王威廉

去年的计算机安全著名会议USENIX Security 2015有篇特别厉害的跨界论文De-anonymizing Programmers via Code Stylometry，用机器学习去侦查匿名代码的作者： http://t.cn/RbQNjb4 准确率目测95%以上，感觉写破烂代码的童鞋以后很难深藏功与名了...

De-anonymizing Programmers via Code Stylometry

Aylin Caliskan-Islam
Drexel University

Richard Harang
U.S. Army Research Laboratory

Andrew Liu
University of Maryland

Arvind Narayanan
Princeton University

Clare Voss
U.S. Army Research Laboratory

Fabian Yamaguchi
University of Goettingen

Rachel Greenstadt
Drexel University

Abstract

Source code authorship attribution is a significant privacy threat to anonymous code contributors. However, it may also enable attribution of successful attacks from code left behind on an infected system, or aid in resolving copyright, copyleft, and plagiarism issues in the programming fields. In this work, we investigate machine learning methods to de-anonymize source code authors of C/C++ using coding style. Our Code Stylometry Feature Set is a novel representation of coding style found in source code that reflects coding style from properties derived from abstract syntax trees.

Our random forest and abstract syntax tree-based approach attributes more authors (1,600 and 250) with significantly higher accuracy (94% and 98%) on a larger data set (Google Code Jam) than has been previously achieved. Furthermore, these novel features are robust, difficult to obfuscate, and can be used in other programming languages, such as Python. We also find that (i) the code resulting from difficult programming tasks is easier to attribute than easier tasks and (ii) skilled programmers (who can complete the more difficult tasks) are easier to attribute than less skilled programmers.

1 Introduction

Do programmers leave fingerprints in their source code? That is, does each programmer have a distinctive "coding style"? Perhaps a programmer has a preference for spaces over tabs, or `while` loops over `for` loops, or, more subtly, modular rather than monolithic code.

These questions have strong privacy and security implications. Contributors to open-source projects may hide their identity whether they are Bitcoin's creator or just a programmer who does not want her employer to know about her side activities. They may live in a regime that prohibits certain types of software, such as censorship circumvention tools. For example, an Iranian programmer was sentenced to death in 2012 for developing photo sharing software that was used on pornographic websites [31].

The flip side of this scenario is that code attribution may be helpful in a forensic context, such as detection of ghostwriting, a form of plagiarism, and investigation of copyright disputes. It might also give us clues about the identity of malware authors. A careful adversary may only leave binaries, but others may leave behind code written in a scripting language or source code downloaded into the breached system for compilation.

While this problem has been studied previously, our work represents a qualitative advance over the state of the art by showing that Abstract Syntax Trees (ASTs) carry authorial 'fingerprints.' The highest accuracy achieved in the literature is 97%, but this is achieved on a set of only 30 programmers and furthermore relies on using programmer comments and larger amounts of training data [12, 14]. We match this accuracy on small programmer sets without this limitation. The largest scale experiments in the literature use 46 programmers and achieve 67.2% accuracy [10]. We are able to handle orders of magnitude more programmers (1,600) while using less training data with 92.83% accuracy. Furthermore, the features we are using are not trivial to obfuscate. We are able to maintain high accuracy while using commercial obfuscators. While abstract syntax trees can be obfuscated to an extent, doing so incurs significant overhead and maintenance costs.

Contributions. First, we use *syntactic features* for code stylometry. Extracting such features requires parsing of incomplete source code using a *fuzzy parser* to generate an *abstract syntax tree*. These features add a component to code stylometry that has so far remained almost completely unexplored. We provide evidence that these features are more fundamental and harder to obfuscate. Our complete feature set consists of a comprehensive set of around 120,000 layout-based, lexical, and syntactic features. With this complete feature set we are

2016年01月22日 14:28 来自 iPhone 6 Plus