Chapter 20
Source Code Authorship Analysis For Supporting the Cybercrime Investigation Process

Georgia Frantzeskou
University of the Aegean, Greece

Stephen G. MacDonell
Auckland University of Technology, New Zealand

Efstathios Stamatatos
University of the Aegean, Greece

ABSTRACT

Nowadays, in a wide variety of situations, source code authorship identification has become an issue of major concern. Such situations include authorship disputes, proof of authorship in court, cyber attacks in the form of viruses, trojan horses, logic bombs, fraud, and credit card cloning. Source code author identification deals with the task of identifying the most likely author of a computer program, given a set of predefined author candidates. We present a new approach, called the SCAP (Source Code Author Profiles) approach, based on byte-level n-grams in order to represent a source code author's style. Experiments on data sets of different programming-language (Java,C++ and Common Lisp) and varying difficulty (6 to 30 candidate authors) demonstrate the effectiveness of the proposed approach. A comparison with a previous source code authorship identification study based on more complicated information shows that the SCAP approach is language independent and that n-gram author profiles are better able to capture the idiosyncrasies of the source code authors. It is also demonstrated that the effectiveness of the proposed model is not affected by the absence of comments in the source code, a condition usually met in cyber-crime cases.

DOI: 10.4018/978-1-60566-836-9.ch020

1. INTRODUCTION

Statement of the Problem

With the increasingly pervasive nature of software systems, cases arise in which it is important to identify the author of a usually limited piece of programming code. Such situations include cyber attacks in the form of viruses, Trojan horses and logic bombs, fraud and credit card cloning, code authorship disputes, and intellectual property infringement.

Why do we believe it is possible to identify the author of a computer program? Humans are creatures of habit and habits tend to persist. That is why, for example, we have a handwriting style that is consistent during periods of our life, although the style may vary, as we grow older. Does the same apply to programming? Could we identify programming constructs that a programmer uses all the time? Spafford and Weber (1993) suggested that a field they called software forensics could be used to examine and analyze software in any form, be it source code for any language or executable programs, to identify the author. Spafford and Weber wrote the following of software forensics:

"It would be similar to the use of handwriting analysis by law enforcement officials to identify the authors of documents involved in crimes or to provide confirmation of the role of a suspect"

The closest parallel is found in computational linguistics. Authorship analysis in natural language texts, including literary works has been widely debated for many years, and a large body of knowledge has been developed. Authorship analysis on computer software, however, is different and more difficult than in natural language texts.

Several reasons make this problem difficult. Programmers reuse code, programs are developed by teams of programmers, and programs can be altered by code formatters and pretty printers.

Identifying the authorship of malicious or stolen source code in a reliable way has become a common goal for digital investigators. Spafford and Weber (1993) have suggested that it might be feasible to analyze the remnants of software after a computer attack, through means such as viruses, worms or Trojan horses, and identify its author through characteristics of executable code and source code. Zheng et al. (2003) proposed the adoption of an authorship analysis framework in the context of cybercrime investigation to help law enforcement agencies deal with the identity tracing problem.

Researchers (Krsul and Spafford, 1995; MacDonell et al. 2001; Ding and Samadzadeh, 2004) addressing the issue of code authorship have tended to adopt a methodology comprising two main steps (Frantzeskou. et al 2004). The first step is the extraction of apparently relevant software metrics and the second step is using these metrics to develop models that are capable of discriminating between several authors, using a statistical or machine learning algorithm. In general, the software metrics used are programming language-dependent. Moreover, the metrics selection process is a non trivial task.

With this in mind, our objective in this chapter is to provide a language independent methodology to source code authorship attribution which is called the SCAP (Source Code Author Profile) approach (Frantzeskou. et al 2008, Frantzeskou et al 2007). The effectiveness of the SCAP method is also demonstrated through a number of experiments (Frantzeskou. et al 2006a, Frantzeskou et al 2006b, Frantzeskou. et al 2005a, Frantzeskou et al 2005b)