Intellectual Property Today™

RFC Express™ New Patent Lawsuits 14-Day Trial Click Here

What, Exactly, Is Software Trade Secret Theft?

By Bob Zeidman and Nikolaus Bear

Bob Zeidman is the president of Zeidman Consulting, a contract R&D firm. He is an experienced expert witness and the developer of CodeSuite®, which is now supported by SAFE Corporation. He can be reached via email at Bob@ZeidmanConsulting.com.

Nikolaus Baer is a research engineer at Zeidman Consulting. He has utilized the CodeSuite® software as an expert witness in litigation involving trade secret theft. He can be reached via email at Nik@ZeidmanConsulting.com.

Introduction

Through our experience working on numerous high tech intellectual property disputes, especially software copyright infringement and trade secret theft cases, we have developed an exact methodology for determining certain types of trade secret theft with regard to software. It has been our observation that experts often rely heavily on the discovery of similarities in source code. These experts often use computer code analysis applications to locate portions of code that look similar and then draw conclusions based upon their analysis of these portions of code. Unfortunately, this method of analysis can vary greatly and can result in the simplistic and incorrect conclusion that mere similarities in themselves establish trade secret theft.

The CodeMatch® program that we use is one of several tools available to experts in the CodeSuite® suite of software analysis tools. It applies a series of algorithms to find correlations between two sets of software source code files based upon several elements of the code and also provides a ranking to help better determine the parts of the code on which to focus (a detailed explanation of the CodeMatch algorithms is given on the website of Software Analysis and Forensic Engineering Corporation at www.SAFE- corp.biz). However, the discovery of these sections of interest is only the first step of analysis, as there are many reasons for software correlation, and it is the job of experts to eliminate other possible explanations before drawing any conclusions about theft. The steps for determining source code trade secret theft are:

1. **Determine trade secrets**. This step requires the owner of the software to determine the intellectual property that comprises their trade secrets and then to point out those sections of code that implement them. The expert must confirm that these sections of code do represent real trade secrets.
2. **Determine source code correlation**. This step requires the determination of similarity between the two sets of code, one containing the trade secrets and the other accused of containing those same trade secrets. Source code correlation is a quantitative measure of software similarity that is defined in the next section.
3. **Determine reasons for correlation**. There are six reasons that source code can be correlated. Of these reasons, some reasons automatically rule out trade secret theft. Other reasons require further analysis.
4. **Determine functionality correlation**. Next, the correlation must be examined to

determine whether actual functionality of the software is correlated.

5. **Determine trade secret theft**. If functional correlation is present, the final step involves verifying that the correlated functions are truly trade secrets and producing a definitive decision about whether that correlation is due to trade secret theft. A public domain literature search is often part of this process, because the information a company treats as confidential is not always so.

Determining trade secrets

Under the Uniform Trade Secrets Act and the Restatement formulations that still provide the law in a few states, a trade secret is information valuable to competitors that is not generally known to others in the industry and that a company has taken precautions to keep secret from its competitors. General or public knowledge cannot be claimed as a trade secret. With regard to computer source code, open source code, third party code, and generally known algorithms cannot be considered trade secrets. Determining trade secrets in computer source code requires gauging first, whether the source code in question is known outside the company and second, how much effort the company puts into protecting the source code.

Two or more parties can claim the same trade secret if each one developed it independently and kept it a secret. For example, researchers at two companies may build on existing video compression techniques to develop a new, more efficient technique. As happens often in the fields of engineering and science, these two techniques use the same algorithms because they are extensions of existing algorithms and both solve the same problem. The two researchers had the same "eureka" moment. Rather than patenting the algorithm, which would require the companies to make them public and would result in time and legal fees for getting through the patent office, the companies decide to keep the algorithms secret. Therefore, similarity of information does not alone signify a trade secret. Along similar lines, employees who change jobs are free to transfer their general skills, knowledge, and experience, and that may translate into different companies using software that shares basic and general similarities. This is why looking only for source code similarities is a faulty method for determining trade secret theft.

Two lines of source code can be similar or even identical, but if the functionality isn't significant then it is not a trade secret because the code contains nothing that is not obvious or easily understood by one of ordinary skill in the art of computer programming. The definition of a trade secret means that finding a correlation between the source code files of two different programs does not necessarily mean that illicit behavior has occurred. Finding the correlations is the next step in determining trade secret theft, but further analysis must be usually performed before conclusions can be drawn.

Determining Source Code Correlation

Source code correlation is a quantitative, repeatable, deterministic measure of the similarity of software source code. Source code correlation takes the most important elements of a program and finds the amount of similarity of each of these elements independently. The overall correlation is then a combination of the individual correlations for each element. In theory, these elements could be any elements that are determined to be important. In practice, these elements are:

- **Statements.** Statements are lines of code that have functionality. If a computer program is a recipe for the computer to follow, statements are the steps of the recipe like, "Mix the flour and water in a large bowl."
- **Comments.** Comments are descriptions for the reader of the source code but have no functionality. In the recipe metaphor, a comment is a description like, "This step will add flavor to your cake."
- **Identifiers.** Identifiers are names of objects in the code such as data structures and routines. Again, using the recipe metaphor, identifiers would be the names of the ingredients, like "flour," "water," and "sugar."
- **Instruction sequences.** Instructions are basic operations for the computer to perform. The sequence of instructions is the basic steps of the program in the order that they are performed. In a recipe, the instruction sequence might be "add, add, add, mix, pour, bake, cool."

The most widely used tool for determining source code correlation in litigation is CodeMatch, which is part of a suite of software analysis tools called CodeSuite, available from Software Analysis and Forensic Engineering Corporation. CodeMatch allows the user to select which elements of source code should be used to determine correlation. For determining copyright infringement, all elements are important. For determining trade secret infringement, statement correlation and instruction sequence correlation are particularly important because these represent actual functionality whereas comment correlation and identifier correlation represent nonfunctional code, though they often point to important sections of code that might otherwise be missed.

Determining reasons for correlation

High correlation between two sets of software source code can be attributed to several factors. These factors are:

- **Common Algorithms.** An algorithm is a procedure or a set of instructions for accomplishing some task. In one programming language there may be an easy or well- understood way of writing a particular algorithm that most programmers use. For example there might be a way to alphabetically sort a list of names. Perhaps this algorithm is taught in most programming classes at universities or is found in a popular programming textbook. These commonly used algorithms will show up in many different programs, resulting in a high degree of correlation between the programs even though there was no direct contact between the programmers.
- **Common Identifier Names.** Certain identifier names are commonly taught in schools or commonly used by programmers in certain industries. For example, the identifier *result* is often used to hold the result of an operation. These identifiers will be found in many unrelated programs and will result in these programs being correlated.
- **Third-Party Source Code.** It is possible that widely available open source code is used in both programs. Also, libraries of source code can be purchased from third-party vendors. If two different programs use the same third-party code, the programs will be correlated.
- **Code Generation Tools.** Automatic code generation tools, such as Microsoft Visual Basic or Adobe Dreamweaver, generate software source code that looks very similar with similar and often identical elements. The structure of the code generated by these tools tends to fit into specific templates with identifiable patterns. Two different programs that were developed using the same code generation tool will be correlated.
- **Common Author.** It is possible that one programmer, or “author,” will create two programs that have correlation simply because that programmer tends to write code in a certain way. This is the programmer’s style of coding. Thus two programs written by the same programmer can be correlated due to the style being similar even though there was no copying and the functionality of each program is different than that of the other.
- **Copied Code (Authorized or Plagiarized).** Code was copied from one program to another, causing the programs to be correlated. The copying may have taken place for only certain sections of the code and may include small or significant changes to the code. When each of the previous reasons for correlation has been eliminated, the reason that remains is copying. If the copying was not authorized by the original owner, then it comprises plagiarism.

Once correlation is found, it is important for an expert to determine which of the above reasons account for the correlation. At that point, the analysis continues into a new phase to determine functional correlation.

Determining Functionality Correlation

The next phase in determining theft of trade secret rests upon the definition of a trade secret and the functionality of the code in question. Correlation due to how the code functions is called functional correlation. For each of the six reasons for correlation, we can look further to determine whether there is also a functional correlation. Some reasons for correlation can be used to rule out trade secret theft while other reasons require still further analysis.

A. Common Algorithms

Two programs that are correlated due to the fact that they both use algorithms that are commonly used in the industry or by programmers in general is a form of functional correlation. However, because these algorithms are commonly used and understood, this type of correlation should not be taken as proof of trade secret theft.

B. Common Identifier Names

Identifiers have no inherent functionality and therefore cannot be a factor in functional correlation. In addition, the use of common identifiers that are widely used throughout the industry or by programmers in general cannot be a form of functional correlation. In cases where software is correlated only because of common identifiers, trade secret theft can usually be ruled out.

C. Third Party Source Code

When the correlation of two programs is due to the use of third party code, such as open source code or purchased libraries of source code, there is no trade secret theft involved (unless of course the third party code contains trade secrets). There may still be a concern, however, if a programmer has stolen a third party's confidential code that was licensed to his former employer. In that case, the third party may have its own trade secret concerns with the programmer.

D. Code Generation Tools

The structure of the code generated by automatic code generation tools tends to fit into specific templates with identifiable patterns. If the correlation between two different programs is solely because they were developed using the same code generation tool, they will be correlated but not due to trade secret theft. Note, however, that the code must be further examined to determine whether the automatically generated code performs the same function in the same way, which means that the code is functionally correlated. If the programmer used enough expertise in guiding the code generation tool to generate the code, it could still comprise a trade secret.

E. Common Author

Two programs can be correlated because they were both written by the same programmer or programmers. If the correlation between two different programs is solely because they were developed by the same people, they will be correlated but not due to trade secret theft. Note, however, that the code must be further examined to determine whether the code also performs the same function in the same way, which means that the code is functionally correlated. Common authorship can be a hint that the functionality will also correlate, but does not necessarily mean that is the case.

F. Copying

When all other reasons for correlation have been eliminated, what remains is copying. Obviously, correlation due to copying means that the two programs are also functionally correlated.

Once functional correlation is found, the next step is for an expert to determine whether that functional correlation is due to trade secret theft.

Determining Trade Secret Theft

The question at this point is whether the functions that the code is performing are actually unique to the company claiming the trade secrets or whether they are within the general knowledge of software engineers. Anything that is general or public knowledge cannot be claimed as a trade secret. This also means that if the functionality isn't significant then it might not be a trade secret.

Finally, a trade secret does not mean that there can only be one owner of the proprietary information. Unlike with a patent, two or more parties can claim the same trade secret on the same code if each one developed it independently and kept it a secret. Two parties could even have identical trade secrets. Therefore, functional correlation does not alone signify a trade secret, and a proper expert examination should also include an analysis of how development occurred and what means the owner used to keep the code secret. The development process and the effort to protect code can be very important in determining trade secrets and trade secret theft.

Conclusion

Because trade secrets can be claimed at many levels of abstraction, trade secret theft may occur in uncorrelated code because the architecture or high-level design of the code, rather than the code itself contains trade secrets. That aspect of trade secret theft is beyond the scope of this article, and it should not be understood that source code correlation is the only way to uncover trade secret theft. However, when dealing with trade secrets in the source code, source code correlation is a very valuable tool.

Most experts use some form of software analysis tool to examine code for trade secret theft, but a lack of a common methodology makes it difficult to draw and discuss succinct conclusions. By viewing the discovery of correlation as an initial step in a method for determining trade secret theft, experts will be able to more quickly and accurately analyze cases and draw more objective conclusions. Experts can move to the essence of a trade secret case by running correlated sections of source code through the various tests described in this article.

The proper use of this methodology will make trade secret theft cases more straightforward and objective. Without the use of correlation tools and a standard for interpreting the results, cases can end up disorganized and subjective. Instead of being able to succinctly narrow a mass of raw source code down to the essence of the trade secret arguments, experts and lawyers can get lost in argument over which metrics to use in order to examine the code; the case can deteriorate into confusing interpretation and spin. Using the proper methodology helps direct the findings in a manner that all parties can agree upon. The relevant code will quickly become apparent. When experts on both sides of litigation use the same methodology, the case is simplified and streamlined because it is based upon the merits of their arguments applied against a set of standards. Correlation tools and standards will enable experts and lawyers to better utilize their skills and knowledge and better support their clients.