DOI: http://dx.doi.org/10.5769/C2011020

BinStat

Ferramenta para Reconhecimento de Executáveis Empacotados

D. Sc. Kil Jin Brandini Park, Rodrigo Ruiz, Antônio Montes
Divisão de Segurança de Sistemas da Informação (DSSI)
Centro de Tecnologia da Informação Renato Archer (CTI)
Campinas – SP, Brasil.
kil.park@cti.gov.br, rodrigo.ruiz@dssi.cti.gov.br, antonio.montes@dssi.cti.gov.br

Resumo — A quantidade de diferentes alvos aliada as diferentes finalidades dos ataques perpetrados por artefatos maliciosos (malware), em conjunto com o vasto arsenal de ferramentas disponíveis aos desenvolvedores, gera uma quantidade de malware que exige o máximo de automatização nas tarefas de análise e prospeção dos mesmos.

Considerando que um dos problemas a serem tratados para a análise estática dos artefatos é o empacotamento destes, apresenta-se uma metodologia de reconhecimento de empacotamento de executáveis, através da utilização de cálculos estatísticos e de teoria da informação.

A ferramenta desenvolvida através do emprego das técnicas apresentadas, batizada de BinStat, gerou uma alta taxa de reconhecimento de empacotamento nos testes realizados, comprovando a aplicabilidade da metodologia abordada.

Palavras-chave: Empacotamento, Executáveis Empacotados, Análise de Malware

Abstract — The quantity of malicious artifacts (malware) generated by the combination of unique attack goals, unique targets and various tools available for the developers, demands the automation of prospecting and analysis of said artifacts.

Considering the fact that one problem handled by experts in analysis of executable code is packing, this paper presents a method of packing detection through the appliance of statistical and information theory metrics.

The tool developed in this study, called BinStat, generated a high recognition rate of executable packing status with the test samples, proving its effectiveness.

Keywords: Packing, Packed Executables, Malware Analysis

1. Introdução

Com o advento da internet e a consequente oferta de serviços sensíveis online, a ação dos artefatos maliciosos foi alçada a um novo patamar. A quebra de confidencialidade dos dados tornou-se mais amplamente perseguida devido ao seu potencial financeiro. A complexidade dos artefatos acompanha a gama de serviços disponíveis, crescendo continuamente e empregando técnicas cada vez mais refinadas de ataque e ofuscação.

Um setor bastante afetado por atividades criminosas é o de serviços bancários online. Dados apresentados por estudo em [1] indicam que no ano de 2009, os prejuízos dos bancos brasileiros com fraudes online atingiram a cifra de novecentos milhões de reais. Além disso, [2] diz que o Brasil é uma das maiores fontes de artefatos maliciosos do tipo trojan voltados para ataques à bancos. Corroborando com esta idéia de protagonismo brasileiro, [3] apresenta dados que pregam que o Brasil é hoje fonte de quatro porcento de todos os novos artefatos maliciosos capturados.

Migrando do setor privado ao público, pode-se observar um movimento intenso das maiores potências mundiais no sentido de construir salvaguardas a ataques cibernéticos contra a segurança nacional. A propagação do worm stuxnet que, de acordo com [4], [5] e [6], foi desenvolvido com o intuito específico de atingir o programa nuclear iraniano ao avariar componentes de controle de suas centrífugas nucleares, foi considerada por muitos especialistas em segurança da informação como o movimento inaugural da ciberguerra (ciberwarfare).

A quantidade de diferentes alvos aliada as diferentes finalidades dos ataques, em conjunto com o vasto arsenal de ferramentas disponíveis aos desenvolvedores, gera uma quantidade de artefatos maliciosos que exige o máximo de automatização das tarefas de análise e prospeção dos mesmos. Assim, o desenvolvimento de ferramentas que extraiam informações de quaisquer etapas das análises dos executáveis, sejam elas estáticas ou dinâmicas, torna-se imprescindível.

Uma ferramenta que pode ser utilizada por desenvolvedores são os empacotadores. Estes podem, além de ofuscar o código-fonte de um executável, reduzir seu tamanho. Portanto, desenvolvedores podem utilizá-los com as finalidades legítimas de reduzir o espaço ocupado por um programa e salvaguardar sua propriedade intelectual.

Porém, no que concerne ao desenvolvimento de malware, os empacotadores são utilizados com o intuito de contornar os mecanismos de reconhecimento de assinaturas presentes nos antivirus e dificultar ou impedir o acesso ao código-fonte dos artefatos, através do emprego de vários métodos como empacotamento de múltiplas camadas e as técnicas anti-desempacotamento: anti-dumping, anti-debbuging e anti-emulating [7].

Assim, um dos problemas a serem tratados por pesquisadores que pretendem realizar, dentro da análise estática, a análise do código-fonte de um determinado executável, é justamente verificar a utilização de técnicas de empacotamento do mesmo.

Dessa forma, o presente trabalho discute o desenvolvimento de uma aplicação batizada de BinStat, cujo intuito é o de, através de análise de estatísticas e fórmulas da teoria da informação, classificar executáveis como empacotados ou não empacotados.

O restante deste trabalho se organiza da seguinte forma:

Em Metodologia de Reconhecimento de Empacotamento, apresenta-se métodos de reconhecimento, entre eles o método estatístico a ser empregado na aplicação apresentada.

Em Arquitetura da Aplicação, discute-se a arquitetura do BinStat e as particularidades de implementação de cada um dos seus módulos.

Em Resultados Preliminares, apresenta-se uma comparação dos cálculos estatísticos e de teoria da informação extraídos de um determinado executável e de uma versão deste mesmo executável empacotado.

Em Resultados e Discussão, analisa-se os resultados obtidos no desenvolvimento e testes da ferramenta BinStat.

Finalizando, seguem as conclusões e referências bibliográficas utilizadas.

2. Metodologia de Reconhecimento de Empacotamento

Algumas ferramentas disponíveis para verificação de empacotamento, tais como o PeiD [8], utilizam-se da metodologia de verificação de empacotamento através do emprego de assinaturas.

Se por um lado o emprego de assinaturas permite apontar não apenas o empacotamento ou não de um executável, mas a ferramenta utilizada para tal, por outro lado, a técnica não reconhece ferramentas que ainda não constem em seu banco de assinaturas, conforme [9] e [10] comprovam. Além disso, o reconhecimento através de assinaturas está sujeito a tentativas de burla, em caso de ferramentas de empacotamento que tentem mascarar sua assinatura, escondendo-a ou tornando-a similar a de outras ferramentas.

Outra técnica apresentada por [10] sugere a adoção de verificação de operações efetuadas em memória através da utilização de um ambiente emulado. A idéia apresentada é a de controlar operações de escrita e execução em um determinado trecho de memória utilizado por um processo dentro de um ambiente emulado. Uma região da memória que sofreu escrita após o início da execução é marcada como "suja".

Para que o artefato malicioso seja executado, seu código original que se apresenta ofuscado deve passar por um processamento e ser escrito em uma determinada posição de memória que conterá as operações originais desempacotadas e eventualmente será lida e enviada para execução. Portanto, regiões de memória "sujas" que serão executadas só podem apresentar ou operações de empacotamento de múltiplos níveis ou operações originalmente ofuscadas.

A desvantagem desta metodologia encontra-se na necessidade de execução dentro de máquinas virtuais do executável a ser analisado.

Uma outra possibilidade é o emprego de cálculos estatísticos e provenientes da teoria da informação com a finalidade de extrair informações sobre executáveis, visando a construção de uma árvore de decisão utilizada para classificá-los como empacotados ou não. Esta metodologia foi aplicada ao presente trabalho, sendo também utilizada por [11] na tentativa de identificação de executáveis com comportamento malicioso, apresentando resultados promissores.

3. Arquitetura da Aplicação

A aplicação é subdividida em quatro módulos, conforme figura 1:

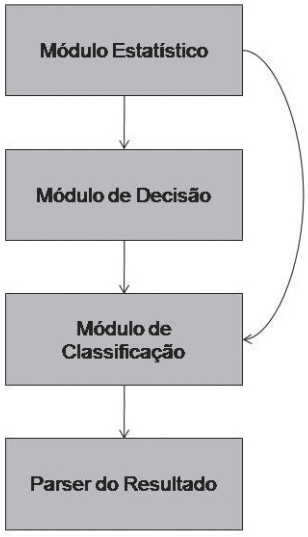

Figura 1. Módulos da Aplicação

É possível verificar que a aplicação funciona de duas maneiras distintas. Na fase de treinamento, os executáveis empacotados e não empacotados serão fornecidos como entrada para o módulo estatístico que por sua vez alimenta o módulo de decisão, onde a árvore de decisão será treinada.

Uma vez que a árvore é construída, esta é passada ao módulo de classificação, que por sua vez estará, então, apto a receber requisições de classificação de executáveis de status de empacotamento desconhecido.

Na fase de produção, o módulo de decisão não atua. Dessa forma, os cálculos a respeito do executável a ser analisado gerados pelo módulo estatístico são repassados ao módulo de classificação, cujo resultado alimenta o parser de resultado, responsável por formatar a saída para uma forma previamente determinada como conveniente para processamentos futuros.

A. Módulo Estatístico

O módulo estatístico segmenta o artefato de entrada em blocos de 1024 bytes. Para cada bloco, serão calculadas o

número de ocorrências e o histograma de frequência dos valores de unigrama, bigrama, trigrama ou quadrigrama, ou seja, as ocorrências dos valores encontrados para as combinações de um byte (0 - 255), dois bytes (0 – 65.535), três bytes (0 – 16.777.215) e quatro bytes (0 – 4.294.967.295).

Este histograma será utilizado como entrada por treze cálculos estatísticos e de teoria da informação [12]:

1) Índice de Simpson

$$S_{b_i} = \frac{\sum_{f=0}^{n} k_f\left(k_f - 1\right)}{N\left(N-1\right)}$$

Onde b_i refere-se ao i-ésimo bloco do executável a ser analisado, n atinge o valor máximo do n-grama a ser analisado menos um, k_f é o número de ocorrência do n-grama de valor f dentro do bloco e N é o número total de bytes do bloco. É importante salientar que o último bloco de um executável a ser analisado pode não possuir todos os 1024 bytes.

2) Distância de Canberra

$$CA_{b_i} = \sum_{f=0}^{n} \frac{\left|X_f - X_{f+1}\right|}{\left|X_f\right| + \left|X_{f+1}\right|}$$

Onde b_i refere-se ao i-ésimo bloco do executável a ser analisado, n atinge o valor máximo do n-grama a ser analisado menos um, X_f refere-se a frequência do n-grama de valor f, X_{f+1} refere-se a frequência do n-grama de valor f + 1.

3) Distância de Ordem de Minkowski

$$M_{b_i} = \sqrt[\lambda]{\sum_{f=0}^{n} \left|X_f - X_{f+1}\right|^{\lambda}}$$

Onde b_i refere-se ao i-ésimo bloco do executável a ser analisado, n atinge o valor máximo do n-grama a ser analisado menos um, X_f refere-se a frequência do n-grama de valor f, X_{f+1} refere-se a frequência do n-grama de valor f + 1. O valor adotado de λ é 3 [11].

4) Distância de Manhattan

$$MH_{b_i} = \sum_{f=0}^{n} \left|X_f - X_{f+1}\right|$$

Onde b_i refere-se ao i-ésimo bloco do executável a ser analisado, n atinge o valor máximo do n-grama a ser analisado menos um, X_f refere-se a frequência do n-grama de valor f, X_{f+1} refere-se a frequência do n-grama de valor f + 1.

5) Distância de Chebyshev

$$CH_{b_i} = max_f \left|X_f - X_{f+1}\right|$$

Onde b_i refere-se ao i-ésimo bloco do executável a ser analisado, X_f refere-se a frequência do n-grama de valor f, X_{f+1} refere-se a frequência do n-grama de valor f + 1.

6) Distância de Bray Curtis

$$BC_{b_i} = \frac{\sum_{f=0}^{n} \left|X_f - X_{f+1}\right|}{\sum_{f=0}^{n} \left(X_f + X_{f+1}\right)}$$

Onde b_i refere-se ao i-ésimo bloco do executável a ser analisado, n atinge o valor máximo do n-grama a ser analisado menos um, X_f refere-se a frequência do n-grama de valor f, X_{f+1} refere-se a frequência do n-grama de valor f + 1.

7) Separação Angular

$$AS_{b_i} = \frac{\sum_{f=0} X_f \cdot X_{f+1}}{\left(\sum_{f=0}^{n} X_f^2 \cdot \sum_{f=0}^{n} X_{f+1}^2\right)^{1/2}}$$

Onde b_i refere-se ao i-ésimo bloco do executável a ser analisado, n atinge o valor máximo do n-grama a ser analisado menos um, X_f refere-se a frequência do n-grama de valor f, X_{f+1} refere-se a frequência do n-grama de valor f + 1.

8) Coeficiente de Correlação

$$CC_{b_i} = \frac{\sum_{f=0}^{n} \left(X_f - \overline{X}_{b_i}\right) \cdot \left(X_{f+1} - \overline{X}_{b_i}\right)}{\left(\sum_{f=0}^{n} \left(X_f - \overline{X}_{b_i}\right)^2 \cdot \sum_{f=0}^{n} \left(X_{f+1} - \overline{X}_{b_i}\right)^2\right)^{1/2}}$$

Onde b_i refere-se ao i-ésimo bloco do executável a ser analisado, n atinge o valor máximo do n-grama a ser analisado menos um, X_{bi} refere-se a média da frequência de n-gramas do bloco b_i, Xf refere-se a frequência do n-grama de valor f, X_{f+1} refere-se a frequência do n-grama de valor f + 1.

9) Entropia

$$E(R) = -\sum_{v \in \Delta_n} t(r_v) \log_2 t(r_v)$$

Onde Δ_n são as imagens, ou seja, todos os valores válidos para um determinado n-grama, e $t(r_v)$ é a frequência do n-grama de valor v.

10) Divergência de Kullback – Leibler

$$KL_{b_i}\left(X_f \parallel X_{f+1}\right) = \sum_{f=0}^{n} X_f \log \frac{X_f}{X_{f+1}}$$

Onde b_i refere-se ao i-ésimo bloco do executável a ser analisado, n atinge o valor máximo do n-grama a ser analisado

menos um, X_f refere-se a frequência do n-grama de valor f, X_{f+1} refere-se a frequência do n-grama de valor f + 1.

11) Divergência de Jensen-Shannon

$$JSD_{b_i}\left(X_f \parallel X_{f+1}\right) = \frac{1}{2} D\left(X_f \parallel M\right) + \frac{1}{2} D\left(X_{f+1} \parallel M\right)$$

Onde

$$M = \frac{1}{2}\left(X_f + X_{f+1}\right)$$

b_i refere-se ao i-ésimo bloco do executável a ser analisado, D refere-se a divergência de Kullback – Leibler, X_f refere-se a frequência do n-grama de valor f, X_{f+1} refere-se a frequência do n-grama de valor f + 1.

12) Divergência de Itakura – Saito

$$BF_{b_i}\left(X_f, X_{f+1}\right) = \sum_{f=0}^{n}\left(\frac{X_f}{X_{f+1}} - \log\left(\frac{X_f}{X_{f+1}}\right) - 1\right)$$

Onde b_i refere-se ao i-ésimo bloco do executável a ser analisado, n atinge o valor máximo do n-grama a ser analisado menos um, X_f refere-se a frequência do n-grama de valor f, X_{f+1} refere-se a frequência do n-grama de valor f + 1.

13) Variação Total

$$\delta_{b_i}\left(X_f, X_{f+1}\right) = \frac{1}{2}\sum_{f}\left|X_f - X_{f+1}\right|$$

Onde b_i refere-se ao i-ésimo bloco do executável a ser analisado, X_f refere-se a frequência do n-grama de valor f, X_{f+1} refere-se a frequência do n-grama de valor f + 1.

Para o treinamento da aplicação (fase de treinamento), para cada um dos n-gramas (unigrama até quadrigrama) utilizados, este módulo calculará as treze medidas apresentadas para cada bloco que compõe os executáveis que integram a base de treinamento. Este conjunto de informações será, então, repassado ao módulo de decisão para a contrução de uma árvores de decisão para cada um dos n-gramas.

Em caso de teste de um executável (fase de produção), o resultado gerado por este módulo será repassado ao módulo de classificação, já previamente alimentado pelas árvores de decisão geradas durante o treinamento.

B. Módulo de Decisão

Para implementar o módulo de decisão, através das técnicas de construção de árvores de decisão, foram utilizadas as versões de código fonte público das ferramentas C5.0/See5, cujo funcionamento encontra-se descrito em [13].

C. Módulo de Classificação

O módulo de classificação recebe como entrada os cálculos de cada um dos blocos do executável a ser testado e as árvores de decisão geradas pelo módulo de decisão. Note que é preciso decidir qual n-grama será testado, pois existe uma árvore de decisão para cada um dos n-gramas utilizados.

Com base nestes dados, o módulo classifica cada um dos blocos como empacotado ("yes") ou não empacotado ("no").

D. Parser do Resultado

O módulo parser do resultado recebe o resultado gerado pelo módulo de decisão e o formata de maneira a proporcionar um formato adequado a utilização posterior do mesmo.

É importante salientar que este módulo pode ser adaptado, possibilitando a integração da aplicação com outros mecanismos.

4. Resultados Preliminares

Como pré avaliação dos cálculos que compõe o módulo estatístico, um determinado executável (identificado pelo md5 e4a18adf075d1861bd6240348a67cce2) foi selecionado e sua versão original foi empacotada com a aplicação upx (identificado pelo md5 745528339c38f3eb1790182db8febee1). O aplicativo original e sua versão empacotada foram utilizadas como entrada deste módulo. As distribuições dos resultados normalizados (para valores no intervalo de 0 a 1), foram comparadas graficamente:

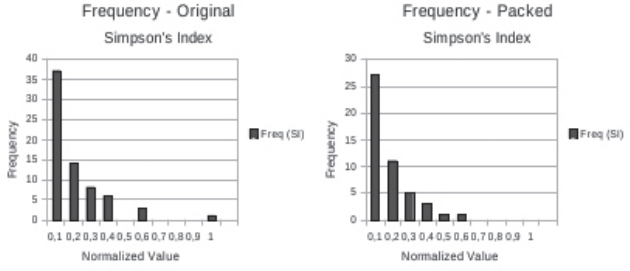

Figura 2. Comparação da Frequência dos Valores Normalizados do Índice de Simpson para Unigrama do Executável Original (Esquerda) e Empacotado (Direita).

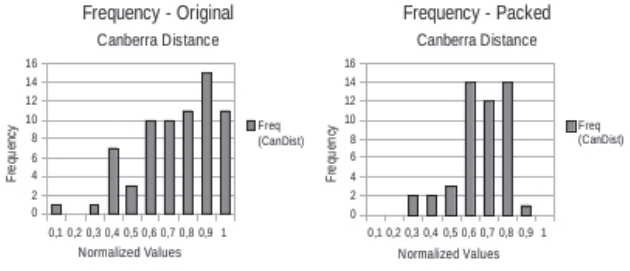

Figura 3. Comparação da Frequência dos Valores Normalizados da Distância de Canberra para Unigrama do Executável Original (Esquerda) e Empacotado (Direita).

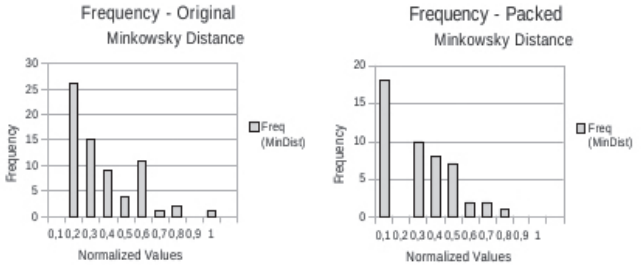

Figura 4. Comparação da Frequência dos Valores Normalizados da Distância de Minkowsky para Unigrama do Executável Original (Esquerda) e Empacotado (Direita).

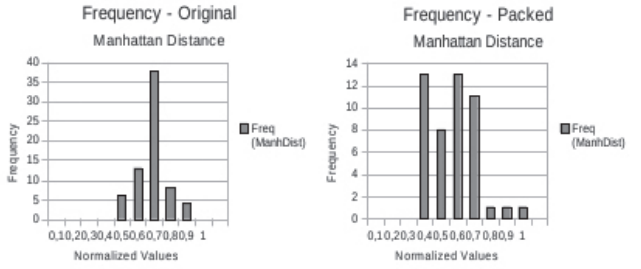

Figura 5. Comparação da Frequência dos Valores Normalizados da Distância de Manhattan para Unigrama do Executável Original (Esquerda) e Empacotado (Direita).

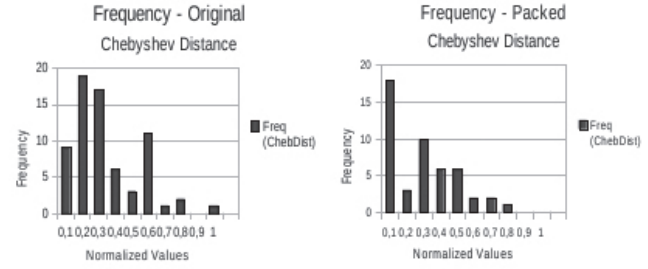

Figura 6. Comparação da Frequência dos Valores Normalizados da Distância de Chebyshev para Unigrama do Executável Original (Esquerda) e Empacotado (Direita).

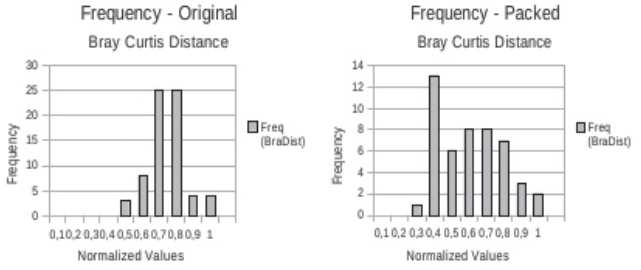

Figura 7. Comparação da Frequência dos Valores Normalizados da Distância de Bray Curtis para Unigrama do Executável Original (Esquerda) e Empacotado (Direita).

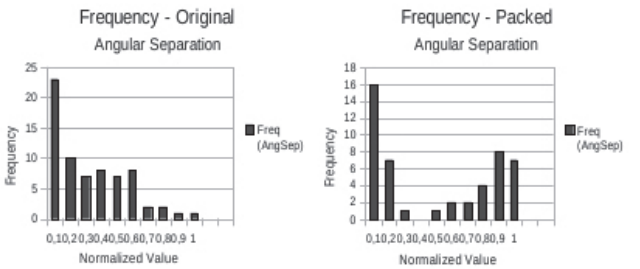

Figura 8. Comparação da Frequência dos Valores Normalizados da Separação Angular para Unigrama do Executável Original (Esquerda) e Empacotado (Direita).

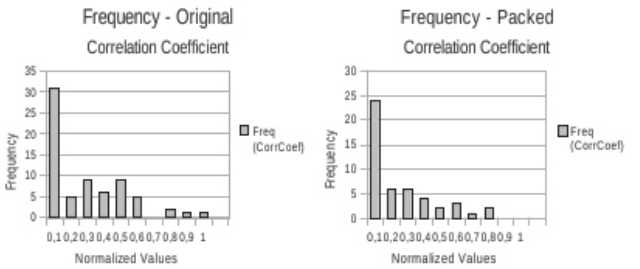

Figura 9. Comparação da Frequência dos Valores Normalizados da Coeficiente de Correlação para Unigrama do Executável Original (Esquerda) e Empacotado (Direita).

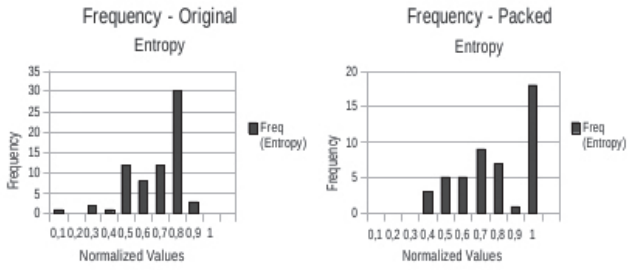

Figura 10. Comparação da Frequência dos Valores Normalizados da Entropia para Unigrama do Executável Original (Esquerda) e Empacotado (Direita).

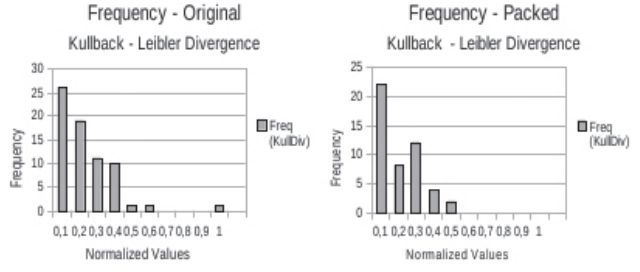

Figura 11. Comparação da Frequência dos Valores Normalizados da Divergência de Kullback - Leibler para Unigrama do Executável Original (Esquerda) e Empacotado (Direita).

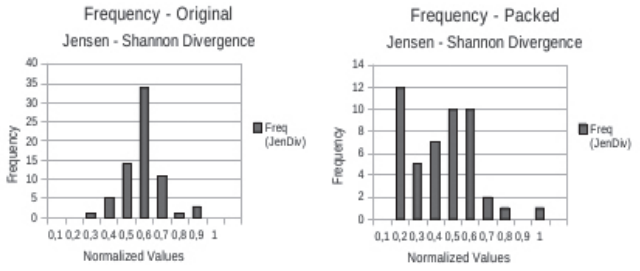

Figura 12. Comparação da Frequência dos Valores Normalizados da Divergência de Jensen - Shannon para Unigrama do Executável Original (Esquerda) e Empacotado (Direita).

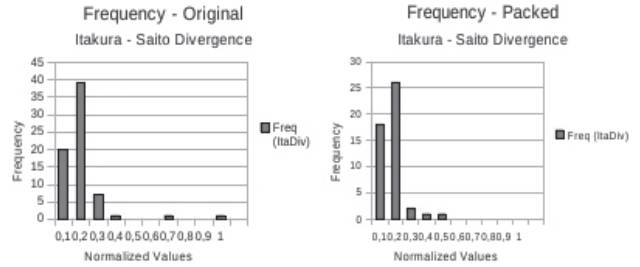

Figura 13. Comparação da Frequência dos Valores Normalizados da Divergência de Itakura - Saito para Unigrama do Executável Original (Esquerda) e Empacotado (Direita).

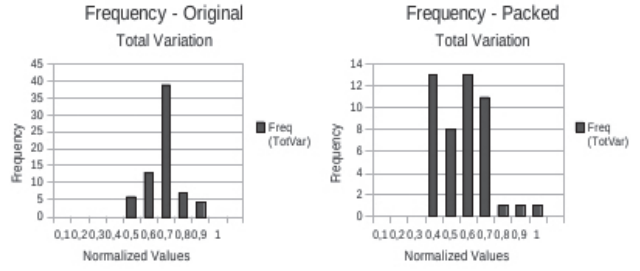

Figura 14. Comparação da Frequência dos Valores Normalizados da Variação Total para Unigrama do Executável Original (Esquerda) e Empacotado (Direita).

O grau de mudança observado nos cálculos entre os executáveis original e empacotado justifica a adoção dos mesmos para a construção da árvore de decisão.

5. Resultados e Discussão

A. Fase de Treinamento

Para a fase de treinamento, foram selecionados quatrocentos e cinquenta e seis (456) executáveis não compactados:

Tabela 1: Dados sobre os executáveis selecionados para a base de treinamento

	Tamanho Mínimo (bytes)	Tamanho Médio (bytes)	Tamanho Máximo (bytes)
Executáveis	817	124981.96	3558912

Tabela 2: Dados sobre os compactadores utilizados

Compactadores	MD5
mew11	cbfbb5517bf4d279cf82d9c4cb4fe259
upx	745528339c38f3eb1790182db8febee1
cexe	fa0e3f80b8e188d90e800cb6a92de28e
fsg	00bd8f44c6176394caf6c018c23ea71b
pecompact	21180116c1bc30cda03befa7be798611
mpress	18cabd06078dc2d7b728dbf888fe9561
xcomp97	e28f888ec49ff180f24c477ca3446315

Após esta etapa, os executáveis originais e compactados são recebidos pelo módulo estatístico, que gera as informações necessárias para o treinamento da árvore de decisão. As informações são armazenadas em um arquivo texto seguindo o padrão de entrada de dados utilizado pelo programa C5.0, conforme [13]:

0.005388,95.159503,0.029534,0.646484,0.019531,0.326108, 0.746169,0.344504,7.592288,0.130506,0.041571,101.373204, 0.323242,yes,BLOCK29,3afb6adccb65b1c4284833080e878db3

Onde cada linha apresenta, separados por vírgula, os treze cálculos estatísticos e de teoria da informação relativos a um determinado bloco, o status do bloco ("yes" caso empacotado, "no" caso contrário), a identificação do bloco dentro do executável e o MD5 do executável original.

Para o teste preliminar, cada um dos conjuntos de treinamentos relativos aos n-gramas utilizados são fornecidos ao C5.0, e duas opções de treinamento são selecionadas. A primeira delas é a opção padrão, e a segunda determina a construção da árvore de decisão utilizando boost constituido de 10 passos, onde os casos de erros das tentativas anteriores são revistos e geram modificações na árvore subsequente, na tentativa de melhorar a árvore de decisão final.

Tabela 3: Dados sobre a árvore de decisão gerada com opções padrão.

N-grama	Tamanho da Árvore de Decisão	Erro da Árvore de Decisão
Unigrama	680	10.40%
Bigrama	618	10.90%
Trigrama	554	11.20%
Quadrigrama	575	11.40%

Tabela 4: Dados sobre a árvore de decisão gerada com opção de boost de 10 passos.

N-grama	Erro da Árvore de Decisão
Unigrama	9.30%
Bigrama	9.70%
Trigrama	10.20%
Quadrigrama	10.50%

Com base nestes dados, as estatísticas geradas sobre a distribuição de unigramas foi adotada para as novas tentativas de treinamento com a adoção de boost constituido de mais passos:

Tabela 5: Dados sobre as árvores de decisão geradas sobre a distribuição de unigrama com distintas opções de boost.

Número de Passos do Boost	Erro da Árvore de Decisão
10	9.30%
20	8.80%
30	8.60%
40	8.60%

Portanto, a árvore de decisão construída sobre a distribuição de unigramas com boost de 30 passos foi adotada para alimentar o módulo de classificação.

Uma parte desta árvore de decisão gerada e adotada pode ser visualizada na figura a seguir:

```
entropy > 6.623186:
|   corrcoef > 0.684858:
|   |   manhdis <= 0.519531: no (128)
|   |   manhdis > 0.519531:
|   |   |   entropy > 7.102338: yes (13)
|   |   |   entropy <= 7.102338:
|   |   |   |   itadiv <= 67.9743: no (14)
|   |   |   itadiv > 67.9743:
|   |   |   |   jendiv <= 0.036151: no (4)
|   |   |   |   jendiv > 0.036151: yes (64/28)
|   corrcoef <= 0.684858:
|   |   candis > 90.49059: yes (33631/425)
|   |   candis <= 90.49059:
|   |   |   angsep <= 0.763649:
|   |   |   |   mindis <= 0.111632:
|   |   |   |   |   itadiv > 68.00925: yes (23209/2354)
|   |   |   |   |   itadiv <= 68.00925:
|   |   |   |   |   |   entropy <= 7.195227: no (19)
|   |   |   |   |   |   entropy > 7.195227: yes (89/13)
|   |   |   |   mindis > 0.111632:
|   |   |   |   |   corrcoef <= 0.113134: yes (92/4)
|   |   |   |   |   corrcoef > 0.113134:
|   |   |   |   |   |   chedis <= 0.10397: no (38)
|   |   |   |   |   |   chedis > 0.10397: yes (22/4)
|   |   |   angsep > 0.763649:
|   |   |   |   candis <= 60.67399:
|   |   |   |   |   itadiv <= 55.78081: no (14)
|   |   |   |   |   itadiv > 55.78081: yes (12/1)
|   |   |   |   candis > 60.67399:
|   |   |   |   |   corrcoef <= 0.488466: yes (77242/1263)
|   |   |   |   |   corrcoef > 0.488466:
|   |   |   |   |   |   candis <= 74.81892: no (13)
|   |   |   |   |   |   candis > 74.81892:
|   |   |   |   |   |   |   jendiv > 0.030838: yes (41/2)
|   |   |   |   |   |   |   jendiv <= 0.030838:
|   |   |   |   |   |   |   |   candis <= 82.95027: yes (3)
|   |   |   |   |   |   |   |   candis > 82.95027:
|   |   |   |   |   |   |   |   |   chedis <= 0.008789: yes (2)
|   |   |   |   |   |   |   |   |   chedis > 0.008789: no (7)
entropy <= 6.623186:
|   chedis <= 0.167969:
|   |   entropy > 6.161778:
|   |   |   manhdis <= 1.005081:
|   |   |   |   jendiv <= 0.035845: no (49)
|   |   |   |   jendiv > 0.035845:
|   |   |   |   |   bradis > 0.44367: yes (450/80)
|   |   |   |   |   bradis <= 0.44367:
|   |   |   |   |   |   itadiv <= 60.18798:
|   |   |   |   |   |   |   entropy <= 6.532473: no (28/1)
|   |   |   |   |   |   |   entropy > 6.532473: yes (3)
|   |   |   |   |   |   itadiv > 60.18798:
|   |   |   |   |   |   |   angsep <= 0.162795: yes (27/1)
|   |   |   |   |   |   |   angsep > 0.162795:
|   |   |   |   |   |   |   |   candis <= 97.74641: no (29/3)
|   |   |   |   |   |   |   |   candis > 97.74641: yes (380/149)
```

Figura 15. Parte da Árvore de Decisão Gerada Durante o Treinamento

B. Fase de Produção

Para os testes desta fase, utilizou-se um conjunto de 22 executáveis não empacotados que não pertenciam a base de treinamento da árvore de decisão.

Além dos sete empacotadores utilizados anteriormente, o empacotador Themida (identificado pelo md5 6e8ef3480f36ce538d386658b9ba011a) foi utilizado na construção de 6 executáveis empacotados, com o intuito de testar a robustez do método proposto na detecção de empacotadores que não fizeram parte do conjunto de treinamento.

Tabela 6: Resultado do teste para executáveis não empacotados.

MD5 do Executável	Blocos Reconhecidos Como Empacotados (falso positivos)	Blocos Reconhecidos Como Não Empacotados
09c7859269563c240ab2aaab574483dd	14,085%	85,915%
1a9b51a0d07be16bc44a4f8ff6f538fd	26,667%	73,333%
1f06d05ef9814e4cb5202c197710d2f5	27,778%	72,222%
1f171553f1138dc0062a71a7d275055a	7,285%	92,715%
2cffa74f01e50f2fc07d45dbe56561bb	11,111%	88,889%
6d12a84c55f20a45c78eb1b5c720619b	31,034%	68,966%
8cace33911b71d63fca920cabda3a63a	40,000%	60,000%
8e93cdf0ea8edba63f07e2898a9b2147	15,556%	84,444%
9a6a653adf28d9d69670b48f535e6b90	41,463%	58,537%
9d1f6b512a3ca51993d60f6858df000d	10,256%	89,744%
41fb70824080b8f9774f688532a89e01	12,903%	87,097%
378da78d3d3c981b38fb4d10b049d493	37,037%	62,963%
5723ccbd541e553b6ca337a296da979f	6,515%	93,485%
97297c74d02e522b6a69d24d4539a359	16,667%	83,333%
9872199bec05c48b903ca87197dc1908	23,077%	76,923%
b65a1a4b606ec35603e98d7ca10d09d7	29,167%	70,833%
b2099fbd58a8f43282d2f7e14d81f97e	11,765%	88,235%
c07f1963e4ff877160ca12bcf0d40c2d	29,167%	70,833%
de7cf7de23de43272e708062d0a049b8	16,667%	83,333%
e8b0a9ecb76aaa0c3519e16f34a49858	21,466%	78,534%
ecef404f62863755951e09c802c94ad5	44,737%	55,263%
fbd6b3bb2a40478df5434a073d571cae	21,739%	78,261%

Tabela 7: Resultado do teste para executáveis empacotados.

MD5 do Executável	Empacotador	Blocos Reconhecidos Como Empacotados	Blocos Reconhecidos Como Não Empacotados (falso negativo)
09c7859269563c240ab2aaab574483dd	upx	93,478%	6,522%
1a9b51a0d07be16bc44a4f8ff6f538fd	upx	70,000%	30,000%
1f06d05ef9814e4cb5202c197710d2f5	upx	69,231%	30,769%
1f171553f1138dc0062a71a7d275055a	upx	93,333%	6,667%
2cffa74f01e50f2fc07d45dbe56561bb	pecompact	86,364%	13,636%
6d12a84c55f20a45c78eb1b5c720619b	pecompact	75,000%	25,000%
8cace33911b71d63fca920cabda3a63a	pecompact	63,636%	36,364%
8e93cdf0ea8edba63f07e2898a9b2147	xcomp97	83,333%	16,667%

MD5 do Executável	Empacotador	Blocos Reconhecidos Como Empacotados	Blocos Reconhecidos Como Não Empacotados (falso negativo)
9a6a653adf28d9d69670b48f535e6b90	xcomp97	87,097%	12,903%
9d1f6b512a3ca51993d60f6858df000d	xcomp97	85,000%	15,000%
41fb70824080b8f9774f688532a89e01	fsg	80,000%	20,000%
378da78d3d3c981b38fb4d10b049d493	fsg	81,250%	18,750%
5723ccbd541e553b6ca337a296da979f	fsg	96,250%	3,750%
97297c74d02e522b6a69d24d4539a359	mew11	85,714%	14,286%
9872199bec05c48b903ca87197dc1908	mew11	76,923%	23,077%
b65a1a4b606ec35603e98d7ca10d09d7	mew11	75,000%	25,000%
b2099fbd58a8f43282d2f7e14d81f97e	mpress	76,190%	23,810%
c07f1963e4ff877160ca12bcf0d40c2d	mpress	82,353%	17,647%
de7cf7de23de43272e708062d0a049b8	mpress	85,185%	14,815%
e8b0a9ecb76aaa0c3519e16f34a49858	cexe	97,273%	2,727%
ecef404f62863755951e09c802c94ad5	cexe	86,667%	13,333%
fbd6b3bb2a40478df5434a073d571cae	cexe	86,364%	13,636%

Tabela 8: Resultado do teste para executáveis empacotados com Themida.

MD5 do Executável	Blocos Reconhecidos Como Empacotados	Blocos Reconhecidos Como Não Empacotados (falso negativo)
01f022417d9a156b3f20fdf2f44c0a03	99,221%	0,779%
4231c10bad00d998dab877b2b773f956	99,385%	0,615%
317211625762902203a6ca9ce1360346	99,392%	0,608%
bd556e1c8ba5ecb55facf78685598eef	99,463%	0,537%
caf0427a22b73226802794adeb7ad36d	99,112%	0,888%
e18b7257604d70ac19efbbd83676d8bc	99,209%	0,791%

Para o caso dos binários originais, vê-se três casos de falsos positivos ao redor de 40 a 45 % dos blocos analisados. Ainda assim, considerando o critério de classificação para empacotamento adotado - 50% de blocos reconhecidos como tal - nenhum dos binários sofreria classificação equivocada.

Já na análise dos binários empacotados, a taxa de falso negativo é menor, atingindo o valor máximo ao redor de 36%. Novamente, com o critério adotado, todos os binários seriam classificados corretamente como empacotados.

Finalmente, pode-se perceber que a aplicação Themida gera artefatos facilmente identificados como empacotados, mesmo não fazendo parte do conjunto de empacotadores utilizados no treinamento da aplicação.

6. Conclusão

O trabalho apresentou uma metodologia de determinação de empacotamento de executáveis através da análise do conteúdo binário destes.

Os dados apresentados demonstram que entre os quatro n-gramas adotados, o que apresentou melhor resultado para a construção da árvore de decisão foi o unigrama, em conjunto com a opção de utilização de boost do algoritmo com 30 passos.

Com estas opções e considerando o critério de classificação para empacotamento adotado - 50% de blocos reconhecidos como tal - a taxa de reconhecimento de executáveis não empacotados e empacotados foi total para o conjunto de teste utilizado.

Além disso, todos os executáveis compactados com o Themida tiveram mais de 99% de seus blocos reconhecidos como compactados, apesar deste compactador não ser utilizado para a base de treinamento da aplicação.

Tais números demonstram a robustez da metodologia adotada.

É importante salientar que o método apresentado, por não utilizar a técnica de assinaturas como a encontrada na ferramenta PEiD, está apto a apontar a utilização de empacotamento mesmo em artefatos que implementem mecanismos para burlar tal técnica.

Para trabalhos futuros, pretende-se ampliar a base de treinamento e testes, com a utilização de outras aplicações de empacotamento além das sete utilizadas.

Além disso, pretende-se investigar o impacto na metodologia descrita da introdução de técnicas de tratamento dos binários, considerando algumas particularidades do formato PE (Portable Executable), antes de submetê-los ao módulo de estatística apresentado.

Referências Bibliográficas

[1] RODRIGUES, R. Febraban: fraudes online somaram prejuízo de R$ 900 mi em 2009. Disponível em: http://computerworld.uol.com.br/seguranca/2010/08/31/febraban-fraudes-online-somaram-prejuizo-de-r-900-mi-em-2009/. Acesso em: 11 jan 2011.

[2] BESTUZHEV, D. Brazil: a country rich in banking Trojans. Disponível em: http://www.securelist.com/en/analysis?pubid=204792084. Acesso em: 11 jan 2011.

[3] MCAFEE. Mcafee Threat Intelligence. Disponível em: http://www.mcafee.com/us/mcafee-labs/threat-intelligence.aspx. Acesso em: 11 jan 2011.

[4] MCMILLAN, R. Was Stuxnet built to attack Iran's nuclear program? Disponível em: http://www.networkworld.com/news/2010/092110-was-stuxnet-built-to-attack.html. Acesso em: 12 jan 2011.

[5] RIEGER, F. Stuxnet: targeting the iranian enrichment centrifuges in Natanz? Disponível em: http://frank.geekheim.de/?p=1189. Acesso em: 12 jan 2011.

[6] SCHNEIER, B. The Stuxnet Worm. Disponível em: http://www.schneier.com/blog/archives/2010/09/the_stuxnet_wor.html. Acesso em: 12 jan 2011

[7] GUO, F., FERRIE, P. CHIUEH, T. A Study of the Packer Problem and Its Solutions. 11th International Symposium On Recent Advances In Intrusion Detection (RAID), 2008, Boston, USA. Anais... Disponível em: https://wiki.smu.edu.sg/w/flyer/images/f/fe/RAID08_Packer.pdf. Acesso em: 22 jul 2011.

[8] JIBZ, QUERTON, SNAKER, XINEOHP., / BOB.PEiD. PEiD. Disponível em: http://www.peid.info/. Acesso em: 21 jan 2011.

[9] KIM, H. C., INOUE, D. ETO, M. TAKAGI, Y. NAKAO, K. Toward Generic Unpacking Techniques for Malware Analysis with Quantification of Code Revelation. Joint Workshop on Information Security (JWIS), 2009, Kaohsiung, Taiwan. Anais... Disponível em: http://jwis2009.nsysu.edu.tw/location/paper/Toward%20Generic%20Unpacking%20Techniques%20for%20Malware%20Analysis%20

with%20Quantification%20of%20Code%20Revelation.pdf. Acesso em: 18 jan 2011.

[10] KANG, M. G., POOSANKAM, P., YIN, H. Renovo: A Hidden Code Extractor for Packed Executables. 2007. 5th ACM Workshop on Recurring Malcode (WORM) 2007, Virginia, USA. Anais... Disponível em:http://bitblaze.cs.berkeley.edu/papers/renovo.pdf. Acesso em: 22 jul 2011.

[11] TABISH, S. M. , SHAFIQ, M. Z., FAROOQ, M. Malware Detection using Statistical Analysis of Byte-Level File Content. In: ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD), Workshop on CyberSecurity and Intelligence Informatics (CSI), 2009, Paris. Anais... Paris: ACM Press, 2009. Disponível em: http://www.cse.msu.edu/~tabishsy/papers/momina_CSIKDD.pdf. Acesso em: 22 jul 2011.

[12] COVER, T. M., THOMAS, J. A. Elements of Information Theory. Hoboken: Wiley, 2006.

[13] RULEQUEST. Data Mining Tools See5 and C5.0. Disponível em: http://www.rulequest.com/see5-info.html. Acesso em: 10 mar 2011.