Authorship Attribution of Obfuscated Binaries

Kamran Morovati, Natalia Stakhanova
Faculty of Computer Science, University of New Brunswick

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# Programming 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.

Op-code n-grams

Our observations showed the op-code 2-grams TF-IDF values were better inputs for classifiers.

- The Naive 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

<table>
<thead>
<tr>
<th>Number of Inputs (op-code n-grams)</th>
<th>54726</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy</td>
<td>94.25%</td>
</tr>
<tr>
<td>Kappa</td>
<td>0.935</td>
</tr>
<tr>
<td>Avg. Class Recall</td>
<td>93.21%</td>
</tr>
<tr>
<td>Avg. Class Precision</td>
<td>92.54%</td>
</tr>
</tbody>
</table>

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%)

<table>
<thead>
<tr>
<th>Correctly Classified Instances</th>
<th>497 (92.21%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Incorrectly Classified Instances</td>
<td>42 (7.79%)</td>
</tr>
<tr>
<td>Kappa</td>
<td>0.9138</td>
</tr>
<tr>
<td>Mean absolute error</td>
<td>0.053</td>
</tr>
<tr>
<td>Root mean squared error</td>
<td>0.1284</td>
</tr>
<tr>
<td>Relative absolute error</td>
<td>29.9741%</td>
</tr>
<tr>
<td>Root relative squared error</td>
<td>43.2081%</td>
</tr>
<tr>
<td>Coverage of cases (0.95 level)</td>
<td>100%</td>
</tr>
<tr>
<td>Mean rel. region size (0.95 level)</td>
<td>42.5046%</td>
</tr>
<tr>
<td>Total Number of Instances</td>
<td>539</td>
</tr>
</tbody>
</table>

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)