Authorship Attribution of Obfuscated Binaries

ISCX Information Security Centre of Excellence

Kamran Morovati, Natalia Stakhanova

Faculty of Computer Science, University of New Brunswick

UNB

Abstract

This research aims at identifying obfuscation resistant features of Windows executables at the binary level. These features potentially can facilitate the authorship attribution of unknown programs. The main objective of this study is an analysis of obfuscators effectiveness in order to hide the author's programming style at the binary level. In this study, we have investigated the efficiency of features such as op-code frequencies, op-code n-grams, API function calls, features driven from program's control flow graph and PE header information in order to detect the obfuscation resistant ones.

Software Authorship Attribution

The goal: attribution of unknown software binaries to an author.

Applications:

- Plagiarism detection
- Resolving legal disputes over authorship of work in courts of law
- Authorship of malware, etc. for identification of cybercriminals.
- Software Forensics
- …

Motivation

Source code is typically obfuscated for protection against detection and reverse engineering of binaries.

Scope and Limitations

- PE files for Windows Platform
- .NET assemblies
- C# Programing Language

Data Set

- We collected our samples from Google Code Jam 2013 competition.
- Samples includes submitted codes from both expert and novice programmers.
- 54 projects from 10 different authors in 13 different categories were selected and obfuscated using several methods.

Research objective

Identifying obfuscation resistant features at binary level.

Research Findings

- Use of obfuscators can not fully protect the authors.
- Opcode frequencies and Opcode n-grams are obfuscation resistant
- Choosing a correct classifier and configuring it properly results in better accuracy.
- API calls, PE header info and CFG related parameters could not contribute much in case of author identification .

Methodology

Source Code Collection

- Source codes from Google Code Jam competition
- Source codes for 13 different programming challenges.
- 10 different authors
- Total number of 540 binaries (including 9 different obfuscated versions of each sample)

Generating Binaries

- Generation of Windows binary files
- Compiled with Visual Studio 2015
- No debugging and code alteration to preserve the author's coding style

Applying different obfuscation Techniques

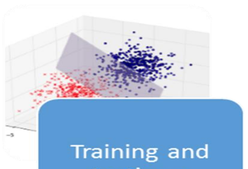
- Generating binaries using *ConfuserEX* software
- 9 different obfuscation techniques
  - CFG obfuscation
  - Invalid Metadata Insertion
  - Variable renaming
  - Anti debug
  - Anti dump
  - Resource protection…

Feature extraction from original and obfuscated binaries

- Op-code Frequencies
- Op-code n-grams
- API Function Calls
- PE Header Info
- CFG
  - #Nodes/Edges
  - #Terminal Nodes
  - #Isolated Nodes
  - Average Node degree
  - # sub graphs

Identification of obfuscation resistant features

- Database Creation
- Use of statistical tests
  - MANOVA test
  - Fisher test
  - Tukey test
- Opcode Patterns
- Calculating Opcode TF-IDF



Training and testing a classifier

- Noise Reduction
- Feature Selection
- Classification
  - SVM
  - Decision Tree
  - KNN
  - Naive Bayes
- Evaluation
  - Accuracy
  - Precision
  - Recall
  - ROC

Op-code n-grams

- Our observations showed the op-code 2-grams TF-IDF values were better inputs for classifiers.
- The Naïve Bayes classifier yielded the best accuracy (94.25%).
- K-NN stands in 2nd place (84% Accuracy).
  - ➢K=3, (K greater than 3 did not produce higher accuracy)
  - ➢Distance Measurement = Cosine Similarity

Number of Inputs (op-code n-grams)	54726
Accuracy	94.25%
Kappa	0.935
Avg. Class Recall	93.21%
Avg. Class Precision	92.54%

Op-code frequencies

- Zero-R classifier (Classification solely by chance) accuracy = ~14%
- Random Forests classifier in combination with Bagging technique yielded the best accuracy (~93%).

Correctly Classified Instances	497 (92.21%)
Incorrectly Classified Instances	42(7.79%)
Kappa statistic	0.9118
Mean absolute error	0.053
Root mean squared error	0.1284
Relative absolute error	29.9741%
Root relative squared error	43.2081%
Coverage of cases (0.95 level)	100%
Mean rel. region size (0.95 level)	42.5046%
Total Number of Instances	539

TP Rate	FP Rate	precision	Recall	F - measure	ROC Area	Class
0.9	0	1	0.9	0.947	1	Author 1
0.45	0.017	0.5	0.45	0.474	0.978	Author 2
0.875	0.022	0.875	0.875	0.875	0.99	Author 3
1	0.002	0.984	1	0.992	1	Author 4
1	0.002	0.98	1	0.99	1	Author 5
0.983	0	1	0.983	0.992	1	Author 6
1	0	1	1	1	1	Author 7
0.82	0.02	0.804	0.82	0.812	0.989	Author 8
0.971	0.002	0.986	0.971	0.978	0.999	Author 9
0.867	0.021	0.839	0.867	0.852	0.988	Author 10
0.922	0.009	0.921	0.922	0.921	0.995	Weighted Avg.

Future Directions

- Analysis of the Frequent Op-codes/API sets and Association Rule Mining.
- Sequential pattern analysis of op-codes and API calls with Time Series and Hidden Markov Models (H.M.M)