# Authorship Attribution of Compiled Malware

## Abstract

In the mostly anonymous world of the Internet, hackers and cyber criminals may often perpetrate crimes with little to no traces leading back to them. Often this is done using malware that the criminal has created. However, researchers can often obtain copies of this malware, and identifying information may be present within these binaries, even if only unintentionally by the authors. Within this project, I analyze various malware binaries using authorship attribution techniques to determine if there are features that can set malicious binaries apart from benign binaries, or families of malware from other malware. We use n-n-gram analysis of the binaries’ bytes, as well as frequency distribution of opcodes and n-gram analysis of opcodes to show that there is an identifying feature set that can be taken advantage of to show distinct authorship between files.

## Introduction

Since the advent of the Internet, humanity has found many ways to use it to increase their productivity, comfort, and enjoyment. However, with all of these benefits we've reaped, some have dedicated their efforts towards undermining others by creating and distributing malware. Anti-virus company Symantec has claimed to have detected over a million unique pieces of malware as of 2008, and the amount of malware has only increased since then [Symantec, 2008].

The fact that these digital attacks can originate from any computer on the globe, with plausible deniability for the owner of that computer, being able to attribute these attacks back to the people who created them would be a useful technique for law enforcement officers and security researchers, on par with fingerprinting and handwriting analysis for attributing physical crimes to criminals. The volume of these digital attacks has steadily increased each year, and thus, automated methods for associating these tools to authors must be created, as attribution of all of these attacks by individual malware researchers is infeasible.

## Background

### Authorship Attribution

The quantitative study of authorship is known as stylometrics. This field has been used in literary fields since 1887 in analyzing the works of Shakespeare [Mendenhall, 1887]. This experiment used word length analysis of Shakespeare’s and Francis Bacon’s work to give evidence that Shakespeare was not simply a pen name used by Bacon. While the semantic aspect of these documents was never analyzed in this study, the lexical information alone proved to be enough evidence to support the theory that Shakespeare and Bacon were two distinct individuals.

This counter-intuitive notion that the meaning of words is of little consequence appears frequently in authorship attribution. One of the key problems that causes this is that if all of a document’s data is used in analysis, much of the document’s domain-specific data will cause a natural skew even when the author is the same. For example, if one author’s story contains several characters, and the author writes a different story with an entirely different set of characters, the disjoint of names might cause the analysis to conclude that there are two authors, or that one of the stories is written by another author who has a story that coincidentally has the same named characters [Binongo, 2003]. Thus, while the addition of more test data is often useful, we must also exercise feature selection to attempt to condense our data to the most useful subset of data.

The field gained more traction with an analysis of the disputed articles in The Federalist Papers [Mosteller & Wallace, 1963]. James Madison, Alexander Hamilton, and John Jay were all believed to be the authors of some of these papers published in The Federalist. Researchers used Bayesian analysis to create models of what each of the three authors undisputed works looked like, then applied these models to the disputed papers to assert which were written by each man, and even which were collaborations between each man. Their analysis again eschewed semantic information because of domain issues, and instead chose to focus on what they called “filler words,” such as articles, prepositions, and conjunctions. The frequency of the usage of these words was enough information to disambiguate these three authors. Later research using other methods corroborates these results [Tweedie et al., 1996].

Another study performed more recently that analyzed the authorship of the 15th book in the Oz series of books. L. Frank Baum passed away during the writing of the 15th book, and the authorship of the subsequent books in the series fell to Ruth Thompson. Many believed that the 15th book should be credited to Thompson instead of Baum, which occurred initially. Again, researchers ignored the semantics of the text, instead focusing on what they decided to call “function words” such as auxiliary verbs, pronouns, prepositions, articles, conjunctions and degree adverbs [Binongo, 2003]. They then performed an additional step of principal component analysis (PCA) to reduce the many variables into a two-dimensional plane for simple visual analysis by humans. Their visualizations show a clear and distinct difference between Baum’s and Thompson’s works, and the 15th book of Oz falls solidly in the area of Thomspon’s works.

### N-gram Analysis

At its core, authorship attribution is a specialized area of document comparison, focusing on the source of the documents, rather than the emotion, topic, or complexity of the document. Thus, we expect that many techniques that are applicable to document comparison might be applicable in some way to binary authorship attribution.

One simple, yet effective, technique that is often used is known as n-gram analysis. The document is parsed into overlapping token groups of size n. The tokens are usually words or characters, but other tokens have been used in some areas of research. Frequency counts of the n-grams of a document are kept and normalized to the unit vector for comparison to other documents’ sets of n-grams. The similarity comparison is also applicable in cases when there are small changes in words or phrases between the two documents. For example, consider the words “described” and “describes” and their character 3-grams. All of their 3-grams will be the same until the last 3-gram, when one will be “bed” and the other will be “bes.” Thus, even though they are different words, there is still a high degree of similarity between the two words.

N-gram analysis has been used in authorship attribution with great success [Kešelj et al., 2003], and has been used in other similar fields such as text categorization with comparable success [Cavnar & Trenkle, 1994]. It is an effective and efficient tool that has proven useful in numerous comparative cases.

### Source Code Analysis

## Methodology

### Datasets Used

For the data in this project, a collection of viruses from the Zeus family of malware was obtained. We reduced this dataset to only those that were uncompressed, post-installation binaries compiled for an 8086 architecture on Windows. From this set, we randomly chose twenty viruses. Next, we added in 16 Windows executables and DLLs ranging across a wide variety of different functions. Finally, we added in three viruses all from a similar source, all compiled for an 8086 architecture on Windows. We feel that this is a dataset of an acceptable size in light of the fact that many digital attacks are very targeted and the most successful pieces of malware are used infrequently and with great precision, so many instances of an author’s work would be difficult to procure.

### N-gram Analysis of Binaries

From a general human perspective, a compiled binary (which is how most malware is seen) is essentially ciphertext. However, this is not the case, the method to change source code into a compiled binary is essentially a lossy encoding scheme. While a good encryption algorithm would create a different cipher for a program’s function in different places in a program, a compiler would create the same machine code for that function regardless of where it is placed in the source code. However, information like variable names and comments would be lost in the transition to machine code.

There is an underlying structure to the machine code that the computer can read, but it is not readily apparent to the naked eye. Most code attribution studies have used uncompiled source code [Frantzeskou et al., 2006; Spafford & Weeber, 1992; Krsul & Spafford, 1997; Lange & Mancoridis, 2007] as their dataset, since it is the actual text that the author created. Compiling creates a layer of obfuscation between the author's original text and the data at hand. However, unlike a cipher text, this compiled data has a roughly deterministic relationship to the source code, and thus, can be used as if it were the author's original text with only a small loss in accuracy.

This compiled data is in the form of machine code, which lacks much of the stylistic choices such as names of functions and variables (though some of this data can still be recovered, as described later), but still retains the functional choices such as program organization and algorithm implementation.

### Opcode Frequency Distribution

Malware is the same as any other program on a computer, except for its malicious activity. This malicious activity might manifest in a different distribution of operations for the CPU to perform. For example, a piece of malware might have many calls to interrupt operations in order to disrupt program flow and obtain control of a privileged process. If it is concerned with cryptographically securing information before sending it back to some sort of controller, there might be an above-average amount of shift operations and operations to send data to external devices.

These opcodes are derived from the same dataset as above by simply taking a variable length subset of the bytes. We will use only the opcodes, not the registers and other arguments passed to the opcodes. We believe that this feature selection process will provide us with a more effective disambiguation of the data into their respective classes.

For ease of programming, we use the Unix objdump utility to obtain a textual representation of all of these opcodes from the code section of the programs. All programs are compiled for the same architecture (8086) so they all draw from the same set of CPU instructions.

### N-gram Analysis of Opcodes

## Results

### N-gram Analysis of Binaries

The n-gram analysis

### Opcode Frequency Distribution

|  |  |  |  |
| --- | --- | --- | --- |
| **Frequency Distribution of Opcodes** | | | |
|  | Zeus | Other Viruses | Benign |
| Zeus | 0.717204086 | 0.726461271 | 0.663736147 |
| Other Viruses | 0.726461271 | 0.92439666 | 0.823485286 |
| Benign | 0.663736147 | 0.823485286 | 0.876942823 |

### N-gram Analysis of Opcodes

## Conclusions

## Future Work

Future work in this field will become more important as digital warfare becomes more widespread and damaging. Hackers will escalate their tools and techniques to counteract whatever countermeasures security researchers come up with, so security researchers should always be creating newer and better techniques. One area that we believe has promise is in analyzing the text strings found within files. While there are some counters to this analysis already, such as the munging of important strings, we believe that many malware creators will not bother to create new munging techniques or keys for each new piece of malware or version of malware. Thus, we will begin to see patterns in the ciphertext that is contained within a program.

Methods that could assist and automate aspects of a security researcher’s analysis of malware would also be an excellent focus for future work. Though the cost of analyzing each piece of malware is prohibitively expensive, the most important and devastating pieces of malware will be analyzed by humans, and any analysis of their process and subsequent improvement of this work would be a boon for security research.

## Acknowledgments

The author wishes to thank his advisor, Charles Nicholas, for his assistance in creating this research paper.

## Works Cited

Binongo, J. N. G. (2003). Who wrote the 15th book of Oz? An application of multivariate analysis to authorship attribution. Chance, 16(2), 9-17.

Burrows, S., Uitdenbogerd, A. L., & Turpin, A. (2009, January). Application of information retrieval techniques for source code authorship attribution. In Database Systems for Advanced Applications (pp. 699-713). Springer Berlin Heidelberg.

Cavnar, W. B., & Trenkle, J. M. (1994). N-gram-based text categorization. *Ann Arbor MI*, *48113*(2), 161-175.

Frantzeskou, G., MacDonell, S., Stamatatos, E., & Gritzalis, S. (2008). Examining the significance of high-level programming features in source code author classification. Journal of Systems and Software, 81(3), 447-460.

Frantzeskou, G., Stamatatos, E., Gritzalis, S., & Katsikas, S. (2006, May). Effective identification of source code authors using byte-level information. In Proceedings of the 28th international conference on Software engineering (pp. 893-896). ACM.

Houvardas, J., & Stamatatos, E. (2006). N-gram feature selection for authorship identification. In *Artificial Intelligence: Methodology, Systems, and Applications* (pp. 77-86). Springer Berlin Heidelberg.

Kešelj, V., Peng, F., Cercone, N., & Thomas, C. (2003, August). N-gram-based author profiles for authorship attribution. In *Proceedings of the conference pacific association for computational linguistics, PACLING* (Vol. 3, pp. 255-264).

Krsul, I., & Spafford, E. H. (1997). Authorship analysis: Identifying the author of a program. *Computers & Security*, *16*(3), 233-257.

Juola, P. (2006). Authorship attribution. Foundations and Trends in information Retrieval, 1(3), 233-334.

Lange, R. C., & Mancoridis, S. (2007, July). Using code metric histograms and genetic algorithms to perform author identification for software forensics. In *Proceedings of the 9th annual conference on Genetic and evolutionary computation* (pp. 2082-2089). ACM.

Mendenhall, T. C. (1887). The characteristic curves of composition. Science, (214S), 237-246.

Mosteller, F., & Wallace, D. L. (1963). Inference in an authorship problem: A comparative study of discrimination methods applied to the authorship of the disputed Federalist Papers. *Journal of the American Statistical Association*, *58*(302), 275-309.

Pektas, A., Eris, M., & Acarman, T. (2011, August). Proposal of n-gram based algorithm for malware classification. In SECURWARE 2011, The Fifth International Conference on Emerging Security Information, Systems and Technologies (pp. 14-18).

Spafford, E. H., & Weeber, S. A. (1993). Software forensics: Can we track code to its authors?. *Computers & Security*, *12*(6), 585-595.

Stamatatos, E. (2009). A survey of modern authorship attribution methods. Journal of the American Society for information Science and Technology, 60(3), 538-556.

Tweedie, F. J., Singh, S., & Holmes, D. I. (1996). Neural network applications in stylometry: The Federalist Papers. *Computers and the Humanities*, *30*(1), 1-10.

Symantec Corporation. (2008). Trends for July-December 07. *Symantec Global Internet Security Threat Report*.