Implicações da ofuscação de código no desenvolvimento de detectores de código malicioso

Davidson Rodrigo Boccardo[1], Aleardo Manacero Júnior[2]
José Nelson Falavinha Júnior[1]

[1]DEE - Departamento de Engenharia Elétrica - FEIS
Universidade Estadual Paulista - Ilha Solteira - SP

[2]DCCE - Departamento de Ciências de Computação e Estatística - IBILCE
Universidade Estadual Paulista - São José do Rio Preto - SP

{flitzdavidson,junior.falavinha}@gmail.com, aleardo@ibilce.unesp.br

Abstract. *In this computer-interconnected world a malware can have disastrous effects. Malware detection is, therefore, an important task and should achieve high hit ratios. This, however, is not the truth when code obfuscation techniques are used. In this work it is performed an evaluation on the performance of the detection tools in the detection of obfuscated codes, aiming the identification of their flaws. The results achieved indicate future challenging research tracks that should provide efficiency improvements for the malware detection tools in presence of obfuscated codes.*

Resumo. *No mundo de computadores interconectados um código malicioso pode causar efeitos desastrosos. A detecção de códigos maliciosos é importante e deveria atingir padrões elevados de acerto. Isso não é verdade quando se usam técnicas de ofuscação. Nesse trabalho se faz a avaliação do desempenho de ferramentas de detecção frente a códigos ofuscados, procurando identificar suas falhas. Os resultados obtidos indicam os grandes desafios de pesquisa para a melhoria da eficiência das ferramentas de detecção na presença de ofuscações de código.*

1. Introdução

Segurança em computadores é atualmente um assunto extremamente importante para as pessoas, negócios e governo. Programadores de códigos maliciosos escrevem programas com intenção de coletar dados privados, distribuir *spam*, quebrar medidas de segurança, etc. Quando bem sucedidos em sua propagação, esses códigos maliciosos representam um efeito desastroso no mundo dos negócios e nas redes de computadores [L. A. Gordon and Richardson 2004].

O primeiro passo para conter ataques maliciosos é a identificação dos programas maliciosos. Diante disto companhias de antivírus (AV) utilizam de várias técnicas de análise dinâmica e estática para identificar um código malicioso [Zeltser 2001]. A maioria das técnicas usadas em AV dependem do conhecimento das assinaturas dos códigos maliciosos, ou seja, de seus padrões de chamadas de sistema e de suas *strings* conhecidas.

Com o aumento da complexidade, quantidade e heterogeneidade dos sistemas de software, aumenta-se também a dificuldade no reconhecimento dos códigos maliciosos baseados em assinaturas. Para agravar ainda mais essa situação, atualmente os programadores hostis contam com ferramentas avançadas de evasão (comerciais ou disponíveis na comunidade *blackhat*). Essas ferramentas utilizam estratégias como ocultação de pontos de chamada (EPO - *Entry Point Obscuring*) [GriYo 2006], polimorfismo [Skulason 1995] e metamorfismo [Szor and Ferrie 2001] para dificultar a detecção do software malicioso.

Dentro deste cenário a atividade de detecção de softwares maliciosos, principalmente em ambientes de computação global como as grades (*grids*) computacionais hoje em uso, é essencial para obtenção de computação segura. Assim, um dos passos para a melhoria da qualidade de software é a identificação dos pontos fracos no processo de detecção e, a partir deles, construir metodologias que permitam a produção de sistemas efetivamente seguros.

Assim, nesse trabalho se faz a avaliação de detectores frente ao uso de ferramentas de ofuscação de código. A partir dos resultados obtidos são indicados possíveis caminhos para a redução dos pontos falhos. Nas próximas páginas isso é feito apresentando-se inicialmente o estado atual da área, os procedimentos metodológicos adotados nesse trabalho, a descrição dos testes e resultados obtidos, fechando-se com as conclusões e perspectivas pertinentes, principalmente aquelas relacionadas aos grandes desafios de pesquisa em computação.

2. Trabalhos relacionados

Enquanto as companhias de antivírus cresciam no começo da década de 90, os procedimentos de teste e revisão cresceram com a disseminação dos códigos maliciosos e os avanços em algoritmos de detecção. Em 1996, Sarah Gordon [Gordon and Ford 1996] apresentou a seguinte observação:

"A avaliação de detectores de códigos maliciosos não é adequadamente tratada por métodos formais. O processo, portanto, tem sido realizado utilizando-se de uma variedade de ferramentas e métodos"

A maior dificuldade em testar detectores de código malicioso é encontrar um conjunto de programas maliciosos. Motivado por esta dificuldade vários pesquisadores classificaram os programas maliciosos conhecidos (M_{known}), em códigos maliciosos que estão atualmente disseminando e infectando novos alvos, os ITW (*In the wild*) e, os que existem somente em laboratório (Zoo), sendo que:

$$M_{known} = M_{ITW} \cup M_{Zoo}$$

O conjunto M_{ITW} representa programas maliciosos, disseminados como resultado de operações do dia-a-dia e entre computadores dos usuários [Wildlist 2006]. O conjunto M_{Zoo} contem códigos maliciosos conhecidos que não estão atualmente se disseminando, ou porque não estão mais infectando novos sistemas ou porque foram apenas exemplos de laboratório que nunca se disseminaram. Em

1995, Joe Weels criou a WildList, uma listagem de códigos maliciosos reportados em um certo período (atualmente são atualizados mensalmente). A WildList é um importante conjunto de testes para detectores de códigos maliciosos e certificações, como ICSA [ICSA-Labs 2006], Checkmark [West-Coast-Labs 2006] e Virus Bulletin [Virus-Bulletin 2006]). Tais certificações requerem dos detectores a identificação de todos códigos maliciosos em M_{ITW} (baseado na versão atual da WildList) e pelo menos 90% dos códigos maliciosos em M_{Zoo}. Contudo, usar M_{ITW} como conjunto de testes não avalia a eficácia dos detectores diante de ofuscações providas por ferramentas evasivas, como as de proteção de propriedade intelectual [S. Chow and Zakharov 2001, C. Collberg and Low 1998a, C. Collberg and Low 1998b, Wroblewski 2002] ou para subversão de detecção.

Marx [Marx 2000] apresentou um conjunto de técnicas e métodos de teste para detectores, com descrições das métricas relevantes e mostrou as possíveis armadilhas que podem invalidar os resultados. Brunnstein [Brunnstein 2004] realizou testes especificamente em algoritmos de detecção baseados em heurísticas que muitos detectores de códigos maliciosos utilizam. Marx [Marx 2002] estendeu seu trabalho propondo o uso de teste retrospectivo para medir a qualidade das heurísticas. O teste retrospectivo é a execução de antigas versões dos detectores contra novos códigos maliciosos que não eram conhecidos na época que o detector foi lançado. Um outro trabalho é de Mihai e Jha [Christodorescu and Jha 2004] que utilizaram alguns tipos comuns de ofuscação de programas como inserção de código irrelevante, reordenação de código, renomeação de variáveis e empacotamento de código e dados, como técnica de avaliação dos detectores.

Já Brosch e Morgenstern [Brosch and Morgenstern 2006] avaliaram detectores diante de várias ferramentas evasivas e um conjunto pequeno de códigos maliciosos, aproximando-se do executado neste trabalho. As diferenças estão relacionadas ao tamanho e variação do conjunto de softwares maliciosos examinados (maior volume neste trabalho) e do acréscimo de ferramentas evasivas de junção[1], inexistentes no trabalho de Brosch e Morgenstern.

3. Metodologia de avaliação

Para a execução das avaliações aqui apresentadas foram adotados procedimentos experimentais necessários para garantir a segurança do processo, uma vez que se trabalhava com elementos potencialmente destrutívos, e a consistência qualitativa dos resultados experimentais. O modelo para o ambiente de testes é detalhado em 3.1. As métricas adotadas para garantir a consistência dos resultados estão apresentadas em 3.2. Finalmente, em 3.3 são relatadas as ferramentas utilizadas, tanto para a criação de códigos evasivos quanto na detecção de softwares maliciosos.

3.1. Modelo experimental

Uma das restrições em um modelo para análise de código malicioso é a instalação de um sistema dedicado, no qual possam ser usadas as estratégias evasivas para subversão de detecção sem nenhum dano. Sem a criação desse ambiente dedicado,

[1]Ferramenta designada para unir dois tipos de arquivo (EXE+EXE) (EXE+JPEG)

desconetado de qualquer rede, poderia levar a ocorrências sérias de infestação caso o software malicioso fosse acidentamente executado.

Na literatura são propostos dois métodos para a criação desses sistemas dedicados: uso de sistemas reais ou uso de sistemas virtuais. O segundo método cria dificuldades para a verificação de efetividade do software ofuscado, uma vez que determinadas funcionalidades não estariam disponíveis. Isso, entretanto, não é um problema quando o que se quer avaliar é a eficiência das ferramentas de detecção e não a capacidade de infestação do software malicioso. Assim, neste trabalho foi utilizado um sistema virtual desconectado da rede.

Tendo sido configurado um ambiente seguro, a etapa seguinte do modelo consiste na aplicação de ferramentas evasivas, em conjuntos de softwares maliciosos e não maliciosos para a geração dos casos de teste. A eficiência das ferramentas de detecção foi então avaliada, através dos resultados de falsos negativos e falsos positivos obtidos com sua aplicação sobre os códigos ofuscados (maliciosos ou não). O modelo experimental utilizado encontra-se descrito na figura 1.

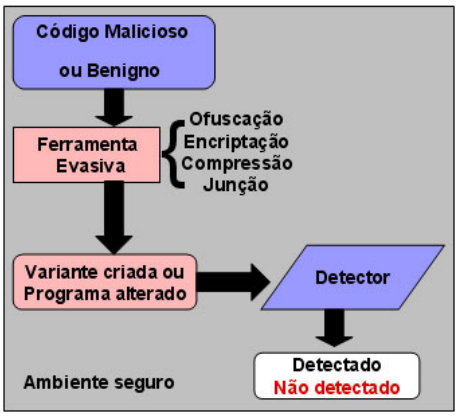

Figura 1. Modelo para avaliação dos detectores de código malicioso

3.2. Métricas para avaliação

Um detector de código malicioso trabalha analisando um objeto de dados (um arquivo, uma mensagem de e-mail ou um pacote de rede) e, determina se esse objeto contém um executável e se é malicioso. Este primeiro teste é geralmente baseado em um método do sistema operacional para descobrir o tipo do dado contido no objeto, que pode ser determinado por cabeçalhos MIME, extensões de arquivo, ou um "número mágico" que é único para um formato de arquivo. Com o uso dessas técnicas pode-se restringir a busca de software malicioso a apenas aqueles objetos que contenham códigos executáveis.

Pode-se definir um detector de código malicioso D como uma função cujo domínio e intervalo são o conjunto dos programas executáveis P e o conjunto *(malicioso, não malicioso)*, respectivamente. Em outras palavras, um detector D é uma função $D : P$! *(malicioso, não malicioso)* definida como:

$$D(p) = \begin{cases} \text{malicioso} & \text{se } p \text{ contêm código malicioso} \\ \text{não malicioso} & \text{caso contrário} \end{cases}$$

A avaliação de um detector D significa fazer interagir todos programas de entrada $p \in P$, verificando a corretude da resposta. Neste contexto, falsos positivos são programas não maliciosos que o detector marca como infectado; e falsos negativos são códigos maliciosos que o detector falha em reconhecer. Também, a taxa de acerto mede a relação entre o número de reais positivos (códigos maliciosos detectados) e o total de códigos maliciosos, desconsiderando assim os códigos não maliciosos, que são usados apenas para controle, pois o usuário pode vir a perder a confiança no detector se o número de falsos positivos for significativo.

Para efeito de avaliação, reduz-se o conjunto P a um conjunto finito (e não de todos os programas existentes, que é virtualmente infinito). Isso permite delimitar de forma exata os subconjuntos de falsos positivos e falsos negativos para um dado conjunto P. Os elementos de um conjunto de teste P_T são classificados em dois subconjuntos disjuntos, um de programas não maliciosos B e outro de programas maliciosos M. A taxa de falsos positivos FP_{P_T}, falsos negativos FN_{P_T}, e a taxa de acerto HR_{P_T} (todos relacionados com o conjunto de testes P_T) são definidos como:

$$FP_{P_T} = \frac{|\ \{\ p\ \in\ B\ :\ D(p)\ =\ malicioso\ \}\ |}{|B|}$$

$$FN_{P_T} = \frac{|\ \{\ p\ \in\ M\ :\ D(p)\ =\ \text{não malicioso}\ \}\ |}{|M|}$$

$$HR_{P_T} = \frac{|\ \{\ p\ \in\ M\ :\ D(p)\ =\ malicioso\ \}\ |}{|M|}$$

A validação dos resultados obtidos ocorreu através de análises de sensibilidade e especificidade em relação ao tamanho dos conjuntos de testes examinados (P_T), adotando-se como critério a obtenção de uma sensibilidade inferior a 4%.

3.3. Testes para seleção de ferramentas

Para a avaliação aqui realizada usou-se dois conjuntos distintos de ferramentas. Um primeiro, o conjunto avaliado, composto por ferramentas comerciais para detecção de software malicioso. O segundo conjunto de ferramentas foi composto por ferramentas para uso em proteção de propriedade intelectual (que tornariam a engenharia reversa mais dispendiosa através da ofuscação de código) e ferramentas disponíveis na comunidade *blackhat* para uso em subversão de detecção.

A escolha de quais ferramentas de evasão e detecção seriam utilizadas no procedimento de avaliação, ocorreu em um processo de seleções sucessivas entre as mais usadas nos dois subgrupos. No processo de seleção foram realizados três pré-testes datados em julho, setembro e novembro de 2006, descritos a seguir. As avaliações foram realizadas sempre com todos os detectores atualizados e avaliados separadamente (somente um detector instalado no sistema por vez, para evitar interferência).

Primeiro pré-teste - julho 2006

Neste teste foram criadas 147 variantes de 22 códigos maliciosos, a partir de 10 ferramentas evasivas (FSG, UPX [UPX 2006], ASPack [ASPACK-Software 2006],

PECompact [Bitsum-Technologies 2007], ASProtect [ASPACK-Software 2006], Armadillo [Silicon-Realms 2007], ExeCryptor [Strongbit-Technology 2007], PEtite [PEtite 2006], fEvicol e PE-Pack). Essas variantes, junto com outras criadas a partir de códigos legítimos (não maliciosos) foram submetidas para exame por sete detectores (ClamAV, Mcafee, Norton, AVG, EZ, BitDefender e Kaspersky). Dentre as ferramentas de evasão, tem-se que FSG, fEvicol e PE-Pack são da comunidade *blackhat*, enquanto as demais são ferramentas comerciais de proteção de código intelectual.

Segundo pré-teste - setembro 2006

Neste teste foram utilizadas as mesmas variantes do primeiro teste, mais 44 variantes criadas através de duas ferramentas de junção da comunidade *blackhat* (os *binders* Mchain e Reductor), totalizando 191 variantes. Elas foram também submetidas para detecção por sete detectores (Mcafee, Norton, AVG, BitDefender, Kaspersky, F-Secure, ESET NOD32). Em relação ao primeiro pré-teste, nesse aqui houve a incorporação das ferramentas de junção e uma alteração nos resultados de detecção provocado por versões atualizadas das ferramentas de detecção (que formaram um subconjunto ligeiramente diferente do anterior).

Terceiro pré-teste - novembro 2006

Neste teste foram utilizadas 433 variantes de códigos maliciosos (agora 32 códigos primários), a partir de 15 ferramentas evasivas (as do segundo pré-teste adicionadas de outras três ferramentas de junção da comunidade *blackhat*, que são simple Binder, Decepction 4.0, MicroJoiner 1.7). Essas variantes foram submetidas a detecção pelos mesmos detectores do segundo pré-teste.

Resultados gerais dos pré-testes

A tabela 1 mostra a taxa de acerto dos detectores diante das variantes geradas nos três primeiros pré-testes. Aqui o objetivo é verificar a confiabilidade de cada ferramenta na presença de código ofuscado. Pode-se notar que as cinco ferramentas que mais detectaram os códigos maliciosos modificados foram Kaspersky, F-Secure, AVG, BitDefender e ESET NOD32.

Tabela 1. Taxas de acerto dos detectores

Ferramenta de Detecção	1º pré-teste % de detecção	2º Pré-teste % de detecção	3º Pré-teste % de detecção
AVG	43,53%	58,63%	51,27%
BitDefender	61,22%	59,16%	49,65%
Kaspersky	63,94%	60,73%	52,19%
ClamAV	27,89%	X	X
Mcafee	63,94%	49,21%	46,65%
Norton	40,81%	41,88%	46,88%
F-Secure	X	60,73%	52,19%
ESET NOD32	X	40,83%	48,96%
eTrust EZ	11,56%	X	X

X = Não foi avaliado neste pré-teste

A tabela 2 exibe as taxas de detecção em cada um dos pré-testes, considerando-se como parâmetro a ferramenta de evasão utilizada. Nela os valores apresentados indicam se as variantes criadas por cada ferramenta foram detectadas por pelo menos uma das ferramentas de detecção. Assim, o valor de 80,64% apresentado no terceiro pré-teste para a ferramenta PECompact2 indica que aproximadamente uma a cada cinco das variantes criadas com ela não foi corretamente identificada por nenhuma ferramenta de detecção. Essa tabela serve, portanto, como indicador da capacidade dessas ferramentas em ocultar a presença de um código malicioso. Nela pode-se notar que as ferramentas mais evasivas foram Armadillo, ExeCryptor, PECompact2 e ASPROTECT.

Tabela 2. Taxas de acerto nos pré-testes diante das ferramentas evasivas

Ferramenta Evasiva	1^o pré-teste % de detecção	2^o Pré-teste % de detecção	3^o Pré-teste % de detecção
ASPACK	100%	100%	100%
ASPROTECT	78,94%	78,94%	95,84%
ExeCryptor	70,58%	70,58%	70,37%
fEvicol	100%	100%	100%
FSG	100%	100%	100%
PECompact2	90,47%	90,47%	80,64%
Armadillo	52,63%	52,63%	49,88%
PE-PaCK	100%	100%	100%
PEtite	100%	100%	100%
UPX	100%	100%	100%
Mchain	X	95,45%	96,99%
Reductor	X	100%	100%
simpleBinder	X	X	100%
Decepction	X	X	100%
MicroJoiner	X	X	100%

X = Não foi utilizado neste pré-teste

Algumas observações interessantes podem ser feitas a respeito dos resultados apresentados na tabela 2. A primeira delas é que as ferramentas da comunidade *blackhat* apresentam pior capacidade de evasão, quando comparadas com ferramentas de proteção de propriedade intelectual. Isso pode ser explicado pelo fato da maioria das ferramentas de detecção fazer a identificação desses códigos a partir da assinatura da ferramenta de evasão, ou seja, identifica a presença da ferramenta e não o código malicioso propriamente dito. Isso leva a segunda observação, que é o fato de a imensa maioria de falsos positivos ter ocorrido nos casos em que se identificava apenas a presença da ferramenta de evasão (tabela 3). Para o teste de falsos positivos foram utilizados oito programas não maliciosos do ambiente Windows (calculadora, prompt de comando DOS, terminal, discador, bloco de notas, wordpad, paint, tetris).

Ferramentas de evasão

Para a execução de testes mais completos, com uma quantidade significativamente maior de códigos maliciosos, foi necessário reduzir-se a quantidade de

Tabela 3. Taxas de falsos positivos

	Propriedade intelectual	*blackhat* Compressor	*blackhat* Junção	*blackhat* Junção	*blackhat* Junção
Ferramenta de Detecção	Armadillo ASPACK ASProtect ExeCryptor PECompact2 PEtite UPX	FSG PE-PaCK	simple-Binder Decepction MicroJoiner Mchain	fEvicol	Reductor
AVG	0%	0%	100%	100%	100%
BitDefender	0%	0%	100%	0%	100%
Kaspersky	0%	0%	100%	100%	100%
Mcafee	0%	0%	100%	87,50%	0%
Norton	0%	0%	100%	0%	100%
F-Secure	0%	0%	100%	100%	100%
ESET NOD32	0%	0%	100%	100%	100%

ferramentas de evasão utilizadas. Assim, a partir dos resultados obtidos com os pré-testes, decidiu-se utilizar apenas as ferramentas com melhor capacidade de evasão (a partir do último teste). As ferramentas escolhidas e aqui descritas foram:

- ExeCryptor versão 2.3.9, software comercial usado para proteção de código, baseado na tecnologia de transformação metamórfica de código, dificultando a engenharia reversa. A ferramenta possui recursos de antidepuração, anti-trace, proteção de importação e compressão de código;
- PECompact2 versão 2.78a, é um compressor comercial de executáveis/módulos Win32 (i.e. *.EXE, *.DLL, *.OCX, *.SCR), que permite em tempo de execução uma rápida descompressão diretamente na memória. A ferramenta provê mecanismos de antivírus e proteção de software;
- SoftwarePassport (antigo *Armadillo*) versão 2.4.2.454, é um software comercial de proteção de propriedade intelectual que inclui diversas formas de proteção de código, dificultando assim a engenharia reversa, análise e modificação de código.

Ferramentas de detecção

Para a avaliação de detecção de códigos maliciosos foram utilizadas cinco ferramentas de detecção conhecidas. A avaliação foi realizada na mesma data, com todos os detectores atualizados e avaliados separadamente. Os detectores avaliados foram:

- AVG Anti-Malware versão 7.5.448;
- BitDefender Av Plus Build 247;
- ESET NOD32 Anti-Virus versão 2.5;
- F-Secure Anti-Virus 2007 versão 7.01;
- Kaspersky Internet Security versão 6.0.0.300e.d.b.

Não foi possível obter informações sobre as estratégias de detecção usadas nessas ferramentas, com exceção do AVG, uma vez que seus fabricantes indicam que o fornecimento dessas informações, mesmo que em linhas gerais, comprometeria a segurança do aplicativo. No caso do AVG as informações foram obtidas através de trocas de mensagens com seu departamento técnico, sendo que em linhas gerais ele faz uso de descompactadores específicos nas várias ferramentas de ofuscação, utilizam de emulador e heurísticas, e de técnicas e algoritmos de pesquisa em strings.

4. Testes e resultados

O processo de avaliação foi realizado de forma separada para cada detector. O volume total de testes foi determinado a partir de análises sobre a sensibilidade e especificidade dos experimentos realizados. Para tanto, considerou-se como significativo um valor diferencial inferior a 4% (na prática bem menor na maioria dos casos).

Como indicado na seção anterior, os testes foram realizados dentro de um ambiente seguro, em que a avaliação dos códigos maliciosos pudesse ocorrer sem qualquer risco de contaminação. Para a construção deste ambiente foi utilizado o software VMWare 5.5 [VMware 2006] em versão trial. Após a instalação deste ambiente passou-se para a fase de geração dos casos de teste.

4.1. Geração dos casos de teste

Para o processo de avaliação, foram coletados 10.311 códigos maliciosos de diversas categorias, tais como vírus, *worms*, *trojans*, *backdoors* e *spywares*. Um total de 6.820 desses códigos não usavam, na versão utilizada, ferramenta evasiva. Para os códigos maliciosos que apresentavam sinais de ofuscação de código (3.491) foram detectadas 41 ferramentas evasivas diferentes (sem contar versões), merecendo destaque as ferramentas UPX e ASPack que foram responsáveis por mais de 80% destas amostras. Os códigos maliciosos foram obtidos através da coleção VX Heavens [VX-Heavens 2006] e os dados foram obtidos através da ferramenta de análise PE Identifier (PEID) [snaker and xineohP 2007], que detecta compressores, encriptadores e compiladores comuns em arquivos *Portable Executable* (PE), que é o formato mais complexo de arquivos EXE e podem ser executados por todas as versões do Windows.

Para a geração de variantes dos códigos maliciosos foram usados apenas os que não utilizavam de nenhuma ferramenta evasiva e, para a geração dos grupos de códigos adotou-se o critério do código malicioso mais recente para o mais antigo. Durante a geração e análise, o critério de parada foi atingido no quinto estágio. O primeiro estágio de testes consistiu em 100 códigos maliciosos, o segundo em 200, o terceiro em 400, o quarto em 800 e o último em 1.600 códigos maliciosos. Dada as três ferramentas evasivas, foram criadas 4.800 variantes de códigos maliciosos.

Para avaliação de possíveis falsos positivos, foram utilizados 80 aplicativos não maliciosos do sistema operacional Windows (calculadora, prompt de comando, terminal, discador, bloco de notas, wordpad, paint, tetris, entre outros), e foi gerado 240 casos de teste através das ferramentas de evasão.

4.2. Avaliação dos detectores

Os resultados aqui apresentados se encontram separados segundo as ferramentas de evasão utilizadas. Apresenta-se também um quadro geral da capacidade de detecção para o teste com 1.600 códigos maliciosos. Vale observar que alguns desses códigos, em suas amostras ITW (In The Wild), não foram detectados por algumas ferramentas. Esse foi o caso dos antívirus ESET NOD32, AVG e BitDefender, que detectaram 98,62%, 99%, e 98,93% das amostras respectivamente. Para as avaliações detalhadas a seguir considerou-se apenas as versões obtidas a partir da aplicação das ferramentas evasivas, desconsiderando-se as versões ITW.

Capacidade de detecção

Na tabela 4 são exibidas as taxas de acerto obtidas para variações produzidas pela ferramenta de evasão ExeCryptor. Através dela se nota um melhor desempenho do detector AVG, seguido pelo detector BitDefender, ambos com percentuais de acerto muito superiores aos obtidos pelas demais ferramentas de detecção. Isso é um forte indicador de de que essas últimas não possuem mecanismos efetivos de detecção para códigos transformados através de criptografia e técnicas de polimorfismo, que são usadas pelo ExeCryptor.

Deve ser observado, entretanto, que mesmo o AVG não conseguiu resultados próximos do ideal. Em sua melhor medida, para o conjunto de 400 códigos maliciosos, sua taxa de acerto ficou pouco acima dos 80%, o que significa deixar de identificar um a cada cinco códigos maliciosos examinados.

Tabela 4. Taxa de acerto dos detectores diante de evasões por ExeCryptor

Ferramenta de Detecção	100 *malwares*	200 *malwares*	400 *malwares*	800 *malwares*	1600 *malwares*
AVG	65%	75,5%	81,25%	79,37%	76%
BitDefender	56%	61%	65%	63,87%	63,06%
Kaspersky	3%	9%	4,5%	2,25%	1,18%
F-Secure	3%	9%	4,5%	2,25%	1,18%
ESET NOD32	16%	20,5%	18,25%	15,12%	11,62%

A tabela 5 mostra as taxas de acerto obtidas na identificação de variações produzidas com a ferramenta SoftwarePassport. Pode-se notar nessa tabela que o desempenho de todos os detectores ficou muito abaixo do necessário, chegando a quase 100% de falhas nos casos do Kaspersky e F-Secure, e não passando de 14% de acerto mesmo em seu melhor caso (AVG). A partir do exame das informações sobre o SoftwarePassport e dos códigos variantes produzidos percebe-se que a capacidade de detecção foi diminuída tanto pela aplicação de criptografia quanto polimorfismo. Como nesse caso, o polimorfismo resultou também em um aumento significativo no tamanho dos arquivos, percebe-se que esse (tamanho do arquivo) é outro fator que dificulta a análise por parte dos detectores.

Já os resultados de detecção para as variações criadas através da ferramenta PECompact podem ser vistos na tabela 6. Para essa ferramenta, ao contrário da duas anteriores, as taxas de acerto foram relativamente boas com todos os detectores.

Tabela 5. Taxa de acerto dos detectores diante de evasões por SoftwarePassport

Ferramenta de Detecção	100 *malwares*	200 *malwares*	400 *malwares*	800 *malwares*	1600 *malwares*
AVG	12%	9%	8%	11,75%	13,56%
BitDefender	3%	2%	2%	1,37%	1,18%
Kaspersky	1%	0,5%	0,75%	0,25%	0,12%
F-Secure	1%	0,5%	0,75%	0,25%	0,12%
ESET NOD32	3%	2,5%	2,75%	3,75%	2,31%

Isso indica que a técnica de evasão por ela utilizada (compactação) é resolvida de forma eficiente pelos detectores, que usam descompactadores capazes de identificar a ação dessa ferramenta. Outro aspecto interessante dos resultados apresentados nesta tabela diz respeito ao desempenho relativo entre detectores, uma vez que o AVG obteve aqui a pior taxa de detecção enquanto os detectores Kaspersky e F-Secure, com resultados sofríveis para criptografia, obtiveram as maiores taxas de acerto. Importante notar, também, que nesse caso o AVG apresentou resultados muito inferiores aos demais detectores.

Tabela 6. Taxa de acerto dos detectores diante de evasões por PECompact

Ferramenta de Detecção	100 *malwares*	200 *malwares*	400 *malwares*	800 *malwares*	1600 *malwares*
AVG	45%	43%	32,25%	41,75%	42%
BitDefender	88%	83,5%	84,5%	83,87%	84,37%
Kaspersky	93%	94%	94,5%	95,25%	95,87%
F-Secure	93%	94%	94,5%	95,25%	95,87%
ESET NOD32	62%	60,5%	61,25%	57,12%	55,43%

Identificação de falsos positivos

Em todos os casos de teste descritos nos parágrafos anteriores foi avaliada, também, a possibilidade de identificação de falsos positivos. Esse resultado, característico da análise de especificidade dos experimentos, pode ser considerado ótimo, uma vez que das 240 variantes criadas para softwares inofensivos apenas seis delas foram consideradas maliciosas por apenas um dos detectores (o AVG). Em particular, todas as variantes detectads foram criadas pela ferramenta SoftwarePassport, o que indica mais fortemente o fraco resultado desses detectores na presença desta ferramenta.

Avaliação geral da taxa de acerto

Estão mostradas na tabela 7 as taxas de detecção obtidas para o teste completo (4.800 variantes). Nessa tabela desconsidera-se a origem da modificação realizada, o que caracterizaria, de certo modo, um cenário mais global das possíveis modificações de código malicioso. É importante notar que nenhuma das ferramentas obteve mais de 50% de detecção, sendo que três conseguiram detectar apenas uma variante a cada três examinadas. Esses resultados foram fortemente orientados pelos

fracos resultados obtidos na detecção de códigos em que foram aplicadas ferramentas de criptografia e polimorfismo.

Tabela 7. Taxa de detecção dos detectores

Ferramenta de Detecção	Nº códigos detectados	% Taxa de detecção
AVG	2105	43,85%
BitDefender	2378	49,54%
Kaspersky	1555	32,39%
F-Secure	1555	32,39%
ESET NOD32	1110	23,12%

5. Conclusões

Os resultados apresentados ao longo das seções 3.3 e 4 indicam de maneira clara que o atual estado da arte na detecção de softwares maliciosos precisa ser melhorado. Resultados como os apresentados na tabela 7, em que a melhor taxa de detecção foi de 49,54%, mostram que os detectores falham quando se ofusca a presença de código malicioso através de criptografia e polimorfismo.

Por outro lado, um resultado interessante é observado na tabela 2, em que se caracterizou que as ferramentas de evasão da comunidade *blackhat*, principalmente as mais aplicadas para evasão de código, têm taxas de detecção elevadas (100% na maioria das vezes). Isso é um bom indicativo sobre os detectores, mesmo quando eles apenas identificam a presença da ferramenta de evasão, causando taxas altas de falsos positivos. Em particular, os falsos positivos para essas ferramentas não é uma preocupação, pois não se imagina alguém aplicando-as para ocultar código não malicioso.

Nessa mesma tabela, aparecem também ferramentas comerciais de proteção de direitos autorais cujas variantes também foram totalmente identificadas. A explicação, nesse caso, também está relacionada ao uso, pois dos 3.491 códigos maliciosos coletados que já apresentavam sinais de ofuscação, uma parte razoável usava tais ferramentas. Isso ocorre provalvelmente pela característica de produção de ferramentas de detecção, que é conduzida com base em demanda pré-existente.

5.1. Computação segura e a detecção de software malicioso

Pode-se definir computação segura como a área da computação preocupada com o funcionamento correto e estável dos sistemas computacionais. Esse objetivo deve ser atingido a partir de duas frentes de atuação: a produção de software que siga princípios seguros de operação, portanto livres de falhas e brechas de segurança, e a obtenção de sistemas capazes de detectar ataques por parte de softwares maliciosos. Em qualquer frente o objetivo essencial é garantir uma alta disponibilidade para sistemas computacionais, em especial aqueles destinados a aplicações de alta demanda, como *grids*. No momento ainda não existem soluções que sejam consideravelmente eficientes para nenhuma das frentes de atuação, muito embora já existam pesquisas em grande volume na engenharia de software e em segurança de computadores.

Isso coloca a obtenção de computação segura como um dos grandes desafios para a pesquisa em ciência da computação nos próximos anos.

Nesse cenário temos que a detecção de softwares maliciosos se torna um importante aspecto a ser resolvido para a obtenção de computação segura. O trabalho até agora desenvolvido permite que detectores façam de forma satisfatória a identificação de softwares maliciosos produzidos de forma convencional. Eles falham fortemente, entretanto, na presença de softwares produzidos com o auxílio de ferramentas de ofuscação. O problema relativo a ofuscações é que elas continuamente são modificadas, tornando difícil a geração de técnicas capazes de identificá-las. As perspectivas sobre esse problema são de aumento contínuo de complexidade, uma vez que cada vez mais temos o uso de aplicações na Internet, assim como a distribuição e compartilhamento de softwares e dados (principalmente na forma de arquivos de vídeo e áudio), o que aumenta o risco de contato com softwares maliciosos.

Desse modo, o desenvolvimento de ferramentas de detecção ainda precisa encontrar formas de se antecipar à demanda. Esse é um dos grandes desafios para a obtenção de sistemas de computação segura, em que falhas produzidas por códigos maliciosos possam ser diagnosticadas e evitadas com antecedência. Para a obtenção de resultados nessa direção podem ser indicados como novos campos de pesquisa os seguintes temas:

1. Criação de modelos universais para descompactação de arquivos, o que resolveria o problema de ofuscações criadas a partir de compactação de código. Esse tema é problemático pois a identificação do padrão usado na compactação não é direto e não pode depender da definição de padrões conhecidos. Isso, portanto, demanda possivelmente a capacidade de identificar autômatos a pilha que representem uma gramática de compressão.
2. Criação de modelos adaptativos para decifragem de códigos, procurando minimizar os efeitos do uso de criptografia em códigos maliciosos. Os problemas a serem resolvidos aqui são semelhantes aos da descompactação de arquivos e, provavelmente, dependem também de soluções semelhantes.
3. Criação de técnicas para identificar a aplicação de mutação (polimorfismo ou metamorfismo), principalmente aquelas relacionadas com reordenação de código e substituição de instruções. As técnicas hoje em uso demandam uma carga computacional bastante elevada, como por exemplo as apresentadas em [Lakhotia and Mohammed 2004, Christodorescu et al. 2005]. Assim, a obtenção de técnicas mais eficientes para a identificação de mutações se torna um problema essencial na detecção de software malicioso.

Cada um desses temas representa desafios diferentes para a obtenção de computação segura. Em particular deve ser destacado que os dois primeiros problemas introduzem, em sua resolução, um problema ainda mais grave de segurança, que é caracterizado pela perda de privacidade sobre informações legítimas. Os autores deste trabalho acreditam, independentemente de aspectos éticos a serem resolvidos, que a solução para a geração de ferramentas mais eficientes para a detecção de malwares dependem de trabalhos de engenharia reversa fundamentados na aplicação de métodos formais em sua especificação.

Referências

ASPACK-Software (2006). *ASPACK: File Compressor*. http://www.aspack.com/. Último acesso, Agosto 2006.

Bitsum-Technologies (2007). *PECompact2: File Compressor*. http://www.bitsum.com/pec2.asp. Último acesso, Fevereiro 2007.

Brosch, T. and Morgenstern, M. (2006). Runtime packers: The hidden problem? In *Blackhat Briefings*.

Brunnstein, K. (2004). *AntiVirus Scanner Tests July 2004*. Virus Test Center, University of Hamburg, Computer Science Department. Publicado em http://agn-www.informatik.uni-hamburg.de/vtc/. Último acesso, Agosto 2006.

C. Collberg, C. T. and Low, D. (1998a). Breaking abstractions and unstructuring data structures. In *Proceedings of the International Conference on Computer Languages*, pages pp. 28–39, Chicago, IL, USA. IEEE Computer Society.

C. Collberg, C. T. and Low, D. (1998b). Manufacturing cheap, resilient, and stealthy opaque constructs. In *Proceedings of the 25th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, San Diego, CA, USA. ACM Press.

Christodorescu, M. and Jha, S. (2004). Testing malware detectors. In *Proceedings of the ACM SIGSOFT International Symposium on Software Testing and Analysis 2004 (ISSTA 04)*, pages pp. 34–44, Boston, MA, USA. ACM SIGSOFT, ACM Press.

Christodorescu, M., Jha, S., Seshia, S., Song, D., and Bryant, R. (2005). Semantics-aware malware detection. In *Proceedings IEEE Symp. Security and Privacy 2005*.

Gordon, S. and Ford, R. (1996). Real world antivirus product reviews and evaluations, the current state of affairs. In *Proceedings of the 19th National Information Systems Security Conference (NISSC 96)*, pages pp. 526–538, Baltimore, MD, USA. National Institute of Standards and Technology (NIST).

GriYo (2006). *EPO: Entry-Point Obscuring*. Publicado online em VX Heaven - http://vx.netlux.org/lib/vgy01.html. Último acesso, Agosto 2006.

ICSA-Labs (2006). *Anti-virus product certification*. http://www.icsalabs.com/. Último acesso, Agosto 2006.

L. A. Gordon, M. P. Loeb, W. L. and Richardson, R. (2004). 2004 csi/fbi computer crime and security survey. Technical report, Technical report, Computer Security Institute.

Lakhotia, A. and Mohammed, M. (2004). Imposing order on program statements and its implication to av scanners. In *Proceedings 11th IEEE Working COnference Reverse Engineering 2004*.

Marx, A. (2000). A guideline to anti-malware-software testing. In *Proc. 9th Annual European Inst.Computer Antivirus Research Conf.(EICAR 00)*.

Marx, A. (2002). Retrospective testing, how good heuristics really work. In *Proceedings of the 2002 Virus Bulletin Conference (VB2002)*, New Orleans, LA, USA. Virus Bulletin.

PEtite (2006). *PEtite: Win32 Executable Compressor*. http://www.un4seen.com/petite. Último acesso, Agosto 2006.

S. Chow, Y. Gu, H. J. and Zakharov, V. (2001). An approach to the obfuscation of control-flow of sequential computer programs. In *Proceedings of the 4th International Information Security Conference*, volume 2200 of Lecture Notes in Computer Science, pages pp. 144–155, Malaga, Spain.

Silicon-Realms (2007). *SoftwarePassport*. http://siliconrealms.com/index.shtml. Último acesso, Fevereiro 2007.

Skulason, F. (1995). Latest trends in polymorphism the evolution of polymorphic computer viruses. In *Proceedings of the 1995 Virus Bulletin Conference (VB1995)*, pages pp. I–VII. Virus Bulletin.

snaker, Qwerton, J. and xineohP (2007). *PE Identifier (PEID)*. http://peid.has.it/. Último acesso, Março 2007.

Strongbit-Technology (2007). *EXECryptor: Stop Crackers and software pirates*. http://www.strongbit.com/. Último acesso, Fevereiro 2007.

Szor, P. and Ferrie, P. (2001). Hunting for metamorphic. In *Proceedings of the 2001 Virus Bulletin Conference (VB2001)*, pages pp. 123–144. Virus Bulletin.

UPX (2006). *UPX: Ultimate Packer for eXecutables*. http://upx.sourceforge.net/. Último acesso, Agosto 2006.

Virus-Bulletin (2006). *Virus-Bulletin, VB 100% Award*. disponível em http://www.virusbtn.com/vb100/about/100use.xml. Último acesso, Agosto 2006.

VMware (2006). *VMware - Virtualization Software*. http://www.vmware.com/. Último acesso, Agosto 2006.

VX-Heavens (2006). *VX Heavens Virus Collection*. http://vx.netlux.org/vl.php. Último acesso, Março 2007.

West-Coast-Labs (2006). *Anti-virus Checkmark*. http://www.westcoastlabs.org/checkmarkcertification.asp. Último acesso, Agosto 2006.

Wildlist (2006). *Frequently asked questions*. The WildList Organization International, http://www.wildlist.org/faq.htm. Último acesso, Agosto 2006.

Wroblewski, G. (2002). *General method of program code obfuscation*. PhD thesis, PhD thesis, Institute of Engineering Cybernetics. Wroclaw, Poland.

Zeltser, L. (2001). *Reverse Engineering Malware*. http://www.zeltser.com/sans/gcih-practical/revmalw.html. Último acesso, Abril 2006.