[0102] At step 180, in response to the likelihood parameter generated by the one or more classifiers being equal to or greater than a predetermined likelihood parameter threshold value, the processor 501 can be configured to identify the given software as being affiliated to the target software. As noted hereinabove, in some non-limiting embodiments of the present technology, the target software may include software of a predetermined software family and/or of a predetermined authorship.

[0103] Further, in certain non-limiting embodiments of the present technology, the processor 501 may be configured to store data indicative of the given software in a dedicated database for further use. For example, the processor 501 can be configured to use the data indicative of the given software to train the one or more classifiers to determine affiliation of other software to the ~~given~~ target software.

[0104] In other non-limiting embodiments of the present technology, where the likelihood parameter is below the predetermined likelihood parameter threshold value, the processor 501 can be configured to determine that the given software is not affiliated to the target software; and thus, the processor 501 would not proceed to store the data indicative of the given software for further use.

[0105] The first method 100 thus terminates.

[0106] According to certain non-limiting embodiments of the present technology, the software source code of the given software can be obfuscated complicating the analysis thereof according to the first method 100. However, according to certain non-limiting embodiments of the present technology, the processor 501 may be configured to determine a specific obfuscation technique having been used for obfuscating the given software. To that end, for example, the processor 501 may be configured to apply (or otherwise have access to) a respective deobfuscator, i.e. a preliminarily prepared program intended to restore initial view of the software source code obfuscated by the specific obfuscation technique.

[0107] However, in accordance with other non-limiting embodiments of the present technology, the processor 501 may not be configured to determine the specific obfuscation technique having been used (for example, due to unavailability of the respective deobfuscator). Accordingly, the processor 501 may be configured to execute a second method 101 for determining the affiliation of the given software to the target software, whose flowchart diagram is depicted in Figure 1B and to which the reference is currently being made.

STEP 111: RECEIVING, BY THE PROCESSOR, A SOFTWARE SOURCE CODE OF THE GIVEN SOFTWARE

[0108] The second method 101 commences at step 111 where the processor 501 can be configured to receive the software source code associated with the given software. According to certain non-limiting

array of request identifiers. Further, the processor 501 can be configured to determine, based on the request likelihood parameter, the affiliation of the given software to the target software.

[0124] The second method 101 thus proceeds to step 161.

STEP 161: IN RESPONSE TO THE LIKELIHOOD PARAMETER BEING EQUAL TO OR GREATER THAN A PREDETERMINED LIKELIHOOD PARAMETER THRESHOLD: IDENTIFYING THE GIVEN SOFTWARE AS BEING AFFILIATED TO THE RESPECTIVE TARGET SOFTWARE; STORING DATA INDICATIVE OF THE GIVEN SOFTWARE IN A DATABASE OF AFFILIATED SOFTWARE; AND USING THE DATA INDICATIVE OF THE GIVEN SOFTWARE FOR FURTHER DETERMINING AFFILIATION TO THE GIVEN TARGET SOFTWARE

[0125] At step 161, in response to the request likelihood parameter generated by the one or more classifiers being equal to or greater than a predetermined request likelihood parameter threshold value, the processor 501 can be configured to identify the given software as being affiliated to the target software. As noted hereinabove, in some non-limiting embodiments of the present technology, the target software may include software of a predetermined software family and/or of a predetermined authorship.

[0126] Further, in certain non-limiting embodiments of the present technology, the processor 501 may be configured to store data indicative of the given software in the dedicated database for further use. For example, the processor 501 can be configured to use the data indicative of the given software to train the one or more classifiers to determine affiliation of other software to the target~~given~~ software.

[0127] In other non-limiting embodiments of the present technology, where the request likelihood parameter is below the predetermined request likelihood parameter threshold value, the processor 501 can be configured to determine that the given software is not affiliated to the target software; and thus, the processor 501 would not proceed to store the data indicative of the given software for further use.

[0128] The second method 101 thus terminates.

[0129] In some non-limiting embodiments of the present technology, after executing steps 130 and 131 of the first method 100 and the second method 101, respectively, each of the methods may further comprise an additional step of identifying, within the software source code, and deleting therefrom source code portions inherently non-indicative of the affiliation to the target software. In the context of the present specification, source code portions can be inherently non-indicative of the affiliation to the target software, by virtue of the portions of the software source code of the given software not being specific enough for further analysis of the given software for the affiliation thereof to the target software (for example, due to the fact that they often occur in different software and, thus, are not indicative

CLAIMS

1. A method for determining an affiliation of a given software with a target software, the method being executable by a processor, the method comprising:

receiving, by the processor, a software source code of the given software;

determining, by the processor, reference data associated with the software source code;

parsing, by the processor, based on the reference data associated with the software source code, the software source code to identify therein at least one function,

the identifying comprising identifying, in the software source code, library functions and deleting the library function by applying, by the processor, a signature analysis to the software source code;

parsing, by the processor, the at least one function to identify therein at least one function command,

the at least one function command comprising a respective action and at least one argument associated with the respective~~at least one~~ action;

generating, by the processor, for the at least one function, based on the respective action and the at least one argument thereof, a respective one of a plurality of function identifiers associated with the given software,

the generating comprising applying, by the processor, a hash function to the respective action and to each value of the at least one argument associated therewith to generate respective hash function values; and

aggregating the respective hash function values;

aggregating, by the processor, the plurality of function identifiers associated with the given software into an aggregated array of function identifiers given an order of occurrence of associated functions within the software source code;

applying, by the processor, at least one classifier to the aggregated array of function identifiers associated with the given software to determine a likelihood parameter indicative of the given software being affiliated to a respective target software,

the at least one classifier having been trained to determine the affiliation with the respective target software;

in response to the likelihood parameter being equal to or greater than a predetermined likelihood parameter threshold:

identifying the given software as being affiliated to the respective target software;

storing data indicative of the given software in a database of affiliated software, wherein the data comprises the aggregated array of function identifiers; and

using the data indicative of the given software for further determining affiliation of an other software to the ~~given~~ respective target software.

2. The method of claim 1, wherein the identifying the library functions further comprises identifying, in the software source code, source code portions inherently non-indicative of the affiliation to the target software, and deleting the source code portions inherently non-indicative of the affiliation to the target software from the software source code.

3. The method of claim 1, wherein the applying the hash function further comprises applying a first hash function to the respective action and applying a second hash function to the at least one argument, the second hash function being different from the first hash function.

4. The method of claim 1, wherein, the generating the respective one of the plurality of identifiers further comprises applying a serialization process to the respective~~at least one~~ action and the at least one argument associated therewith.

5. A method for training a classifier to determine an affiliation with a given target software, the method being executable by a processor, the method comprising:

receiving, by the processor, a plurality of target software source codes associated with the given target software;

determining, by the processor, for each one of the plurality of target software source codes, respective reference data;

identifying, by the processor, in each one of the plurality of target software source codes, based on the respective reference data, at least one target function;

parsing, by the processor, the at least one target function to identify therein at least one target function command;

generating, by the processor, based on each of the at least one target function command, a respective target function identifier associated with the at least one target function, the respective target function identifier comprising an associated number sequence;

aggregating, by the processor, number sequences associated with respective target functions over the plurality of target software source codes, thereby generating a number array associated with the ~~respective~~ given target software;

identifying, by the processor, in the number array associated with the given target software, at least one pattern, wherein:

the at least one pattern comprises a predetermined repetitive number sequence within the number array, and

the predetermined repetitive number sequence is indicative of a frequency of occurrence of at least one associated target function command within the given target software;

storing the at least one pattern, thereby generating a training set of data,

the training set of data comprising a plurality of patterns, a given pattern of the plurality of patterns including a respective label being indicative of an association between the given pattern and the given target software;

training the classifier, based on the training set of data, to determine the affiliation of a given software to the given target software.

6. The method of claim 5, wherein the identifying the at least one target function further comprises identifying, in each of the plurality of target software source codes, library functions and deleting the library functions.

7. The method of claim 6, wherein the identifying the library functions further comprises identifying, in each of the plurality of target software source codes, source code portions inherently non-indicative of the affiliation to the given target software, and deleting the source code portions inherently non-indicative of the affiliation to the given target software from each of the plurality of target software source codes.

8. The method of claim 5, wherein the at least one target function command comprises at least one action and at least one argument associated with the at least one action; and

the generating the respective target function identifier further comprises:

applying a hash function to the at least one action and to each value of the at least one argument to generate respective hash function values, each one of the respective hash function values being a respective number sequence; and

concatenating the respective hash function values to form the associated number sequence.

9. The method of claim 5, wherein the given target software is associated with at least one of: a predetermined software family, and a predetermined authorship.

10. The method of claim 5, further comprising determining a frequency occurrence value associated with the at least one pattern, the determining being according to the following formula:

$$\lambda = \frac{L}{K},$$

where L is a frequency of occurrence of the at least one pattern within the number array associated with the given~~respective~~ target software, and K is a number of software source codes in the plurality of target software source codes associated with the given~~respective~~ target software used for generating the training set of data.

11. The method of claim 5, further comprising assigning a weight value to the at least one pattern.

12. The method of claim 11, wherein the weight value is further determined based on the frequency of occurrence value associated with the at least one pattern.

13. A method for determining an affiliation of a given software with a target software, the method being executable by a processor, the method comprising:

receiving, by the processor, a software source code of the given software;

executing, by the processor, the software source code in an isolated program environment to identify at least one of a plurality of outgoing requests of the given software,

the at least one of the plurality of outgoing requests comprising a respective action and at least one argument associated with the respective action; and

generating, for the at least one of the plurality of outgoing requests, based on the respective action and the at least one argument, a respective one of a plurality of identifiers associated with the given software;

aggregating, by the processor, the plurality of identifiers associated with the given software into an aggregated array of identifiers given a time order of identifying associated outgoing requests;

applying, by the processor, at least one classifier to the aggregated array of identifiers to determine a likelihood parameter indicative of the given software being affiliated to a respective target software;

in response to the likelihood parameter being equal to or greater than a predetermined likelihood parameter threshold:

identifying the given software as being affiliated to the respective target software;

storing data indicative of the given software in a database of affiliated software, wherein the data comprises the aggregated array of identifiers; and

using the data indicative of the given software for further determining affiliation of an other software to the respective target software.

14. The method of claim 13, wherein the receiving the software source code further comprises identifying therein source code portions inherently non-indicative of the affiliation to the target software, and deleting the source code portions inherently non-indicative of the affiliation to the target software from the software source code.

15. The method of claim 13, wherein:

the generating the respective one of the plurality of identifiers further comprises:

applying a hash function to the respective action and to each value of the at least one argument associated with the at least one of the plurality of outgoing requests to generate respective hash function values, each one of the respective hash function values being a respective number sequence;

concatenating the respective hash function values.

16. The method of claim 15, wherein the applying the hash function further comprises applying a first hash function to the respective action and applying a second hash function to the at least one argument, the second hash function being different from the first hash function.

17. The method of claim 15, wherein, the generating the respective one of the plurality of identifiers further comprises applying a serialization process to the respective~~at least one~~ action and the at least one argument associated therewith.

18. The method of claim 13, wherein the respective target software is associated with at least one of: a predetermined software family, and a predetermined authorship.

19. The method of claim 13, wherein the at least one classifier has been trained to determine the affiliation to the respective target software based on a training set of data, and the method further comprising generating the training set of data, the generating comprising:

receiving, by the processor, a plurality of target software source codes associated with the respective target software;

executing, by the processor, a given one of the plurality of target software source codes in the isolated program environment to identify at least one of a respective plurality of target outgoing requests of the respective target software,

the at least one of the respective plurality of outgoing requests comprising a respective target action and at least one target argument associated with the respective target action;

generating, for the at least one of the respective plurality of target outgoing requests, based on the respective target action and the at least one target argument, a respective one of a respective plurality of target identifiers associated with the respective target software;

aggregating, by the processor, target identifiers associated with the respective target software over respective pluralities of target identifiers into an aggregated array of target identifiers given a time order of identifying associated target outgoing requests;

identifying, by the processor, in the aggregated array of target identifiers associated with the respective target software, at least one pattern, wherein:

the at least one pattern comprises a predetermined repetitive sequence of a given target identifier within the aggregated array of target identifiers, and

the predetermined repetitive sequence of the given target identifier is indicative of a frequency of the respective target software executing the respective target action in the isolated program environment;

storing the at least one pattern with a label indicative of an association between the at least one pattern and the respective target software for inclusion thereof into the training set of data.

20. The method of claim 19, wherein a length of the predetermined repetitive sequence is determined as a constant number.

21. The method of claim 20, wherein the length of the predetermined repetitive sequence is further determined iteratively, based on a current number thereof within the aggregated array of target identifiers.

22. The method of claim 19, further comprising determining a frequency occurrence value associated with the at least one pattern, the determining being according to the following formula:

$$\lambda = \frac{L}{K},$$

where L is a frequency of occurrence of the at least one pattern within the aggregated array of target identifiers associated with the respective target software, and K is a number of software source codes in the plurality of software source codes associated with the respective target software used for generating the training set of data.

23. The method of claim 19, further comprising assigning a weight value to the at least one pattern.

24. The method of claim 23, wherein the weight value is indicative of a type of the respective action associated with the at least one of the plurality of outgoing requests of the respective target software.

25. The method of claim 23, wherein the weight value is further determined based on the frequency of occurrence value associated with the at least one pattern.