Poster: Stylometry of Author-Specific and Country-Specific Style Features in JavaScript

Dennis Röllke
Ruhr-Universität Bochum
Dennis.Roellke@ruhr-uni-bochum.de

Aviel J. Stein
Drexel University
ajs568@drexel.edu

Edwin Dauber
Drexel University
egd34@drexel.edu

Mosfiqur Rahman
Drexel University
mr986@drexel.edu

Michael J. Weisman
U.S. Army Research Laboratory
michael.j.weisman2.civ@mail.mil

Gregory G. Shearer
ICF International
gregory.g.shearer.ctr@mail.mil

Frederica Nelson
U.S. Army Research Laboratory
frederica.f.nelson.civ@mail.mil

Aylin Caliskan
Princeton University
aylinc@princeton.edu

Richard Harang
Sophos, Data Science Team
rich.harang@sophos.com

Rachel Greenstadt
Drexel University
rag59@drexel.edu

***Abstract*—Stylometry is the study of writing style, and is often used for authorship attribution. Published papers have shown the usefulness of this technique for code and compiled programs written in languages such as C++. We show that these abstract syntax tree focused techniques can be adapted to JavaScript, an interpreted scripting language common on the World Wide Web. Using Google Code Jam submissions, we show that individual authors from a small suspect set (17 authors) can be attributed with over 99% accuracy. We also presesnt a proof-of-concept experiment which shows that we can differentiate the country of origin of a code author with over 91% accuracy, using Canada and China as our example countries.**

I. Introduction

Stylometry has proven useful for identifying the writers of code and compiled programs written in languages such as C++. We show that similar techniques can be used on JavaScript, using Google Code Jam submissions. JavaScript is an interpreted language commonly used on the World Wide Web.

II. Motivation

The number of attacks and variations on networks via malware increase every year. To pursue those that use such code to attack the privacy and security of others, analysts must find new methods of defense. One of the new ways that cyber-attacks are performed are via JavaScript, which is used in 94.9% of websites on the internet according to W3Techs Web Technology Surveys [1]. The December 2017 McAfee Labs Threat Report showed a steady increase of JavaScript attacks and the second quarter of 2017 set the record for new JavaScript attacks [6].

III. Related Work

One of the means that security analysts pursue to stop and prevent further attacks is to find and identify criminals, which in the case of malware can be understood as the writers of the malicious code. One approach to attribute authors is the employment of stylometry. Modern techniques work by analyzing the source code of malicious software and comparing to known work by a group of suspect authors. Using abstract syntax trees (AST) from source code in C++ previous work was able to attribute authors of data sets of sizes 1,600 authors and 250 authors with 94% and 98% accuracy respectively [3]. This was much higher than had previously been achieved. Similar techniques were used to identify authors of both C++ code and Python code with accuracy above 80% with long-term short term memory (LSTM) models [2]. It has also been applied to executable binaries of C++ code on datasets of 100 authors with 96% accuracy and datasets of 600 authors with 83% accuracy [4].

The aforementioned work examines author attribution for C/C++ software, which is commonly used for server exploits. In this scenario, it happens that source code may remain on the attacked system after it is compiled. Forensic means can discover the injected code and can then further analyze it using various techniques, including stylometry. In the modern environment, web-application attacks are another common type of attack. Many of these are written in JavaScript, a programming language that highly differs from C/C++. In contrast to C/C++, JavaScript is an interpreted scripting language, rather than a compiled language. Further, the JavaScript landscape is very diverse, with a great variety of frameworks, libraries and flavors available. Therefore, it is important to empirically

Network and Distributed Systems Security (NDSS) Symposium 2018
18-21 February 2018, San Diego, CA, USA
ISBN 1-1891562-49-5
http://dx.doi.org/10.14722/ndss.2018.23xxx
www.ndss-symposium.org

determine if these existing techniques are applicable to this different variety of programming language.

IV. Methodology

Our first task was to adjust the machine learning algorithm capable of attributing C/C++ to attribute JavaScript. This means adapting the features used for learning on C/C++ code to features which can be used to learn on JavaScript. The algorithm is trained by collecting features on three levels: lexical, layout and syntax. A translation of lexical and layout features is straight-forward. Syntactic feature extraction, which focuses on the AST, is both critical to the success of these methods and is highly language dependent. We used the JavaScript parser Esprima to generate the AST for our JavaScript programs [5]. Figure 1 shows the overall workflow.

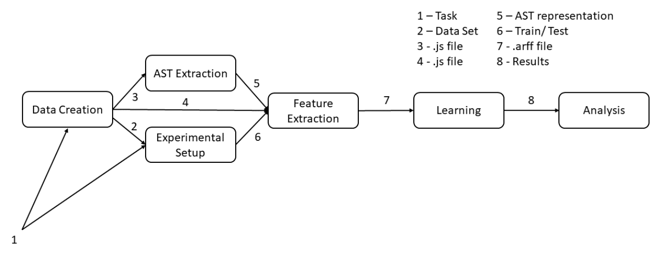


Fig. 1. This summarizes the stylometry workflow. Given an attribution task, we build a dataset, setup experiments and extract ASTs, extract feature vectors, do the machine learning, and then analyze the results.

To train and test our machine learning algorithm we decided to gather data from Google Code Jam (GCJ), an ideal source of data sets, given the free-to use policy under which contestants submit their assignments: "Your submitted source code may be made available for anyone to view on the Internet and download and use at the end of the Contest."[1] Another advantage of GCJ data is the availability of statistical information, such as popularity of programming language, contestants skill-levels, and nationalities.

V. Results

We used the data files organized by author and trained the system by collecting features of the code and associating each with its authors. These features were extracted by analyzing abstract syntax trees derived from the JavaScript code. We then applied learning algorithms to these to find associations. This technique was developed before for C/C++ but we show that by replacing the AST extraction component and modifying the feature extraction component it can be adapted to JavaScript. Our dataset contained 242 files over 17 authors. We performed a 9-fold cross-validation and achieved 99.1% accuracy.

Within our international team, we noticed that sentence structure is one of the main challenges when communicating in a foreign language. Sentence structure is syntax in native language, and it is well understood that this is an important feature for attribution in natural language, and is often used to determine the origin of the author. Therefore, we hypothesize that our features, especially the syntactic features, can also determine the origin of the authors of code. For a proof-of-concept experiment, we selected two countries with a large amount of JavaScript GCJ data: Canada and China. Instead of individual authors from those countries, we combined the authors from Canada into one class and from China into another, with a combined total of 84 files. We performed 4-fold cross-validation and obtained an accuracy of 91.9%.

VI. Conclusion and Future Work

While still small scale and only proof-of-concept, our results support our hypotheses that we can attribute JavaScript code using the same techniques developed for other languages and that we can attribute code to the origin of the programmer using stylometric techniques. The underlying hypothesis behind origin attribution is that our native language and culture influences our thought process, how we think about problems, and the way in which we solve them. While this is interesting itself, our findings also have some other interesting implications. Lately there have been concerns about entities launching international cyber-attacks. The ability to attribute these attacks to the country of origin can be helpful because it gives insight into the attacker's means and motives.

The obvious future direction, for which we are currently building a dataset, is to extend this work to actual malicious code. Expanding the method to other languages used in many modern attacks is also of interest.

Acknowledgment

The authors would like to thank DARPA Contract No. FA8750-17-C-0142 and the United States Army Research Laboratory for funding this research.

References

[1] Usage of javascript for websites, Jan 2018.
[2] Bander Alsulami, Edwin Dauber, Richard Harang, Spiros Mancoridis, and Rachel Greenstadt. Source code authorship attribution using long short-term memory based networks. In *European Symposium on Research in Computer Security*, pages 65–82. Springer, 2017.
[3] Aylin Caliskan-Islam, Richard Harang, Andrew Liu, Arvind Narayanan, Clare Voss, Fabian Yamaguchi, and Rachel Greenstadt. De-anonymizing programmers via code stylometry. In *24th USENIX Security Symposium (USENIX Security), Washington, DC*, 2015.
[4] Aylin Caliskan-Islam, Fabian Yamaguchi, Edwin Dauber, Richard Harang, Konrad Rieck, Rachel Greenstadt, and Arvind Narayanan. When coding style survives compilation: De-anonymizing programmers from executable binaries. *arXiv preprint arXiv:1512.08546*, 2015.
[5] Ariya Hidayat. Esprima.
[6] Niamh Minihane, Francisca Moreno, Eric Peterson, Raj Samani, Craig Schmugar, Dan Sommer, and Bing Sun. Mcafee labs threat report december 2017, Dec 2017.

[1]https://code.google.com/codejam/

Stylometry of Author-Specific and Country-Specific Style Features in JavaScript

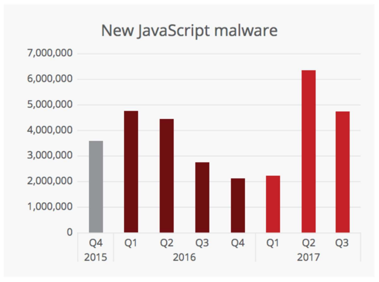
Dennis Röllke[2], Aviel J. Stein[1], Edwin Dauber[1], Mosfiqur Rahman[1], Michael J. Weisman[3], Gregory G. Shearer[4], Frederica Nelson[3], Aylin Caliskan[5], Richard Harang[6], Rachel Greenstadt[1]

1. Drexel University 2. Ruhr-Universitat Bochum 3. U.S. Army Research Laboratory 4. ICF International 5. Princeton University 6. Sophos Data Science Team

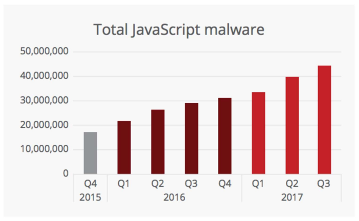
Motivation

The number and variations of attacks on networks via malware increase every year.

Using abstract syntax trees (AST) from source code in C++ pervious work was able to attribute authors of data sets of size 1,600 and 250 with 94% and 98% accuracy respectively. This was much higher than had previously been achieved.



Source: McAfee Labs, 2017.



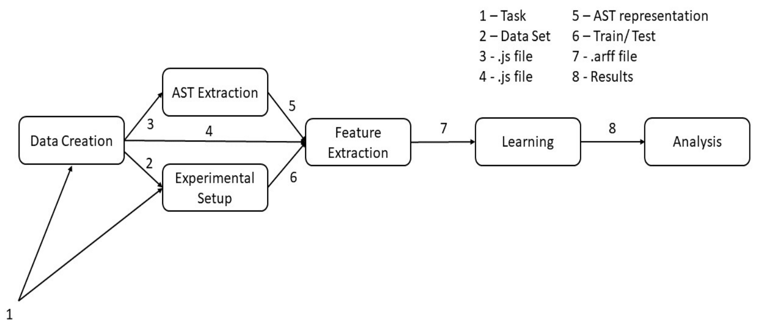
Source: McAfee Labs, 2017.

Web-application attacks are common. Thus, we evaluated the applicability of the former shown methodology to JavaScript, a programming language that highly differs from C/C++.

In the current climate, there is great concern about international cyber-attacks. Therefore, in addition to attributing authors we also decided to attempt attribution to country of origin.

Methodology

To train our Machine Learning algorithm, we decided to gather data from Google Code Jam (GCJ), an ideal source of data sets, given its free-to use policy.

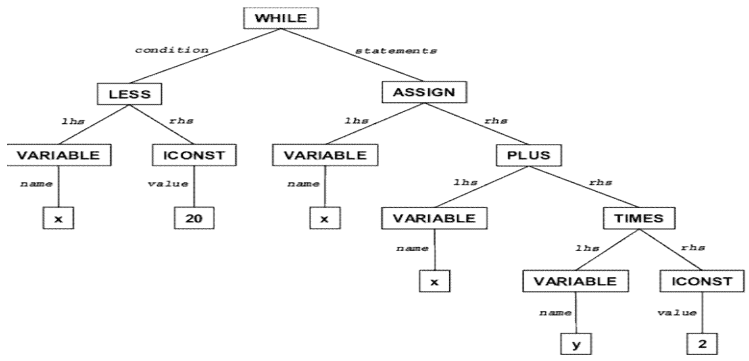


Our first task was to adjust the machine learning algorithm capable of C/C++ attribution to cover JavaScript. This means adapting AST and feature extraction from C/C++ to JavaScript.

References

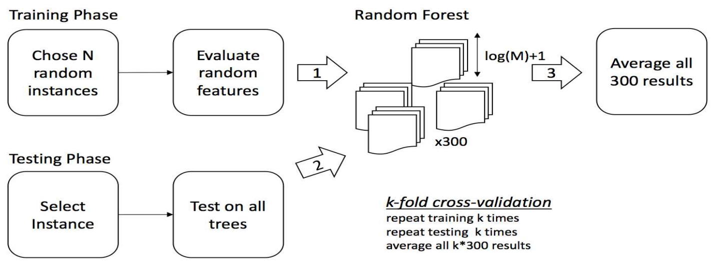
1. Caliskan-Islam, Aylin, et al. "De-anonymizing programmers via code stylometry." 24th USENIX Security Symposium (USENIX Security), Washington, DC. 2015.
2. Alsulami, Bander, et al. "Source Code Authorship Attribution Using Long Short-Term Memory Based Networks." European Symposium on Research in Computer Security. Springer, Cham, 2017.
3. Caliskan-Islam, Aylin, et al. "When coding style survives compilation: De-anonymizing programmers from executable binaries." arXiv preprint arXiv:1512.08546 (2015).
4. Fritzson, Peter & Privitzer, Pavol & Sjölund, Martin & Pop, Adrian. (2009). Towards a Text Generation Template Language for Modelica, 10.3384/ecp09430124. (Fig 3)
5. McAfee Labs Threat Report, December 2017
6. Ariya Hidayat. Esprima

The technique requires collection of features on three levels: lexical, layout and syntactic. While translating lexical and layout features is straightforward, JavaScript requires different AST parsers than C/C++, and thus additional feature translation. As these features are the most important to the success of the technique, this is the most important step of porting the algorithm from one language to another.



Syntax analysis is based on constructing an Abstract Syntax Tree (AST), a lossless and unique representation of source code as a tree. Former employed AST parsers for C/C++ are not capable of parsing JavaScript code and thus had to be replaced.

After implementing this step based on Esprima, we successfully showed JavaScript authorship attribution based on random forest machine learning algorithms.



Results

Author: Our dataset contained 242 files written by 17 authors. We ran several tests with training and testing separated by a 9-fold cross-validation and achieved 99.1% accuracy.

Country: We looked at files written by programmers from Canada and China, which had a combined total of 84 files. Our 4-fold cross-validation showed an accurate distinguishability of 91.9%.

Conclusion

This supports the case that our culture and language effect how we think and therefore how we code.

We plan to use similar techniques to see if we can detect malware family grouping based on source code, as well as if we can still succeed at authorship attribution on actual malware.

Acknowledgements

The authors would like to thank DARPA Contract No. FA8750-17-C-0142 and the United States Army Research Laboratory for funding this research.