

Systems and Methods for Detecting Copied Computer Code Using Fingerprints

BACKGROUND OF THE INVENTION

[0001] Software theft has been, and continues to be, pervasive. Individuals and companies typically try various techniques to combat software theft, including requiring a unique software key to install software, requiring online activation of software, requiring an active online connection to use software, encryption of software, and the like. Although these techniques typically prevent casual users from installing unauthorized copies, the techniques can typically be overcome by sophisticated users.

[0002] Another way to combat software theft is to try to identify the source of the stolen software using watermarks. This involves applying unique watermarks to each copy of the software so that when a stolen piece of software is found, the watermark in the stolen software will correspond to one of the unique watermarks in the authorized software. This requires modification of the computer code, which is undesirable. Further, this technique can be overcome by removing the watermark from the stolen software or removing the watermark from the authorized software so that all further copies do not contain the unique watermark.

SUMMARY OF THE INVENTION

[0003] In addition to the issues identified above with the known techniques for combating software theft, these techniques focus on the software as a whole, and thus cannot identify when only portions of the underlying code are stolen. For example, if a watermark is applied to the software, the watermark would not appear in the stolen software if less than the entire code were used. Similarly, if software theft were identified by comparing hash values generated from the authorized and stolen software, the hash values would not match when less than the entire underlying code is present in the stolen software. Thus, a thief could simply modify some portion of the code to defeat these techniques. Further, it is often the

case that only a portion of the underlying code is truly unique and provides the overall value to the software, and accordingly a thief may only want to use this unique portion in different software.

[0004] Exemplary embodiments of the present invention are directed to techniques for combating software theft by identifying whether at least a portion of one piece of software appears in another piece of software. Thus, the present invention allows the identification of whether portions of one piece of software appear in a different piece of software, even when the overall operation of the two pieces of software is different. The inventive technique is particularly useful because it operates using compiled computer binaries, and thus does not require access to the underlying source code.

[0005] In accordance with exemplary embodiments of the present invention, fingerprints are generated using compiled computer binaries and the fingerprints are compared to determine whether there are sufficient similarities so as to indicate theft of at least a portion of one of the compiled computer binaries in the other compiled computer binary.

[0006] The fingerprints are generated by disassembling the compiled computer binaries and generating a control flow graph and function call graphs for each function in the control flow graph. Each function is then processed to identify unique spectra. These unique spectra are used to identify similarities between the different compiled computer binaries.

BRIEF DESCRIPTION OF THE DRAWINGS

[0007] Figure 1 is a flow diagram of the overall process of exemplary embodiments of the present invention;

[0008] Figure 2 is a block diagram of an exemplary system for generating and matching fingerprints in accordance with the present invention;

[0009] Figure 3 is a flow diagram of an exemplary process for generating a fingerprint using binary code in accordance with the present invention;

[0010] Figure 4 is a block diagram of an exemplary call graph in accordance with the present invention;

5 [0011] Figure 5 is a block diagram of an exemplary control flow graph in accordance with the present invention;

[0012] Figure 6 is a block diagram of an exemplary fingerprint in accordance with the present invention; and

[0013] Figures 7A and 7B are flow diagrams of an exemplary process for matching
10 fingerprints in accordance with the present invention.

DETAILED DESCRIPTION OF THE DRAWINGS

[0014] Figure 1 is a flow diagram of the overall process of exemplary embodiments of the present invention. As illustrated in Figure 1, the overall process involves generating a fingerprint from compiled computer binaries (step 300) and matching the generated
15 fingerprints to determine whether there is sufficient similarity (step 700). It should be recognized that fingerprint generation from compiled computer binaries that are later compared for matching can be performed at approximately the same time or can be performed at different times. For example, a fingerprint of a first compiled computer binary can be generated for purposes of identifying theft of the underlying code. Other compiled
20 computer binaries can then be collected over a period of time, and then fingerprints can be generated using the other compiled computer binaries for comparison with the first compiled computer binaries. These other compiled computer binaries can be obtained in any manner, such as, for example, using a web spider that crawls across the Internet and collects compiled computer binaries. The other compiled computer binaries can also be manually input. For
25 example, the owner of a first compiled computer binary may suspect that a second compiled

computer binary contains code stolen from the first compiled computer binary. In this case, the fingerprints can be generated from the first and second compiled computer binaries at approximately the same time and then compared using the inventive fingerprint matching technique.

5 [0015] Figure 2 is a block diagram of an exemplary system for generating and matching fingerprints in accordance with the present invention. The system 200 can comprise one or more computers that include a processor 205 coupled to memory 210, input 215, and output 220. The disclosed processes can be performed by processor 205 executing computer code stored in memory 210. The processor 205 can be any type of processor, including a
10 microprocessor, field programmable gate array (FPGA), and/or an application specific integrated circuit (ASIC). Memory 210 can be any type of non-transitory memory. In addition to storing computer code for executing the processes described herein, memory 210 can also store the generated fingerprints. Alternatively or additionally, a separate storage medium can store the generated fingerprints. For example, the computer binaries,
15 fingerprints, and comparison scores can be stored in a distributed file system and non-relational, distributed database. Input 215 provides mechanisms for controlling the disclosed processes, including, for example, a keyboard, mouse, trackball, trackpad, touchscreen, etc. Further, input 215 can include a connection to an external storage device for providing compiled computer binaries, such as an external hard drive or flash storage memory, as well
20 as a network connection. Output 220 can include a display, printer, and/or the like. Additionally, output 220 can include a network connection for notifying an owner of a compiled computer binary of any identified potential infringement, such as by electronic mail, posting on a website or webpage, a text message, and/or the like.

[0016] Figure 3 is a flow diagram of an exemplary process for generating a fingerprint using
25 binary code in accordance with the present invention. This process is performed using the

system of Figure 2. Initially, processor 205 receives a compiled computer binary (step 305) via input 215 and/or memory 210 and measures bulk file characteristics and meta-data (step 310). Next, processor 205 disassembles the compiled computer binary, generates a control flow graph from the disassembled binary, and generates function call graphs for each function in the control flow graph (step 315). Once the compiled computer binary is disassembled, the remainder of the processing can be performed independent of the particular language, operating system, or architecture that the code was written for.

[0017] Figure 4 is a block diagram of an exemplary call graph in accordance with the present invention. As will be appreciated by those skilled in the art, a call graph describes the relationship between various functions in a compiled binary. Thus, in Fig. 4, a main function 405 (also commonly referred to as a routine) has calls to sub-functions 410 and 415 (also commonly referred to as sub-routines). In turn, function 415 has calls to functions 420, 425, and 430, and function 425 has calls to functions 435 and 440. Those skilled in the art will recognize that in a call graph each function is commonly referred to as a node and the connections between functions are commonly referred to as edges. It will be recognized that the call graph of Figure 4 is a highly simplified graph and that compiled computer binaries typically will be disassembled into much more extensive call graphs.

[0018] Figure 5 is a block diagram of an exemplary control flow graph in accordance with the present invention. Those skilled in the art will recognize that a control flow graph for a function describes all possible paths that may be traversed during execution of the particular function. Examples of paths that may be executed in the function of Figure 5 include blocks 505, 515, 520, 525, 530, 535, and 545; 505, 515, 520, 525, 530, 535, 540, 550, 560, and 565; 505, 510, 515, 550, 560, and 565; and 505, 510, 515, 550, 555, and 565. In addition to these paths, any path with connections illustrated in Figure 5 can be traversed during the execution

of the function. Further, it should be recognized that this is merely an exemplary function and that other functions are within the scope of the invention.

[0019] Returning to Figure 3, after processor 205 generates the control flow graphs for each function (step 315), processor 205 selects one of the functions for processing (step 320). As

5 illustrated, this processing is performed along two parallel paths. It should be recognized, however, that these two paths can be performed serially, if desired. It should also be recognized that this parallel processing does not require all of the control flow graphs to be generated, and accordingly this parallel processing can be performed as control flow graphs are generated. Turning to the path on the left-hand side of Figure 3, first processor 205

10 calculates the function out-degree using the call graph for the particular function (step 325).

The function out-degree represents the number of paths or calls from a particular function to other functions in the call graph. Next processor 205 calculates the total number of blocks within the selected function's control flow graph (step 330), and finally processor 205 calculates the total number of edges within the selected function's control flow graph (step

15 335). The edges in the control flow graph are the connections between the different blocks within this graph.

[0020] Turning now to the path on the right-hand side of Figure 3, first processor 205 calculates the leading Eigenvector of the adjacency matrix (step 340). The adjacency matrix is comprised of the function coordinates 620 (described below in connection with Figure 6).

20 Next, processor 205 calculates the Markov chain (step 345). The Markov chain is calculated starting with the leading Eigenvector and then appending connected nodes corresponding with successively smaller elements of the leading Eigenvector. The Markov chain provides a good low-rank approximation, or serialization, of the control flow graph that is relatively unique and particularly well-suited for further statistical analysis of the control flow graph.

[0021] Processor 205 then calculates count block size, in-degree, and out-degree along the Markov chain (step 350). These three spectra are relatively unique among and within compiled computer binaries. An example of the count block size, in-degree, and out-degree will now be described in connection with Figure 5, and assuming a chain between the blocks as follows: 505=>515=>550=>560=>565. In this example the block count spectrum would be [12, 2, 3, 3, 9] because block 505 has 12 instructions, block 515 has two instructions, blocks 550 and 560 each have three instructions, and block 565 has nine instructions. The in-degree spectrum would be [0, 2, 3, 1, 2] because block 505 does not have any incoming edges, block 515 has two incoming edges, block 550 has three incoming edges, block 560 has one incoming edge, and block 565 has two incoming edges. The out-degree spectrum would be [2, 2, 2, 1, 1] because blocks 505, 515, and 550 each have two outgoing edges and blocks 560 and 565 each have two outgoing edges. The values for each spectra and ordering of values provides a unique signature for a particular control flow graph that can be used to identify other functions that have the same or similar unique signatures.

[0022] After the processing of the two parallel paths is complete, processor 205 determines whether there are any further functions to process (step 355). If there are ("Yes" path out of decision step 355), then the next function is selected (step 360) and the parallel processing is repeated. If not ("No" path out of decision step 355), then processor 205 generates the fingerprint of the binary using the calculated information (step 365).

[0023] Figure 6 is a block diagram of an exemplary fingerprint in accordance with the present invention. As illustrated in Figure 6, the fingerprint includes bulk file characteristics and meta-data 610, function coordinates 620, and unique spectra 630. The function coordinates 620 includes, for each function, the calculated out-degree, number of blocks, and number of edges. The unique spectra includes, for each function, the calculated block size, in-degree, and out-degree. These fingerprints are then used for comparison against

fingerprints of other compiled computer binaries as described below in connection with Figures 7A and 7B.

[0024] Figures 7A and 7B are flow diagrams of an exemplary process for matching fingerprints in accordance with the present invention. This process is performed using the system of Figure 2. Initially, processor 205 receives two fingerprints of compiled computer binaries via input 215 and/or memory 210 (step 705) and compares their bulk file characteristic and meta-data (step 710). The bulk file characteristic can include measuring the Jaccard Similarity, which is a numerical score representing the amount of overlap between strings contained in both fingerprints, and is typically between 0 and 1.

[0025] Next, processor 205 computes distances between each pair of possible functions of the two fingerprints using the function coordinates 620 of the respective fingerprints (step 720). Processor 205 then sorts the distances sorted (step 720) and discards function pairs having distances greater than a threshold distance (step 725). This step reduces the processing load because only the most-likely related functions will have distances below the threshold. Thus, the particular threshold value can be selected depending upon the available processing power of the computer and the desired run-time of the fingerprint comparison process. Furthermore, those skilled in the art will recognize that above a certain distance it is highly unlikely that functions will be related, and thus at least some thresholding should be performed to reduce unnecessary processing.

[0026] Processor 205 then generates a list using the remaining function pairs (step 730), and one of the function pairs from the reduced list is selected for further processing of the unique spectra 630 (step 735). Specifically, processor 205 calculates a cross-correlation for the block-size spectra (step 740), the in-degree spectra (step 745), and the out-degree spectra (step 750). It will be recognized that the cross-correlation is a measure of how closely the spectra of the two fingerprints are related. Next processor 205 determines whether any

function pairs remain to be processed (step 755). If so (“Yes path out of decision step 755), then the next function pair is selected from the reduced list (step 735) and the cross-correlation of the unique spectra are calculated (steps 740-750). The cross-correlation can produce a correlation coefficient indicating the degree of similarity or correlation. For example, a coefficient of -1 indicates complete anti-correlation and 1 indicates complete correlation (i.e., the two fingerprints have the same control flow graph).

[0027] If there are no remaining function pairs to process (“No” path out of decision step 755), then processor 205 calculates a block-size, in-degree and out-degree spectra ratios (step 760-770). These ratios are calculated by dividing a total number of respective correlation coefficients above a threshold by a total number of correlation coefficients. The threshold used for the calculation of the three ratios can be the same or different. Processor 205 then selects the maximum ratio of the unique spectra ratios (step 775) and generates a comparison score based on the selected maximum ratio (step 780). The generated comparison score is then used by processor 205 to identify infringement of one of the compiled computer binaries (step 785). The comparison score is generated by comparing the selected maximum ratio of the unique spectra ratios to a threshold, and accordingly infringement is identified when the selected maximum ratio is above the threshold. The threshold can be set, for example, by training the system using known data and a particular compiled computer binary for which it is to be determined whether there are other compiled computer binaries infringing the particular compiled computer binary. This training identifies commonalities between the known data and the particular compiled computer binary so that the threshold can be set to avoid false positives indicating infringement due to code commonly used across different pieces of software that would not be an indicator of infringement.

[0028] When, based on the generated comparison score, there is sufficient similarity between the compiled computer binaries or portions of the compiled computer binaries,

processor 205 can notify the owner of one of the compiled computer binaries of the potential infringement via output 220 (step 790). The notification can include details of the regions of the allegedly infringing computer binary that is most likely involved in the infringement.

[0029] The collection of compiled computer binaries, fingerprint generation, and fingerprint

5 matching can be automated and scheduled to execute using any type of task scheduling technique. Thus, the present invention provides a particularly cost- and time-effective way to discover, remediate, and enforce intellectual property rights, and accordingly acts as a deterrence against the theft of software code. Further, by identifying infringement based on the functions contained within compiled computer binaries, the present invention can identify
10 an entirely copied compiled computer binary, as well as copied portions of a compiled computer binary.

[0030] Although exemplary embodiments have been described above as generating fingerprints using compiled computer binaries, the present invention is equally applicable to computer source code, byte code, and the like.

15 [0031] The foregoing disclosure has been set forth merely to illustrate the invention and is not intended to be limiting. Since modifications of the disclosed embodiments incorporating the spirit and substance of the invention may occur to persons skilled in the art, the invention should be construed to include everything within the scope of the appended claims and equivalents thereof.

CLAIMS

1. A method comprising:
 - receiving, by a computer, a first compiled computer binary;
 - generating, by the computer, a first fingerprint of the first compiled computer binary;
 - receiving, by the computer, a second compiled computer binary;
 - generating, by the computer, a second fingerprint of the second compiled computer binary;
 - comparing, by the computer, the first fingerprint of the first compiled computer binary with the second fingerprint of the second compiled computer binary; and
 - determining, by the computer, whether at least some of the first compiled computer binary is present in the second compiled computer binary based on the comparison,wherein the generation of the first fingerprint of the first compiled computer binary comprises
 - generating, by the computer, a call graph using the first compiled computer binary;
 - generating, by the computer, a control flow graph for each of a plurality of functions in the call graph;
 - selecting, by the computer, one of the plurality of functions;
 - calculating, by the computer for the selected one of the plurality of functions, a leading Eigenvector of an adjacency matrix of the selected function's control flow graph;
 - generating, by the computer for the selected one of the plurality of functions, an edge-connected path starting from a node corresponding to a largest element in the leading Eigenvector of the adjacency matrix and appending connected nodes corresponding to successively smaller elements of the leading Eigenvector; and
 - calculating, by the computer for the selected one of the plurality of functions, unique spectra of the control flow graph for the selected function using the generated edge-connected path.

REPLACEMENT SHEETS

2. The method of claim 1, wherein the generation of the first fingerprint of the first compiled computer binary further comprises:
 - selecting, by the computer, one of the plurality of functions;
 - calculating, by the computer for the selected one of the plurality of functions
 - a function out-degree using the call graph;
 - a total number of blocks within the selected function's control flow graph;and
 - a total number of edges within the selected function's control flow graph.
3. The method of claim 1, wherein the unique spectra are count block size, in-degree, and out-degree along the edge-connected path.
4. The method of claim 1, wherein the generation of the first fingerprint of the first compiled computer binary comprises:
 - measuring, by the computer, bulk file characteristics and meta-data from the first compiled computer binary.
5. The method of claim 1, wherein the second compiled computer binary is collected using a spider that crawls across websites on the Internet.
6. The method of claim 1, wherein the method determines that the second compiled computer binary infringes the first compiled computer binary when the at least some of the first compiled computer binary is present in the second compiled computer binary but not all of the first compiled computer binary is present in the second compiled computer binary.
7. A method comprising:
 - receiving, by a computer, a first compiled computer binary;
 - generating, by the computer, a first fingerprint of the first compiled computer binary;
 - receiving, by the computer, a second compiled computer binary;

REPLACEMENT SHEETS

generating, by the computer, a second fingerprint of the second compiled computer binary;

comparing, by the computer, the first fingerprint of the first compiled computer binary with the second fingerprint of the second compiled computer binary; and

determining, by the computer, whether at least some of the first compiled computer binary is present in the second compiled computer binary based on the comparison,

wherein the comparison of the first fingerprint of the first compiled computer binary with the second fingerprint of the second compiled computer binary comprises

computing, by the computer, distances between each pair of possible functions of the first and second compiled computer binaries;

selecting, by the computer, a subset of the pairs of possible functions; and

calculating, by the computer, a cross-correlation of unique spectra for each pair of functions of the subset of pairs of possible functions,

wherein the unique spectra are count block size, in-degree, and out-degree along the edge-connected path,

wherein the cross-correlation of unique spectra produces, for each pair of functions of the subset of pairs of possible function

a block size correlation coefficient;

an in-degree correlation coefficient; and

an out-degree correlation coefficient.

8. The method of claim 7, wherein the determination of whether at least some of the first compiled computer binary is present in the second compiled computer binary comprises:

calculating, by the computer, a block size ratio by dividing a number of block size correlation coefficients above a first threshold by a total number of block size correlation coefficients;

calculating, by the computer, an in-degree ratio by dividing a number of in-degree correlation coefficients above a second threshold by a total number of in-degree correlation coefficients;

REPLACEMENT SHEETS

calculating, by the computer, an out-degree ratio by dividing a number of out-degree correlation coefficients above a third threshold by a total number of out-degree correlation coefficients;

selecting, by the computer, one of the block size, in-degree, and out-degree ratios having a highest value; and

comparing, by the computer, the selected one of the block size, in-degree, and out-degree ratios having the highest value with a fourth threshold.

9. The method of claim 8, wherein the first, second, and third thresholds are the same.

10. The method of claim 8, wherein the first, second, and third thresholds are different.

11. The method of claim 7, wherein the generation of the first fingerprint of the first compiled computer binary comprises:

measuring, by the computer, bulk file characteristics and meta-data from the first compiled computer binary.

12. The method of claim 7, wherein the second compiled computer binary is collected using a spider that crawls across websites on the Internet.

13. The method of claim 7, wherein the method determines that the second compiled computer binary infringes the first compiled computer binary when the at least some of the first compiled computer binary is present in the second compiled computer binary but not all of the first compiled computer binary is present in the second compiled computer binary.

14. A method comprising:
receiving, by a computer, a first compiled computer binary;

REPLACEMENT SHEETS

disassembling, by the computer, the first compiled computer binary into a form independent of programming language, operating system and architecture for which the first compiled computer binary was written;

generating, by the computer, a first fingerprint of the first compiled computer binary using the disassembled first compiled computer binary by

generating, by the computer, a call graph using the disassembled first compiled computer binary, wherein the call graph describes relationships between functions and sub-functions in the disassembled first compiled computer binary;

generating, by the computer, a control flow graph for each of a plurality of functions in the call graph, wherein the control flow graph describes all possible paths that may be traversed during execution of each of the plurality of functions in the call graph;

receiving, by the computer, a second compiled computer binary;

generating, by the computer, a second fingerprint of the second compiled computer binary;

comparing, by the computer, the first fingerprint of the first compiled computer binary with the second fingerprint of the second compiled computer binary; and

determining, by the computer, whether at least some of the first compiled computer binary is present in the second compiled computer binary based on the comparison.

15. The method of claim 14, wherein the generation of the first fingerprint of the first compiled computer binary further comprises:

selecting, by the computer, one of the plurality of functions;

calculating, by the computer for the selected one of the plurality of functions

a function out-degree using the call graph;

a total number of blocks within the selected function's control flow graph;

and

a total number of edges within the selected function's control flow graph.

REPLACEMENT SHEETS

16. The method of claim 14, wherein the generation of the first fingerprint of the first compiled computer binary further comprises:

- selecting, by the computer, one of the plurality of functions;
- calculating, by the computer for the selected one of the plurality of functions, a leading Eigenvector of an adjacency matrix of the selected function's control flow graph;
- generating, by the computer for the selected one of the plurality of functions, an edge-connected path starting from a node corresponding to a largest element in the leading Eigenvector of the adjacency matrix and appending connected nodes corresponding to successively smaller elements of the leading Eigenvector; and
- calculating, by the computer for the selected one of the plurality of functions, unique spectra of the control flow graph for the selected function using the generated edge-connected path.

17. The method of claim 16, wherein the unique spectra are count block size, in-degree, and out-degree along the edge-connected path.

18. The method of claim 14, wherein the generation of the first fingerprint of the first compiled computer binary comprises:

- measuring, by the computer, bulk file characteristics and meta-data from the first compiled computer binary.

19. The method of claim 14, wherein the comparison of the first fingerprint of the first compiled computer binary with the second fingerprint of the second compiled computer binary comprises:

- computing, by the computer, distances between each pair of possible functions of the first and second compiled computer binaries;
- selecting, by the computer, a subset of the pairs of possible functions; and
- calculating, by the computer, a cross-correlation of unique spectra for each pair of functions of the subset of pairs of possible functions.

REPLACEMENT SHEETS

20. The method of claim 19, wherein the unique spectra are count block size, in-degree, and out-degree along the edge-connected path.
21. The method of claim 20, wherein the cross-correlation of unique spectra produces, for each pair of functions of the subset of pairs of possible function:
- a block size correlation coefficient;
 - an in-degree correlation coefficient; and
 - an out-degree correlation coefficient.
22. The method of claim 21, wherein the determination of whether at least some of the first compiled computer binary is present in the second compiled computer binary comprises:
- calculating, by the computer, a block size ratio by dividing a number of block size correlation coefficients above a first threshold by a total number of block size correlation coefficients;
 - calculating, by the computer, an in-degree ratio by dividing a number of in-degree correlation coefficients above a second threshold by a total number of in-degree correlation coefficients;
 - calculating, by the computer, an out-degree ratio by dividing a number of out-degree correlation coefficients above a third threshold by a total number of out-degree correlation coefficients;
 - selecting, by the computer, one of the block size, in-degree, and out-degree ratios having a highest value; and
 - comparing, by the computer, the selected one of the block size, in-degree, and out-degree ratios having the highest value with a fourth threshold.
23. The method of claim 22, wherein the first, second, and third thresholds are the same.
24. The method of claim 22, wherein the first, second, and third thresholds are different.

REPLACEMENT SHEETS

25. The method of claim 14, wherein the second compiled computer binary is collected using a spider that crawls across websites on the Internet.

26. The method of claim 14, wherein the method determines that the second compiled computer binary infringes the first compiled computer binary when the at least some of the first compiled computer binary is present in the second compiled computer binary but not all of the first compiled computer binary is present in the second compiled computer binary.

REPLACEMENT SHEETS