Modelagem de um Sistema Distribuído orientado à Agentes para Coleta e Análise de Código Malicioso da Internet

André R. A. Grégio, *RA079779, FEEC – Unicamp*

***Abstract*—Atualmente, a maior ameaça aos sistemas computacionais são os códigos maliciosos (malware) advindos da Internet, como os virus, worms, Trojans, bankers e bots. A facilidade de disseminação e modificação destes códigos de forma a enganar os mecanismos antivirus gerou uma grande demanda de coleta – a fim de catalogar as inúmeras variantes de um determinado tipo de malware – e de análise – a fim de categorizar comportamentos maliciosos e auxiliar no desenvolvimento de vacinas. Para lidar com a complexidade de um sistema distribuído de coleta e análise, é apropriado o uso de agentes, dos mais variados tipos, em cada uma das etapas necessárias para o funcionamento de tal sistema. Neste trabalho apresenta-se uma proposta de modelagem de um sistema distribuído de coleta e análise de malware baseado em agentes, detalhando-se os tipos de agentes utilizados e as interações realizadas para o cumprimento da tarefa.**

***Index Terms*—Agentes computacionais, códigos maliciosos, segurança de sistemas, sistemas distribuídos.**

I. Introdução

AGENTES computacionais podem ser muito versáteis para resolver uma infinidade de problemas para os humanos, sendo sofisticados o bastante para lidar com ambientes hostis (que oferecem risco de vida) ou complexos (como redes de computadores) [1]. Essa abrangência de aplicações envolve desde a atuação em assistentes pessoais, passando pela busca paralela de informações, mineração de dados, manufatura e sistemas de produção, controle de tráfego aéreo, até o gerenciamento de redes [2]. Neste ultimo, agentes colaborativos podem obter e intercambiar informações sobre hardware de rede e carga dos sistemas a fim de promover uma utilização otimizada de recursos e melhor administração.

Na área de segurança de sistemas de informação, os agentes podem ser utilizados na automatização de tarefas frequentes de coleta e análise de dados, de modo a melhorar a rapidez de resposta em casos de ataques, provendo estatísticas, identificando ameaças, analisando grandes massas de dados ou observando tendências. Isto auxilia na tomada de decisões na ocorrência de incidentes e pode, inclusive, ajudar na prevenção da disseminação de ataques.

Artigo apresentado para a disciplina IA009 (Introdução à Teoria de Agentes) ministrada pelo Prof. Dr. Ricardo Gudwin em junho de 2010 na FEEC – Faculdade de Engenharia Elétrica e de Computação / Unicamp.

O grande problema de segurança de sistemas hoje é a profusão de código malicioso – conhecido por *malware* – através da Internet. Eles podem ser de diversas classes, cada qual com uma determinada característica quanto ao seu comportamento malicioso. Algumas das principais classes de *malware* e suas respectivas descrições são, segundo [3]:

- Vírus: um pedaço de código que replica recursivamente uma cópia possivelmente evoluída de si mesmo, infectando um arquivo ou área do sistema, ou simplesmente modificando uma referência a tais objetos de modo a tomar o controle e multiplicar-se para formar novas gerações;
- *Worm*: um tipo de código malicioso que se replica primariamente em redes de computadores, se executando automaticamente em uma máquina remota. Tipicamente, *worms* são aplicações *standalone* sem a necessidade de um programa hospedeiro;
- *Trojan Horses*: programas que tentam apelar para a curiosidade ou interesse do usuário com algum conteúdo chamativo ou funcionalidade útil de forma a levar o usuário a executá-lo;
- *Backdoor*: também chamado de porta dos fundos, pode ser tanto uma ferramenta para fins maliciosos (geralmente permitindo conexões remotas a sistemas), quanto uma falha de implementação em um programa;
- *Downloader*: um programa malicioso que instala um conjunto de outros itens em uma máquina sob ataque, como por exemplo, conteúdo malicioso proveniente de sites Web;
- *Keyloggers*: programa para capturar pressionamentos de teclas em sistemas comprometidos, obtendo assim informações sensíveis como senhas, números de cartão de crédito e outras informações bancárias;
- *Spammers*: são programas utilizados para enviar mensagens não solicitadas para grupos de notícias, *Instant Messaging* ou outros tipos de dispositivo móvel em forma de mensagens de email ou SMS;
- *Rootkits*: um conjunto especial de ferramentas de ataque que são utilizadas após um ataque com sucesso em um sistema e a obtenção de privilégios administrativos. Normalmente ocultam-se do usuário e modificam o comportamento do sistema

comprometido.

Estatísticas recentes do CERT.br (Centro de Estudos, Resposta e Tratamento de Incidentes de Segurança do Brasil) [REF] mostram que a grande maioria de incidentes reportados (cerca de 87%) está, ou pode estar, relacionado à atuação de código malicioso. A Fig. 1 mostra a porcentagem de incidentes reportados ao CERT.br entre janeiro e março de 2010, separados por tipo de ataque.

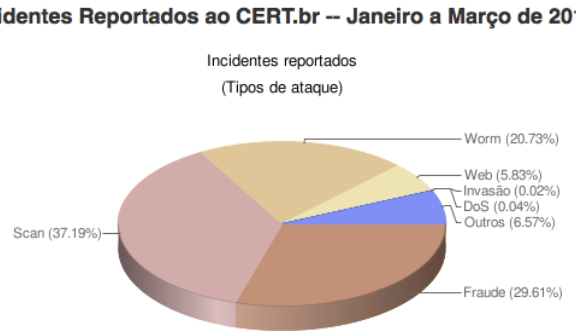

Fig. 1. Incidentes reportados ao CERT.br entre janeiro e março de 2010 divididos por tipos de ataque. Os ataques que podem ser relacionados à atuação de código malicioso são, principalmente, *Scan* (37.19%), *Worm* (20.73%) e *Fraude* (29.61%), totalizando 87.53%.
Fonte: http://www.cert.br/stats/incidentes/2010-jan-mar/tipos-ataque.html

Na figura acima pode-se observar que a soma dos incidentes identificados como *scan* (varredura de sistemas em busca de vulnerabilidades), *fraude* (tentativa de roubo de alguma informação do usuário ou do sistema, especialmente bancárias) e *worm* (códigos maliciosos que se propagam de modo automatizado) corresponde à maior parte dos ataques, e todos são relacionados a atividades executadas por *malware*.

A compreensão das ações efetuadas por *malware* em um sistema auxilia na recuperação de sistemas comprometidos, na detecção de ataques e na prevenção da futura atuação deste mesmo *malware* ou de algum exemplar com ações similares. Para isto, é necessário gerar inteligência sobre os comportamentos de *malware* por meio da análise e execução destes.

Uma vez que o processo de análise esteja bem definido, os procedimentos tornam-se repetitivos, e, dado o número de passos para se efetuá-lo por completo, o uso de agentes torna-se indispensável para liberar o analista humano das tarefas.

Assim, este trabalho visa mostrar algumas soluções de sistema para análise automatizada de análise de *malware*, propor uma arquitetura distribuída de análise baseada em agentes e mostrar como os agentes devem ser implementados para o sistema entrar em produção.

II. Revisão da Literatura

Sistemas distribuídos que envolvem a interação de diversos componentes diferentes em busca da solução de algum problema podem ser difíceis de implementar. A utilização de agentes facilita muito na questão de compartimentalizar as tarefas de maneira específica, paralelizando-as e possibilitando até mesmo a busca de maneiras inteligentes para otimizar o processo. Nesta seção serão discutidas algumas abordagens que representam o estado da arte atual na coleta e análise de malware, bem como serão descritos os tipos de agentes utilizados para o funcionamento do sistema.

Existem diversas abordagens para coleta e análise de *malware* disponíveis publicamente através da Internet. Nessas abordagens, a coleta é feita principalmente através da submissão via Web de amostras que os usuários desejam analisar e, em contrapartida, o sistema devolve um relatório com o resultado da análise. Dentre os sistemas citados, os mais conhecidos são o Anubis [4], o CWSandBox [5] e o JoeBox [6]. O objetivo final de todos eles é analisar código malicioso, cada qual com soluções próprias para realizar a análise estática ou dinâmica. A seguir, uma breve descrição de funcionamento de cada um deles.

- Anubis: trata-se de uma ferramenta para analisar dinamicamente o comportamento de executáveis de Windows XP. Para isso, o binário é executado em ambiente emulado e suas ações são monitoradas. Esses binários são submetidos para o sistema via Web, enviados para análise e, após geração de resultados, um relatório é gerado, em um processo seqüencial.
- CWSandBox: é uma ferramenta similar a Anubis, monitorando as chamadas de sistema feitas por *malware* em sistemas operacionais Windows a fim de gerar um relatório de análise. A diferença está na utilização de tecnologia de virtualização.
- JoeBox: Plataforma de análise de executáveis de Windows XP e Vista cujo diferencial é o uso de sistemas operacionais reais em arquitetura cliente-servidor.

Neste trabalho, o objetivo é propor uma arquitetura de coleta e análise baseada em agentes para completar, de maneira inteiramente automatizada, a análise de um dado *malware*. Com isto, aproveita-se dos conceitos de autonomia, flexibilidade e reatividade dos agentes para resolver uma tarefa complexa de modo mais fácil e eficiente.

Há vários tipos de agentes que podem ser considerados para cada tarefa, e cada um tem uma determinada classificação. A Tabela 1 contém uma classificação de agentes baseada nas propriedades destes, a qual foi retirada de [7].

TABELA I
CLASSIFICAÇÕES DE AGENTES

Propriedade	Outros Nomes	Significado
Reativo	Sensor e atuador	Responde em tempo à mudanças no ambiente.
Autônomo		Exercita o controle sobre suas próprias ações.
Orientado à metas	Com propósito proativo	Não atua simplesmente em resposta ao ambiente.
Temporalmente contínuo		É um processo continuamente em execução.
Comunicativo	Socialmente capaz	Comunica-se com outros agentes; pode se comunicar com pessoas.
De Aprendizado	Adaptativo	Muda seu comportamento conforme experiências prévias.
Móvel		Capaz de se transportar de uma máquina a outra.
Flexível		Suas ações não seguem roteiro.
Personagem		Personalidade crível e estado emocional presente.

Na seção IV, os agentes utilizados na arquitetura proposta adiante serão detalhados de acordo com sua classificação, bem como discutir-se-á as vantagens do uso desses agentes dadas suas propriedades inerentes. A seguir, detalha-se a arquitetura modelada para que se compreenda o escopo do sistema e onde os agentes serão aplicados.

III. Arquitetura do Sistema

Nesta seção será mostrada uma visão geral do sistema, detalhando-se cada componente da arquitetura projetada de acordo com sua funcionalidade e com o tipo de agente utilizado. Tal arquitetura está ilustrada na Fig. 2 e consiste dos seguintes componentes: Módulo de Captura de Spam, Módulo de Captura por Ataques, Módulo de Coleta de Malware, Módulo de Análise (Estática e Dinâmica) e Módulo de Gerenciamento.

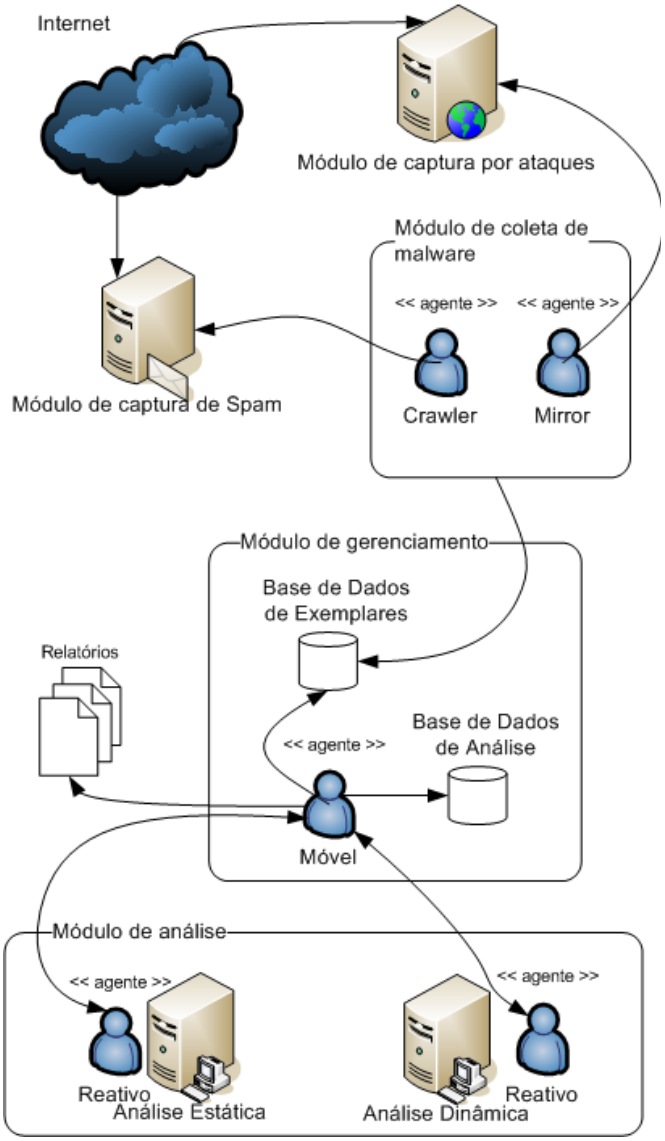

Fig. 2. Arquitetura proposta para o sistema distribuído de coleta e análise de malware orientado à agentes.

A dinâmica de entrada e saída de dados do sistema funciona da seguinte maneira:

- Existe uma conta de e-mail em um servidor de e-mail institucional a qual não é difundida nem faz parte de listas e, portanto não deve receber nenhum e-mail. Desta forma, qualquer e-mail recebido nesta conta é, por definição, não solicitado – também conhecido como Spam. Além disso, para esta conta são redirecionados os Spams recebidos em contas de e-mail particulares, os quais podem conter anexos maliciosos ou links para download de conteúdo malicioso.
- Há também sensores de segurança instalados em algumas redes cujos endereços não são publicados, fazendo com que qualquer acesso seja um ataque em potencial. Estes sensores emulam vulnerabilidades conhecidas em sistema operacional Windows XP e, ao receber ataques, tentam efetuar o download da ferramenta, em geral um código malicioso.
- Os e-mails são processados por um agente, o qual retira os anexos e busca por binários nos links e os insere em uma base de dados. Em paralelo, outro agente faz *polling* nos sensores e, a cada novo binário recebido, o busca e insere na base de dados de exemplares.
- Os exemplares recebidos e armazenados, caso ainda não tenham sido analisados, são remetidos para análise dinâmica e estática em analisadores ociosos, e os dados da análise são inseridos na base de dados de análise. Caso todos os analisadores estejam ocupados, é formada uma fila.
- A partir dos dados de análise, é gerado um relatório com as informações da atuação do malware, permitindo uma avaliação do dano e a comparação com outros códigos, identificando-o.

Cada etapa do fluxo é feita por um modulo e seu respectivo agente. Esses módulos serão explicados nas subseções a seguir.

A. *Módulo de Captura de Spam*

O módulo de captura de Spam opera a partir de um agente de e-mail, para o qual são passadas as informações do servidor de e-mail (POP ou IMAP), conta e senha. Este agente fica em estado dormente no Módulo de Coleta de Malware e periodicamente acessa o servidor de e-mail em busca de mensagens (marcadas como não lidas). Ao serem acessadas, as mensagens são marcadas como lidas no servidor a fim de não serem processadas novamente.

Quando há mensagens não lidas, o agente executa uma *thread* para analisá-las e volta à dormência, aguardando o próximo período de acesso ao servidor de e-mail.

As mensagens de e-mail obtidas são analisadas, uma a uma,,e se separa o corpo da mensagem e os possíveis anexos. O agente procura por links no corpo da mensagem e os acessa, em busca de arquivos. É feito o download destes arquivos em um diretório temporário e é avaliado se são arquivos executáveis. Caso sejam, o agente conecta-se ao banco de dados de exemplares e insere o arquivo, de acordo com seu *hash sha-1* [8]. Se o *hash* já estiver no banco é porque o arquivo já foi obtido anteriormente e este é então descartado, juntamente com os outros arquivos que não são executáveis.

Os links no corpo do e-mail não são seguidos em profundidade, apenas efetua-se o download do link direto disponível.

O mesmo processo é feito com os anexos: verifica-se se o arquivo é um binário executável e, se verdadeiro, insere-o no banco. Se falso, o arquivo é descartado. Passa-se então à próxima mensagem, até que não hajam mais e-mails para se analisar. Neste ponto, esta *thread* é terminada.

B. *Módulo de Captura por Ataques*

Este módulo é dividido em duas partes: um sensor de ataques e o agente que obtém os exemplares coletados pelo sensor.

O sensor é implantado a partir de um *honeypot* de baixa interatividade conhecido como *Dionaea* [9]. *Honeypots* são sistemas computacionais com serviços reais ou emulados contendo mecanismos para captura e *logging* de atividades, com a finalidade de serem sondados, atacados e comprometidos [10]. O objetivo disso é monitorar, estudar e compreender as técnicas e motivações dos atacantes bem como obter as ferramentas de ataque utilizadas.

A *Dionaea* é um *honeypot* com o intuito específico de emular vulnerabilidades remotas conhecidas de sistemas operacionais Windows, possibilitando o ataque através da rede e o *download* do código malicioso presente na efetivação do ataque. Quando há o sucesso no ataque e no download do malware, este é armazenado em um diretório chamado *binaries*.

O agente responsável por esse módulo, também despachado pelo Módulo de Coleta de Malware, monitora periodicamente o sensor *Dionaea* e, caso hajam binários no diretório remoto *binaries*, efetua o *download* destes e insere os não repetidos no banco de dados de exemplares. Estes binários são então removidos do sensor para liberação de espaço.

C. *Módulo de Coleta de Malware*

O Módulo de Coleta de Malware é composto de uma máquina com grande capacidade de armazenamento para os binários, a qual hospeda dois agentes, um para captura de Spam e outro para captura de binários obtidos de ataques efetuados no sensor de ataques.

Neste modulo são disparados os agentes de Spam e de Captura de Ataques, e é onde é efetuado o processamento dos arquivos temporários, o armazenamento dos binários inéditos obtidos e inserção dos dados no banco de dados de exemplares, situado no Módulo de Gerenciamento.

D. *Módulo de Análise*

Este módulo tem por finalidade analisar o *malware* estática e dinamicamente. Ele consiste de um agente estacionário que aplica um determinado conjunto de ações sobre o *malware* para fins da análise estática, a qual visa obter informações do binário sem executá-lo, e de um agente controlador de um *pool* de máquinas virtuais para a análise dinâmica, que executa o binário para verificar seu comportamento.

Todo *malware* novo que é inserido no banco de dados de exemplares e que ainda não foi analisado é enviado ao módulo de análise, por intermédio do Módulo de Gerenciamento. Este *malware* é então, inserido em duas filas, a de análise estática e a de análise dinâmica. As duas análises ocorrem em paralelo, mas, dado que a análise estática é mais rápida, o mesmo exemplar pode sair de uma fila e ainda estar na outra.

Para a análise estática, são retiradas algumas informações gerais para identificá-lo, tais como:

- seus *hashes* (MD5, SHA-1 e SHA-256);
- tamanho do arquivo
- se está cifrado com algum *packer* conhecido;
- se é identificado por algum antivírus.

Já para a análise dinâmica, primeiro é verificado se há máquinas ociosas no *pool* de máquinas virtuais. Caso haja ociosidade ou a fila dinâmica for menor que um determinado limiar, o *malware* é colocado na fila. Cada máquina virtual ociosa acessa um *malware* e o obtém para análise, retirando-o da fila.

Esse *malware* é então executado dentro da máquina virtual, cujo sistema operacional é monitorado por uma ferramenta que observa mudanças no sistema de arquivos. Tais mudanças são registradas em um arquivo, bem como o tráfego de rede gerado pela máquina durante a execução do referido *malware*.

A execução do *malware* é limitada a um período de tempo curto (cerca de 3 minutos), a fim de se capturar o comportamento inicial deste sem deixá-lo executar muito tempo.

Após o fim de cada análise, o agente responsável por esta análise conecta-se ao banco de dados de análise, no Módulo de Gerenciamento, e insere os dados obtidos.

E. *Módulo de Gerenciamento*

Este modulo é responsável pelo banco de dados de análise e pelo banco de dados de exemplares, bem como por enviar os exemplares de *malware* para o Módulo de Análise e por fim, gerar os relatórios de cada um deles, após a análise completar.

Nele, há um agente cuja função é, de tempos em tempos, verificar os exemplares de código malicioso que ainda não foram analisados e enviá-los ao Módulo de Análise. No caso dos exemplares já analisados, o agente verifica se há relatório disponível, senão ele gera o relatório correspondente.

Este agente verifica as filas no Módulo de Análise, controlando a transferência de exemplares e, quando necessário, serializa a transmissão dos binários via rede para aguardar a análise.

IV. AGENTES UTILIZADOS

Estudando-se a arquitetura proposta, nota-se a vantagem clara da aplicação de agentes para interagir com os diversos módulos do sistema. Alguns atributos importantes de agentes são especificamente úteis para este trabalho, como rastreabilidade, robustez, auto-gerenciabilidade, reatividade, continuidade, comunicabilidade, flexibilidade e mobilidade, conforme visto em [11]. A seguir será apresentada uma breve discussão dos atributos citados no escopo da arquitetura modelada.

O atributo da rastreabilidade permite que se monitore se os agentes estão em operação e o que estão fazendo, se estão

ocupados e qual o tamanho da fila para uma determinada tarefa de análise, enquanto que o da robustez auxilia nos momentos em que os exemplares de *malware* não gerarem comportamento algum, ou algum erro faça com que alguma outra etapa da análise não gere informação para ser inserida nas bases de dados ou nos relatórios.

A auto-gerenciabilidade possibilita a cada agente gerir seu próprio ciclo de vida, decidindo se irá dar uma análise por terminada, enfileirar um *malware* ou ficar ocioso. No caso da reatividade, o surgimento de novos exemplares para análise, saída de *malware* da fila ou finalização de alguma atividade de módulo que seja requisito para o agente operar faz com que este reaja prontamente e execute suas funções, dentro de seu ciclo de vida próprio (continuidade).

Já o atributo da comunicabilidade é útil para que o agente possa avisar aos analistas humanos da chegada de novo material para análise, geração de relatórios ou informações sobre o estado do sistema como um todo, auxiliando no gerenciamento em geral.

Por fim, a flexibilidade e mobilidade permitem a um agente o transporte entre módulos, carregando consigo *malware* ou resultados.

Os agentes necessários para a implementação da arquitetura estão detalhados na Tabela 2.

TABELA 2
DEFINIÇÃO DOS AGENTES DA ARQUITETURA POR MÓDULO

Agente	Ambiente	Ciclo Operacional	Objetivo
Internet (*crawler*) [12]	Situado no Mód. de Coleta de Malware, atua sobre o Mód. de Captura de Spam e Mód. de Gerenciamento.	De hora em hora, acessa o servidor de e-mail, coleta os malware e insere no banco de exemplares.	Implícito – varre e-mails em busca de malware em links ou anexos.
Internet / Reflexivo (*mirror*)	Situado no Mód. de Coleta de Malware, atua sobre o Módulo de Captura de Ataques e de Gerenciamento.	De hora em hora, acessa o sensor de segurança e sincroniza os novos malware com o banco de exemplares.	Implícito – verifica se há malware inédito no sensor e o transfere.
Reativo / Comunicativo	Situado no Mód. de Análise, aguarda *input* do Mód. de Gerenciamento	Aguarda por interação do agente de Gerenciamento e, sob demanda, enfileira os *malware*, realiza a **análise estática/dinâmica** e a insere no banco de análises.	Implícito – recebe sinal de um agente externo e executa sua atividade.
Móvel / Reativo / Comunicativo	Situado no Módulo de Gerenciamento	Verifica (*polling*) o banco de dados de exemplares, transporta os exemplares novos para o Módulo de Análise e avisa os agentes de análise que há material de trabalho. Recebe os dados e insere no banco de análises.	Implícito – faz checagem por *malware*, transporta-os via rede, comunica-se com outros agentes e gera relatórios de análise para o usuário.

A linguagem escolhida para implementação dos agentes foi Java, pois além de ser orientada à objetos, o que é ideal para o desenvolvimento de agentes, provê vários mecanismos para comunicação entre processos, contém toda a infra-estrutura para programação *multi-thread* e é portável.

V. RESULTADOS E CONSIDERAÇÕES FINAIS

A arquitetura modelada do sistema distribuído orientado a agentes neste trabalho mostra a viabilidade do uso de vários tipos de agentes executando em paralelo, sob demanda e/ou interagindo entre si com o objetivo comum de resolver um problema complexo.

Um protótipo do sistema foi desenvolvido para analisar *malware* em sistema operacional Windows XP contido em um emulador (QEMU [13]), sobre plataforma-base Linux. A Fig. 3 mostra o resultado do Agente de Análise Estática, após relatório gerado pelo Agente de Gerenciamento.

```
ANÁLISE ESTÁTICA

MD5: ff6b5e9d994730d8be026e8fba19b5c8
SHA-1: 6bed294c55bb70f1ac643a80784417abae248e38
SHA-256: 1cf9d39ecc403ccc2508cec885a0cb42dc5ee8deb67aacb1ff24d33ce507a700
Tamanho: 1.5M
Packer: Xtreme-Protector v1.05
Antivírus: Trojan Horse
```

Fig. 3. Informações gerais do *malware* coletadas pelo Agente de Análise Estática e organizadas pelo Agente de Gerenciamento.

Na Fig. 4 é possível observar um trecho do comportamento apresentado pelo *malware* após a execução pelo Agente de Análise Dinâmica e geração de relatório pelo Agente de Gerenciamento.

```
ANÁLISE DINÂMICA

1: "C:\malware.exe created process C:\WINDOWS\explorer.exe"
2: "C:\malware.exe terminated process C:\WINDOWS\explorer.exe"
3: "C:\malware.exe SetValueKey registry HKCU\Software\Microsoft\Windows\
     CurrentVersion\Explorer\Shell Folders\Cache"
4: "C:\malware.exe SetValueKey registry HKLM\SOFTWARE\Microsoft\Windows\
     CurrentVersion\Internet Settings\Cache\Paths\Directory"
[...]
109: "C:\malware.exe terminated process C:\WINDOWS\system32\cmd.exe"
110: "C:\WINDOWS\system32\javax.exe Write file C:\WINDOWS\Atualizada"
111: "C:\WINDOWS\explorer.exe terminated process C:\WINDOWS\system32\cmd.exe"
```

Fig. 4. Informações de comportamento coletadas pelo Agente de Análise Dinâmica e organizadas pelo Agente de Gerenciamento.

A implementação de um sistema de coleta e análise de *malware* baseada nesta arquitetura tem muitas vantagens sobre os demais sistemas disponíveis, dentre as quais pode-se citar:

- Com agentes distintos para realizar as tarefas de análise estática e dinâmica, resultados parciais podem ser obtidos mais rapidamente, de acordo com a necessidade do usuário (ou do analista). Como as informações de análise estática são geradas mais rapidamente pelo agente específico, obtém-se informações sobre se o exemplar é identificado por antivírus ou está cifrado com algum *packer*, o que já o classifica como *malware*. Desta forma, não é necessário aguardar o fim da análise dinâmica para saber se determinado executável é malicioso.
- No que diz respeito à flexibilidade do sistema, para escalá-lo, caso sejam necessárias mais análises sendo

feitas ao mesmo tempo, basta que se inicialize novos agentes nos módulos respectivos;

- O fato de o sistema de coleta ser automatizado por um agente de coleta e um de *crawling* (*honeypot* e e-mail) permite que o processo inteiro, do início (submissão) ao fim (geração de relatório) seja feito sem interação humana, diminuindo a carga de trabalho;
- A arquitetura permite que se adicione novos módulos e novos tipos de agentes, como por exemplo, um agente inteligente para efetuar mineração de dados nas informações coletadas e armazenadas no banco de dados de análise.

REFERÊNCIAS

[1] J. Hendler, "Intelligent Agents: Where AI Meets Information Technology," *IEEE Expert Systems*, pp. 20–23, Dez. 1996.

[2] M. T. C. da Costa, "Uma Arquitetura Baseada em Agentes para Suporte ao Ensino à Distância," Tese de Doutorado, Depto. Eng. De Prod. E Sist., Univ. Federal de Santa Catarina, Florianópolis, Brasil, 1999.

[3] P. Szor, *The Art of Computer Virus Research and Defense*. New Jersey: Addison-Wesley Professional, 2005.

[4] U. Bayer, C. Kruegel, E. Kirda, "TTAnalyze: A Tool for Analyzing Malware," *15th EICAR Annual Conference*, Hamburg, Germany, 2006.

[5] T. Holz, F. Freiling, C. Willems, "Toward Automated Dynamic Malware Analysis using CWSandBox," *IEEE Security & Privacy*, v. 5, n. 2, pp. 32–39, Mar. 2007.

[6] S. Buehlmann, "JoeBox – Analyse your Malware on Windows Simply and Quickly," *disponível em: http://www.joebox.org*.

[7] S. Franklin, A. Graesser, "Is it an Agent, or just a Program?: A Taxonomy for Autonomous Agents," *Proc. of 3rd International Workshop on Agent Theories, Architectures and Languages*, Springer-Verlag, 2006.

[8] FIPS, "Secure Hash Standard," Federal Information Processing Standards Publication 180-1, Abr. 1995. *Disponível em: http://www.itl.nist.gov/fipspubs/fip180-1.htm*.

[9] THP, "Dionaea – catches bugs," The Honeynet Project, 2009. *Disponível em: http://dionaea.carnivore.it*

[10] L. Spitzner, *Honeypots: Tracking Hackers*, Boston: Pearson Education, 2002.

[11] R. Gudwin, "IA009 – Introdução à Teoria de Agentes," Notas de aula, 2010. *Disponível em:* http://www.dca.fee.unicamp.br/~gudwin/courses/IA009/Aulas.html

[12] M. Schrenk, *Webbots, Spiders, and Screen Scrapers: A Guide to Developing Internet Agents with PHP/CURL*. San Francisco: No Starch Press, 2007.

[13] F. Bellard, "QEMU, a Fast and Portable Dynamic Translator," *Proc. of Usenix 2005 Annual Technical Conference*, *FREENIX Track*, pp. 41–46, 2005.

André R. A. Grégio é Doutorando em Engenharia da Computação na FEEC/Unicamp. Obteve o título de Mestre em Computação Aplicada pelo INPE/MCT (São José dos Campos/SP) na área de redes e segurança de sistemas de informação em 2007 e o de Bacharel em Ciência da Computação pela UNESP (São José do Rio Preto/SP) em 2005.

É Tecnologista Pleno no Centro de Tecnologia da Informação Renato Archer (CTI/MCT) em Campinas, onde realiza Pesquisa e Desenvolvimento na Divisão de Segurança de Sistemas de Informação (DSSI).