Análise Visual de Comportamento de Código Malicioso

Alexandre Or Cansian Baruque[1,2,+], André Ricardo Abed Grégio[1,2], Paulo Lício de Geus[2]

[1] Centro de Tecnologia da Informação Renato Archer (CTI)

[2]Universidade Estadual de Campinas (Unicamp)

[+] *Bolsista do CNPq — Brasil*

orcansian@gmail.com, argregio@cti.gov.br, paulo@las.ic.unicamp.br

Abstract. *Malware attacks are a major threat to computer systems. To develop counter-measures, it is necessary to understand the behavior presented by malware, i.e., the actions performed in the targets. Dynamic analysis systems are used to trace malware behaviors, but they generate a massive amount of data that can confuse the analyst. Visualization techniques can be applied to these data to identify useful patterns and help in the analysis process. In this paper, we propose a visual and interactive tool to analyze malware behavior.*

Resumo. *Ataques por programas maliciosos constituem uma das principais ameaças aos sistemas computacionais atuais. Para criar contra-medidas, é necessário entender o comportamento destes programas, isto é, as ações realizadas nos alvos. Sistemas de análise dinâmica existem para traçar tais comportamentos, mas geram muitos dados textuais que podem confundir o analista. Técnicas de visualização podem ser utilizadas na tentativa de se identificar padrões que sirvam no auxílio à análise, possibilitando a descoberta de informações úteis. Neste artigo, apresenta-se uma ferramenta interativa e visual para análise de comportamento de código malicioso.*

1. Introdução

Programas maliciosos constituem uma grande ameaça aos usuários de sistemas computacionais. Também conhecidos como malware, esses programas englobam os vírus, worms, cavalos-de-tróia, e podem infectar uma máquina através de arquivos anexos em mensagens de e-mail, do acesso à links de páginas Web servindo conteúdo malicioso e do compartilhamento de mídias contaminadas. A monitoração da execução deste tipo de programa provê uma grande quantidade de informações, que devem ser analisadas de forma a produzir resultados úteis que possam auxiliar na tomada de contra-medidas. Entretanto, muitas variantes de malware surgem a cada dia, causando uma sobrecarga para os mecanismos de defesa e para os analistas de segurança.

As informações obtidas a partir das atividades efetuadas por um programa malicioso podem ser confusas para um analista e, devido à quantidade, pode ser difícil encontrar rapidamente o que é realmente relevante para o tratamento de um incidente deste tipo. Com a finalidade de facilitar a análise das ações nocivas executadas por malware, é possível se aplicar técnicas de visualização, as quais podem permitir a observação de padrões e identificação de comportamentos de ataque de maneira mais intuitiva. Neste trabalho, é apresentada uma ferramenta interativa tridimensional para ajudar na análise

das atividades que um malware efetua durante a infecção de uma máquina-alvo, a qual foi desenvolvida e testada com exemplares reais coletados.

2. Conceitos e Trabalhos Relacionados

Visualização de dados pode ser utilizada para vários objetivos visando a análise [6], tais como:

- **Exploração**, na qual não há uma hipótese definida sobre os fenômenos que podem ocorrer nos dados analisados, envolvendo a busca visual por tendências, exceções ou estruturas visando a definição das hipóteses.
- **Confirmação**, que usa dados de natureza conhecida e hipóteses sobre os fenômenos relacionados de forma a confirmá-las ou rejeitá-las, por meio de visualização.
- **Apresentação**, onde é feita a demonstração visual dos dados, fenômenos relacionados a estes ou hipóteses, de modo a permitir sua interpretação.

Há muitas técnicas de visualização de dados, as quais variam a complexidade e generalidade desde um simples gráfico de área até o fatiamento de volumes tridimensionais. Estas técnicas podem ser agrupadas por categorias, como por exemplo, geométricas, baseadas em ícones, pixels ou grafos, hierárquicas, tridimensionais, ou que se utilizam de mapas. Muitas delas foram utilizadas em trabalhos voltadas à visualização de eventos de segurança.

Quist e Liebrock [8] aplicaram técnicas de visualização para compreender o comportamento de executáveis compilados. O *framework* criado por eles, VERA (*Visualization of Executables for Reversing and Analysis*), auxilia os analistas a terem um melhor entendimento do fluxo de execução de um executável, tornando o processo de engenharia reversa mais rápido.

Conti et al. [2] deselvolveram um sistema que facilita uma análise livre de contexto de arquivos de tipos diversos, fornecendo um rápido panorama do contexto e das estruturas internas dos arquivos. Isto é especialmente útil em um ambiente de análise forense, quando se analisa arquivos em formatos não documentados e busca-se por mensagens de texto ocultas em arquivos binários.

Trinius et al. [10] apresentam de métodos visualização para aprimorar a compreensão do comportamento de *malware*. Em seu trabalho, é mostrado o uso de *treemaps* e *thread graphs* para mostrar as ações de um executável e classificar seu comportamento como malicioso.

O framework DEViSE [9] (*Data Exchange for Visualizing Security Events*) permite ao analista um meio de passar dados de uma ferramenta para outra, obtendo assim uma compreensão maior dos dados ao agregar mais informações extraídas de várias origens.

Existem diversas ferramentas que fazem uso da visualização para fins de análise voltada à segurança, cada uma delas utilizando uma abordagem própria das técnicas, com vantagens e desvantagens de acordo com a situação em que é utilizada. Como visto, há também muita pesquisa na tentativa de superar as dificuldades causadas pela grande quantidade de dados presentes em dados de eventos de segurança.

A principal limitação dos trabalhos nesta área é que parte da pesquisa não é aberta ao público, as ferramentas muitas vezes não são interativas ou intuitivas o suficiente, e

a interpretação pode ser muito complexa, tirando a vantagem trazida pela visualização. Um dos objetivos do trabalho proposto neste artigo é superar algumas destas limitações, provendo interatividade e utilizando técnicas de visualização tridimensionais e baseadas em ícones a fim de produzir um resultado mais compreensível.

Por exemplo, em um dos trabalhos já citados [10], é proposta a visualização de comportamentos de *malware* através de *treemaps*, mostrando a frequência e distribuição das ações maliciosas capturadas. Entretanto, ainda existe o excesso de dados e a falta de interatividade. Para resolver o problema do excesso de dados, a proposta deste artigo é visualizar apenas as ações que causam mudanças em um sistema alvo. Quanto a falta de interatividade, foi proposto um gráfico de comportamento em espiral, representando todas essas atividades escolhidas de forma temporal e que pode ser aumentado, rotacionado e ter ícones específicos selecionados de forma a detalhar a ação. Estas características serão explicadas na seção a seguir.

3. Descrição da ferramenta

A ferramenta de visualização proposta tem como objetivo principal receber um arquivo de comportamento e exibir as informações contidas nele de uma forma interativa por meio de um gráfico em três dimensões no formato de uma espiral como visto na Figura 1. A visualização gráfica em espiral permite uma análise interativa e mais compreensível de dados complexos.

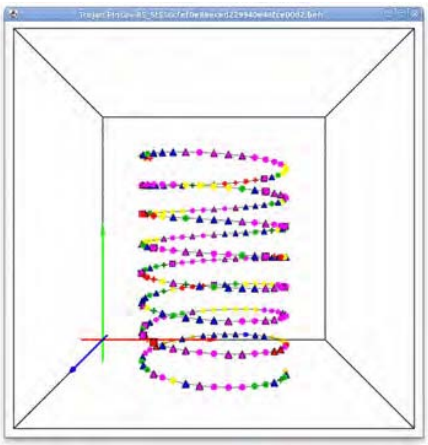


Figura 1. Visão geral do gráfico em espiral

Nota-se que, devido à forma ser em espiral, esta ferramenta visual permite a interpretação de uma grande quantidade de informações, o que seria muito mais difícil através da análise manual, sem o auxílio de *software* de análise específico. Um outro ponto para a escolha visual é poder comparar rapidamente os padrões presentes em comportamentos distintos. Caso a apresentação fosse bidimensional, com os ícones dispostos em uma matriz (como mostrado em [3], pequenas variações nas ações poderiam gerar gráficos bem diferentes para comportamentos muito similares, o que é indesejável na análise de *malware*.

Dentre as principais características da ferramenta, destacam-se:

- A capacidade de manipular livremente o ângulo de visão do gráfico para obter mais detalhes de uma de uma determinada área do gráfico.
- A possibilidade de destacar pontos relevantes do gráfico.
- A flexibilidade em aceitar como entrada diversos tipos arquivos de entrada através da configuração adequada dos parâmetros.
- A facilidade em personalizar características do gráfico criado, como por exemplo o raio da espiral.

3.1. Arquitetura

A arquitetura da ferramenta é dividida em dois módulos: Módulo GUI e o Módulo Visualização. O usuário interage com o Módulo GUI, e este por sua vez encaminha as escolhas do usuário para o Módulo Visualização, que é responsável pela apresentação dos resultados.

3.2. Módulo GUI

O Módulo GUI é uma interface gráfica, conforme pode ser visto na Figura 2, que foi criada através do uso da biblioteca *Swing* da linguagem *Java*.

Figura 2. Interface gráfica criada pelo Módulo GUI

Através do uso desta interface, o usuário fornece as informações a respeito da formatação dos arquivos (*logs*) a serem analisados, bem como determina qual será a representação gráfica das palavras-chave presentes nestes *logs* que serão criadas pelo Módulo Visualização.

A vantagem do uso deste Módulo está na flexibilidade da interpretação dos arquivos de logs genéricos, proporcionando uma melhor filtragem da palavra-chave de interesse, pois somente serão visualizadas na espiral as formas geométricas e cores relacionadas às palavras-chave indicadas pelos usuário. Tanto o formato esperado de um arquivo de comportamento, quanto uma explicação melhor a respeito das palavras-chave estão na Seção 3.3.1

3.3. Módulo Visualização

O Módulo Visualização utiliza a biblioteca *j3d* do *Java*. Esta biblioteca foi escolhida por permitir um rápido desenvolvimento do Módulo, e também por facilitar a implementação da computação gráfica, que por sua vez gera o ambiente em 3D, exibindo assim o gráfico para o usuário.

Ao ser iniciado, o Módulo Visualização executa as seguintes tarefas: recebe os parâmetros do Módulo GUI, cria a cena, renderiza a cena e exibe a imagem do gráfico. A seguir são detalhados os mecanismos que permitem a execução destas tarefas.

3.3.1. Estrutura do arquivo de entrada

A Figura 3 exemplifica uma linha de um arquivo de entrada válido, no qual o caracter separador é o "**;**", **open** é a primeira palavra-chave e **process** a segunda. Estas palavras referem-se, respectivamente, à cor e à forma geométrica. Vale lembrar que a posição das palavras-chave e o caracter separador são escolhidas pelo usuário no Módulo GUI.

%HOMEPATH%\desktop\malware.exe;**open**;**process**;proc.exe

Figura 3. Exemplo de uma linha de um arquivo log válido

Cada ponto no gráfico é composto simultaneamente por uma cor e uma forma geométrica. A cor corresponde a uma palavra-chave e a forma a outra palavra-chave. Portanto, é necessário que cada linha do arquivo log contenha duas palavras-chave para que a composição gráfica seja feita corretamente.

As palavras-chave, no caso específico deste trabalho, são as ações (criar, escrever, remover) e os tipos de subsistema influenciados por estas (arquivo, registros, processos) quando da atividade de um programa malicioso. A Tabela 1 mostra as ações, seus tipos e os respectivos ícones (formas geométricas) e cores que representam tais informações.

3.3.2. Adição de um ponto no gráfico da cena

A biblioteca *j3d* possui algumas poucas fomas geométricas nativas, entre elas o cubo e a esfera. Portanto, para tornar possível o uso de formas não nativas foram criados vários métodos que encapsulam a criação de formas complexas (tais como, a pirâmide e o asterisco utilizados neste artigo) a partir da composição de retas e planos. Além disso, também foi criado um método que encapsula a criação de um objeto "ponto" a partir de uma cor e uma forma geométrica definida.

Com o intuito de facilitar o gerenciamento das informações pelo Módulo Visualização foi criado o objeto "ponto" mencionado, o qual contém todas as informações relevantes a um ponto do gráfico, tais como a cor, a forma geométrica e a linha correspondente do arquivo de entrada. Para definir em qual posição (x, y, z) um ponto será inserido, utilizam-se as fórmulas abaixo:

$$x = \cos(\alpha y)$$

$$z = \sin(\alpha y)$$

Observa-se que uma vez definida a coordenada "y", o resto do vetor posição (x, y, z) também estará definido. A coordenada "y" depende de dois fatores: a linha na qual o ponto corresponde e a constante "α".

Tabela 1. Ações, tipos possíveis de visualização e ícones que as representam.

Action / Type	MUTEX	FILE	PROC	REG	NET
READ					
QUERY					
RECEIVE					
WRITE					
SEND					
CONNECT					
CREATE					
DISCONNECT					
DELETE					
TERMINATE					
RELEASE					

Um ponto referente à enésima linha do arquivo possui a coordenada "y" definida pela fórmula $y = \frac{10n}{\alpha}$, na qual "α" é uma constante escolhida pelo usuário no Módulo GUI, e "n" é o número da linha a qual o ponto se refere.

3.3.3. Criação da cena

Durante o método de criação da cena, cada linha do arquivo de entrada é percorrida. Caso o método encontre um par de palavras-chave, este adiciona um ponto correspondente no gráfico da cena, como já descrito na Seção 3.3.2. Em seguida, são adicionados ao gráfico os detalhes, isto é, os eixos e a curva da espiral.

3.3.4. Renderização da cena

A renderização é o processamento das informações providas na cena para gerar, de fato, a imagem visível ao usuário. A renderização é feita quase que integralmente pelos métodos nativos da biblioteca *j3d*, com exceção de duas classes customizadas: a classe *CanvasOverlay* e a classe *MouseBehavior*.

A classe *CanvasOverlay* estende a classe nativa *Canvas*, e tem como objetivo implementar a capacidade de se escrever texto sobre a camada do plano principal (*canvas*). Isto é feito para mostrar ao usuário informações adicionais sobre um ponto específico no gráfico, conforme ilustrado na Figura 4.

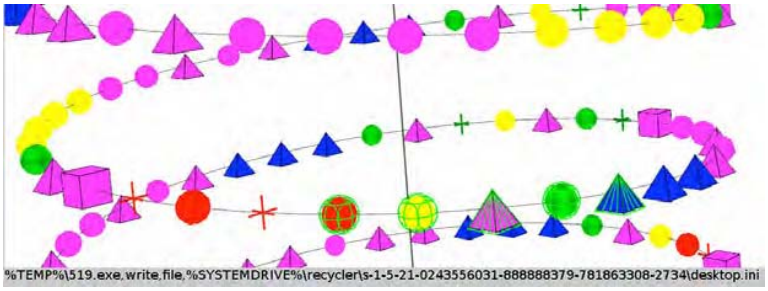


Figura 4. Detalhes das marcações nos pontos (grade verde) e texto inserido através da classe *CanvasOverlay*

A classe *MouseBehavior* estende a classe nativa *Behavior* e tem como objetivo adicionar ao Módulo Visualização a capacidade de reconhecer comandos do *mouse* diretamente sobre a tela, como a posição e o clique do mouse sobre o *canvas*.

Com a classe *MouseBehavior* é possível controlar a câmera com o *mouse* e também criar marcações para destacar pontos do gráfico (Figura 4). As marcações são criadas por métodos implementados dentro da classe *MouseBehavior*. Assim, quando for detectado um clique do *mouse* sobre algum ponto do gráfico, este método irá adicionar ou remover, alternadamente, a marcação correspondente ao ponto.

4. Testes e Resultados

A representação de comportamentos maliciosos envolve diversas categorias de ações e tipos, para os quais foram definidos cores e formas geométricas, com a finalidade de facilitar sua identificação e representação (Tabela 1).

Através do módulo GUI, é possível filtrar as facetas do comportamento que se deseja visualizar. Por exemplo, supondo que o usuário deseje verificar apenas as atividades relacionadas à processos (criação e finalização), este deve escolher somente a caixa de seleção “process”, determinado pela cor verde na Figura 2, desmarcando as outras. Isto faz com que a espiral produzida contenha apenas o tipo de informação selecionada, permitindo uma análise mais detalhada, conforme pode-se observar na Figura 5.

Para fins de teste, foram obtidos os comportamentos de mais de 400 exemplares de malware coletados pela arquitetura apresentada em [4]. Estes exemplares foram submetidos a um sistema de análise dinâmica de *malware* [5], para que fossem extraídos os perfis comportamentais. Os arquivos com comportamentos foram utilizados na geração das espirais, através da ferramenta desenvolvida apresentada neste artigo. É interessante notar que exemplares identificados pelo antivírus ClamAV [1] como pertencentes à família “Allaple” apresentam padrões similares, mesmo quando o comportamento é incompleto. Isto pode ser observado na Figura 6.

Dado que mesmo uma família denominada por um mecanismo de antivírus pode ter variantes diversificadas em diferentes grupos internos, ou sub-famílias, a análise dessas diferenças é um processo relevante na compreensão das atividades maliciosas. Por exemplo, a família de *worms* anteriormente mencionada, conhecida como “Allaple” é bastante popular, contém dezenas de variantes e ainda está em atividade. Os exemplares se caracterizam por realizar atividades de varredura em redes visando atacar outros sistemas e se disseminar.

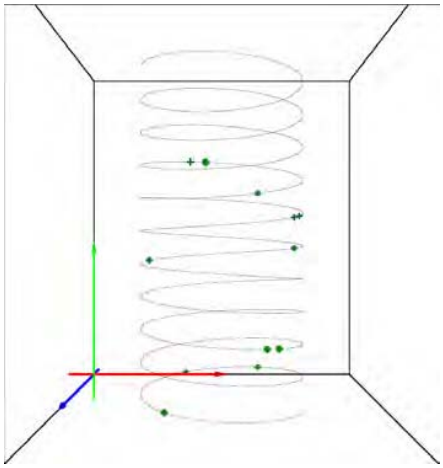


Figura 5. Espiral gerada através da seleção, no módulo GUI, de visualização filtrada por atividades relacionadas aos processos presentes no comportamento

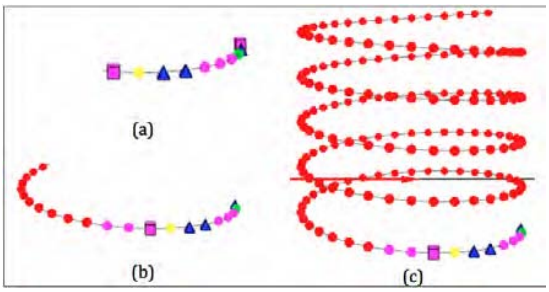


Figura 6. Comportamento de três exemplares da família "Allaple". Em (a), o malware parou sua execução antes de gerar tráfego de rede; em (b), pode-se notar a presença de algum tráfego enquanto que em (c), atividades de rede ocorreram em grande quantidade

Entretanto, podem haver mudanças no comportamento que levem à atividades mais sofisticadas, como downloads ou obtenção de informações sobre as máquinas sondadas em uma varredura. Na Figura 7 é mostrada uma variante de "Allaple" cujo comportamento difere visivelmente das variantes da Figura 6, indicando uma possível sub-família. Além disso, pode-se notar uma alternância entre as atividade de conexão com a rede e criação de arquivos, representadas por esferas vermelhas e rosas, respectivamente.

Em um outro caso, foi possível classificar um exemplar não identificado pela semelhança com a espiral de um cavalo-de-tróia conhecido (identificado pelo ClamAV como uma variante de Trojan.Agent), conforme mostrado na Figura 8. Isto mostra que, visualmente, foi possível detectar um comportamento de um programa até então classificado como inofensivo por um mecanismo antivírus, como sendo malicioso. É interessante notar que o comportamento do programa desconhecido contém mais ações do que as do identificado como um *trojan*, inclusive atividades de rede diversas.

5. Conclusão

Devido ao problema causado pelos programas maliciosos em sistemas de computadores e redes, é necessário criar meios que facilitem a compreensão da atuação destes e a tomada

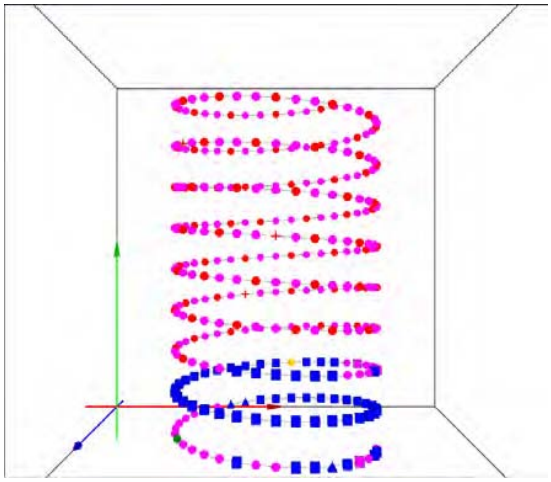
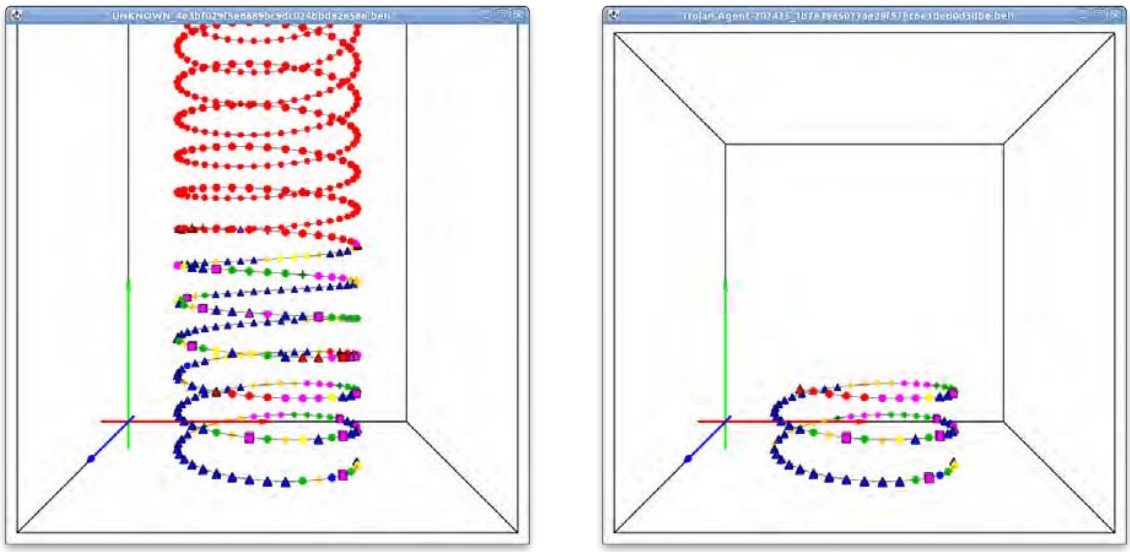


Figura 7. Variante de exemplar de malware da família "Allaple", cujo comportamento predominante envolve a alternância entre as atividades de conexões de rede (esferas vermelhas) e criação de arquivos (esferas rosas)



(a) Exemplar não identificado. (b) *Trojan.Agent*

Figura 8. Exemplar não identificado pelo antivírus ClamAV (a), cujo comportamento inicial é similar ao de um cavalo-de-tróia da classe "Agent" (b)

de medidas de proteção. Técnicas de visualização podem ser aplicadas com sucesso no auxilio à análise de comportamento malicioso, pois permitem a visualização de padrões que poderiam estar ocultos em uma massa muito grande de informações textuais.

Com a finalidade de ajudar na análise de malware, foi desenvolvida uma ferramenta para visualização de comportamento de execução de programas a qual é também interativa, permitindo a um usuário ou analista de segurança a verificação das atividades de forma detalhada e informativa. Esta ferramenta utiliza-se de tecnologias tridimensionais providas por pacotes em Java e apresenta os dados dispostos sob a forma de uma espiral.

Para validar a ferramenta, foram feitos testes que produziram mais de 400 espirais de programas maliciosos e permitiram identificar, visualmente, padrões comuns em famílias (tornando possível seu agrupamento e classificação posterior). Além disso, mostrou-se que é possível associar programas maliciosos não identificados à malware que já é conhecido. Como trabalho futuro, propõe-se uma extensão que permita abrir e manipular diversos arquivos de comportamentos ao mesmo tempo, possibilitando a comparação em paralelo mais rápida de várias instâncias de *malware*.

A fim de disseminar o conhecimento científico e prover transparência, uma versão "beta" da ferramenta está disponível em [7], bem como as figuras de todas as espirais geradas.

Referências

[1] Clam antivirus. `http://www.clamav.net`.

[2] G. Conti, E. Dean, M. Sinda and B. Sangster. Visual Reverse Engineering of Binary and Data Files. *Proceedings of the 5th international workshop on Visualization for Computer Security*(VizSec), 2008, pp. 1-17.

[3] S. G. Eick, J. L. Steffen and E. E. Sumner, Jr. Seesoft—A Tool for Visualizing Line Oriented Software Statistics. In *IEEE Transactions on Software Engineering*, vol. 18, no. 11, pp. 957-968, 1992.

[4] A. R. A. Grégio, I. L. Oliveira, R. D. C. dos Santos, A. M. Cansian and P. L. de Geus. Malware distributed collection and pre-classification system using honeypot technology. *Proceedings of SPIE* , vol. 7344, pp. 73440B-73440B-10, 2009.

[5] D. S. Fernandes Filho, V. M. Afonso, A. R. A. Grégio, R. D. C. dos Santos, M. Jino and P. L. de Geus. Análise Comportamental de Código Malicioso através da Monitoração de Chamadas de Sistema e Tráfego de Rede. *Anais do X Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais* (SBSeg), pp. 311-324, 2010.

[6] D. Keim. Visual Data Mining, Tutorial. In *23rd International Conference on Very Large Data Bases (VLDB '97)*, 1997. Visitado em Agosto de 2011.

[7] Malicious Behavior Spiral. `http://www.las.ic.unicamp.br/~gregio/mbs`

[8] D. Quist and L. Liebrock. Visualizing Compiled Executables for Malware Analysis. *Proceedings of the Workshop on Visualization for Cyber Security*, 2009, pp. 27-32.

[9] H. Read, K. Xynos and A. Blyth. Presenting DEViSE: Data Exchange for Visualizing Security Events. *IEEE Computer Graphics and Applications*, vol. 29,pp. 6-11, 2009.

[10] P. Trinius, T. Holz, J. Gobel and F. C. Freiling. Visual analysis of malware behavior using treemaps and thread graphs. *International Workshop on Visualization for Cyber Security*(VizSec), 2009, pp. 33-38.