(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
20 January 2022 (20.01.2022)

WIPO | PCT

(10) International Publication Number
WO 2022/012821 A1

(51) International Patent Classification:
G06F 21/56 (2013.01)

(21) International Application Number:
PCT/EP2021/065635

(22) International Filing Date:
10 June 2021 (10.06.2021)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
2010898.1 15 July 2020 (15.07.2020) GB

(71) Applicant: **BRITISH TELECOMMUNICATIONS PUBLIC LIMITED COMPANY** [GB/GB]; 81 Newgate Street, London EC1A 7AJ (GB).

(72) Inventor: **EL-MOUSSA, Fadi**; Ground Floor, Faraday Building, 1 Knightrider Street, London EC4V 5BT (GB).

(74) Agent: **BRITISH TELECOMMUNICATIONS PUBLIC LIMITED COMPANY, INTELLECTUAL PROPERTY DEPARTMENT**; Ground Floor, Faraday Building 1 Knightrider Street, London EC4V 5BT (GB).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DJ, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, IT, JO, JP, KE, KG, KH, KN, KP, KR, KW, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, WS, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, ST, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

Published:
— *with international search report (Art. 21(3))*

(54) Title: COMPUTER-IMPLEMENTED AUTOMATIC SECURITY METHODS AND SYSTEMS

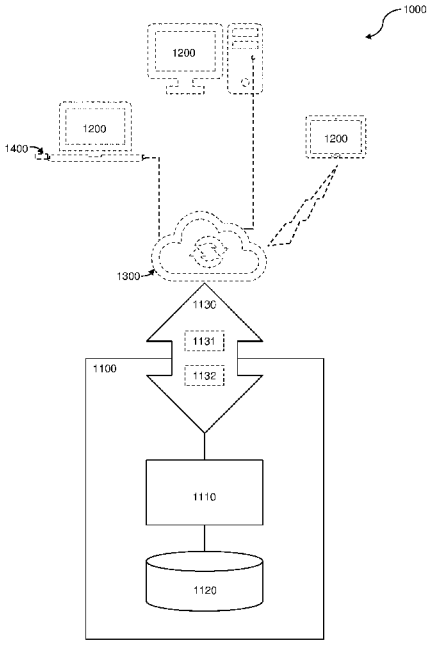

Figure 1

(57) Abstract: Computer-implemented automatic security methods and systems One aspect relates to a computer-implemented method of automatically securing a computer system or network against a suspect binary file (SBF) by, in response to detection of the SBF, initiating an automatic defence strategy comprising an action known to mitigate a known threat posed by a closest known malicious binary file (KMBF), the method further comprising: identifying the closest KMBF by comparing an SBF application programming interface (API) profile generated in respect of the SBF with respective KMBF API profiles generated in respect of each of a plurality of KMBFs, the SBF and KMBF API profiles being generated by: identifying any API calls in the respective binary file; and assigning each of said identified API calls to one of a plurality of API call categories defined by one or more actions known to be effective in mitigating one or more possible threats posed by the respective API call category.

WO 2022/012821 A1

COMPUTER-IMPLEMENTED AUTOMATIC SECURITY METHODS AND SYSTEMS

FIELD

The present disclosure relates to defending computer systems and networks against malware.

More specifically, aspects relate to computer-implemented methods of automatically securing a computer system or network against suspect binary files, data processing systems configured to perform such methods, computer programs comprising instructions which, when the program is executed by a computer, cause the computer to carry out such methods, computer-readable data carriers having stored thereon such computer programs and data carrier signals carrying such computer programs.

BACKGROUND

Hackers constantly update malware code to add new attack vectors (e.g. new flooding types, propagation methods, ways of stealing credentials, ransomware, etc.), exploit new vulnerabilities, target new operating systems and make the code smaller and more optimised. This results in a high volume of new types of malware entering circulation on an on-going basis.

The typical approach of intrusion detection systems to unknown files introduced into a computer system or network is to attempt to classify them into a known malware “family” (e.g. by functionality or signature), then implement a defence strategy known to work against malware in that family. If an unknown file cannot be classified into a known malware family then some intrusion detection systems assume it is benign, introducing a security risk. In other intrusion detection systems such unclassifiable unknown files are flagged for consideration by a human analyst. There can then be a delay in defensive action until the human analyst has completed their assessment, again introducing a security risk since an attack could be allowed to proceed during that delay. Alternatively, the

system/network can be locked down until the human analyst has completed their assessment, restricting its functionality to what may be an unnecessary extent.

What is needed is an automatic way to determine and implement a suitable defence strategy against unknown files entering a computer system or network.

SUMMARY

According to a first aspect, there is provided a computer-implemented method of automatically securing a computer system or network against a suspect binary file (SBF) by, in response to detection of the SBF, initiating an automatic defence strategy comprising an action known to mitigate a known threat posed by a closest known malicious binary file (KMBF), the method further comprising:

identifying the closest KMBF from a plurality of KMBFs by comparing an SBF branch map generated in respect of the SBF with respective KMBF branch maps generated in respect of each of the plurality of KMBFs, the SBF and KMBF branch maps being generated by breaking each of the respective binary files down into a respective sequence of blocks and determining how each block of the sequence branches to one or more other blocks of the sequence.

The closest KMBF can be identified as the one of the plurality of KMBFs with the highest branch map matching score, the method optionally further comprising:

allocating each of the plurality of KMBFs a branch map matching score by performing tree pattern matching between the respective KMBF and the SBF.

The method can further comprise generating the SBF and KMBF branch maps by identifying any branch instructions in the respective binary file, each of said branch instructions:

delineating the end of a block; and

indicating both:

one or more other blocks said block branches to, and

whether said block branches to each of the one or more other blocks conditionally or unconditionally.

The closest KMBF can be identified as the one of the plurality of KMBFs with the highest branch map matching score, the method optionally further comprising, for each of the KMBFs:

allocating each identified KMBF branch instruction a branch instruction matching score with respect to a corresponding SBF branch instruction according to how close the number of one or more other blocks of the KMBF the KMBF branch instruction indicates branching to is to the number of one or more other blocks of the SBF the corresponding SBF branch instruction indicates branching to; and

allocating the KMBF a branch map matching score by combining all of said branch instruction matching scores.

The method can further comprise, for each of the KMBFs:

allocating each identified KMBF branch instruction its branch instruction matching score further according to whether said KMBF branch instruction indicates branching to a block of the KMBF that corresponds to a block of the SBF which said corresponding SBF branch instruction indicates branching to.

The method can further comprise, for each of the KMBFs:

allocating each identified KMBF branch instruction its branch instruction matching score further according to, when said KMBF branch instruction and said corresponding SBF branch instruction both indicate branching to a plurality of other blocks, whether an alternative block of the KMBF the KMBF branch instruction indicates branching to corresponds to an alternative block of the SBF the SBF branch instructions indicates branching to.

Identifying the closest KMBF can be further performed by comparing an SBF application programming interface (API) profile generated in respect of the SBF with respective KMBF API profiles generated in respect of each of the plurality of KMBFs, the SBF and KMBF API profiles optionally being generated by:

identifying any API calls in the respective binary file; and

assigning each of said identified API calls to one of a plurality of API call categories defined by one or more actions known to be effective in mitigating one or more possible threats posed by the respective API call category.

The method can further comprise:

counting the number of API calls of the SBF assigned to each of the API call categories;

for each of the KMBFs:

counting the number of API calls of the KMBF assigned to each of the API call categories;

allocating each of the API call categories an API call category matching score according to how close the number of API calls of the SBF assigned to that category is to the number of API calls of the KMBF assigned to that category;

allocating the KMBF an API profile matching score by combining all of said API call category matching scores;

allocating a branch map matching score in any of the manners set out above; and

allocating a combined matching score by combining the API profile matching score with the branch map matching score;

wherein the closest KMBF can be identified as the one of the plurality of KMBFs with the highest combined matching score.

The plurality of API call categories can comprise:

API calls which can be stymied by encrypting and/or deleting one or more categories of data;

API calls which can be stymied by enforcing file and/or application access controls;

API calls which can be stymied by blocking one or more categories of transmission and/or reception;

API calls which can be stymied by enforcing process locks and/or memory access controls; and

API calls which can be stymied by raising one or more alerts.

The automatic defence strategy can further comprise a further action predicted to mitigate a predicted threat posed by a discrepant function present in the SBF but not the KMBF.

The method can further comprise, in response to detection of the SBF and prior to initiating the automatic defence strategy:

identifying the discrepant function;

assigning the discrepant function to one of a plurality of function categories defined by one or more actions known to be effective in mitigating one or more possible threats posed by the respective function category; and

selecting the further action from said one or more actions known to be effective in mitigating the one or more threats posed by said one of the plurality of function categories.

The plurality of function categories can comprise:

functions which can be stymied by encrypting and/or deleting one or more categories of data;

functions which can be stymied by enforcing file and/or application access controls;

functions which can be stymied by blocking one or more categories of transmission and/or reception;

functions which can be stymied by enforcing process locks and/or memory access controls; and

functions which can be stymied by raising one or more alerts.

The method can further comprise, in response to detection of the SBF and prior to initiating the automatic defence strategy, determining the further action predicted to mitigate the predicted threat posed by the discrepant function by separating out a portion of the SBF corresponding to the discrepant function and running that portion of the SBF in a controlled virtual environment.

The discrepant function can be identified and assigned to one of the plurality of function categories based on results of running the portion of the SBF in the controlled virtual environment.

The method can further comprise identifying the discrepant function by identifying a discrepant branch of the SBF branch map having no corresponding branch in the closest KMBF branch map.

The portion of the SBF corresponding to the discrepant function which is separated out can be the discrepant branch of the SBF branch map.

The method can further comprise:

generating an SBF application programming interface (API) profile in respect of the SBF and a closest KMBF API profile in respect of the closest KMBF by:

identifying any API calls in the respective binary file; and

assigning each of said identified API calls to one of a plurality of API call categories defined by one or more actions known to be effective in mitigating one or more possible threats posed by the respective API call category; and

identifying the discrepant function by identifying an API call category to which the number of API calls assigned from the closest KMBF is lower than the number of API calls assigned from the SBF.

The action and, when present, the further action, can be selected from:

encrypting and/or deleting one or more categories of data;

enforcing one or more file and/or application access controls;

blocking one or more categories of transmission and/or reception;

enforcing one or more process locks and/or memory access controls; and

raising one or more alerts.

According to a second aspect, there is provided a computer-implemented method of automatically securing a computer system or network against a suspect binary file (SBF) by, in response to detection of the SBF, initiating an automatic defence strategy comprising an action known to mitigate a known threat posed by a closest known malicious binary file (KMBF), the method further comprising:

identifying the closest KMBF from a plurality of KMBFs by comparing an SBF application programming interface (API) profile generated in respect of the

SBF with respective KMBF API profiles generated in respect of each of the plurality of KMBFs, the SBF and KMBF API profiles being generated by:

identifying any API calls in the respective binary file; and

assigning each of said identified API calls to one of a plurality of API call categories defined by one or more actions known to be effective in mitigating one or more possible threats posed by the respective API call category.

The closest KMBF can be identified as the one of the plurality of KMBFs with the highest API profile matching score, the method optionally further comprising:

counting the number of API calls of the SBF assigned to each of the API call categories; and

for each of the KMBFs:

counting the number of API calls of the KMBF assigned to each of the API call categories;

allocating each of the API call categories an API call category matching score according to how close the number of API calls of the SBF assigned to that category is to the number of API calls of the KMBF assigned to that category; and

allocating the KMBF an API profile matching score with respect to the SBF by combining all of said API call category matching scores.

The plurality of API call categories can comprise:

API calls which can be stymied by encrypting and/or deleting one or more categories of data;

API calls which can be stymied by enforcing file and/or application access controls;

API calls which can be stymied by blocking one or more categories of transmission and/or reception;

API calls which can be stymied by enforcing process locks and/or memory access controls; and

API calls which can be stymied by raising one or more alerts.

Identifying the closest KMBF can be further performed by comparing an SBF branch map generated in respect of the SBF with respective KMBF branch maps

generated in respect of each of the plurality of KMBFs, the SBF and KMBF branch maps optionally being generated by breaking each of the respective binary files down into a respective sequence of blocks and determining how each block of the sequence branches to one or more other blocks of the sequence.

The method can further comprise, for each of the KMBFs:

allocating an API profile matching score in the manner set out above;

allocating a branch map matching score by performing tree pattern matching between the KMBF and the SBF; and

allocating a combined matching score by combining the API profile matching score with the branch map matching score;

wherein the closest KMBF can be identified as the one of the plurality of KMBFs with the highest combined matching score.

The method can further comprise generating the SBF and KMBF branch maps by identifying any branch instructions in the respective binary file, each of said branch instructions:

delineating the end of a block; and

indicating both:

one or more other blocks said block branches to, and

whether said block branches to each of the one or more other blocks conditionally or unconditionally.

The method can further comprise, for each of the KMBFs:

allocating an API profile matching score in the manner set out above;

allocating each identified KMBF branch instruction a branch instruction matching score with respect to a corresponding SBF branch instruction according to how close the number of one or more other blocks of the KMBF the KMBF branch instruction indicates branching to is to the number of one or more other blocks of the SBF the corresponding SBF branch instruction indicates branching to;

allocating the KMBF a branch map matching score by combining all of said branch instruction matching scores; and

allocating a combined matching score by combining the API profile matching score with the branch map matching score;

wherein the closest KMBF can be identified as the one of the plurality of KMBFs with the highest combined matching score.

The method can further comprise, for each of the KMBFs:

allocating each identified KMBF branch instruction its branch instruction matching score further according to whether said KMBF branch instruction indicates branching to a block of the KMBF that corresponds to a block of the SBF which said corresponding SBF branch instruction indicates branching to.

The method can further comprise, for each of the KMBFs:

allocating each identified KMBF branch instruction its branch instruction matching score further according to, when said KMBF branch instruction and said corresponding SBF branch instruction both indicate branching to a plurality of other blocks, whether an alternative block of the KMBF the KMBF branch instruction indicates branching to corresponds to an alternative block of the SBF the SBF branch instructions indicates branching to.

The automatic defence strategy can further comprise a further action predicted to mitigate a predicted threat posed by a discrepant function present in the SBF but not the KMBF.

The method can further comprise, in response to detection of the SBF and prior to initiating the automatic defence strategy:

identifying the discrepant function;

assigning the discrepant function to one of a plurality of function categories defined by one or more actions known to be effective in mitigating one or more possible threats posed by the respective function category; and

selecting the further action from said one or more actions known to be effective in mitigating the one or more threats posed by said one of the plurality of function categories.

The plurality of function categories can comprise:

functions which can be stymied by encrypting and/or deleting one or more categories of data;

functions which can be stymied by enforcing file and/or application access controls;

functions which can be stymied by blocking one or more categories of transmission and/or reception;

functions which can be stymied by enforcing process locks and/or memory access controls; and

functions which can be stymied by raising one or more alerts.

The method can further comprise, in response to detection of the SBF and prior to initiating the automatic defence strategy, determining the further action predicted to mitigate the predicted threat posed by the discrepant function by separating out a portion of the SBF corresponding to the discrepant function and running that portion of the SBF in a controlled virtual environment.

The discrepant function can be identified and assigned to one of the plurality of function categories based on results of running the portion of the SBF in the controlled virtual environment.

The method can further comprise:

generating an SBF branch map in respect of the SBF and a closest KMBF branch map in respect of the closest KMBF by breaking each of the respective binary files down into a respective sequence of blocks and determining how each block of the sequence branches to one or more other blocks of the sequence; and

identifying the discrepant function by identifying a discrepant branch of the SBF branch map having no corresponding branch in the closest KMBF branch map.

The portion of the SBF corresponding to the discrepant function which is separated out can be the discrepant branch of the SBF branch map.

The method can further comprise identifying the discrepant function by identifying an API call category to which the number of API calls assigned from the closest KMBF is lower than the number of API calls assigned from the SBF.

The action and, when present, the further action, can be selected from:

encrypting and/or deleting one or more categories of data;

enforcing one or more file and/or application access controls;

blocking one or more categories of transmission and/or reception;

enforcing one or more process locks and/or memory access controls; and

raising one or more alerts.

According to a third aspect, there is provided a computer-implemented method of automatically securing a computer system or network against a suspect binary file (SBF) by, in response to detection of the SBF, initiating an automatic defence strategy comprising:

a first action known to mitigate a known threat posed by a known malicious binary file (KMBF); and

a further action predicted to mitigate a predicted threat posed by a discrepant function present in the SBF but not the KMBF.

The method can further comprise, in response to detection of the SBF and prior to initiating the automatic defence strategy:

identifying the discrepant function;

assigning the discrepant function to one of a plurality of function categories defined by one or more actions known to be effective in mitigating one or more possible threats posed by the respective function category; and

selecting the further action from said one or more actions known to be effective in mitigating the one or more threats posed by said one of the plurality of function categories.

The plurality of function categories can comprise:

functions which can be stymied by encrypting and/or deleting one or more categories of data;

functions which can be stymied by enforcing file and/or application access controls;

functions which can be stymied by blocking one or more categories of transmission and/or reception;

functions which can be stymied by enforcing process locks and/or memory access controls; and

functions which can be stymied by raising one or more alerts.

The first and further actions can be selected from:

encrypting and/or deleting one or more categories of data;

enforcing one or more file and/or application access controls;

blocking one or more categories of transmission and/or reception;

enforcing one or more process locks and/or memory access controls; and

raising one or more alerts.

The method can further comprise, in response to detection of the SBF and prior to initiating the automatic defence strategy, determining the further action predicted to mitigate the predicted threat posed by the discrepant function by separating out a portion of the SBF corresponding to the discrepant function and running that portion of the SBF in a controlled virtual environment.

The discrepant function can be identified and assigned to one of the plurality of function categories based on results of running the portion of the SBF in the controlled virtual environment.

The method can further comprise:

generating an SBF branch map in respect of the SBF and a KMBF branch map in respect of the KMBF by breaking each of the respective binary files down into a respective sequence of blocks and determining how each block of the sequence branches to one or more other blocks of the sequence; and

identifying the discrepant function by identifying a discrepant branch of the SBF branch map having no corresponding branch in the KMBF branch map.

The portion of the SBF corresponding to the discrepant function which is separated out can be the discrepant branch of the SBF branch map.

The method can further comprise selecting the KMBF from a plurality of KMBFs by identifying it as the closest of the plurality of KMBFs to the SBF, said identifying optionally being performed by comparing an SBF branch map generated in respect of the SBF with respective KMBF branch maps generated in respect of each of the plurality of KMBFs, the SBF and KMBF branch maps being generated in the manner set out above.

The closest KMBF can be identified as the one of the plurality of KMBFs with the highest branch map matching score, the method optionally further comprising:

allocating each of the plurality of KMBFs a branch map matching score by performing tree pattern matching between the respective KMBF and the SBF.

The method can further comprise generating the SBF and KMBF branch maps by identifying any branch instructions in the respective binary file, each of said branch instructions:

delineating the end of a block; and

indicating both:

one or more other blocks said block branches to, and

whether said block branches to each of the one or more other blocks conditionally or unconditionally.

The closest KMBF can be identified as the one of the plurality of KMBFs with the highest branch map matching score, the method optionally further comprising, for each of the plurality of KMBFs:

allocating each identified KMBF branch instruction a branch instruction matching score with respect to a corresponding SBF branch instruction according to how close the number of one or more other blocks of the respective KMBF the KMBF branch instruction indicates branching to is to the number of one or more other blocks of the SBF the corresponding SBF branch instruction indicates branching to; and

allocating the respective KMBF a branch map matching score by combining all of said branch instruction matching scores.

The method can further comprise, for each of the KMBFs:

allocating each identified KMBF branch instruction its branch instruction matching score further according to whether said KMBF branch instruction indicates branching to a block of the respective KMBF that corresponds to a block of the SBF which said corresponding SBF branch instruction indicates branching to.

The method can further comprise, for each of the KMBFs:

allocating each identified KMBF branch instruction its branch instruction matching score further according to, when said KMBF branch instruction and said corresponding SBF branch instruction both indicate branching to a plurality of other blocks, whether an alternative block of the respective KMBF the KMBF branch instruction indicates branching to corresponds to an alternative block of the SBF the SBF branch instructions indicates branching to.

Identifying the closest KMBF can be further performed by comparing an SBF application programming interface (API) profile generated in respect of the SBF with respective KMBF API profiles generated in respect of each of the plurality of KMBFs, the SBF and KMBF API profiles optionally being generated by:

identifying any API calls in the respective binary file; and

assigning each of said identified API calls to one of a plurality of API call categories defined by one or more actions known to be effective in mitigating one or more possible threats posed by the respective API call category.

The method can further comprise:

counting the number of API calls of the SBF assigned to each of the API call categories;

for each of the plurality of KMBFs:

counting the number of API calls of the respective KMBF assigned to each of the API call categories;

allocating each of the API call categories an API call category matching score according to how close the number of API calls of the SBF assigned to that category is to the number of API calls of the respective KMBF assigned to that category;

allocating the respective KMBF an API profile matching score by combining all of said API call category matching scores;

allocating a branch map matching score in any of the manners set out above; and

allocating a combined matching score by combining the API profile matching score with the branch map matching score;

wherein the closest KMBF can be identified as the one of the plurality of KMBFs with the highest combined matching score.

The method can further comprise:

generating an SBF application programming interface (API) profile in respect of the SBF and a KMBF API profile in respect of the KMBF by:

identifying any API calls in the respective binary file; and

assigning each of said identified API calls to one of a plurality of API call categories defined by one or more actions known to be effective in mitigating one or more possible threats posed by the respective API call category; and

identifying the discrepant function by identifying an API call category to which the number of API calls assigned from said KMBF is lower than the number of API calls assigned from the SBF.

The method can further comprise selecting the KMBF from a plurality of KMBFs by identifying it as the closest of the plurality of KMBFs to the SBF, said identifying being performed by comparing an SBF application programming interface (API) profile generated in respect of the SBF with respective KMBF API profiles generated in respect of each of the plurality of KMBFs, the SBF and KMBF API profiles optionally being generated by:

identifying any API calls in the respective binary file; and

assigning each of said identified API calls to one of a plurality of API call categories defined by one or more actions known to be effective in

mitigating one or more possible threats posed by the respective API call category.

The closest KMBF can be identified as the one of the plurality of KMBFs with the highest API profile matching score, the method further comprising:

counting the number of API calls of the SBF assigned to each of the API call categories; and

for each of the plurality of KMBFs:

counting the number of API calls of the respective KMBF assigned to each of the API call categories;

allocating each of the API call categories an API call category matching score according to how close the number of API calls of the SBF assigned to that category is to the number of API calls of the respective KMBF assigned to that category; and

allocating the respective KMBF an API profile matching score with respect to the SBF by combining all of said API call category matching scores.

The plurality of API call categories can comprise:

API calls which can be stymied by encrypting and/or deleting one or more categories of data;

API calls which can be stymied by enforcing file and/or application access controls;

API calls which can be stymied by blocking one or more categories of transmission and/or reception;

API calls which can be stymied by enforcing process locks and/or memory access controls; and

API calls which can be stymied by raising one or more alerts.

According to a fourth aspect, there is provided a data processing system configured to perform the method of any of the first to third aspects.

According to a fifth aspect, there is provided a computer program comprising instructions which, when the program is executed by a computer, cause the computer to carry out the method of any of the first to third aspects.

According to a sixth aspect, there is provided a computer-readable data carrier having stored thereon the computer program of the fifth aspect.

According to a seventh aspect, there is provided a data carrier signal carrying the computer program of the fifth aspect.

BRIEF DESCRIPTION OF THE FIGURES

Aspects of the present disclosure will now be described by way of example with reference to the accompanying figures. In the figures:

Figure 1 schematically illustrates an example system in which the methods described herein could be employed;
Figure 2 is a flowchart illustrating a first computer-implemented method of automatically securing a computer system or network against an SBF;
Figure 3A illustrates an example branch map for a KMBF;
Figure 3B illustrates an example branch map for an SBF;
Figure 4 is a flowchart illustrating a second computer-implemented method of automatically securing a computer system or network against an SBF; and
Figure 5 is a flowchart illustrating a third computer-implemented method of automatically securing a computer system or network against an SBF.

DETAILED DESCRIPTION OF THE FIGURES

The following description is presented to enable any person skilled in the art to make and use the system, and is provided in the context of a particular application. Various modifications to the disclosed embodiments will be readily apparent to those skilled in the art.

Several novel approaches to automatically securing computer systems and networks against suspect binary files (SBFs) will now be described. All of these methods involve comparing an SBF to one or more known malware binary files (KMBFs) in order to predict possible threats posed by the SBF and implement a

suitable automatic defence strategy accordingly, even where the SBF defies classification into an existing KMBF family.

One approach is to perform malware matching by branch mapping. That is, an SBF is matched to a closest KMBF by comparing maps of how blocks of code branch to one another in the SBF and each of a plurality of candidate closest KMBFs. (Branching can for example be via unconditional branch instructions such as “jump” or conditional branch instructions such as “compare”.) A defence strategy known to work against the identified closest KMBF is then implemented.

Another approach is to perform malware matching by application programming interface (API) profiling. That is, an SBF is matched to a closest KMBF by comparing profiles of the categories of API calls present in the SBF and each of a plurality of candidate closest KMBFs. The API call categories are defined by defensive actions known to work against those types of API calls, i.e. known to mitigate possible threats posed by those types of API calls. A defence strategy known to work against the identified closest KMBF is then implemented.

Yet another approach is to implement a combined matching and non-matching function defence strategy. This involves identifying one or more discrepant functions (e.g. API calls) in an SBF not present in a particular KMBF. Those discrepant functions are classified into categories defined by defensive actions known to work against those types of function, i.e. known to mitigate possible threats posed by those types of function. A defence strategy is then implemented comprising both mitigation actions known to work against the KMBF, and mitigation actions known to work against the categories of functions into which the discrepant functions fall.

The above approaches can be combined in various ways. A closest KMBF can be identified by a combination of branch mapping and API profiling. The KMBF compared to the SBF for the purposes of determining a combined matching and non-matching defence strategy can be a closest KMBF of a plurality of candidate closest KMBFs identified with the aid of branch mapping and/or API profiling. The

discrepant functions can be identified with the aid of branch mapping and/or API profiling of the SBF and KMBF.

The defence strategies according to any of the above approaches can for example comprise actions selected from:

- encrypting and/or deleting one or more categories of data;
- enforcing one or more file and/or application access controls;
- blocking one or more categories of transmission and/or reception;
- enforcing one or more process locks and/or memory access controls; and
- raising one or more alerts.

Figure 1 schematically illustrates an example system 1000 in which these methods could be employed.

The system 1000 comprises a data processing system 1100, such as a server, capable of performing the methods described herein. It comprises a processor 1110 operably coupled to both a memory 1120 and an interface 1130. The memory 1120 can optionally store a computer program comprising instructions which, when the program is executed by the processor 1110, cause the data processing system 1100 to carry out some or all steps of the methods described herein. Alternatively or additionally, the interface 1130 can optionally comprise one or both of a physical interface 1131 configured to receive a data carrier having such instructions stored thereon and a receiver 1132 configured to receive a data carrier signal carrying such instructions. The receiver 1132 (when present) can comprise one or more wireless receiver modules and/or one or more wired receiver modules.

The data processing system 1100 can be communicably coupled to one or more other computing systems such as user devices 1200, for example via a network 1300. The methods described herein can for example be used to protect a network 1300 and computing systems 1100, 1200 connected to it from malicious binary files introduced to one of those computing systems. Introduction of such binary files could for example be via downloading of email attachments on one of

the user devices 1200 or connection of a peripheral file storage device 1400 such as a universal serial bus (USB) memory stick to one of the user devices 1200.

Branch mapping

Figure 2 is a flowchart illustrating a computer-implemented method 200 of automatically securing a computer system or network against an SBF.

The method 200 comprises identifying a closest KMBF to an SBF from a plurality of KMBFs at step 240 and then initiating an automatic defence strategy comprising an action known to mitigate a known threat posed by that closest KMBF at step 290.

In response to identification of the closest KMBF at step 240, the action known to mitigate the known threat posed by that closest KMBF is selected at step 250. Step 250 may be performed by the computing system which performs steps 240 and 290, or may be outsourced to another computing system communicably coupled to it, for example a remote computing system which stores a database of defensive actions known to mitigate certain threats.

Steps 240 to 290 are performed in response to detection of an SBF at step 220. Step 220 may be performed by the computing system which performs steps 240 and 290, or that computing system may be informed of detection of the SBF through a network or user interface. For example, in the system 1000 one of the user devices 1200 could detect the SBF, e.g. while scanning email attachments or files stored on a USB memory stick 1400 connected to it. The user device 1200 could then alert the server 1100, which performs step 240 in response to receiving this alert.

Step 240 comprises comparing an SBF branch map generated in respect of the SBF with respective KMBF branch maps generated in respect of each of the plurality of KMBFs. The SBF and KMBF branch maps are generated by breaking each of the respective binary files down into a respective sequence of blocks and

determining how each block of the sequence branches to one or more other blocks of the sequence.

The SBF branch map is generated at step 230, in response to detection of the SBF at step 220. Step 230 could be performed by the same computing system that performs steps 240 and 290, the computing system that performs step 220 (if different), or another computing system communicably coupled to both. In the example described above in relation to the system 1000 of Figure 1 the user device 1200 that detects the SBF could also generate the SBF branch map and communicate it to the server within its alert message. Alternatively, the user device 1200 could communicate the SBF itself in the alert message so that the server 1100 can generate the SBF branch map. The former approach will generally result in lower traffic demands on the network 1300 than the latter, while the latter approach will generally result in lower processing, memory and electrical power demands on the user device 1200 (which can be particularly beneficial if the user device 1200 is a mobile device).

The KMBF branch maps are generated at step 210. This can be done at any time prior to step 240. Step 210 could be performed by the same computing system that performs steps 240 and 290 or another computing system communicably coupled to it. Step 210 can for example be performed on an ongoing basis, with a new KMBF branch map being generated in respect of each new KMBF identified (e.g. by security analysts), in response to that identification. In this way, a database of KMBF branch maps can be built up over time, for example in the memory 1120 of the data processing system 1100.

The closest KMBF can be identified at step 240 as the one of the plurality of KMBFs with the highest branch map matching score.

Step 240 can optionally comprise allocating each of the plurality of KMBFs a branch map matching score using pattern matching techniques, for example by performing tree pattern matching between the respective KMBF and the SBF.

Generating the SBF and KMBF branch maps at steps 230 and 210 respectively can optionally be performed by identifying any branch instructions in the respective binary file, wherein each of said branch instructions delineates the end of a block and indicates both one or more other blocks said block branches to and whether said block branches to each of the one or more other blocks conditionally or unconditionally. In that case, step 240 can comprise, for each of the KMBFs, allocating each identified KMBF branch instruction a branch instruction matching score with respect to a corresponding SBF branch instruction according to how close the number of one or more other blocks of the KMBF the KMBF branch instruction indicates branching to is to the number of one or more other blocks of the SBF the corresponding SBF branch instruction indicates branching to. That KMBF can then be allocated a branch map matching score by combining all of said branch instruction matching scores.

Each branch instruction matching score can be allocated further according to whether the respective KMBF branch instruction indicates branching to a block of the KMBF that corresponds to a block of the SBF which said corresponding SBF branch instruction indicates branching to. When said KMBF branch instruction and said corresponding SBF branch instruction both indicate branching to a plurality of other blocks, the branch instruction matching score can be allocated further according to whether an alternative block of the KMBF the KMBF branch instruction indicates branching to corresponds to an alternative block of the SBF the SBF branch instructions indicates branching to.

Corresponding blocks of the SBF and each KMBF can for example be identified according to where they fall in their respective sequences and/or in other ways, e.g. according to their content. For example correspondence may be implied by similar functionality such as the presence of functions (e.g. API calls) which fall into the same category, for example according to a categorisation of the type described below in relation to API profiling and/or discrepant function classification.

Figures 3A and 3B respectively illustrate example branch maps for a KMBF and an SBF. Each binary file comprises seven blocks of code, K1 to K7 and S1 to S7

respectively. However, the branching between these blocks differs. Conditional branching is indicated by dashed arrows for the positive branch and dotted arrows for the negative branch. Unconditional branching is indicated by dot-dash arrows.

As shown in Figure 3A, block K1 of the KMBF branches conditionally to block K2 in the positive condition and block K3 in the negative. Block K2 branches unconditionally to block K5. Block K3 branches conditionally to block K4 in the positive condition and block K6 in the negative. Block K4 branches unconditionally to block K5. Block K5 branches unconditionally to block K7. Block K6 branches unconditionally to block K7. Block K7 is the final block.

As shown in Figure 3B, block S1 of the SBF branches conditionally to block S2 in the positive condition and block S3 in the negative. Block S2 branches conditionally to block S3 in the positive condition and block S5 in the negative. Block S3 branches conditionally to block S4 in the positive condition and block S6 in the negative. Block S4 branches conditionally to block S5 in the positive condition and block S6 in the negative. Block S5 branches conditionally to block S7 in the positive condition and block S6 in the negative. Blocks S6 and S7 are alternative final blocks.

Comparing the KMBF branch map of Figure 3A with the SBF branch map of Figure 3B, we can arrive at the following example analysis, wherein branch instruction matching scores between 0 and 1 are allocated for each block according to how similar the immediate branching from that block to the next block(s) is between the KMBF and the SBF. In this scheme:

- a branch instruction matching score of 0 indicates no match between the immediate route of the respective branch map onwards from a respective block of the KMBF and a corresponding block of the SBF;
- a branch instruction matching score of 0.5 indicates that one of two branches from a respective block match between the KMBF block and the corresponding SBF block; and
- a branch instruction matching score of 1 indicates a perfect match between the immediate route of the respective branch map onwards from a respective block of the KMBF and a corresponding block of the SBF.

A branch map matching score between 0 and 1 is then allocated as the mean of all of the branch instruction matching scores, wherein a branch map matching score of 1 would indicate identical branch maps.

Block no.	**KMBF block(s) branched to**	**SBF block(s) branched to**	**Branch instruction matching score**
1	2, 3	2, 3	(1+1)/2 = 1
2	5	3, 5	(0+1)/2 = 0.5
3	4, 6	4, 6	(1+1)/2 = 1
4	5	5, 6	(1+0)/2 = 0.5
5	7	7, 6	(1+0)/2 = 0.5
6	7	none	0
	Branch map matching score		(1+0.5+1+0.5+0.5+0)/6 =0.583

Table 1

Table 1 illustrates one example scheme for allocating a branch map matching score; other schemes could alternatively be used.

Note that two binary files having identical branch maps does not necessarily imply that the binary files themselves are identical, since the contents of one or more of the blocks could differ. A defence strategy selected based solely on branch map comparison therefore won't always be optimal. However, most novel malware is simply existing malware with only minor modifications, which translate to little or no change to the branch map. If a novel malicious binary file has been created by making some minor modifications to a KMBF, then its branch map matching score with respect to that KMBF is likely to be high. In this way, branch mapping can be used to help identify such novel malicious binary files and provide a fast route to a defence strategy which is likely to be suitable; i.e. this approach leads to 'quick wins' in novel malware defence.

API profiling

Figure 4 is a flowchart illustrating another computer-implemented method 400 of automatically securing a computer system or network against an SBF.

The method 400 comprises identifying a closest KMBF to an SBF from a plurality of KMBFs at step 440 and then initiating an automatic defence strategy comprising an action known to mitigate a known threat posed by that closest KMBF at step 490.

In response to identification of the closest KMBF at step 440, the action known to mitigate the known threat posed by that closest KMBF is selected at step 450. Step 450 may be performed by the computing system which performs steps 440 and 490, or may be outsourced to another computing system communicably coupled to it, for example a remote computing system which stores a database of defensive actions known to mitigate certain threats.

Steps 440 to 490 are performed in response to detection of an SBF at step 420. Step 420 may be performed by the computing system which performs steps 440 and 490, or that computing system may be informed of detection of the SBF through a network or user interface. For example, in the system 1000 one of the user devices 1200 could detect the SBF, e.g. while scanning email attachments or files stored on a USB memory stick 1400 connected to it. The user device 1200 could then alert the server 1100, which performs step 440 in response to receiving this alert.

Step 440 comprises comparing an SBF API profile generated in respect of the SBF with respective KMBF API profiles generated in respect of each of the plurality of KMBFs. The SBF and KMBF API profiles are generated by identifying any API calls in the respective binary file and assigning each of said identified API calls to one of a plurality of API call categories defined by one or more actions known to be effective in mitigating one or more possible threats posed by the respective API call category.

The SBF API profile is generated at step 430, in response to detection of the SBF at step 420. Step 430 could be performed by the same computing system that performs steps 440 and 490, the computing system that performs step 420 (if different), or another computing system communicably coupled to both. In the example described above in relation to the system 1000 of Figure 1 the user device 1200 that detects the SBF could also generate the SBF API profile and communicate it to the server within its alert message. Alternatively, the user device 1200 could communicate the SBF itself in the alert message so that the server 1100 can generate the SBF API profile. The former approach will generally result in lower traffic demands on the network 1300 than the latter, while the latter approach will generally result in lower processing, memory and electrical power demands on the user device 1200 (which can be particularly beneficial if the user device 1200 is a mobile device).

The KMBF API profiles are generated at step 410. This can be done at any time prior to step 440. Step 410 could be performed by the same computing system that performs steps 440 and 490 or another computing system communicably coupled to it. Step 410 can for example be performed on an ongoing basis, with a new KMBF API profile being generated in respect of each new KMBF identified (e.g. by security analysts), in response to that identification. In this way, a database of KMBF API profiles can be built up over time, for example in the memory 1120 of the data processing system 1100.

Step 440 can comprise counting the respective number of API calls of the SBF and of each of the KMBFs assigned to each of the API call categories. For each of the KMBFs, an API call category matching score can then be assigned to each of the API call categories according to how close the number of API calls of the SBF assigned to that category is to the number of API calls of the KMBF assigned to that category. Each KMBF can be assigned an API profile matching score with respect to the SBF by combining all of said API call category matching scores. The closest KMBF can be identified as the one of the plurality of KMBFs with the highest API profile matching score.

Combination of the API call category matching scores to assign an API profile matching score could for example involve summing the API call category matching scores together. Such a summing operation could be weighted such that higher weighting can be given to the API call categories which are most commonly found in malware and/or which generally result in the most detrimental effects when implemented by malware.

The plurality of API call categories can for example be defined by types of mitigation action as follows.

API call category	Mitigation action	Example API call type(s)	Category weight
1	encrypt/delete one or more categories of data	• file read • file transmit • file copy	0.5
2	enforce file/application access controls	• file modify • file delete • file encrypt	0.7
3	block one or more categories of transmission and/or reception (e.g. communication over a certain port or a certain protocol interface) - can be implemented at host e.g. via API or on network e.g. via firewall	• socket • network	0.2
4	process lock/memory access control	• process injection (e.g. creating virtual memory)	0.8

Table 2

This categorisation enables fast defence strategy development. For example, if the closest KMBF identified at step 440 is known to comprise an API call in category 3 then step 450 can comprise selecting a transmission block action.

Identification of the closest KMBF at step 440 could optionally further comprise comparison of branch maps for the SBF and each of the plurality of KMBFs as described above.

Combined matching and non-matching defence strategy

Figure 5 is a flowchart illustrating another computer-implemented method 500 of automatically securing a computer system or network against an SBF.

The method 500 comprises initiating an automatic defence strategy comprising a first action known to mitigate a known threat posed by a KMBF and a further action predicted to mitigate a predicted threat posed by a discrepant function present in the SBF but not the KMBF at step 590. In this way, the defence strategy is tailored according to how the SBF differs from the KMBF.

A suitable KMBF is identified at step 540, either by the computing system which performs step 590 or another computing system. In response to identification of the KMBF at step 540, the action known to mitigate the known threat posed by that KMBF is selected at step 550. Step 550 may be performed by the computing system which performs step 540, the computing system which performs step 590 (if different), or may be outsourced to another computing system communicably coupled to both of them, for example a remote computing system which stores a database of defensive actions known to mitigate certain threats.

Step 590 is performed in response to detection of an SBF at step 520. Step 520 may be performed by the computing system which perform step 590, or that computing system may be informed of detection of the SBF through a network or user interface. For example, in the system 1000 one of the user devices 1200 could detect the SBF, e.g. while scanning email attachments or files stored on a USB memory stick 1400 connected to it. The user device 1200 could then alert the server 1100, which performs step 590 in response to receiving this alert.

The discrepant function is identified at step 560, in response to identification of the KMBF at step 540. Step 560 may be performed by the computing system

which performs step 540, or another computing system communicably coupled to it.

The further action predicted to mitigate the predicted threat posed by the discrepant function present in the SBF but not the KMBF is selected at step 580. Step 580 may be performed by the computing system which performs step 590, or may be outsourced to another computing system communicably coupled to it, for example a remote computing system which stores a database of defensive actions known to mitigate certain threats. This could for example be the same computing system that performs step 550.

Step 580 can be informed by optional step 570, wherein the discrepant function identified at step 560 is assigned to one of a plurality of function categories defined by one or more actions known to be effective in mitigating one or more possible threats posed by the respective function category. The further action selected at step 580 can then be selected from said one or more actions. Step 570, when present, can be performed by the computing system which performs step 560, the computing system which performs step 580 (if different), or another computing system communicably coupled to both of them.

The plurality of function categories can for example be defined by types of mitigation action as follows.

Function category	Mitigation action	Example function type(s)
1	encrypt/delete one or more categories of data	• file read • file transmit • file copy
2	enforce file/application access controls	• file modify • file delete • file encrypt
3	block one or more categories of transmission and/or reception (e.g. communication over a certain port or a certain protocol interface) - can be implemented at host e.g. via API or on network e.g. via firewall	• socket • network
4	process lock/memory access control	• process injection (e.g. creating virtual memory)

Table 3

This categorisation enables fast defence strategy development. For example, if the KMBF identified at step 540 is known to comprise functions in categories 1 and 3 then step 550 can comprise selecting an encryption action and a transmission block action. If the discrepant function identified at step 560 is categorised into category 4 then step 580 can comprise selecting a process lock action.

The discrepant function could be identified at step 560 in various ways. For example, each function in the SBF and the KMBF could be categorised (e.g. according to the scheme of Table 3 or similar) to form a function profile (similar to the API profiles described above) and any SBF function in a category which does not appear in the KMBF function profile could be identified as discrepant. The discrepant function can for example be an API call. Identification of the discrepant

function at step 560 can optionally involve one or both of branch mapping and API profiling as described above.

Identification of the KMBF at step 540 can optionally comprise identifying a closest KMBF by one or both of branch mapping and API profiling as described above.

Between detection of the SBF at step 520 and initiation of the automatic defence strategy at step 590, the further action predicted to mitigate the predicted threat posed by the discrepant function can be determined by separating out a portion of the SBF corresponding to the discrepant function and running that portion of the SBF in a controlled virtual environment such as a sandbox. This is faster and less resource-intensive than running the entire SBF in a sandbox but yields almost as much information. Identification and categorisation of the discrepant function at steps 560 and 570 could for example be performed based on results of running the separated out potion of the SBF in a controlled virtual environment in this way. The portion of the SBF corresponding to the discrepant function can for example be identified by means of branch mapping.

Variations

Other embodiments will be apparent to those skilled in the art from consideration of the specification and practice of the embodiments disclosed herein. It is intended that the specification and examples be considered as exemplary only.

In addition, where this application has listed the steps of a method or procedure in a specific order, it could be possible, or even expedient in certain circumstances, to change the order in which some steps are performed, and it is intended that the particular steps of the method or procedure claims set forth herein not be construed as being order-specific unless such order specificity is expressly stated in the claim. That is, the operations/steps may be performed in any order, unless otherwise specified, and embodiments may include additional or fewer operations/steps than those disclosed herein. It is further contemplated that executing or performing a particular operation/step before,

contemporaneously with, or after another operation is in accordance with the described embodiments.

The methods described herein may be encoded as executable instructions embodied in a computer readable medium, including, without limitation, non-transitory computer-readable storage, a storage device, and/or a memory device. Such instructions, when executed by a processor (or one or more computers, processors, and/or other devices) cause the processor (the one or more computers, processors, and/or other devices) to perform at least a portion of the methods described herein. A non-transitory computer-readable storage medium includes, but is not limited to, volatile memory, non-volatile memory, magnetic and optical storage devices such as disk drives, magnetic tape, compact discs (CDs), digital versatile discs (DVDs), or other media that are capable of storing code and/or data.

Where a processor is referred to herein, this is to be understood to refer to a single processor or multiple processors operably connected to one another. Similarly, where a memory is referred to herein, this is to be understood to refer to a single memory or multiple memories operably connected to one another.

The methods and processes can also be partially or fully embodied in hardware modules or apparatuses or firmware, so that when the hardware modules or apparatuses are activated, they perform the associated methods and processes. The methods and processes can be embodied using a combination of code, data, and hardware modules or apparatuses.

Examples of processing systems, environments, and/or configurations that may be suitable for use with the embodiments described herein include, but are not limited to, embedded computer devices, personal computers, server computers (specific or cloud (virtual) servers), hand-held or laptop devices, multiprocessor systems, microprocessor-based systems, set top boxes, programmable consumer electronics, mobile telephones, network personal computers (PCs), minicomputers, mainframe computers, distributed computing environments that include any of the above systems or devices, and the like. Hardware modules or

apparatuses described in this disclosure include, but are not limited to, application-specific integrated circuits (ASICs), field-programmable gate arrays (FPGAs), dedicated or shared processors, and/or other hardware modules or apparatuses.

User devices can include, without limitation, static user devices such as PCs and mobile user devices such as smartphones, tablets, laptops and smartwatches.

Receivers and transmitters as described herein may be standalone or may be comprised in transceivers. Communicable coupling as described herein involves at least one communication link comprising at least one transmitter capable of transmitting data to at least one receiver over one or more wired or wireless communication channels. Wired communication channels can be arranged for electrical or optical transmission. Such a communication link can optionally further comprise one or more relaying transceivers.

User input devices can include, without limitation, microphones, buttons, keypads, touchscreens, touchpads, trackballs, joysticks, mice, gesture control devices and brain control (e.g. electroencephalography, EEG) devices. User output devices can include, without limitation, speakers, buzzers, display screens, projectors, indicator lights, haptic feedback devices and refreshable braille displays. User interface devices can comprise one or more user input devices, one or more user output devices, or both.

CLAIMS

1. A computer-implemented method of automatically securing a computer system or network against a suspect binary file, 'SBF', by, in response to detection of the SBF, initiating an automatic defence strategy comprising an action known to mitigate a known threat posed by a closest known malicious binary file, 'KMBF', the method further comprising:

identifying the closest KMBF from a plurality of KMBFs by comparing an SBF application programming interface, 'API', profile generated in respect of the SBF with respective KMBF API profiles generated in respect of each of the plurality of KMBFs, the SBF and KMBF API profiles being generated by:

identifying any API calls in the respective binary file; and

assigning each of said identified API calls to one of a plurality of API call categories defined by one or more actions known to be effective in mitigating one or more possible threats posed by the respective API call category.

2. The method of claim 1, wherein the closest KMBF is identified as the one of the plurality of KMBFs with the highest API profile matching score, the method further comprising:

counting the number of API calls of the SBF assigned to each of the API call categories; and

for each of the KMBFs:

counting the number of API calls of the KMBF assigned to each of the API call categories;

allocating each of the API call categories an API call category matching score according to how close the number of API calls of the SBF assigned to that category is to the number of API calls of the KMBF assigned to that category; and

allocating the KMBF an API profile matching score with respect to the SBF by combining all of said API call category matching scores.

3. The method of either of claims 1 or 2, wherein the plurality of API call categories comprise:

API calls which can be stymied by encrypting and/or deleting one or more categories of data;

API calls which can be stymied by enforcing file and/or application access controls;

API calls which can be stymied by blocking one or more categories of transmission and/or reception;

API calls which can be stymied by enforcing process locks and/or memory access controls; and

API calls which can be stymied by raising one or more alerts.

4. The method of any preceding claim, wherein identifying the closest KMBF is further performed by comparing an SBF branch map generated in respect of the SBF with respective KMBF branch maps generated in respect of each of the plurality of KMBFs, the SBF and KMBF branch maps being generated by breaking each of the respective binary files down into a respective sequence of blocks and determining how each block of the sequence branches to one or more other blocks of the sequence.

5. The method of claim 4, further comprising, for each of the KMBFs:

allocating an API profile matching score in the manner set out in claim 2;

allocating a branch map matching score by performing tree pattern matching between the KMBF and the SBF; and

allocating a combined matching score by combining the API profile matching score with the branch map matching score;

wherein the closest KMBF is identified as the one of the plurality of KMBFs with the highest combined matching score.

6. The method of claim 4, further comprising generating the SBF and KMBF branch maps by identifying any branch instructions in the respective binary file, each of said branch instructions:

delineating the end of a block; and

indicating both:

one or more other blocks said block branches to, and

whether said block branches to each of the one or more other blocks conditionally or unconditionally.

7. The method of claim 6, further comprising, for each of the KMBFs:

allocating an API profile matching score in the manner set out in claim 2;

allocating each identified KMBF branch instruction a branch instruction matching score with respect to a corresponding SBF branch instruction according to how close the number of one or more other blocks of the KMBF the KMBF branch instruction indicates branching to is to the number of one or more other blocks of the SBF the corresponding SBF branch instruction indicates branching to;

allocating the KMBF a branch map matching score by combining all of said branch instruction matching scores; and

allocating a combined matching score by combining the API profile matching score with the branch map matching score;

wherein the closest KMBF is identified as the one of the plurality of KMBFs with the highest combined matching score.

8. The method of claim 7, further comprising, for each of the KMBFs:

allocating each identified KMBF branch instruction its branch instruction matching score further according to whether said KMBF branch instruction indicates branching to a block of the KMBF that corresponds to a block of the SBF which said corresponding SBF branch instruction indicates branching to.

9. The method of claim 8, further comprising, for each of the KMBFs:

allocating each identified KMBF branch instruction its branch instruction matching score further according to, when said KMBF branch instruction and said corresponding SBF branch instruction both indicate branching to a plurality of other blocks, whether an alternative block of the KMBF the KMBF branch instruction indicates branching to corresponds to an alternative block of the SBF the SBF branch instructions indicates branching to.

10. The method of any preceding claim, wherein the automatic defence strategy further comprises a further action predicted to mitigate a predicted threat posed by a discrepant function present in the SBF but not the KMBF.

11. The method of claim 10, further comprising, in response to detection of the SBF and prior to initiating the automatic defence strategy:

identifying the discrepant function;

assigning the discrepant function to one of a plurality of function categories defined by one or more actions known to be effective in mitigating one or more possible threats posed by the respective function category; and

selecting the further action from said one or more actions known to be effective in mitigating the one or more threats posed by said one of the plurality of function categories.

12. The method of claim 11, wherein the plurality of function categories comprise:

functions which can be stymied by encrypting and/or deleting one or more categories of data;

functions which can be stymied by enforcing file and/or application access controls;

functions which can be stymied by blocking one or more categories of transmission and/or reception;

functions which can be stymied by enforcing process locks and/or memory access controls; and

functions which can be stymied by raising one or more alerts.

13. The method of any of claims 10 to 12, further comprising, in response to detection of the SBF and prior to initiating the automatic defence strategy, determining the further action predicted to mitigate the predicted threat posed by the discrepant function by separating out a portion of the SBF corresponding to the discrepant function and running that portion of the SBF in a controlled virtual environment.

14. The method of claim 13 as dependent directly or indirectly on claim 11, wherein the discrepant function is identified and assigned to one of the plurality of function categories according to claim 11 based on results of running the portion of the SBF in the controlled virtual environment according to claim 13.

15. The method of any of claims 10 to 14, further comprising:

generating an SBF branch map in respect of the SBF and a closest KMBF branch map in respect of the closest KMBF by breaking each of the respective binary files down into a respective sequence of blocks and determining how each block of the sequence branches to one or more other blocks of the sequence; and

identifying the discrepant function by identifying a discrepant branch of the SBF branch map having no corresponding branch in the closest KMBF branch map.

16. The method of claim 15, wherein the portion of the SBF corresponding to the discrepant function which is separated out according to claim 10 is the discrepant branch of the SBF branch map identified according to claim 15.

17. The method of any of claims 10 to 16, further comprising identifying the discrepant function by identifying an API call category to which the number of API calls assigned from the closest KMBF is lower than the number of API calls assigned from the SBF.

18. The method of any preceding claim, wherein the action and, when dependent on any of claims 10 to 17, the further action, are selected from:

encrypting and/or deleting one or more categories of data;

enforcing one or more file and/or application access controls;

blocking one or more categories of transmission and/or reception;

enforcing one or more process locks and/or memory access controls; and

raising one or more alerts.

19. A data processing system configured to perform the method of any preceding claim.

20. A computer program comprising instructions which, when the program is executed by a computer, cause the computer to carry out the method of any of claims 1 to 18.

21. A computer-readable data carrier having stored thereon the computer program of claim 20.

22. A data carrier signal carrying the computer program of claim 20.

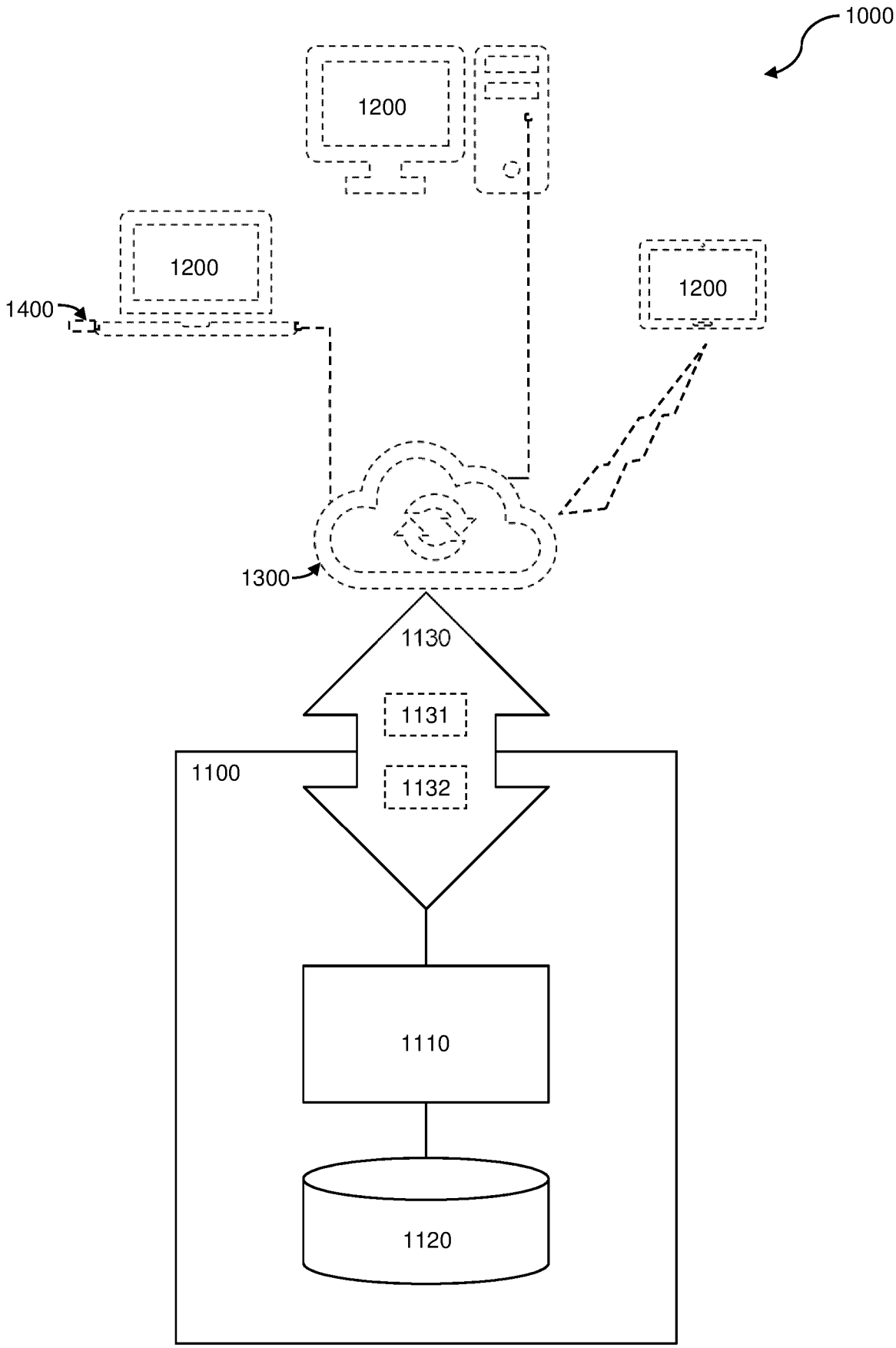

Figure 1

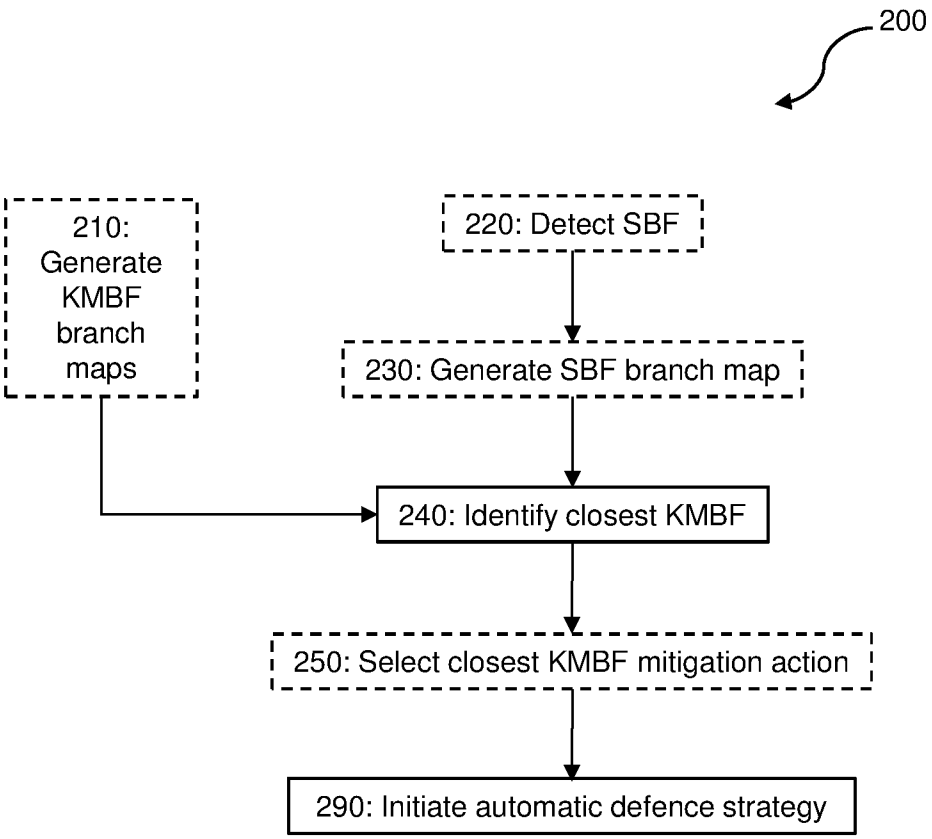
200
210: Generate KMBF branch maps
220: Detect SBF
230: Generate SBF branch map
240: Identify closest KMBF
250: Select closest KMBF mitigation action
290: Initiate automatic defence strategy

Figure 2

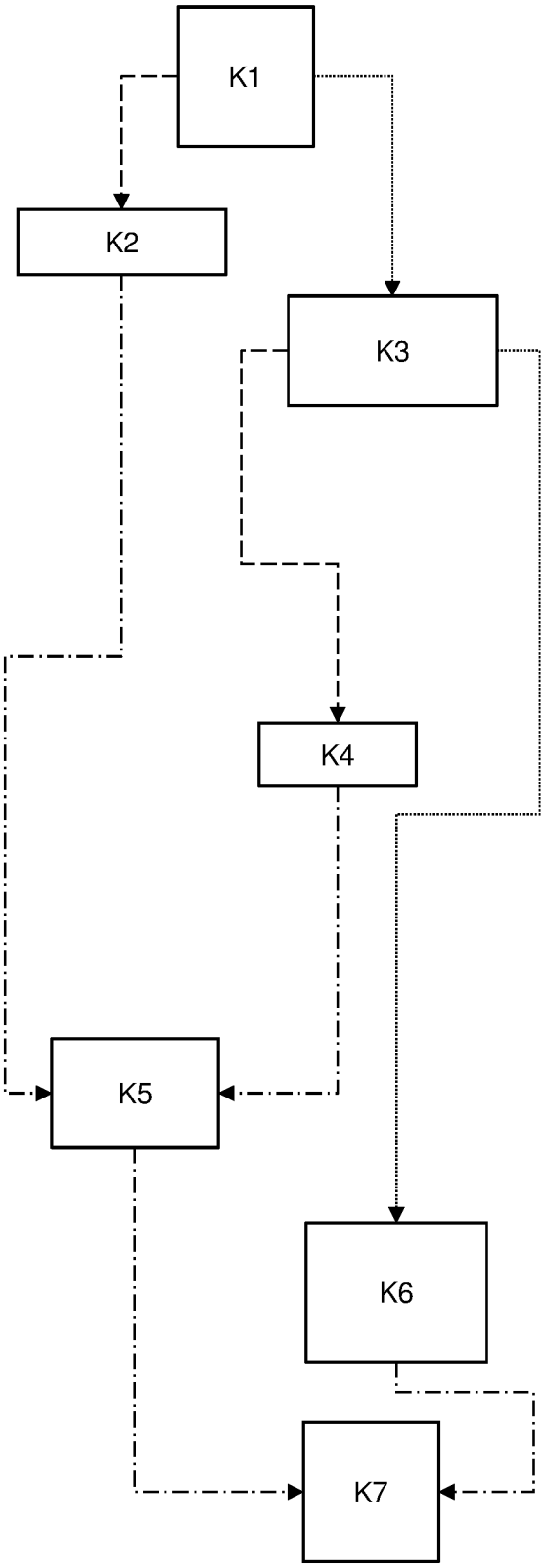

Figure 3A

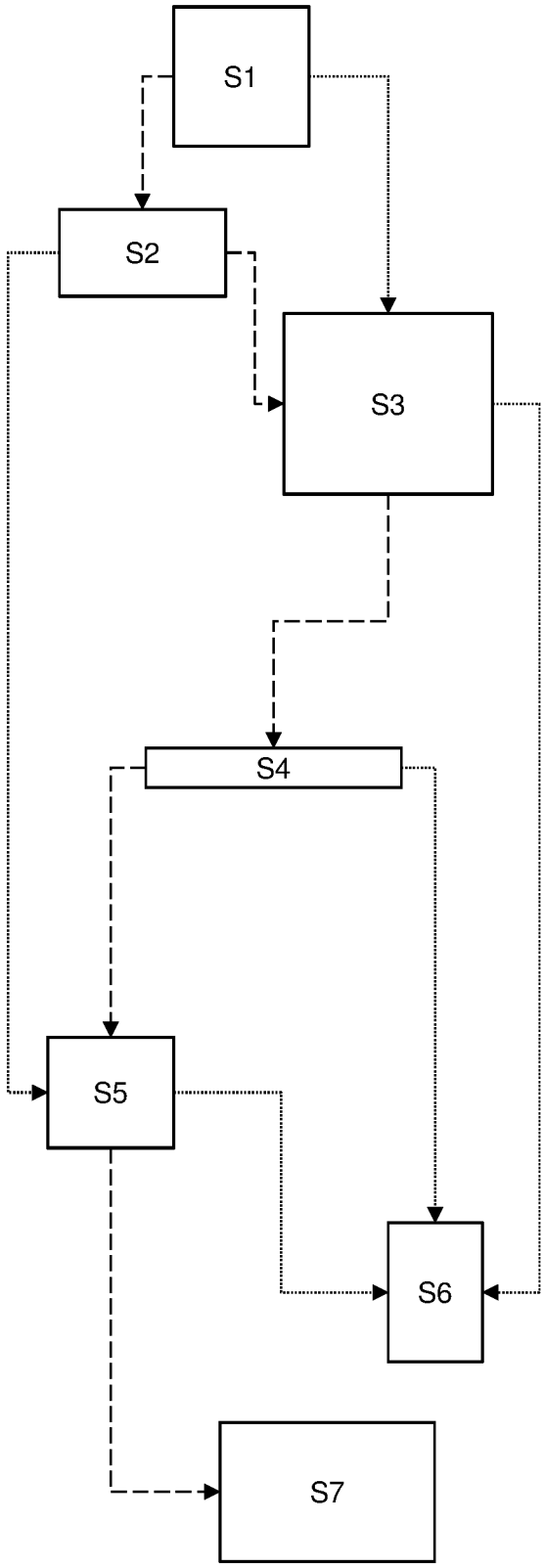

Figure 3B

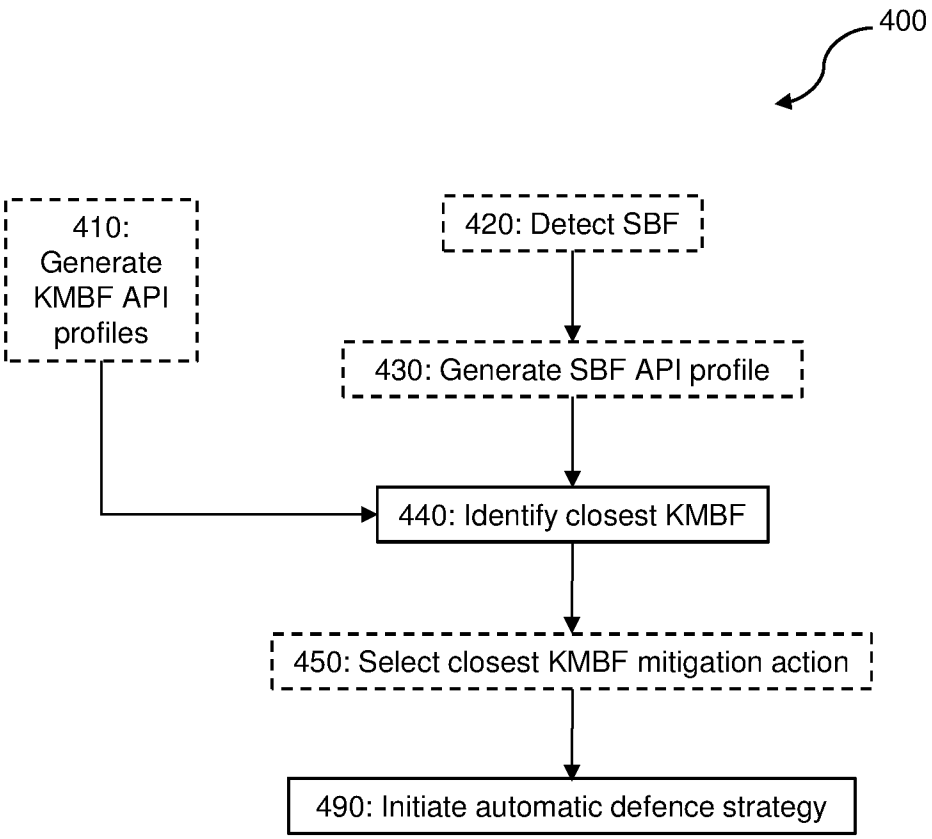
400
410: Generate KMBF API profiles
420: Detect SBF
430: Generate SBF API profile
440: Identify closest KMBF
450: Select closest KMBF mitigation action
490: Initiate automatic defence strategy

Figure 4

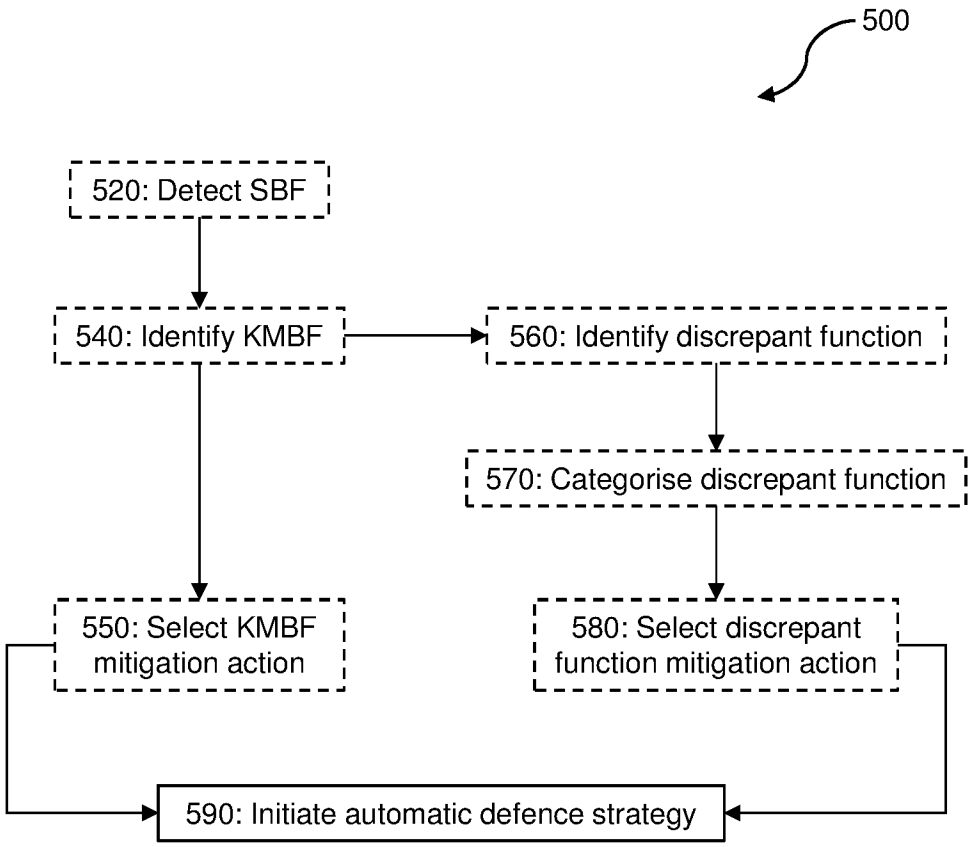
500
520: Detect SBF
540: Identify KMBF
560: Identify discrepant function
570: Categorise discrepant function
550: Select KMBF mitigation action
580: Select discrepant function mitigation action
590: Initiate automatic defence strategy

Figure 5

INTERNATIONAL SEARCH REPORT

International application No
PCT/EP2021/065635

A. CLASSIFICATION OF SUBJECT MATTER
INV. G06F21/56
ADD.

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
EPO-Internal, WPI Data

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X Y	US 2019/121978 A1 (KRAEMER JEFFREY ALBIN [US] ET AL) 25 April 2019 (2019-04-25) paragraphs [0006], [0017] - [0019], [0057] - [0061], [0153], [0174] - [0181] ----- -/--	1-3,12, 13,18-22 4-11, 14-17

[X] Further documents are listed in the continuation of Box C.

[X] See patent family annex.

* Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier application or patent but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
19 August 2021	30/08/2021

Name and mailing address of the ISA/	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Fax: (+31-70) 340-3016	Veillas, Erik

2

Form PCT/ISA/210 (second sheet) (April 2005)

C(Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	ALAM SHAHID ET AL: "Annotated Control Flow Graph for Metamorphic Malware Detection", COMPUTER JOURNAL., vol. 58, no. 10, 15 December 2014 (2014-12-15), pages 2608-2621, XP055781962, GB ISSN: 0010-4620, DOI: 10.1093/comjnl/bxu148 Retrieved from the Internet: URL:https://www.researchgate.net/profile/Shahid-Alam-3/publication/282287967_Annotated_Control_Flow_Graph_for_Metamorphic_Malware_Detection/links/562e095c08ae518e34825a05/Annotated-Control-Flow-Graph-for-Metamorphic-Malware-Detection.pdf> sections 3, 3.1, 3.2, 3.4, 3.5	4-9
Y	MOJTABA ESKANDARI ET AL: "A graph mining approach for detecting unknown malwares", JOURNAL OF VISUAL LANGUAGES & COMPUTING, ACADEMIC PRESS, UNITED KINGDOM, vol. 23, no. 3, 19 February 2012 (2012-02-19), pages 154-162, XP028486346, ISSN: 1045-926X, DOI: 10.1016/J.JVLC.2012.02.002 [retrieved on 2012-03-08] abstract; figure 1 sections 3.1 and 3.2	4-9
Y	WO 2017/003580 A1 (MCAFEE INC [US]) 5 January 2017 (2017-01-05) paragraphs [0037] - [0039], [0044] - [0051]; figures 1B,2-3, 4-5	10,11, 14-17
Y	US 9 659 176 B1 (ROTER MICHELE [US] ET AL) 23 May 2017 (2017-05-23) column 9, line 14 - column 13, line 8; claims 1-3	10,11, 14-17

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No
PCT/EP2021/065635

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 2019121978 A1	25-04-2019	US 2019121978 A1	25-04-2019
		US 2021232685 A1	29-07-2021
		WO 2019051507 A1	14-03-2019
WO 2017003580 A1	05-01-2017	CN 108064384 A	22-05-2018
		EP 3314509 A1	02-05-2018
		JP 6668390 B2	18-03-2020
		JP 2018524720 A	30-08-2018
		JP 2020113290 A	27-07-2020
		WO 2017003580 A1	05-01-2017
US 9659176 B1	23-05-2017	NONE	