistema proposto, a fórmula de Jaccard mostrou-se mais adequada para o cálculo de similaridade entre os casos de análise *live* e obteve sucesso na recuperação de casos contendo os mesmos processos em execução conforme apresentado na Seção 4.4.

Conclui-se que o SRBC desenvolvido mostrou-se uma ferramenta capaz de auxiliar peritos durante a realização de uma análise em um equipamento computacional ligado. De forma automatizada, é possível extrair e analisar informações de um caso em busca de alguma ameaça ou oportunidade. Caso algum aplicativo de interesse à análise *live* seja identificado, o SRBC consegue, através de cálculos de similaridade, recuperar casos semelhantes ao problema atual. Esses casos recuperados contêm procedimentos periciais de sucesso e fornecem valiosas informações que são sugeridas aos usuários do Sistema.

O SRBC não substitui e nem tem a intenção de substituir o perito durante a análise *live*. Trata-se de uma ferramenta pericial para auxiliar peritos durante uma análise notoriamente complexa, sendo que cabe ainda aos peritos interpretarem e validarem os resultados do Sistema. As maiores contribuições do SRBC consistem na

padronização dos procedimentos periciais e na difusão dos conhecimentos relativos a análise *live* a todos os peritos.

A solução proposta de SRBC apresenta uma arquitetura capaz de analisar diversas informações durante a análise de um computador ligado. Todavia, o protótipo analisa apenas duas dessas informações: processos em execução e DLL carregadas em memória. Portanto, como trabalhos futuros sugerimos a inclusão no Sistema de novos tipos de informações possíveis de serem coletadas durante a análise *live*, tais como, arquivos abertos, aplicativos instalados, conexões estabelecidas entre outros.

O protótipo implementado processa os dados extraídos do arquivo de *dump* da memória RAM de um computador investigado. É possível adaptar o Sistema para receber relatórios gerados por ferramentas periciais comumente utilizada em resposta à incidentes – análise por *live* response. Uma nova análise pode ser feita a partir da inclusão no SRBC de casos reais de análise *live.*

Pode-se também utilizar em trabalhos futuros um algoritmo de aprendizagem por reforço na fase de revisão do SRBC para validação automática dos procedimentos periciais com auxílio dos *feedbacks* dos peritos.

Finalmente, consideramos que o SRBC alcançará a maturidade quando inclusões de casos resultarem em um aumento insignificante no número de processos e DLLs cadastrados no Sistema. Desta forma poucos novos aplicativos de interesse à análise *live* serão identificados na fase *Revisar* do ciclo RBC.

A. Lista completa de parâmetros do Volatility 1.3a

i. *connections*: exibe a lista de conexões estabelecidas com endereço local, endereço remoto e o *Pid* (*Process id*).

ii. *connscan*: além da lista de conexões estabelecidas, exibe conexões escondias e histórico de conexões, caso estejam ainda em memória.

iii. *connscan2*: exibe as mesmas informações da opção *connscan.*

iv. *datetime:* exibe a hora e data do Sistema em que foi feita a cópia da memória RAM.

v. *dlllist:* lista as *DLLs* na memória referenciadas para cada processo.

vi. *dmp2raw:* converte um arquivo *dump* de falha de Sistema em um arquivo de *dump* da memória.

vii. *dmpchk:* exibe as informações de um arquivo *dump* de falha.

viii. *files:* lista os arquivos abertos para cada processo.

ix. *hibinfo:* converte o arquivo de hibernação em arquivo de *dump* da memória.

x. *ident:* exibe propriedades do arquivo de *dump* da memória, tais como: nome da arquivo, o tipo da sistema operacional e hora da criação do arquivo.

xi. *memdmp:* extrai a memória endereçada para um processo.

xii. *memmap:* exibe o mapa de memória.

xiii. *modscan:* busca módulos com os atributos de nome, base e tamanho.

xiv. *modscan2:* exibe as mesmas informações da opção *modscan.*

xv. *modules:* exibe a lista com os módulos carregados.

xvi. *procdump:* extrai o conteúdo de um processo para um arquivo executável.

xvii. *pslist:* exibe a lista de processos em execução.

xviii. *psscan:* busca por objetos EPROCESS.

xix. *psscan2:* exibe as mesmas informações da opção *psscan.*

xx. *regobjkeys:* lista as chaves de registro abertas para cada processo.

xxi. *sockets:* lista as *sockets* abertas, contendo os atributos Pid, porta, código do protocolo e data de criação.

xxii. *sockscan:* busca por *sockets* abertas.

xxiii. *sockscan2:* exibe as mesmas informações da opção *sockscan.*

xxiv. *strings:* realiza a correspondência entre os deslocamentos físicos e os endereços virtuais.

xxv. *thrdscan:* busca por objetos ETHREAD.

xxvi. *thrdscan2:* exibe as mesmas informações da opção *thrdscan.*

xxvii. *vaddump:* extrai as seções VAD (*Virtural Address Descriptors*) para um arquivo.

xxviii. *vadinfo:* extai as informações VAD.

xxix. *vadwalk:* percorre a árvore VAD.

B. Matrizes de Similaridades

SRBC contendo 16 aplicativos cadastrados

Sequência binária representativa dos casos:

Caso 01: 0000000000000000
Caso 02: 0000010000000000
Caso 03: 0000000100000010
Caso 04: 0000010100000010
Caso 05: 0000000000000000
Caso 06: 0000100010000000
Caso 07: 0000110000000000
Caso 08: 0000000100000000
Caso 09: 0000000100000000
Caso 10: 0000000100000000
Caso 11: 0000000000000000
Caso 12: 0000010000000000
Caso 13: 0000010100000010
Caso 14: 0000000100000010
Caso 15: 0000010000000000
Caso 16: 0000010000000010
Caso 17: 0000000100000000
Caso 18: 0000000000000000
Caso 19: 0000000000000000
Caso 20: 0000000000000000
Caso 21: 0000000100000000
Caso 22: 0000010100000000
Caso 23: 0000100100000000
Caso 24: 0000000000000000
Caso 25: 0000000000000000
Caso 26: 0000000000000000
Caso 27: 0000010000000000
Caso 28: 0000010000000000
Caso 29: 0000010000000000
Caso 30: 0000000000000010
Caso 31: 0000010000000010
Caso 32: 0000010000000000
Caso 33: 0000010100000000
Caso 34: 0000010000000000
Caso 35: 0000000000000000
Caso 36: 0000000000000000
Caso 37: 0000010000000010
Caso 38: 0000000100000000
Caso 39: 0000000000000000
Caso 40: 0000000100000000
Caso 41: 0000010100000000
Caso 42: 0000000000000000
Caso 43: 0000010100000011
Caso 44: 0000000000000010
Caso 45: 0000010100000000
Caso 46: 0000000000000000
Caso 47: 0000000000000000
Caso 48: 0000000100000000
Caso 49: 0000000000000000
Caso 50: 0000000100000000

Matriz de similaridade para 16 aplicativos – Parte 1 de 2:

Caso	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
1	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	-	0	0.33	0	0	0.5	0	0	0	0	1	0.33	0	1	0.5	0	0	0	0	0	0.5	0	0	0
3	0	0	-	0.67	0	0	0	0.5	0.5	0.5	0	0	0.67	1	0	0.33	0.5	0	0	0	0.5	0.33	0.33	0	0
4	0	0.33	0.67	-	0	0	0.25	0.33	0.33	0.33	0	0.33	1	0.67	0.33	0.67	0.33	0	0	0	0.33	0.67	0.25	0	0
5	0	0	0	0	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	-	0.33	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.33	0	0
7	0	0.5	0	0.25	0	0.33	-	0	0	0	0	0.5	0.25	0	0.5	0.33	0	0	0	0	0	0.33	0.33	0	0
8	0	0	0.5	0.33	0	0	0	-	1	1	0	0	0.33	0.5	0	0	1	0	0	0	1	0.5	0.5	0	0
9	0	0	0.5	0.33	0	0	0	1	-	1	0	0	0.33	0.5	0	0	1	0	0	0	1	0.5	0.5	0	0
10	0	0	0.5	0.33	0	0	0	1	1	-	0	0	0.33	0.5	0	0	1	0	0	0	1	0.5	0.5	0	0
11	0	0	0	0	0	0	0	0	0	0	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12	0	1	0	0.33	0	0	0.5	0	0	0	0	-	0.33	0	1	0.5	0	0	0	0	0	0.5	0	0	0
13	0	0.33	0.67	1	0	0	0.25	0.33	0.33	0.33	0	0.33	-	0.67	0.33	0.67	0.33	0	0	0	0.33	0.67	0.25	0	0
14	0	0	1	0.67	0	0	0	0.5	0.5	0.5	0	0	0.67	-	0	0.33	0.5	0	0	0	0.5	0.33	0.33	0	0
15	0	1	0	0.33	0	0	0.5	0	0	0	0	1	0.33	0	-	0.5	0	0	0	0	0	0.5	0	0	0
16	0	0.5	0.33	0.67	0	0	0.33	0	0	0	0	0.5	0.67	0.33	0.5	-	0	0	0	0	0	0.33	0	0	0
17	0	0	0.5	0.33	0	0	0	1	1	1	0	0	0.33	0.5	0	0	-	0	0	0	1	0.5	0.5	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-	0	0	0	0	0	0	0
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-	0	0	0	0	0	0
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-	0	0	0	0	0
21	0	0	0.5	0.33	0	0	0	1	1	1	0	0	0.33	0.5	0	0	1	0	0	0	-	0.5	0.5	0	0
22	0	0.5	0.33	0.67	0	0	0.33	0.5	0.5	0.5	0	0.5	0.67	0.33	0.5	0.33	0.5	0	0	0	0.5	-	0.33	0	0
23	0	0	0.33	0.25	0	0.33	0.33	0.5	0.5	0.5	0	0	0.25	0.33	0	0	0.5	0	0	0	0.5	0.33	-	0	0
24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-	0
25	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-
26	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
27	0	1	0	0.33	0	0	0.5	0	0	0	0	1	0.33	0	1	0.5	0	0	0	0	0	0.5	0	0	0
28	0	1	0	0.33	0	0	0.5	0	0	0	0	1	0.33	0	1	0.5	0	0	0	0	0	0.5	0	0	0
29	0	1	0	0.33	0	0	0.5	0	0	0	0	1	0.33	0	1	0.5	0	0	0	0	0	0.5	0	0	0
30	0	0	0.5	0.33	0	0	0	0	0	0	0	0	0.33	0.5	0	0.5	0	0	0	0	0	0	0	0	0
31	0	0.5	0.33	0.67	0	0	0.33	0	0	0	0	0.5	0.67	0.33	0.5	1	0	0	0	0	0	0.33	0	0	0
32	0	1	0	0.33	0	0	0.5	0	0	0	0	1	0.33	0	1	0.5	0	0	0	0	0	0.5	0	0	0
33	0	0.5	0.33	0.67	0	0	0.33	0.5	0.5	0.5	0	0.5	0.67	0.33	0.5	0.33	0.5	0	0	0	0.5	1	0.33	0	0
34	0	1	0	0.33	0	0	0.5	0	0	0	0	1	0.33	0	1	0.5	0	0	0	0	0	0.5	0	0	0
35	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
36	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
37	0	0.5	0.33	0.67	0	0	0.33	0	0	0	0	0.5	0.67	0.33	0.5	1	0	0	0	0	0	0.33	0	0	0
38	0	0	0.5	0.33	0	0	0	1	1	1	0	0	0.33	0.5	0	0	1	0	0	0	1	0.5	0.5	0	0
39	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
40	0	0	0.5	0.33	0	0	0	1	1	1	0	0	0.33	0.5	0	0	1	0	0	0	1	0.5	0.5	0	0
41	0	0.5	0.33	0.67	0	0	0.33	0.5	0.5	0.5	0	0.5	0.67	0.33	0.5	0.33	0.5	0	0	0	0.5	1	0.33	0	0
42	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
43	0	0.25	0.5	0.75	0	0	0.2	0.25	0.25	0.25	0	0.25	0.75	0.5	0.25	0.5	0.25	0	0	0	0.25	0.5	0.2	0	0
44	0	0	0.5	0.33	0	0	0	0	0	0	0	0	0.33	0.5	0	0.5	0	0	0	0	0	0	0	0	0
45	0	0.5	0.33	0.67	0	0	0.33	0.5	0.5	0.5	0	0.5	0.67	0.33	0.5	0.33	0.5	0	0	0	0.5	1	0.33	0	0
46	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
47	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
48	0	0	0.5	0.33	0	0	0	1	1	1	0	0	0.33	0.5	0	0	1	0	0	0	1	0.5	0.5	0	0
49	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
50	0	0	0.5	0.33	0	0	0	1	1	1	0	0	0.33	0.5	0	0	1	0	0	0	1	0.5	0.5	0	0

Matriz de similaridade para 16 aplicativos – Parte 2 de 2:

Caso	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	1	1	1	0	0.5	1	0.5	1	0	0	0.5	0	0	0	0.5	0	0.25	0	0.5	0	0	0	0	0
3	0	0	0	0	0.5	0.33	0	0.33	0	0	0	0.33	0.5	0	0.5	0.33	0	0.5	0.5	0.33	0	0	0.5	0	0.5
4	0	0.33	0.33	0.33	0.33	0.67	0.33	0.67	0.33	0	0	0.67	0.33	0	0.33	0.67	0	0.75	0.33	0.67	0	0	0.33	0	0.33
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0.5	0.5	0.5	0	0.33	0.5	0.33	0.5	0	0	0.33	0	0	0	0.33	0	0.2	0	0.33	0	0	0	0	0
8	0	0	0	0	0	0	0	0.5	0	0	0	0	1	0	1	0.5	0	0.25	0	0.5	0	0	1	0	1
9	0	0	0	0	0	0	0	0.5	0	0	0	0	1	0	1	0.5	0	0.25	0	0.5	0	0	1	0	1
10	0	0	0	0	0	0	0	0.5	0	0	0	0	1	0	1	0.5	0	0.25	0	0.5	0	0	1	0	1
11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12	0	1	1	1	0	0.5	1	0.5	1	0	0	0.5	0	0	0	0.5	0	0.25	0	0.5	0	0	0	0	0
13	0	0.33	0.33	0.33	0.33	0.67	0.33	0.67	0.33	0	0	0.67	0.33	0	0.33	0.67	0	0.75	0.33	0.67	0	0	0.33	0	0.33
14	0	0	0	0	0.5	0.33	0	0.33	0	0	0	0.33	0.5	0	0.5	0.33	0	0.5	0.5	0.33	0	0	0.5	0	0.5
15	0	1	1	1	0	0.5	1	0.5	1	0	0	0.5	0	0	0	0.5	0	0.25	0	0.5	0	0	0	0	0
16	0	0.5	0.5	0.5	0.5	1	0.5	0.33	0.5	0	0	1	0	0	0	0.33	0	0.5	0.5	0.33	0	0	0	0	0
17	0	0	0	0	0	0	0	0.5	0	0	0	0	1	0	1	0.5	0	0.25	0	0.5	0	0	1	0	1
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
21	0	0	0	0	0	0	0	0.5	0	0	0	0	1	0	1	0.5	0	0.25	0	0.5	0	0	1	0	1
22	0	0.5	0.5	0.5	0	0.33	0.5	1	0.5	0	0	0.33	0.5	0	0.5	1	0	0.5	0	1	0	0	0.5	0	0.5
23	0	0	0	0	0	0	0	0.33	0	0	0	0	0.5	0	0.5	0.33	0	0.2	0	0.33	0	0	0.5	0	0.5
24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
25	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
26	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
27	0	-	1	1	0	0.5	1	0.5	1	0	0	0.5	0	0	0	0.5	0	0.25	0	0.5	0	0	0	0	0
28	0	1	-	1	0	0.5	1	0.5	1	0	0	0.5	0	0	0	0.5	0	0.25	0	0.5	0	0	0	0	0
29	0	1	1	-	0	0.5	1	0.5	1	0	0	0.5	0	0	0	0.5	0	0.25	0	0.5	0	0	0	0	0
30	0	0	0	0	-	0.5	0	0	0	0	0	0.5	0	0	0	0	0	0.25	1	0	0	0	0	0	0
31	0	0.5	0.5	0.5	0.5	-	0.5	0.33	0.5	0	0	1	0	0	0	0.33	0	0.5	0.5	0.33	0	0	0	0	0
32	0	1	1	1	0	0.5	-	0.5	1	0	0	0.5	0	0	0	0.5	0	0.25	0	0.5	0	0	0	0	0
33	0	0.5	0.5	0.5	0	0.33	0.5	-	0.5	0	0	0.33	0.5	0	0.5	1	0	0.5	0	1	0	0	0.5	0	0.5
34	0	1	1	1	0	0.5	1	0.5	-	0	0	0.5	0	0	0	0.5	0	0.25	0	0.5	0	0	0	0	0
35	0	0	0	0	0	0	0	0	0	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
36	0	0	0	0	0	0	0	0	0	0	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0
37	0	0.5	0.5	0.5	0.5	1	0.5	0.33	0.5	0	0	-	0	0	0	0.33	0	0.5	0.5	0.33	0	0	0	0	0
38	0	0	0	0	0	0	0	0.5	0	0	0	0	-	0	1	0.5	0	0.25	0	0.5	0	0	1	0	1
39	0	0	0	0	0	0	0	0	0	0	0	0	0	-	0	0	0	0	0	0	0	0	0	0	0
40	0	0	0	0	0	0	0	0.5	0	0	0	0	1	0	-	0.5	0	0.25	0	0.5	0	0	1	0	1
41	0	0.5	0.5	0.5	0	0.33	0.5	1	0.5	0	0	0.33	0.5	0	0.5	-	0	0.5	0	1	0	0	0.5	0	0.5
42	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-	0	0	0	0	0	0	0	0
43	0	0.25	0.25	0.25	0.25	0.5	0.25	0.5	0.25	0	0	0.5	0.25	0	0.25	0.5	0	-	0.25	0.5	0	0	0.25	0	0.25
44	0	0	0	0	1	0.5	0	0	0	0	0	0.5	0	0	0	0	0	0.25	-	0	0	0	0	0	0
45	0	0.5	0.5	0.5	0	0.33	0.5	1	0.5	0	0	0.33	0.5	0	0.5	1	0	0.5	0	-	0	0	0.5	0	0.5
46	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-	0	0	0	0
47	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-	0	0	0
48	0	0	0	0	0	0	0	0.5	0	0	0	0	1	0	1	0.5	0	0.25	0	0.5	0	0	-	0	1
49	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-	0
50	0	0	0	0	0	0	0	0.5	0	0	0	0	1	0	1	0.5	0	0.25	0	0.5	0	0	1	0	-

SRBC contendo 29 aplicativos cadastrados

Sequência binária representativa dos casos:

Caso 01: 00000000000000001000000000000
Caso 02: 00000100000000000000000000000
Caso 03: 00000001000000100000000000000
Caso 04: 00000101000000100000000001010
Caso 05: 00000000000000000000000000000
Caso 06: 00001000100000000000010000000
Caso 07: 00001100000000000000000000000
Caso 08: 00000001000000000000000000000
Caso 09: 00000001000000000000100000000
Caso 10: 00000001000000000000000000000
Caso 11: 00000000000000000000000000000
Caso 12: 00000100000000000000000000001
Caso 13: 00000101000000101000000000000
Caso 14: 00000001000000100000000000000
Caso 15: 00000100000000000110001000000
Caso 16: 00000100000000100000000000000
Caso 17: 00000001000000000001000000000
Caso 18: 00000000000000000000000000000
Caso 19: 00000000000000000000000000000
Caso 20: 00000000000000000000000000000
Caso 21: 00000001000000000000000000000
Caso 22: 00000101000000000110000000000
Caso 23: 00001001000000000000000000000
Caso 24: 00000000000000000000000100000
Caso 25: 00000000000000000000000000000
Caso 26: 00000000000000000001000000000
Caso 27: 00000100000000000000000000000
Caso 28: 00000100000000000000000000000
Caso 29: 00000100000000000000000000000
Caso 30: 00000000000000100000000000000
Caso 31: 00000100000000100000001000000
Caso 32: 00000100000000000000000000000
Caso 33: 00000101000000000000000000000
Caso 34: 00000100000000000000000000000
Caso 35: 00000000000000000000000000000
Caso 36: 00000000000000000000000000000
Caso 37: 00000100000000100110000010000
Caso 38: 00000001000000000000000000000
Caso 39: 00000000000000000000000000000
Caso 40: 00000001000000000000000000000
Caso 41: 00000101000000000000000000100
Caso 42: 00000000000000000000000000000
Caso 43: 00000101000000111000000000000
Caso 44: 00000000000000100000000000000
Caso 45: 00000101000000000000000000000
Caso 46: 00000000000000000000000000000
Caso 47: 00000000000000000000000000000
Caso 48: 00000001000000001000000000000
Caso 49: 00000000000000000000100000000
Caso 50: 00000001000000000000000000000

Matriz de similaridade para 29 aplicativos – Parte 1 de 2:

Caso	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
1	-	0	0	0	0	0	0	0	0	0	0	0	0.25	0	0	0	0	0	0	0	0	0	0	0	0
2	0	-	0	0.2	0	0	0.5	0	0	0	0	0.5	0.25	0	0.25	0.5	0	0	0	0	0	0.25	0	0	0
3	0	0	-	0.4	0	0	0	0.5	0.33	0.5	0	0	0.5	1	0	0.33	0.33	0	0	0	0.5	0.2	0.33	0	0
4	0	0.2	0.4	-	0	0	0.17	0.2	0.17	0.2	0	0.17	0.5	0.4	0.13	0.4	0.17	0	0	0	0.2	0.29	0.17	0	0
5	0	0	0	0	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	-	0.25	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.25	0	0
7	0	0.5	0	0.17	0	0.25	-	0	0	0	0	0.33	0.2	0	0.2	0.33	0	0	0	0	0	0.2	0.33	0	0
8	0	0	0.5	0.2	0	0	0	-	0.5	1	0	0	0.25	0.5	0	0	0.5	0	0	0	1	0.25	0.5	0	0
9	0	0	0.33	0.17	0	0	0	0.5	-	0.5	0	0	0.2	0.33	0	0	0.33	0	0	0	0.5	0.2	0.33	0	0
10	0	0	0.5	0.2	0	0	0	1	0.5	-	0	0	0.25	0.5	0	0	0.5	0	0	0	1	0.25	0.5	0	0
11	0	0	0	0	0	0	0	0	0	0	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12	0	0.5	0	0.17	0	0	0.33	0	0	0	0	-	0.2	0	0.2	0.33	0	0	0	0	0	0.2	0	0	0
13	0.25	0.25	0.5	0.5	0	0	0.2	0.25	0.2	0.25	0	0.2	-	0.5	0.14	0.5	0.2	0	0	0	0.25	0.33	0.2	0	0
14	0	0	1	0.4	0	0	0	0.5	0.33	0.5	0	0	0.5	-	0	0.33	0.33	0	0	0	0.5	0.2	0.33	0	0
15	0	0.25	0	0.13	0	0	0.2	0	0	0	0	0.2	0.14	0	-	0.2	0	0	0	0	0	0.6	0	0	0
16	0	0.5	0.33	0.4	0	0	0.33	0	0	0	0	0.33	0.5	0.33	0.2	-	0	0	0	0	0	0.2	0	0	0
17	0	0	0.33	0.17	0	0	0	0.5	0.33	0.5	0	0	0.2	0.33	0	0	-	0	0	0	0.5	0.2	0.33	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-	0	0	0	0	0	0	0
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-	0	0	0	0	0	0
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-	0	0	0	0	0
21	0	0	0.5	0.2	0	0	0	1	0.5	1	0	0	0.25	0.5	0	0	0.5	0	0	0	-	0.25	0.5	0	0
22	0	0.25	0.2	0.29	0	0	0.2	0.25	0.2	0.25	0	0.2	0.33	0.2	0.6	0.2	0.2	0	0	0	0.25	-	0.2	0	0
23	0	0	0.33	0.17	0	0.25	0.33	0.5	0.33	0.5	0	0	0.2	0.33	0	0	0.33	0	0	0	0.5	0.2	-	0	0
24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-	0
25	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-
26	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.5	0	0	0	0	0	0	0	0
27	0	1	0	0.2	0	0	0.5	0	0	0	0	0.5	0.25	0	0.25	0.5	0	0	0	0	0	0.25	0	0	0
28	0	1	0	0.2	0	0	0.5	0	0	0	0	0.5	0.25	0	0.25	0.5	0	0	0	0	0	0.25	0	0	0
29	0	1	0	0.2	0	0	0.5	0	0	0	0	0.5	0.25	0	0.25	0.5	0	0	0	0	0	0.25	0	0	0
30	0	0	0.5	0.2	0	0	0	0	0	0	0	0	0.25	0.5	0	0.5	0	0	0	0	0	0	0	0	0
31	0	0.33	0.25	0.33	0	0	0.25	0	0	0	0	0.25	0.4	0.25	0.4	0.67	0	0	0	0	0	0.17	0	0	0
32	0	1	0	0.2	0	0	0.5	0	0	0	0	0.5	0.25	0	0.25	0.5	0	0	0	0	0	0.25	0	0	0
33	0	0.5	0.33	0.4	0	0	0.33	0.5	0.33	0.5	0	0.33	0.5	0.33	0.2	0.33	0.33	0	0	0	0.5	0.5	0.33	0	0
34	0	1	0	0.2	0	0	0.5	0	0	0	0	0.5	0.25	0	0.25	0.5	0	0	0	0	0	0.25	0	0	0
35	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
36	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
37	0	0.2	0.17	0.25	0	0	0.17	0	0	0	0	0.17	0.29	0.17	0.5	0.4	0	0	0	0	0	0.5	0	0	0
38	0	0	0.5	0.2	0	0	0	1	0.5	1	0	0	0.25	0.5	0	0	0.5	0	0	0	1	0.25	0.5	0	0
39	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
40	0	0	0.5	0.2	0	0	0	1	0.5	1	0	0	0.25	0.5	0	0	0.5	0	0	0	1	0.25	0.5	0	0
41	0	0.33	0.25	0.33	0	0	0.25	0.33	0.25	0.33	0	0.25	0.4	0.25	0.17	0.25	0.25	0	0	0	0.33	0.4	0.25	0	0
42	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
43	0.2	0.2	0.4	0.43	0	0	0.17	0.2	0.17	0.2	0	0.17	0.8	0.4	0.13	0.4	0.17	0	0	0	0.2	0.29	0.17	0	0
44	0	0	0.5	0.2	0	0	0	0	0	0	0	0	0.25	0.5	0	0.5	0	0	0	0	0	0	0	0	0
45	0	0.5	0.33	0.4	0	0	0.33	0.5	0.33	0.5	0	0.33	0.5	0.33	0.2	0.33	0.33	0	0	0	0.5	0.5	0.33	0	0
46	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
47	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
48	0.5	0	0.33	0.17	0	0	0	0.5	0.33	0.5	0	0	0.5	0.33	0	0	0.33	0	0	0	0.5	0.2	0.33	0	0
49	0	0	0	0	0	0	0	0	0.5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
50	0	0	0.5	0.2	0	0	0	1	0.5	1	0	0	0.25	0.5	0	0	0.5	0	0	0	1	0.25	0.5	0	0

Matriz de similaridade para 29 aplicativos – Parte 2 de 2:

Caso	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.2	0	0	0	0	0.5	0	0
2	0	1	1	1	0	0.33	1	0.5	1	0	0	0.2	0	0	0	0.33	0	0.2	0	0.5	0	0	0	0	0
3	0	0	0	0	0.5	0.25	0	0.33	0	0	0	0.17	0.5	0	0.5	0.25	0	0.4	0.5	0.33	0	0	0.33	0	0.5
4	0	0.2	0.2	0.2	0.2	0.33	0.2	0.4	0.2	0	0	0.25	0.2	0	0.2	0.33	0	0.43	0.2	0.4	0	0	0.17	0	0.2
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0.5	0.5	0.5	0	0.25	0.5	0.33	0.5	0	0	0.17	0	0	0	0.25	0	0.17	0	0.33	0	0	0	0	0
8	0	0	0	0	0	0	0	0.5	0	0	0	0	1	0	1	0.33	0	0.2	0	0.5	0	0	0.5	0	1
9	0	0	0	0	0	0	0	0.33	0	0	0	0	0.5	0	0.5	0.25	0	0.17	0	0.33	0	0	0.33	0.5	0.5
10	0	0	0	0	0	0	0	0.5	0	0	0	0	1	0	1	0.33	0	0.2	0	0.5	0	0	0.5	0	1
11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12	0	0.5	0.5	0.5	0	0.25	0.5	0.33	0.5	0	0	0.17	0	0	0	0.25	0	0.17	0	0.33	0	0	0	0	0
13	0	0.25	0.25	0.25	0.25	0.4	0.25	0.5	0.25	0	0	0.29	0.25	0	0.25	0.4	0	0.8	0.25	0.5	0	0	0.5	0	0.25
14	0	0	0	0	0.5	0.25	0	0.33	0	0	0	0.17	0.5	0	0.5	0.25	0	0.4	0.5	0.33	0	0	0.33	0	0.5
15	0	0.25	0.25	0.25	0	0.4	0.25	0.2	0.25	0	0	0.5	0	0	0	0.17	0	0.13	0	0.2	0	0	0	0	0
16	0	0.5	0.5	0.5	0.5	0.67	0.5	0.33	0.5	0	0	0.4	0	0	0	0.25	0	0.4	0.5	0.33	0	0	0	0	0
17	0.5	0	0	0	0	0	0	0.33	0	0	0	0	0.5	0	0.5	0.25	0	0.17	0	0.33	0	0	0.33	0	0.5
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
21	0	0	0	0	0	0	0	0.5	0	0	0	0	1	0	1	0.33	0	0.2	0	0.5	0	0	0.5	0	1
22	0	0.25	0.25	0.25	0	0.17	0.25	0.5	0.25	0	0	0.5	0.25	0	0.25	0.4	0	0.29	0	0.5	0	0	0.2	0	0.25
23	0	0	0	0	0	0	0	0.33	0	0	0	0	0.5	0	0.5	0.25	0	0.17	0	0.33	0	0	0.33	0	0.5
24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
25	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
26	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
27	0	-	1	1	0	0.33	1	0.5	1	0	0	0.2	0	0	0	0.33	0	0.2	0	0.5	0	0	0	0	0
28	0	1	-	1	0	0.33	1	0.5	1	0	0	0.2	0	0	0	0.33	0	0.2	0	0.5	0	0	0	0	0
29	0	1	1	-	0	0.33	1	0.5	1	0	0	0.2	0	0	0	0.33	0	0.2	0	0.5	0	0	0	0	0
30	0	0	0	0	-	0.33	0	0	0	0	0	0.2	0	0	0	0	0	0.2	1	0	0	0	0	0	0
31	0	0.33	0.33	0.33	0.33	-	0.33	0.25	0.33	0	0	0.33	0	0	0	0.2	0	0.33	0.33	0.25	0	0	0	0	0
32	0	1	1	1	0	0.33	-	0.5	1	0	0	0.2	0	0	0	0.33	0	0.2	0	0.5	0	0	0	0	0
33	0	0.5	0.5	0.5	0	0.25	0.5	-	0.5	0	0	0.17	0.5	0	0.5	0.67	0	0.4	0	1	0	0	0.33	0	0.5
34	0	1	1	1	0	0.33	1	0.5	-	0	0	0.2	0	0	0	0.33	0	0.2	0	0.5	0	0	0	0	0
35	0	0	0	0	0	0	0	0	0	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
36	0	0	0	0	0	0	0	0	0	0	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0
37	0	0.2	0.2	0.2	0.2	0.33	0.2	0.17	0.2	0	0	-	0	0	0	0.14	0	0.25	0.2	0.17	0	0	0	0	0
38	0	0	0	0	0	0	0	0.5	0	0	0	0	-	0	1	0.33	0	0.2	0	0.5	0	0	0.5	0	1
39	0	0	0	0	0	0	0	0	0	0	0	0	0	-	0	0	0	0	0	0	0	0	0	0	0
40	0	0	0	0	0	0	0	0.5	0	0	0	0	1	0	-	0.33	0	0.2	0	0.5	0	0	0.5	0	1
41	0	0.33	0.33	0.33	0	0.2	0.33	0.67	0.33	0	0	0.14	0.33	0	0.33	-	0	0.33	0	0.67	0	0	0.25	0	0.33
42	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-	0	0	0	0	0	0	0	0
43	0	0.2	0.2	0.2	0.2	0.33	0.2	0.4	0.2	0	0	0.25	0.2	0	0.2	0.33	0	-	0.2	0.4	0	0	0.4	0	0.2
44	0	0	0	0	1	0.33	0	0	0	0	0	0.2	0	0	0	0	0	0.2	-	0	0	0	0	0	0
45	0	0.5	0.5	0.5	0	0.25	0.5	1	0.5	0	0	0.17	0.5	0	0.5	0.67	0	0.4	0	-	0	0	0.33	0	0.5
46	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-	0	0	0	0
47	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-	0	0	0
48	0	0	0	0	0	0	0	0.33	0	0	0	0	0.5	0	0.5	0.25	0	0.4	0	0.33	0	0	-	0	0.5
49	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-	0
50	0	0	0	0	0	0	0	0.5	0	0	0	0	1	0	1	0.33	0	0.2	0	0.5	0	0	0.5	0	-

Referências Bibliográficas

Aamodt, A.; Plaza, E. (1994). Case-Based Reasoning: Foundational Issues, Methological Variations, and System Approaches. *AI Communications*, 7(1), 39-59.

Adelstein, F. (2006). Diagnosing your system without killing it first. *Communications of the ACM*, 49, 63-66.

Apache Project (2011). Disponível em: http://httpd.apache.org/ . Acesso em : 12/05/2011.

Anson, S.; Bunting, S. (2007). *Mastering Windows Network Forensics and Investigation*: Sybex.

Aquilina, J. M.; Casey, E.; Malin, C. H. (2008). *Malware Forensics - Investigating and Analyzing Malicious Code*. Syngress.

Beebe, N. L.; Clark, J. G. (2005). Dealing with Terabyte Dataset in Digital Investigations. *Research Advances in Digital Forensics*, 1, 3-16. Springer

Bogaerts, S.; Leake, D. (2004). Facilitating CBR for Incompletely-Described Cases: Distance Metrics for Partial Problem Descriptions. *ECCBR*, 62-76. Springer.

Boileau, A. (2006). Hit By A Bus: Physical Access Attacks with Firewire. *RUXCON*. Sydney, Australia.

Carrier, B. D. (2004). A Hardware-Based Memory Acquisition Procedure for Digital Investigations. *Digital Investigation Journal*, 1, 50-60.

Carrier, B. D.; Spafford, E. H. (2005). Automated Digital Evidence Target Definition Using Outlier Analysis and Existing Evidence. *Digital Forensic Research Workshop (DFRWS)*. New Orleans, LA.

Carrier, B. D.; Spafford, E. H. (2003). Getting Physical with the Digital Investigation Process. *International Journal of Digital Evidence,* 2:2.

Carrier, B. D. (2006). Risks of live digital forensic analysis. *Communications of the ACM*, 49, 56-61.

Carvey, H. (2011). *Windows Registry Forensics: Advanced Digital Forensic Analysis of the Windows Registry*. Syngress.

Chan, E.; Wan, W.; Chaugule, A.; Campbell, R. (2010). Forenscope: A framework for live forensics. *In Proceedings of the 26th Annual Computer Security Applications Conference*, 307-316.

Craw, S. (2003). Introspective Learning to Build Case-Based Reasoning (CBR) Knowledge Containers. *In Proceedings of the International Workshop on Machine Learning and Data Mining in Pattern Recognition (MLDM2003),* 1-6. Springer.

Faries, J.; Schlossberg, K. (1994). *The effect of similarity on memory for prior problems. Sixteenth Annual Conference of the Cognitive Science Society*, 278-282. Lawrence Erlbaum.

Farmer, D.; Venema, W. (2006). *Forensic Discovery*. 3 ed. Addison-Wesley

Hart, J. M. (2004). *Windows System Programming*. 3 ed. Addison Wesley.

Hay, B.; Nance, K.; Bishop, M. (2009). *Live Analysis: Progress and Challenges. Digital Forensics,* 30-37. IEEE Security and Privacy.

Hoelz, B. W. (2009). MADIK: Uma Abordagem Multiagente para o Exame Pericial de Sistemas Computacionais. Dissertação (Mestrado em Informática) - Universidade de Brasília, Brasília.

Hoelz, B.; Ralha, C.; Mesquita, F. (2011a). A Case-Based Reasoning Framework for Live Forensics. *In Proceedings of the Seventh Annual IFIP WG 11.9 International Conference*. Orlando.

Hoelz, B.; Ralha, C.; Mesquita, F. (2011b). Case-based reasoning in live forensics. In: Advances in Digital Forensics VII, 45-56. Springer.

Kolodner, J. L. (1992). An Introduction to Case-Based Resoning. *Artificial Intelligence Review* 6(1), 3-34.

Kruse II, W. G.; Heiser, J. G. (2002). *Computer Forensics: Incident Response Essentials*. Addison Wesley.

Lancaster, J.; Kolodner, J. (1987). Problem solving in a natural task as a function of experience. *In Proceedings of the Ninth Annual Conference of the Cognitive Science Society.* 727-796. Lawrence Erbaum.

Leake, D. B. (1996). *CBR in Context: The Present and Future*. Menlo Park: AAAI Press/MIT Press.

Liao, T. W.; Zhang, Z.; Mount, C. R. (1998). Similarity Measures For Retrieval in Case-Based Reasoning Systems. *Applied Artificial Intelligence*. 267-288.

Lópes, R.; Mcsherry, D.; Bridge, D.; Smyth, B.; Craw, S.; Faltings, B. (2005). *Retrieval, reuse, revision, and retention in case-based reasoning*. Cambridge University Press.

Lourenço, F.; Lobo, V.; Bação, F. (2004). Binary-based similarity measures for categorical data and their application in Self-Organizing Maps. In: JOCLAD 2004 - XI Jornadas de Classificação e Análise de Dados. Lisboa.

Lowman, S. (2010). The Effect of File and Disk Encryption on Computer Forensics. Disponível em:http://lowmanio.co.uk/. Acesso em: 07/05/2011

Mandia, K.; Prosise, C.; Pepe, M. (2003). *Incident Response & Computer Forensics*. 2 ed. McGraw-Hill.

ManTech Memory DD (2011). Disponível em: http://www.mantech.com/capabilities /mdd.asp. Acesso em: 10/05/2011.

Marling, C.; Squalli, M.; Rissland, E.; Muños-Avila, H.; Aha, D. (2002). Case-based reasoning integrations. AI Magazine, 23:1:69-86.

Marshall, A. M. (2008). Digital Forensics - Digital Evidence in Criminal Investigation. Wiley-Blackwell.

Mead, S. (2006). Unique file identification in the National Software Reference Library. Digital Investigation 3, 138-150.

Petroni, N. L.; Walters, A.; Fraser, T.; Arbaugh, W. A. (2006). FATKit: A Framework for the Extraction and Analysis of Digital Forensic Data from Volatile System Memory. *Digital Investigation Journal* , 197-210.

PHP (2011).Disponível em: http://php.net/. Acesso em: 12/05/2011.

Pinkas, B.; Sander, T. (2002). Securing Passwords Against Dictionary Attacks. *In Proceeding of the 9th ACM Conference on Computer and Communications Security* 161-170. ACM.

Polastro, M.; Eleuterio, P. M. (2010). NuDetective: A Forensic Tool to Help Combat Child Pornography through Automatic Nudity Detection. *Workshops on Database and Expert Systems Applications*. 349-353. Dexa.

PostgresSQL (2011). Disponível em: http://www.postgresql.org/. Acesso em: 12/05/2011.

Real, R.; Vargas, J. M. (1996). The Probabilistic Basis of Jaccard's Index of Similarity. Systematic Biology, 380-385.

Reith, M.; Carr, C.; Gunsch, G. (2002). An Examination of Digital Forensics Models. *International Journal of Digital Evidence*. IJDE, 1(3):1-13

Rijsbergen, V. (1979). *Information Retrieval*. Butterworth-Heinemann Pres.

Rogers, M. K.; Mislan, R.; Goldman, J.; Wedge, T.; Debrota, S. (2006). Computer Forensics Field Triage Process Model. *In Proceedings of the Conference on Digital Forensics, Security and Law,* 27-40.

Russel, S. J.; Norvig, P. (2010). *Artificial Intelligence: A Modern Approach*. 3 ed. Prentice Hall.

Russinovich, M. E.; Solomon, D. A.; Ionescu, A. (2009). Windows Internals. 5 ed. Microsoft.

Schmidt, H.; Norman, G.; Boshuizen, H. (1990). A cognitive perspective on medical expertise: Theory and implications. Academic Medicine(65), 611-621.

Schuster, A. (2006). Searching for processes and threads in Microsoft Windows memory dumps. Digital Investigation, 510-516.

Schweitzer, D. (2003). Incident Response. Indianapolis: Wisley Publishing Inc.

Slade, S. (1991). Case-Based Reasoning: A Research Paradigm. AI Magazine, 42-55.

Solomon, M. G.; Barrett, D.; Broom, N. (2005). Computer Forensics Jump Start. Alameda: Sybex.

Steel, C. (2006). Windows Forensics - The Field Guide for Conducting Corporate Computer Investigations. Wiley.

Tanenbaum, A. S. (2003). Sistemas Operacionais Modernos. 2 ed. Prentice Hall.

Vidas, T. (2006). The Acquisition and Analysis of Random Access Memory. *Journal of Digital Forensic*. Practice, 1(4), 315-323.

VMware (2011). Disponível em: http://www.vmware.com/. Acesso em: 10/05/2011.

Volatile Systems (2011). Disponível em: https://www.volatilesystems.com/default/volatility. Acesso em: 10/05/2011.

Waits, C.; Akinyele, J. A.; Nolan, R; Rogers, L. (2008). Computer Forensics: Results of Live Response Inquiry vs. Memory Image Analysis. CERT Program.

Walters, A. (2006). FATKit: Detecting Malicious Library Injection and Upping the "Anti". Technical report. 4TF ResearchLaboratories.

Walters, A.; Petroni, N. L. (2007). Volatools: Integrating Volatile Memory Forensics into the Digital Investigation Process. Black Hat DC.

Watson, I. (1999). Case-based reasoning is a methodology not a technology. Knowledge-Based Systems, 12, 303-308.

Woodward, A.; Hannay, P. (2008). Forensic implications of using the firewire memory exploit with Microsoft Windows XP. Security and Management, 593-597.