US 20120124667A1

(19) **United States**

(12) **Patent Application Publication** (10) **Pub. No.: US 2012/0124667 A1**

Chiang et al. (43) **Pub. Date: May 17, 2012**

(54) **MACHINE-IMPLEMENTED METHOD AND SYSTEM FOR DETERMINING WHETHER A TO-BE-ANALYZED SOFTWARE IS A KNOWN MALWARE OR A VARIANT OF THE KNOWN MALWARE**

(75) Inventors: **Yi-Ta Chiang**, Hsinchu County (TW); **Ying-Dar Lin**, Taipei City (TW); **Yu-Sung Wu**, Hsinchu (TW); **Yuan-Cheng Lai**, Hsinchu (TW)

(73) Assignee: **National Chiao Tung University**, Hsinchu (TW)

(21) Appl. No.: **13/112,249**

(22) Filed: **May 20, 2011**

(30) **Foreign Application Priority Data**

Nov. 12, 2010 (TW) 099139009

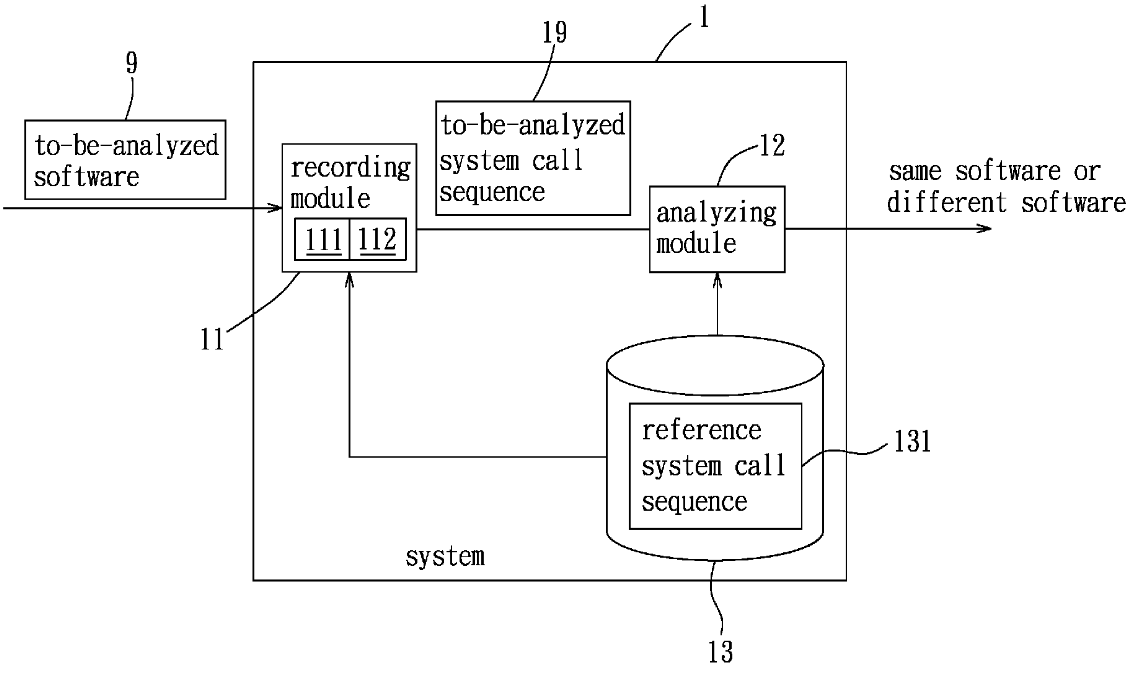
Publication Classification

(51) **Int. Cl.**
G06F 11/00 (2006.01)

(52) **U.S. Cl.** **726/24**

(57) **ABSTRACT**

A machine-implemented method for determining whether a to-be-analyzed software is a known malware or a variant of the known malware includes the steps of: (A) configuring a processor to execute the to-be-analyzed software, and obtain a to-be-analyzed system call sequence that corresponds to the to-be-analyzed software with reference to a plurality of system calls made in sequence as a result of executing the to-be-analyzed software; (B) configuring the processor to determine a degree of similarity between the to-be-analyzed system call sequence and a reference system call sequence that corresponds to the known malware; and (C) configuring the processor to determine that the to-be-analyzed software is neither the known malware nor a variant of the known malware when the degree of similarity determined in step (B) is not greater than a predefined similarity threshold value.


9
to-be-analyzed software
1
19
to-be-analyzed system call sequence
recording module
111
112
11
12
analyzing module
same software or different software
reference system call sequence
131
system
13

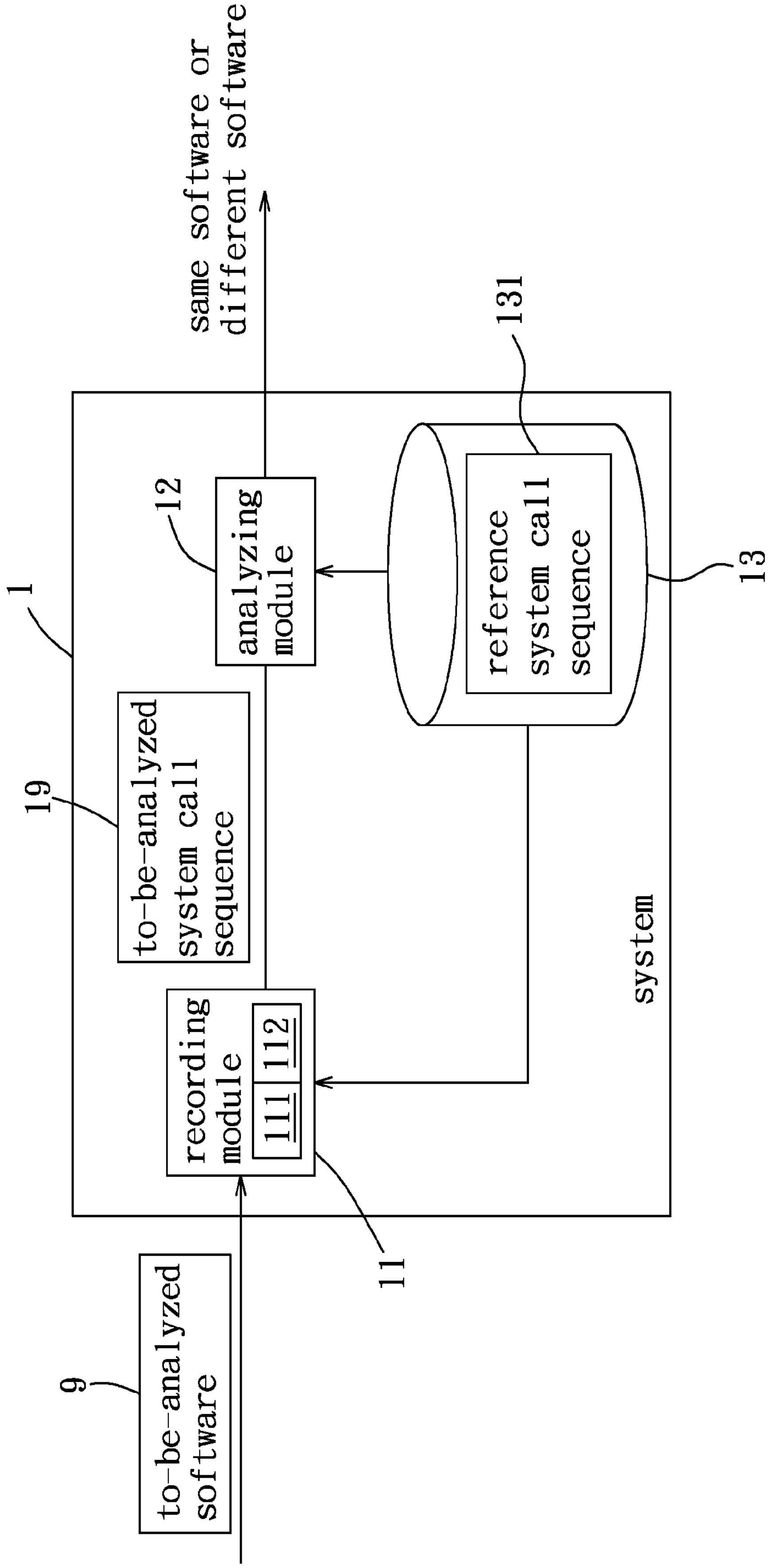

to-be-analyzed software
9
recording module
111
112
11
to-be-analyzed system call sequence
19
1
analyzing module
12
reference system call sequence
131
13
system
same software or different software

FIG. 1

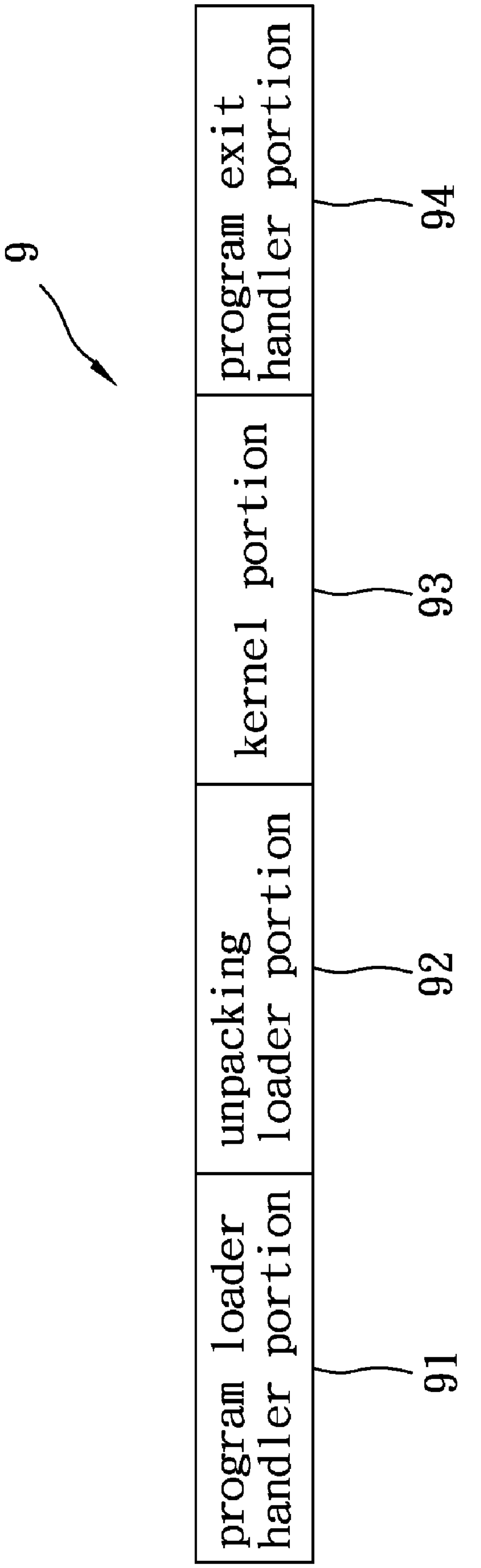

9
program loader handler portion
unpacking loader portion
kernel portion
program exit handler portion
91
92
93
94

FIG. 2

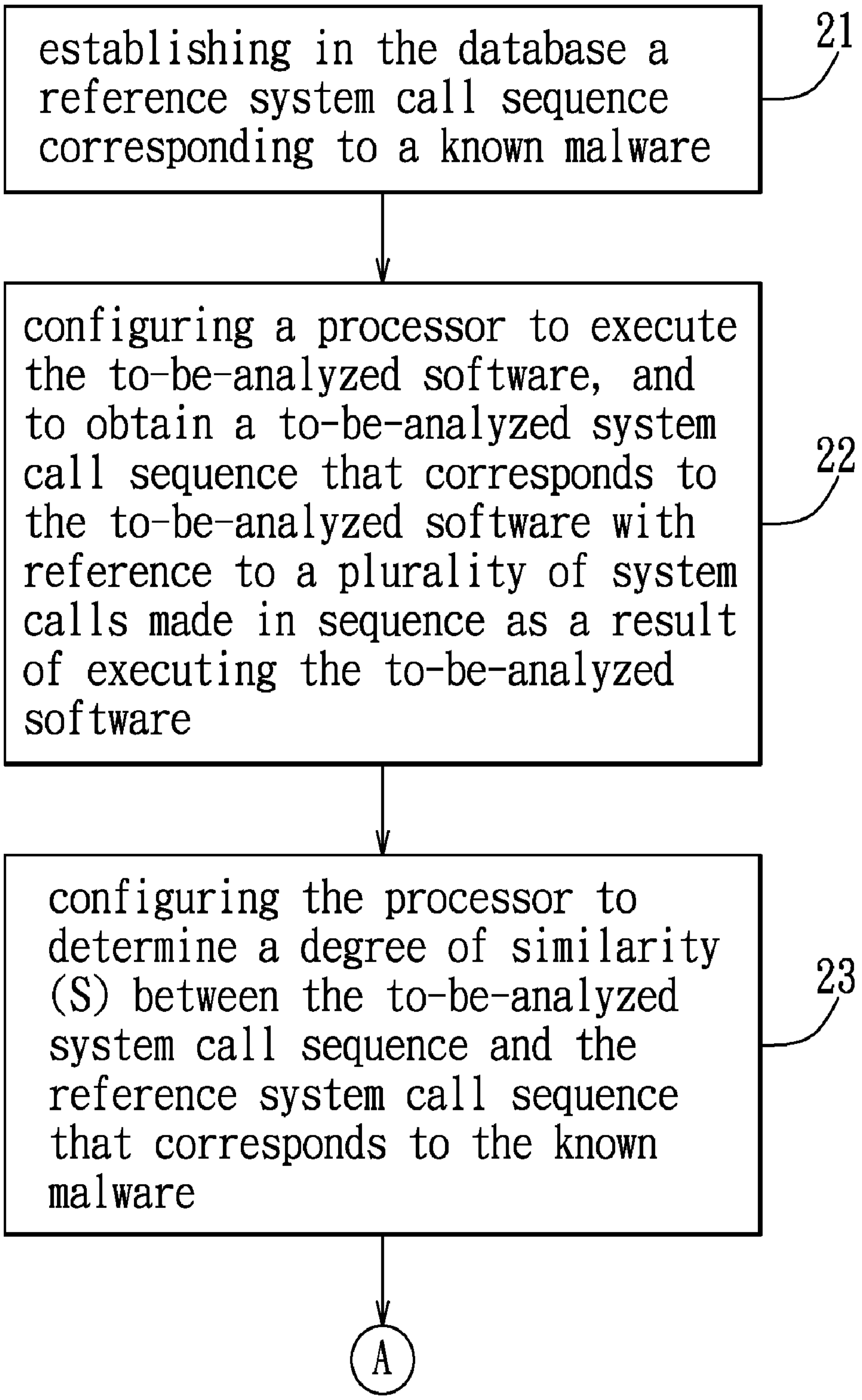

establishing in the database a reference system call sequence corresponding to a known malware
21
configuring a processor to execute the to-be-analyzed software, and to obtain a to-be-analyzed system call sequence that corresponds to the to-be-analyzed software with reference to a plurality of system calls made in sequence as a result of executing the to-be-analyzed software
22
configuring the processor to determine a degree of similarity (S) between the to-be-analyzed system call sequence and the reference system call sequence that corresponds to the known malware
23
A

FIG. 3A

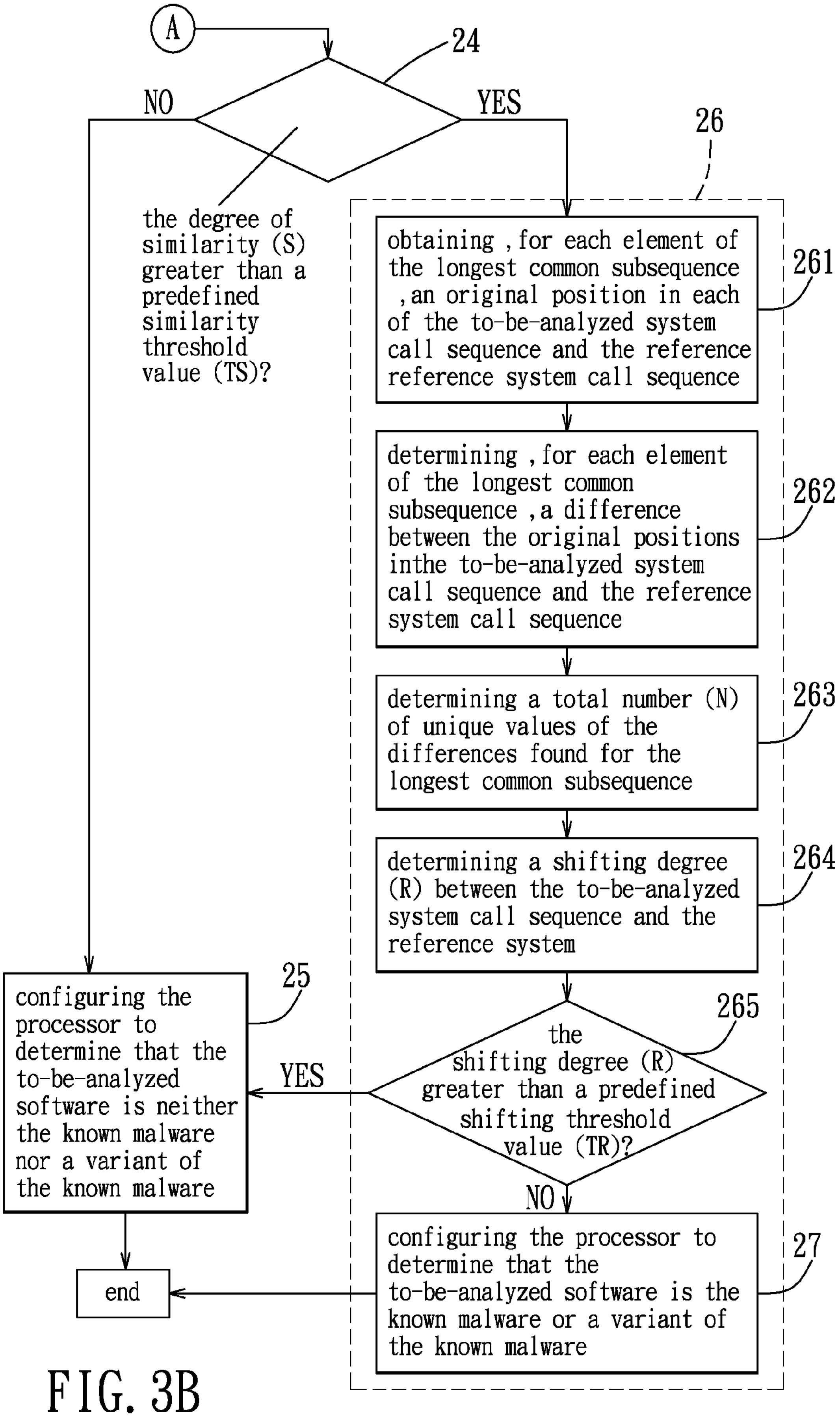
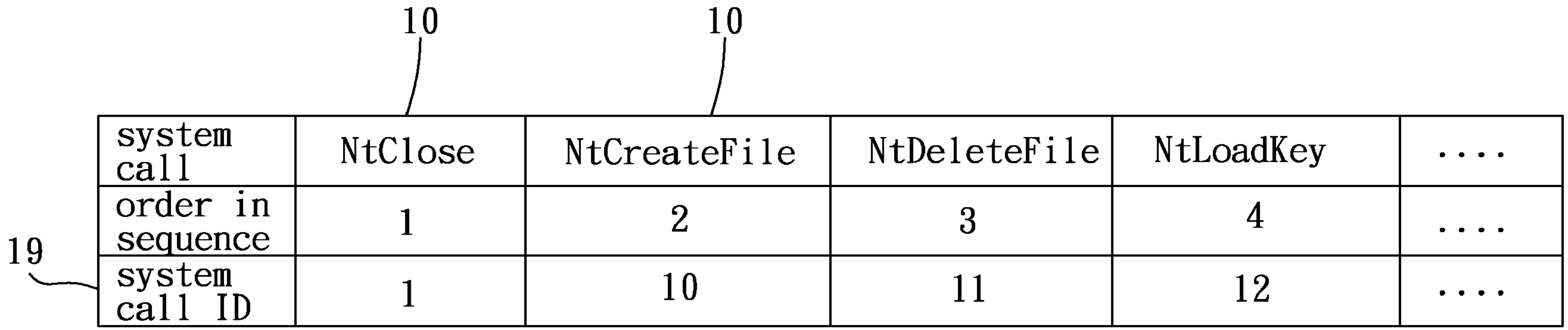

A
24
NO
YES
26
the degree of similarity (S) greater than a predefined similarity threshold value (TS)?
obtaining ,for each element of the longest common subsequence ,an original position in each of the to-be-analyzed system call sequence and the reference reference system call sequence
261
determining ,for each element of the longest common subsequence ,a difference between the original positions inthe to-be-analyzed system call sequence and the reference system call sequence
262
determining a total number (N) of unique values of the differences found for the longest common subsequence
263
determining a shifting degree (R) between the to-be-analyzed system call sequence and the reference system
264
configuring the processor to determine that the to-be-analyzed software is neither the known malware nor a variant of the known malware
25
YES
the shifting degree (R) greater than a predefined shifting threshold value (TR)?
265
NO
configuring the processor to determine that the to-be-analyzed software is the known malware or a variant of the known malware
27
end

FIG. 3B



system call	NtClose	NtCreateFile	NtDeleteFile	NtLoadKey	....
order in sequence	1	2	3	4	....
system call ID	1	10	11	12	....

FIG. 4

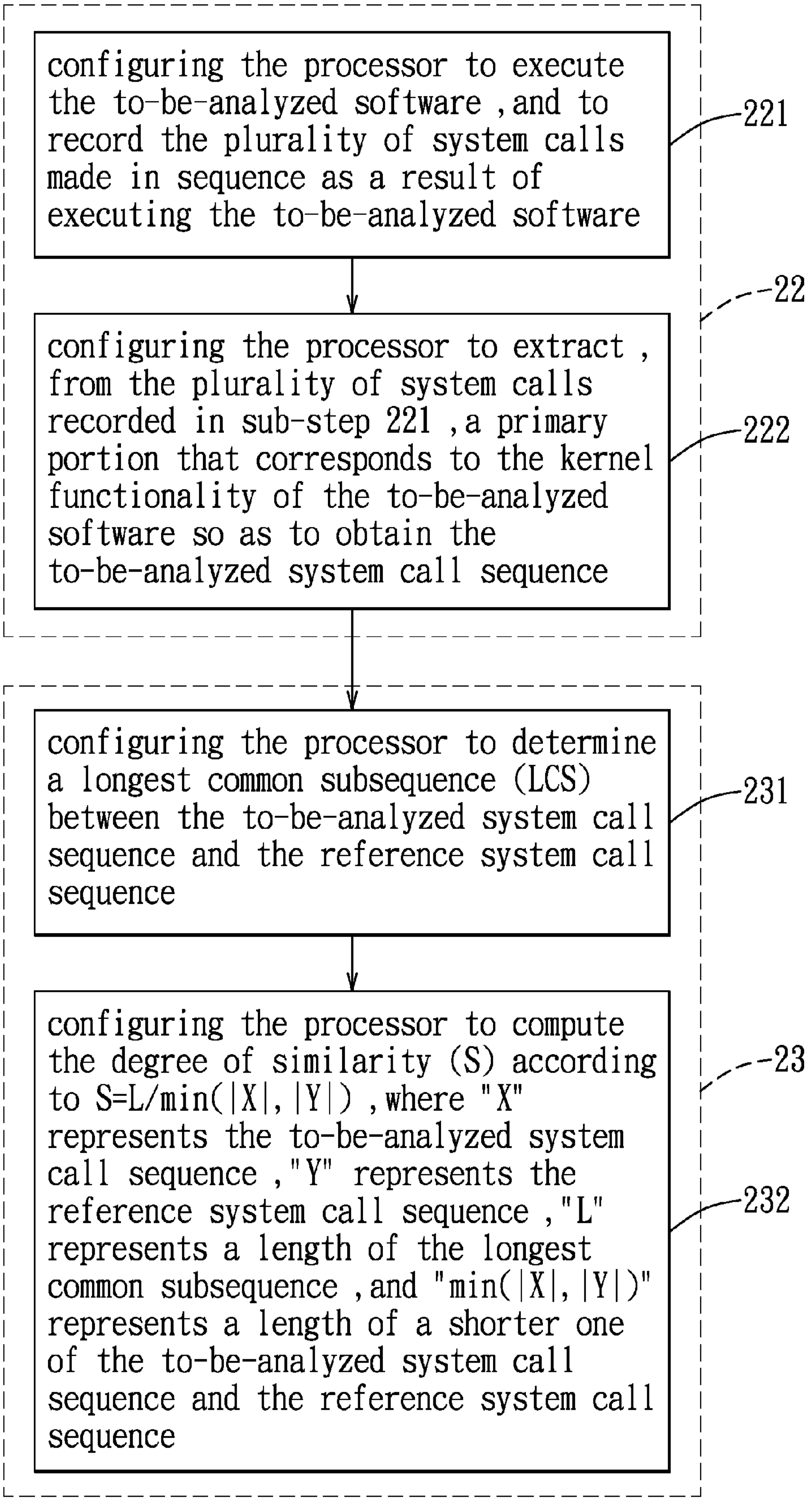

configuring the processor to execute the to-be-analyzed software ,and to record the plurality of system calls made in sequence as a result of executing the to-be-analyzed software
221
22
configuring the processor to extract , from the plurality of system calls recorded in sub-step 221 ,a primary portion that corresponds to the kernel functionality of the to-be-analyzed software so as to obtain the to-be-analyzed system call sequence
222
configuring the processor to determine a longest common subsequence (LCS) between the to-be-analyzed system call sequence and the reference system call sequence
231
configuring the processor to compute the degree of similarity (S) according to S=L/min(|X|, |Y|) ,where "X" represents the to-be-analyzed system call sequence ,"Y" represents the reference system call sequence ,"L" represents a length of the longest common subsequence ,and "min(|X|, |Y|)" represents a length of a shorter one of the to-be-analyzed system call sequence and the reference system call sequence
23
232

FIG. 5

MACHINE-IMPLEMENTED METHOD AND SYSTEM FOR DETERMINING WHETHER A TO-BE-ANALYZED SOFTWARE IS A KNOWN MALWARE OR A VARIANT OF THE KNOWN MALWARE

CROSS-REFERENCE TO RELATED APPLICATION

[0001] This application claims priority of Taiwanese Application No. 099139009, filed on Nov. 12, 2010.

BACKGROUND OF THE INVENTION

[0002] 1. Field of the Invention

[0003] The invention relates to a machine-implemented method for determining whether a to-be-analyzed software is a known malware, more particularly to a machine-implemented method for determining whether a to-be-analyzed software is a known malware or a variant of the known malware.

[0004] 2. Description of the Related Art

[0005] With the convenience of the Internet also come safety threats posed by malicious software and programs (collectively referred to as malware).

[0006] A botnet is an autonomous network of compromised zombie computers running software agents, commonly referred to as robots or bots, under the control of an attacker. Botnets are generally for nefarious purposes, such as sending spam messages and conducting information theft. These attacks might lead to crippling of the Internet or even financial losses. Therefore, preventive measures such as botnet detection and removal are constantly under study and research in the relevant field.

[0007] Conventionally, there are two approaches to detecting botnets, namely a static analysis approach and a dynamic analysis approach. In the static analysis approach, a to-be-analyzed binary (or code) is analyzed to determine if there are suspicious instruction sequences or if there are well-known signatures of known botnets. The static analysis approach does not consider what happens after the to-be-analyzed binary is executed, and does not produce accurate results if the to-be-analyzed binary is a botnet agent binary that has undergone obfuscation (e.g., that has been encrypted or compressed). On the other hand, the dynamic analysis approach executes the to-be-analyzed binary and monitors the runtime behavior (e.g., calling of application program interface (API), modifying system registry) of the to-be-analyzed binary in order to determine if it resembles a known botnet. However, the conventional dynamic analysis approach is rough and does not generate highly accurate results.

SUMMARY OF THE INVENTION

[0008] Therefore, the object of the present invention is to provide a system and a machine-implemented method for determining whether a to-be-analyzed software is a known malware or a variant of the known malware with increased accuracy.

[0009] According to one aspect of the present invention, there is provided a machine-implemented method for determining whether a to-be-analyzed software is a known malware or a variant of the known malware. The machine-implemented method includes the steps of: (A) configuring a processor to execute the to-be-analyzed software, and obtain a to-be-analyzed system call sequence that corresponds to the to-be-analyzed software with reference to a plurality of system calls made in sequence as a result of executing the to-be-analyzed software; (B) configuring the processor to determine a degree of similarity between the to-be-analyzed system call sequence and a reference system call sequence that corresponds to the known malware; and (C) configuring the processor to determine that the to-be-analyzed software is neither the known malware nor a variant of the known malware when the degree of similarity determined in step (B) is not greater than a predefined similarity threshold value.

[0010] According to another aspect of the present invention, there is provided a system for determining whether a to-be-analyzed software is a known malware or a variant of the known malware. The system includes a database, a recording module, and an analyzing module. The database has a reference system call sequence that corresponds to the known malware established therein. The recording module is for executing the to-be-analyzed software, and obtains a to-be-analyzed system call sequence that corresponds to the to-be-analyzed software with reference to a plurality of system calls made in sequence as a result of executing the to-be-analyzed software. The analyzing module is coupled to the database and the recording module for determining a degree of similarity between the to-be-analyzed system call sequence and the reference system call sequence, and further determines that the to-be-analyzed software is neither the known malware nor a variant of the known malware when the degree of similarity thus determined is not greater than a predefined similarity threshold value.

BRIEF DESCRIPTION OF THE DRAWINGS

[0011] Other features and advantages of the present invention will become apparent in the following detailed description of the preferred embodiment with reference to the accompanying drawings, of which:

[0012] FIG. **1** is a block diagram illustrating a system for determining whether a to-be-analyzed software is a known malware or a variant of the known malware according to the preferred embodiment of the present invention;

[0013] FIG. **2** is a schematic diagram illustrating four portions of a sequence of system calls made by a program;

[0014] FIGS. **3**A and **3**B collectively illustrate a flow chart illustrating a machine-implemented method for determining whether a to-be-analyzed software is a known malware or a variant of the known malware according to the preferred embodiment of the present invention; and

[0015] FIG. **4** is a schematic diagram illustrating a table with exemplary system calls in a to-be-analyzed system call sequence corresponding to the known malware; and

[0016] FIG. **5** is a flow chart illustrating sub-steps of step **22** and step **23** of the machine-implemented method shown in FIG. **2**.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

[0017] With reference to FIG. **1**, according to the preferred embodiment of the present invention, a system **1** for determining whether a to-be-analyzed software **9** is a known malware (e.g., a bot) or a variant of the known malware includes a database **13**, a recording module **11** and an analyzing module **12**. The to-be-analyzed software **9** may originally be resident in a storage space (not shown). According to the present invention, the system **1** achieves the determination by using three primary principles, namely segment identification of system call sequence, a similarity matching algorithm based on longest common subsequence (LCS), and shift analysis. These will become clear in the following description with reference to the preferred embodiment and the accompanying drawings. The system **1** is capable of determining not only if the to-be-analyzed software **9** is a known malware, but also whether the to-be-analyzed software **9** is a variant of a known malware. The variant may come as a result of using an obfuscation tool (e.g., a packer) to insert additional system calls, or modifying the source code of an existing (ancestor) bot.

[0018] The database **13** has established therein at least one reference system call sequence **131** that corresponds to the known malware.

[0019] The recording module **11** is for executing the to-be-analyzed software **9**, and obtains a to-be-analyzed system call sequence **19** that corresponds to the to-be-analyzed software **9** with reference to a plurality of system calls made in sequence as a result of executing the to-be-analyzed software **9**. Preferably, the recording module **11** records the plurality of system calls made in sequence as a result of executing the to-be-analyzed software **9**, and extracts, from the plurality of system calls thus recorded, a primary portion that corresponds to the kernel functionality of the to-be-analyzed software **9** so as to obtain the to-be-analyzed system call sequence **19**.

[0020] With reference to FIG. **2**, generally speaking, the system calls made by a program can be divided into four portions. The first portion is a program loader portion **91**, where system calls are made by the operating system (e.g., Windows®) to load the necessary dynamic link libraries (DLLs) and allocate the required memory space, etc. The second portion is an unpacking loader portion **92**, where system calls are made to prepare a suitable environment for execution of the source program, such as unpacking compressed binary into a text segment. The third portion is the kernel portion **93** referred to above, where system calls are made to perform the underlying kernel functionality of the program. The fourth portion is a program exit handler portion **94**, where system calls are used to release the allocated resource and to exit the program. If the program is obfuscated by an obfuscation tool such that additional system calls are introduced, in order to maintain the kernel functionality of the program, the system calls made during the kernel portion **93** are always kept intact. The present invention utilizes this particular characteristic to determine whether the to-be-analyzed software **9** is a known malware or a variant of the known malware. In this embodiment, the kernel portion **93** is the primary portion that is extracted to serve as the to-be-analyzed system call sequence **19**. The extraction of the to-be-analyzed system call sequence **19** is referred to above as the segment identification of system call sequence.

[0021] Since the unpacking loader portion **92** varies when the same malware is obfuscated by different obfuscation tools, a profile needs to be built for each different obfuscation tool for proper identification of the unpacking loader portion **92** and for effective extraction of the kernel portion **93**. In order to increase the accuracy in extracting the kernel portion **93** from the sequence of system calls made by the to-be-analyzed software **9** to serve as the to-be-analyzed system call sequence **19**, a profile for each different obfuscation tool needs to be established to effectively remove the non-relevant segments **91**, **92**, **94**.

[0022] The analyzing module **12** is coupled to the database **13** and the recording module **11** for determining a degree of similarity (S) between the to-be-analyzed system call sequence **19** and the reference system call sequence **131**. The analyzing module **12** further determines that the to-be-analyzed software **9** is neither the known malware nor a variant of the known malware when the degree of similarity (S) thus determined is not greater than a predefined similarity threshold value (T_S).

[0023] Specifically, the analyzing module **12** determines the degree of similarity (S) by determining a longest common subsequence (LCS) between the to-be-analyzed system call sequence **19** and the reference system call sequence **131**, and computes the degree of similarity (S) according to $S=L/\min(|X|, |Y|)$, where "X" represents the to-be-analyzed system call sequence **19**, "Y" represents the reference system call sequence **131**, "L" represents a length of the longest common subsequence, and "$\min(|X|, |Y|)$" represents a length of a shorter one of the to-be-analyzed system call sequence **19** and the reference system call sequence **131**. This is the similarity matching algorithm referred to above.

[0024] The analyzing module **12** further performs the following steps when the degree of similarity (S) determined thereby is greater than the predefined similarity threshold value (T_S): obtaining, for each element of the longest common subsequence, an original position in each of the to-be-analyzed system call sequence **19** and the reference system call sequence **131**; determining, for each element of the longest common subsequence, a difference between the original positions in the to-be-analyzed system call sequence **19** and the reference system call sequence **131**; determining a total number of unique values of the differences found for the longest common subsequence; determining a shifting degree (R) between the to-be-analyzed system call sequence **19** and the reference system call sequence **131** according to $R=N/L$, where "N" represents the total number of unique values of the differences and "L" represents the length of the longest common subsequence; determining that the to-be-analyzed software **9** is neither the known malware nor a variant of the known malware when the shifting degree (R) thus determined is greater than a predefined shifting threshold value (T_R); and determining that the to-be-analyzed software **9** is the known malware or a variant of the known malware when the shifting degree (R) thus determined is not greater than the predefined shifting threshold value (T_R). This is the shifting analysis referred to above.

[0025] The predefined similarity threshold value (T_S) ranges between 0.58 and 0.63, and the predefined shifting threshold value (T_R) ranges between 0.05 and 0.08. Preferably, the predefined similarity threshold value (T_S) is 0.6, and the predefined shifting threshold value (T_R) is 0.06.

[0026] Referring to FIGS. **3**A and **3**B, the present invention will be more clearly understood with reference to the following descriptions in connection with the machine-implemented method according to the preferred embodiment of the present invention. The machine-implemented method is for determining whether the to-be-analyzed software **9** is a known malware or a variant of the known malware, and includes the following steps.

[0027] First, in step **21**, a reference system call sequence **131** corresponding to a known malware is established in the database **13**. The machine-implemented method is then performed to determine whether the to-be-analyzed software **9** is the known malware or a variant of the known malware. One should readily appreciate that there may be multiple reference system call sequences **131** respectively corresponding to multiple different known malwares established in the database **13**, and the machine-implemented method of the present invention is for determining whether the to-be-analyzed software **9** is one of the known malwares or a variant of one of the known malwares.

[0028] Next, in step **22**, a processor (not shown) (or the recording module **11** of the system **1** shown in FIG. **1**) is configured to execute the to-be-analyzed software **9**, and to obtain a to-be-analyzed system call sequence **19** that corresponds to the to-be-analyzed software **9** with reference to a plurality of system calls made in sequence as a result of executing the to-be-analyzed software **9**.

[0029] With reference to FIG. **5**, step **22** includes two sub-steps in this embodiment. In sub-step **221**, the processor (or the recording module **11** of the system **1** shown in FIG. **1**) is configured to execute the to-be-analyzed software **9**, and to record the plurality of system calls made in sequence as a result of executing the to-be-analyzed software **9**. In sub-step **222**, the processor (or the recording module **11** of the system **1** shown in FIG. **1**) is configured to extract, from the plurality of system calls recorded in sub-step **221**, a primary portion **111** that corresponds to the kernel functionality of the to-be-analyzed software **9** so as to obtain the to-be-analyzed system call sequence **19**. In this embodiment, the primary portion **111** corresponds to the kernel portion **93** shown in FIG. **2**. The remaining of the plurality of system calls recorded in sub-step **221** include the program loader portion **91**, the unpacking loader portion **92**, and the program exit handler portion **94** of FIG. **2**, and are collectively referred to as a secondary portion **112**.

[0030] The reason behind taking only the kernel portion **93** as the primary portion **111** to serve as the to-be-analyzed system call sequence **19** and neglecting the secondary portion **112** is that, as described previously with reference to FIG. **2**, the program loader portion **91** and the program exit handler portion **94** of the secondary portion **112** of the system calls are common to nearly all programs and executable files and are irrelevant to the identification of the known malware, and that while the unpacking loader portion **92** of the secondary portion **112** may vary among programs obfuscated using different obfuscation tools, the system calls made to perform the underlying kernel functionality of the program (i.e., the kernel portion **93**) remain substantially unchanged for a known malware and its variants. As such, the reference system call sequence **131** established in the database **13** in step **21** also only corresponds to the kernel portion **93** of the system calls made by the known malware . Using only the primary portion **111** as the basis for the determination increases both the speed and the success rate of the identification.

[0031] In the system **1** described above, after sub-step **222** is performed by the recording module **11**, the to-be-analyzed system call sequence **19** is transmitted to the analyzing module **12**. FIG. **4** shows a table with exemplary system calls **10** in the to-be-analyzed system call sequence **19** (e.g., NtClose, NtCreateFile, NtDeleteFile, NtLoadKey, etc.). The actual to-be-analyzed system call sequence **19** in this embodiment is a sequence of the system call identifications (IDs) corresponding to the system calls **10**. Therefore, in this example, the to-be-analyzed system call sequence **19** is (1, 10, 11, 12, . . .). In this embodiment, the processor (or the recording module **11** shown in FIG. **1**) utilizes the "Pin" tool, a dynamic binary instrumentation tool developed by Intel® for dynamic instrumentation of programs, to record the system calls and the corresponding system call IDs made as a result of executing the to-be-analyzed software **9**.

[0032] Subsequently, in step **23**, the processor (or specifically, the analyzing module **12** shown in FIG. **1**) is configured to determine a degree of similarity (S) between the to-be-analyzed system call sequence **19** and the reference system call sequence **131** that corresponds to the known malware. With reference to FIG. **5**, step **23** includes two sub-steps in this embodiment. In sub-step **231**, the processor (or the analyzing module **12** shown in FIG. **1**) is configured to determine a longest common subsequence (LCS) between the to-be-analyzed system call sequence **19** and the reference system call sequence **131**. In sub-step **232**, the processor (or the analyzing module **12** shown in FIG. **1**) is configured to compute the degree of similarity (S) according to the following equation: $S=L/\min(|X|, |Y|)$, where "X" represents the to-be-analyzed system call sequence **19**, "Y" represents the reference system call sequence **131**, "L" represents a length of the longest common subsequence, and "min (|X|, |Y|)" represents a length of a shorter one of the to-be-analyzed system call sequence **19** and the reference system call sequence **131**. The value of the degree of similarity (S) ranges between 0 and 1, and S=1 indicates that X is a variant of Y or that Y is a variant of X.

[0033] For instance, assuming that X (i.e., the to-be-analyzed system call sequence **19**) is (1, 10, 11, 12, 2, 3, 18, 4, 20, 21, 5), while Y (i.e., the reference system call sequence **131**) is (1, 2, 3, 4, 5), the longest common subsequence (LCS) between X and Y is (1, 2, 3, 4, 5) with a length "L" of 5, and min (|X|, |Y|) is 5. Therefore, S=1, and X is a variant of Y.

[0034] Next, in step **24**, the processor (or the analyzing module **12** shown in FIG. **1**) is configured to determine whether or not the degree of similarity (S) determined in step **23** is greater than a predefined similarity threshold value (T_S). In this embodiment, the predefined similarity threshold value (T_S) ranges between 0.58 and 0.63. Preferably, the predefined similarity threshold value (T_S) is 0.6. In the negative, i.e., if it is determined in step **24** that the degree of similarity (S) is smaller than or equal to the predefined similarity threshold value (T_S), the flow goes to step **25**, where the processor is configured to determine that the to-be-analyzed software **9** is neither the known malware nor a variant of the known malware. If affirmative, i.e., if it is determined in step **24** that the degree of similarity (S) is greater than the predefined similarity threshold value (T_S), the flow goes to step **26**, where the processor (or the analyzing module **12** shown in FIG. **1**) is configured to perform the following sub-steps.

[0035] In sub-step **261**, it is obtained, for each element of the longest common subsequence, an original position in each of the to-be-analyzed system call sequence **19** and the reference system call sequence **131**. In the following description, let the sequence ($a_1, a_2, a_3, \ldots, a_L$) represent the original positions of the elements of the longest common subsequence in the to-be-analyzed system call sequence **19**, and let the sequence ($b_1, b_2, b_3, \ldots, b_L$) represent the original positions of the elements of the longest common subsequence in the reference system call sequence **131**.

[0036] In sub-step **262**, it is determined, for each element of the longest common subsequence, a difference between the original positions in the to-be-analyzed system call sequence **19** and the reference system call sequence **131**. In other words, the differences (a_1-b_1, a_2-b_2, a_3-b_3, . . . , a_L-b_L) are determined in sub-step **262**.

[0037] In sub-step **263**, a total number (N) of unique values of the differences found for the longest common subsequence is determined. The total number (N) is a positive integer.

[0038] In sub-step **264**, a shifting degree (R) between the to-be-analyzed system call sequence **19** and the reference system call sequence **131** is determined according to R=N/L.

[0039] For the above-described example, where the to-be-analyzed system call sequence **19** (X) is (1, 10, 11, 12, 2, 3, 18, 4, 20, 21, 5), the reference system call sequence **131** (Y) is (1, 2, 3, 4, 5), and the longest common subsequence (LCS) between X and Y is (1, 2, 3, 4, 5), the original positions of the elements of the longest common subsequence in the to-be-analyzed system call sequence **19** ($a_1, a_2, a_3, \ldots, a_L$) is (1, 5, 6, 8, 11), and the original positions of the elements of the longest common subsequence in the reference system call sequence **131** ($b_1, b_2, b_3, \ldots, b_L$) is (1, 2, 3, 4, 5). Therefore, the differences between the original positions in the to-be-analyzed system call sequence **19** and the reference system call sequence **131** ($a_1-b_1, a_2-b_2, a_3{}^-b_3, \ldots, a_L{}^-b_L$) are (0, 3, 3, 4, 6), and the total number (N) of unique values of the differences found for the longest common subsequence is 4. As such, the shifting degree (R) between the to-be-analyzed system call sequence **19** and the reference system call sequence **131** is determined by R=4/5=0.8.

[0040] Subsequently, in sub-step **265**, it is determined whether or not the shifting degree (R) determined in sub-step **264** is greater than a predefined shifting threshold value (T_R). In this embodiment, the predefined shifting threshold value (T_R) ranges between 0.05 and 0.08. Preferably, the predefined shifting threshold value (T_R) is 0.06.

[0041] If affirmative, i.e., the shifting degree (R) determined in sub-step (**264**) is greater than the predefined shifting threshold value (T_R), the flow goes to step **25**, where it is determined that the to-be-analyzed software **9** is neither the known malware nor a variant of the known malware. On the other hand, in the negative, i.e, the shifting degree (R) determined in sub-step (**264**) is smaller than or equal to the predefined shifting threshold value (T_R), the flow goes to step **27**, where it is determined that the to-be-analyzed software **9** is the known malware or a variant of the known malware.

[0042] Taking the previous example, where the degree of similarity (S) is **1**, and the shifting degree (R) is 0.8, although there is a 100% similarity between the to-be-analyzed system call sequence **19** and the reference system call sequence **131**, there is a shifting degree (R) that is far greater than the predefined shifting threshold value (T_R), meaning that as there are multiple additional system calls (to be exact, six additional system calls in this example, namely system call IDs **10**, **11**, **12**, **18**, **20** and **21**) in the to-be-analyzed system call sequence **19** as compared to the reference system call sequence **131**, the shiftings resulted in the system calls that are made for both the to-be-analyzed software **9** and the known malware (or a variant thereof) are too great so as to render the to-be-analyzed software **9** be deemed as neither the known malware nor a variant of the known malware. In other words, even if the degree of similarity (S) between the to-be-analyzed system call sequence **19** and the reference system call sequence **131** is very high, or even indicating 100% similarity, the processor (or the analyzing module **12** shown in FIG. **1**) will determine that the to-be-analyzed software **9** is neither the known malware nor a variant of the known malware when the shifting degree (R) exceeds the predefined shifting threshold value (T_R).

[0043] It should be noted herein that in practice, a virtual machine, such as VirtualBox, may be used for execution of the to-be-analyzed software **9** in order to obtain the to-be-analyzed system call sequence **19** so that in case where the to-be-analyzed software **9** is a malware, the system **1** is not contaminated. In addition, the virtual machine may be connected to the Internet through a firewall to allow connection access to the to-be-analyzed software **9** while preventing malicious traffic from interfering with the determination process.

[0044] In summary, the present invention utilizes segment identification of system call sequence, a similarity matching algorithm based on longest common subsequence (LCS), and a shift analysis to determine whether a to-be-analyzed software **9** is a known malware or a variant of the known malware. In addition, by extracting, from a plurality of system calls recorded as a result of executing the to-be-analyzed software **9**, a primary portion **111** that corresponds to the kernel functionality of the to-be-analyzed software **9** so as to obtain the to-be-analyzed system call sequence **19**, both the speed and the success rate of the identification can be increased.

[0045] while the present invention has been described in connection with what is considered the most practical and preferred embodiment, it is understood that this invention is not limited to the disclosed embodiment but is intended to cover various arrangements included within the spirit and scope of the broadest interpretation so as to encompass all such modifications and equivalent arrangements.

What is claimed is:

1. A machine-implemented method for determining whether a to-be-analyzed software is a known malware or a variant of the known malware, the machine-implemented method comprising the steps of:

(A) configuring a processor to execute the to-be-analyzed software, and obtain a to-be-analyzed system call sequence that corresponds to the to-be-analyzed software with reference to a plurality of system calls made in sequence as a result of executing the to-be-analyzed software;

(B) configuring the processor to determine a degree of similarity between the to-be-analyzed system call sequence and a reference system call sequence that corresponds to the known malware; and

(C) configuring the processor to determine that the to-be-analyzed software is neither the known malware nor a variant of the known malware when the degree of similarity determined in step (B) is not greater than a predefined similarity threshold value.

2. The machine-implemented method as claimed in claim **1**, wherein step (B) includes the sub-steps of:

(B-1) determining a longest common subsequence (LCS) between the to-be-analyzed system call sequence and the reference system call sequence; and

(B-2) computing the degree of similarity (S) according to S=L/min (|X|, |Y|) , where “X” represents the to-be-analyzed system call sequence, “Y” represents the reference system call sequence, “L” represents a length of the longest common subsequence, and “min (|X|, |Y|)” represents a length of a shorter one of the to-be-analyzed system call sequence and the reference system call sequence.

3. The machine-implemented method as claimed in claim **2**, further comprising the step of:

(D) configuring the processor to perform the following sub-steps when the degree of similarity determined in step (B.) is greater that the predefined similarity threshold value:

(D-1) obtaining, for each element of the longest common subsequence, an original position in each of the to-be-analyzed system call sequence and the reference system call sequence,

(D-2) determining, for each element of the longest common subsequence, a difference between the original positions in the to-be-analyzed system call sequence and the reference system call sequence,

(D-3) determining a total number of unique values of the differences found for the longest common subsequence,

(D-4) determining a shifting degree (R) between the to-be-analyzed system call sequence and the reference system call sequence according to R=N/L, where "N" represents the total number determined in sub-step (D-3) and "L" represents the length of the longest common subsequence,

(D-5) determining that the to-be-analyzed software is neither the known malware nor a variant of the known malware when the shifting degree (R) determined in sub-step (D-4) is greater than a predefined shifting threshold value, and

(D-6) determining that the to-be-analyzed software is the known malware or a variant of the known malware when the shifting degree determined in sub-step (D-4) is not greater than the predefined shifting threshold value.

4. The machine-implemented method as claimed in claim **3**, wherein the predefined similarity threshold value ranges between 0.58 and 0.63, and the predefined shifting threshold value ranges between 0.05 and 0.08.

5. The machine-implemented method as claimed in claim **4**, wherein the predefined similarity threshold value is 0.6, and the predefined shifting threshold value is 0.06.

6. The machine-implemented method as claimed in claim **1**, wherein step (A) includes the sub-steps of:

(A-1) configuring the processor to execute the to-be-analyzed software, and record the plurality of system calls made in sequence as a result of executing the to-be-analyzed software; and

(A-2) configuring the processor to extract, from the plurality of system calls recorded in sub-step (A-1), a primary portion that corresponds to the kernel functionality of the to-be-analyzed software so as to obtain the to-be-analyzed system call sequence.

7. A system for determining whether a to-be-analyzed software is a known malware or a variant of the known malware, said system comprising:

a database having a reference system call sequence that corresponds to the known malware established therein;

a recording module for executing the to-be-analyzed software, and obtaining a to-be-analyzed system call sequence that corresponds to the to-be-analyzed software with reference to a plurality of system calls made in sequence as a result of executing the to-be-analyzed software; and

an analyzing module coupled to said database and said recording module for determining a degree of similarity between the to-be-analyzed system call sequence and the reference system call sequence, and further determining that the to-be-analyzed software is neither the known malware nor a variant of the known malware when the degree of similarity thus determined is not greater than a predefined similarity threshold value.

8. The system as claimed in claim **7**, wherein said analyzing module determines the degree of similarity by determining a longest common subsequence (LCS) between the to-be-analyzed system call sequence and the reference system call sequence, and computing the degree of similarity (S) according to S=L/min (|X|, |Y|), where "X" represents the to-be-analyzed system call sequence, "Y" represents the reference system call sequence, "L" represents a length of the longest common subsequence, and "min (|X|, |Y|)" represents a length of a shorter one of the to-be-analyzed system call sequence and the reference system call sequence.

9. The system as claimed in claim **8**, wherein said analyzing module further performs the following steps when the degree of similarity determined thereby is greater than the predefined similarity threshold value:

obtaining, for each element of the longest common subsequence, an original position in each of the to-be-analyzed system call sequence and the reference system call sequence;

determining, for each element of the longest common subsequence, a difference between the original positions in the to-be-analyzed system call sequence and the reference system call sequence;

determining a total number of unique values of the differences found for the longest common subsequence;

determining a shifting degree (R) between the to-be-analyzed system call sequence and the reference system call sequence according to R-N/L, where "N" represents the total number of unique values of the differences and "L" represents the length of the longest common subsequence;

determining that the to-be-analyzed software is neither the known malware nor a variant of the known malware when the shifting degree (R) thus determined is greater than a predefined shifting threshold value; and

determining that the to-be-analyzed software is the known malware or a variant of the known malware when the shifting degree thus determined is not greater than the predefined shifting threshold value.

10. The system as claimed in claim **9**, wherein the predefined similarity threshold value ranges between 0.58 and 0.63, and the predefined shifting threshold value ranges between 0.05 and 0.08.

11. The system as claimed in claim **10**, wherein the predefined similarity threshold value is 0.6, and the predefined shifting threshold value is 0.06.

12. The system as claimed in claim **7**, wherein said recording module records the plurality of system calls made in sequence as a result of executing the to-be-analyzed software, and extracts, from the plurality of system calls thus recorded, a primary portion that corresponds to the kernel functionality of the to-be-analyzed software so as to obtain the to-be-analyzed system call sequence.

* * * * *