Learning and Clustering Obfuscated Software According to The Obfuscation Transformations Employed to Obtain Them

Master Thesis

Supervisors: Prof. Dr. Alexander Pretschner, Sebastian Banescu
Email: alexander.pretschner, sebastian.banescu @ in.tum.de
Phone: +49 89 289 – 17, 314
Starting date: immediately

Context
Obfuscation is continuously summoned by a myriad of individuals and organizations to protect the internal structure and data of their programs against reverse engineering. In essence, obfuscation is the process of transforming code into an unintelligible form that conceals the internal structure—and sometimes behavior—of an application. Effectively, obfuscation aspire to render the reverse engineering process impractical for the majority of attackers.

Unfortunately, malware authors are also leveraging obfuscation in writing their instances to evade detection by common detection mechanisms and to prolong the time taken to analyze their instances and come up with defensive techniques to contain them. In this context, malware authors deploy layers upon layers of obfuscation transformations regardless of the complexity or performance cost that might yield.

With that in mind, shortening the time taken to reverse engineer a malware instance might reduce the devastation it inflicts. However, this can only be achieved after breaking through the layers of obfuscation utilized by a malware instance in a process known as deobfuscation. Automatic deobfuscation attempt to relieve malware analysts of such burden by automating this process. By and large, automatic deobfuscation techniques adopt obfuscation-specific approaches that are designed to unravel particular obfuscation transformations i.e. those that fit their assumptions.

Goal
The goal of this thesis is to investigate the applicability and effectiveness of machine learning within the field of automatic deobfuscation. One research question is whether the patterns inflicted by obfuscation transformations can be extracted by machine learning. Another research question is whether and which patterns persist after applying multiple obfuscation transformations. To answer these research questions, we plan to generate a dataset of programs obfuscated using different obfuscation transformations and develop an automated clustering technique based on machine learning.

The primary contribution of this approach is saving malware analysts a noticeable amount of time via automatically identifying the sequence of transformations used by an obfuscated instance. With such valuable information in hand, malware analysts can skip directly to transformation-specific deobfuscation techniques. Hence, the “obscurity” (choice) of the obfuscation transformations chosen by malware authors will be eliminated.
Work-plan

1. Develop an understanding of automated deobfuscation techniques and the use of machine learning within similar domains (see references).

2. Design tool and prepare training dataset
   a. Gather a collection of C/C++ programs of varying complexities.
   b. Decide upon list of numerical features to extract from obfuscated programs
   c. Generate programs obfuscated using random combinations of Tigress transformations.

3. Implement the remaining modules.
   a. Implement the feature extraction module.
   b. Design experiments i.e. objective(s), inputs, expected outputs, et cetera.
   c. Implement the evaluation module.

4. Review implementation and re-run experiments

5. The final thesis document must contain:
   a. Description of the problem and motivation
   b. Description of the theoretical background
   c. Description of the proposed approach
   d. Design and implementation details of the tool
   e. Description of the accuracy and validity of the proposed approach
   f. Conclusions and future work.

Deliverables

- Source code of the original and obfuscated C/C++ programs along with their Tigress obfuscation scripts
- Source code of the implemented Python tool
- Technical report with comprehensive documentation of the tool’s design and implementation.
- Final thesis report written in conformance with TUM guidelines.
References


