Internal distribution code:

(A) [-] Publication in OJ
(B) [-] To Chairmen and Members
(C) [-] To Chairmen
(D) [X] No distribution

Datasheet for the decision of 5 November 2019

Case Number: T 2296/13 - 3.5.06

Application Number: 11193638.1

Publication Number: 2515250

IPC: G06F21/00

Language of the proceedings: EN

Title of invention:
System and method for detection of complex malware

Applicant:
Kaspersky Lab, ZAO

Headword:
Detection of complex malware/KASPERSKY

Relevant legal provisions:
EPC Art. 56

Keyword:
Inventive step - (no)

Decisions cited:

Catchword:

Europäisches Patentamt
European Patent Office
Office européen des brevets

Beschwerdekammern

Boards of Appeal

Chambres de recours

Boards of Appeal of the
European Patent Office
Richard-Reitzner-Allee 8
85540 Haar
GERMANY
Tel. +49 (0)89 2399-0
Fax +49 (0)89 2399-4465

Case Number: T 2296/13 - 3.5.06

D E C I S I O N
of Technical Board of Appeal 3.5.06
of 5 November 2019

Appellant: (Applicant)	Kaspersky Lab, ZAO 39A/3 Leningradskoe Shosse Moscow 125212 (RU)
Representative:	Sloboshanin, Sergej V. Füner, Ebbinghaus, Finck, Hano Mariahilfplatz 3 81541 München (DE)
Decision under appeal:	**Decision of the Examining Division of the European Patent Office posted on 26 July 2013 refusing European patent application No. 11193638.1 pursuant to Article 97(2) EPC.**

Composition of the Board:

Chairman	M. Müller
Members:	A. Teale
	B. Müller

Summary of Facts and Submissions

I. The appeal is against the decision, dispatched with reasons on 26 July 2013, to refuse European patent application No. 11 193 638.1 on the basis that the subject-matter of the independent claims lacked inventive step, Article 56 EPC, in view of the document

D1: US 2008/0066179 A1

and that claims 7 and 14 were unclear, Article 84 EPC.

II. The following document was mentioned in the European Search Report but not relied upon in the reasons for the decision:

D3: EP 2 219 130 A1.

III. A notice of appeal and the appeal fee were received on 19 September 2013, the appellant requesting that the decision be set aside and that a patent be granted based on the documents on file. Oral proceedings were requested as an auxiliary measure.

IV. With a statement of grounds of appeal, dated and received on 28 October 2013, the appellant filed claims according to a main and an auxiliary request. The appellant requested that a patent be granted on the basis of the claims according to said main and auxiliary requests and the description and drawings on file.

V. In an annex to a summons to oral proceedings the board set out its preliminary opinion that the claims of both requests overcame the clarity objection in the decision and were sufficiently clear for the assessment of

inventive step. The subject-matter of claims 1 and 8 of the main request appeared to lack inventive step, Article 56 EPC, in view of D1. The subject-matter of claims 1 and 8 of the auxiliary request appeared to lack inventive step, Article 56 EPC, in view of the combination of D1 and D3.

VI. With a response, received on 4 October 2019, the appellant submitted amended pages of the description, but did not amend the claims.

VII. At the oral proceedings, held on 5 November 2019, the appellant filed amended claims according to a second auxiliary request and requested that the decision under appeal be set aside and that a patent be granted in the following version:

Claims:
- No. 1 to 14 according to the main request, filed with the letter of 28 October 2013, or
- No. 1 to 14 according to the first auxiliary request, filed as auxiliary request with the letter of 28 October 2013, or
- No. 1 to 12 according to the second auxiliary request, filed during the oral proceedings of 5 November 2019.

Description (for all requests):
- Substitute pages 1 and 2, filed with the letter of 4 October 2019;
- substitute page 13, filed with the letter of 1 June 2012;
- pages 3 to 12, as originally filed.

Drawings (for all requests):
Sheets 1/5 to 5/5 with figures 1 to 4, as originally filed.

VIII. At the end of the oral proceedings the board announced its decision.

IX. Claim 1 of the main request reads as follows:

"A method for detection of computer malware, the method comprising: monitoring execution of processes or threads (201, 202) of one or more software objects (101); determining if the one or more objects (101) are trusted objects or non-trusted objects; creating for each non-trusted object a separate object context (203, 204); storing in a plurality of separate object contexts (203, 204) events of execution of the monitored processes or threads (201, 202) of each non-trusted object; determining if the processes or threads (201, 202), whose events of execution are stored in separate object contexts (203, 204), are related to each other, wherein processes are related when a process initiates another process, a process creates a new object from which a new process is initiated, or a process embeds a thread in another process; creating for two or more related processes or threads (201, 202) a new separate common context (205); merging events stored in the separate object contexts (203, 204) of the two or more related processes or threads (201, 202) into the separate common context (205); independently analyzing events of the processes or threads (201, 202) stored in each of the plurality of separate object contexts (203, 204) using malware behavior rules (107) to identify one or more malicious software objects having malicious behavior patterns; and independently analyzing merged events of the related processes or

threads (201, 202) stored in the separate common context (205) using malware behavior rules (107) to identify one or more malicious software objects having complex malicious behavior patterns."

X. Claim 1 of the first auxiliary request differs from that of the main request in the addition of the following passages:

"when a monitored process or thread terminates, discontinuing monitoring said process or thread and deleting a separate object context (203, 204) associated with said process or thread;" and

[independently analyzing merged events] "involving a plurality of related processes or threads even when one of the plurality of related processes or threads has terminated and a separate object context of said terminated process or thread was deleted".

XI. Claim 1 of the second auxiliary request differs from that of the first auxiliary request in the addition, after the expression "non-trusted objects" (line 3), of the following expression:

", by computing digital signatures of the objects and checking if the digital signatures are associated with a trusted object; and discontinuing monitoring of one or more processes or threads of a trusted object".

Reasons for the Decision

1. The admissibility of the appeal

The appeal fulfills the admissibility requirements under the EPC.

2. A summary of the invention

2.1 The invention relates to protecting computers from complex malware, also termed "multi-component" malware, consisting of several components performing actions which, taken individually, appear innocuous, but which, in concert with others, damage the computer. For instance, a first component may perform actions with files, a second component may modify the system registry and a third component may perform networking functions; see page 1, lines 21 to 27.

2.2 When an object (101), for instance an executable file, is launched on the protected computer, it creates a process; see figure 1 and sentence bridging pages 3 and 4. The process has a virtual address space accessed by a plurality of threads, and each process may be associated with one or more objects. If a process or thread performs a malicious action, then the object that launched that process or thread is also treated as malicious; see page 4, lines 6 to 8.

2.3 The invention monitors the execution of the processes or threads of an object by storing execution events in a separate "object context". Malware behaviour rules are used to separately analyse the execution events in each object context to identify malicious objects, for instance identifying objects which attempt to modify the system registry or to create or modify executable files; see page 5, lines 16 to 19.

2.4 The system also determines whether objects are related to each other, for instance by detecting whether a parent-child relationship exists between two objects or their processes; see page 6, lines 1 to 8. If so, their

contexts are merged into a "common context"; see figure 2 and figure 3A; step 315, and the paragraph bridging pages 5 and 6. From then on, events for each object context are also added to the common context. The system analyses events stored in the common context using malware behaviour rules to identify malicious objects with complex behaviour; see page 2, lines 2 to 16, and the paragraph bridging pages 6 and 7.

2.5 The features added to claim 1 according to the first auxiliary request are based on page 7, line 30, to page 8, line 7, and concern identifying malicious objects by means of a common context even after the individual processes have terminated and the separate object contexts have been deleted.

2.6 The features added to claim 1 in the second auxiliary request, disclosed on page 4, lines 14 to 21, concern ceasing to monitor an object if it is identified as trustworthy by computing a digital signature for the object and checking whether this signature is in a list of signatures of trusted objects; see figure 1; 109 and figure 3; 310. An object is no longer monitored if it is listed as trustworthy.

3. Clarity, Article 84 EPC

3.1 According to the appealed decision, then claims 7 and 14 were inconsistent with claims 1 and 8, respectively, in making continued monitoring of trusted objects conditional. This objection has been overcome by changing the dependencies of claims 7 and 14 to 3 and 10, respectively, in the main and first auxiliary requests, and by making claims 6 and 12 dependent on claims 1 and 7, respectively, in the second auxiliary request.

3.2 Claim 1 of all three requests is therefore sufficiently clear for the purposes of assessing inventive step. Whether all requirements of Article 84 EPC are fulfilled need not be decided.

4. The prior art on file

4.1 Document D1

4.1.1 The appealed decision assessed inventive step starting from D1, which relates to an antivirus protection system for computers; see title and figure 1. A "Process Behavior-Evaluating Unit" identifies the programs in the computer and classifies them as "normal" or "suspect". "Normal" programs are, for instance, those known to the "Program-Behavior Knowledge Base", whilst "suspect" programs include those of unknown origin; see [9, 89].

4.1.2 A program monitoring unit monitors and records the behaviour of programs. The actions which are monitored are those which may affect computer safety, namely file operations, network operations, creation of processes or threads, registry operations and injecting threads; see [13]. "Dangerous actions" are those monitored actions which may threaten a computer's safety and are rarely executed by normal programs but often by viruses or Trojans, for instance modifying a program file, promoting itself to a higher running layer (e.g. application to system) or calling the shell program; see [15, 95].

4.1.3 A correlation analysing unit creates a "correlative tree", comprising a "process tree" (see figure 2) and a "file tree" (see figure 3), and performs a correlation

analysis of the behaviour of the programs using the correlative tree. Each node of the process tree stores the actions/behaviour of a running process, the parent node of each node corresponding to its parent process; see [19-20, 23 and 98]. Each node of the tree file represents a program, the parent node of each node corresponding to its creator.

4.1.4 The system also comprises a virus-identifying knowledge base (see [10, 109, 111, 115]) comprising a program-behaviour knowledge base and a database of attack-identifying rules; see [27-49, 116-117 and 131-151]. The knowledge base contains a list of programs known to be "normal"; see [10].

4.1.5 A virus-identifying unit compares captured program behaviour with information in the virus-identifying knowledge base to determine whether a program is a virus program; see [152].

4.1.6 As shown in figure 5, the "dangerous" actions executed by a known program are compared to legal actions or behaviours in the program behaviour knowledge base to determine whether the program has been attacked. If so, the program is terminated; see [61]. As shown in figure 6, the "dangerous" actions executed by an unknown program are compared to rules stored in the database of attack-identifying rules in the virus-identifying knowledge base to determine whether the program is harmful. If so, the program is terminated; see [62-63].

4.1.7 Hence, in the terms of claim 1 of the main request, D1 discloses a method for detection of computer malware, the method comprising: monitoring execution of processes or threads of one or more software objects;

determining if the one or more objects are trusted objects ("normal"/"known") or non-trusted objects ("suspect"/"unknown"); creating for each non-trusted object (node in process tree) a separate object context; storing in a plurality of separate object contexts events of execution of the monitored processes or threads of each non-trusted object (see [98]); determining if the processes or threads, whose events of execution are stored in separate object contexts, are related to each other, wherein processes are related when a process initiates another process (implicit in the structure of the process tree in figure 2), a process creates a new object from which a new process is initiated, or a process embeds a thread in another process and independently analyzing events of the processes or threads (virus-identifying unit) stored in each of the plurality of separate object contexts using malware behavior rules (attack-identifying rules; [116]) to identify one or more malicious software objects having malicious behavior patterns.

4.2 Document D3

D3 discusses malware detection based on correlating the behaviour of several processes. Paragraph [8] mentions a complex malware attack ("Backdoor.GPigeon") involving three processes, the first (A.exe) having terminated before the "infringing" action occurs. Processes are linked in a "process set" by the fact that one launches another; see [43]. The process set allows events to be more effectively correlated to detect malicious behaviour; see [81].

5. Inventive step, Article 56 EPC

5.1 The main request

5.1.1 Present claim 1 is the same as that in the decision, according to which the subject-matter of claim 1 differed from the disclosure of D1 in that:

1. [the method] stored in separate object contexts events of execution of monitored processes or threads only for non-trusted objects and

2. the common context was a new context separated from the object contexts of the two or more related processes or threads.

Since the two difference features were unrelated and lacked any synergistic effect, the examining division considered their contributions to inventive step separately. The first difference was obvious because D1 hinted at only recording execution events for the non-trusted objects; see [9-10]. D1 stated that suspect and/or unknown programs potentially required monitoring, thus prompting a skilled person to avoid storing execution events of trusted programs. As to the second difference, the creation of a common context as a separate entity from the object contexts was an obvious design choice having the same technical effect as the method of D1, namely analysing the correlation of programs and/or the behaviours of programs.

5.1.2 In contrast to the appealed decision, the board does not construe claim 1 as excluding objects being created for trusted objects. On the contrary, claim 1 merely requires that objects be created for non-trusted objects. Hence the board does not agree with the

appealed decision (see point 3.2) that difference "1" exists. The appellant argued (see response of 4 October 2019, point 3) that claim 1 only positively recited the creation of separate object contexts for non-trusted objects and, construing claim 1 in the context of the underlying application, in particular figure 3A (see decision 310) and page 10, lines 7 to 10, in which a context was only generated (step 313) if an object was not trusted; see also page 5, line 26, to page 6, line 15. The board disagrees that claim 1 must be construed narrowly according to the cited embodiments, since claim 1 as originally filed was also broader than that, stating that execution events of monitored processes or threads "of each non-trusted object" are stored in a plurality of separate object contexts.

5.1.3 A claim which leaves out an option cannot, as a rule, be construed as requiring that the feature **not** be present, even if the feature was absent from the main (or only) embodiment, unless the absence of that feature is directly and unambiguously derivable from the original application.

5.1.4 In the present case, the board considers that it does not follow from the claim context that trusted objects not be monitored. Moreover, monitoring trusted objects is not unreasonable in the present case. Even though **separate** monitoring of a trusted object may appear redundant, it would be entirely plausible to analyse a trusted object in the context of related processes because even a trusted object could, in principle, form part of complex malware according to the definition in the application (see page 1, lines 24-27).

5.1.5 It is common ground between the board and the appellant that the subject-matter of claim 1 also differs from the disclosure of D1 in that:

3. events associated with related processes that are stored in the object context and in the common context as merged events are each analysed independently from each other.

5.1.6 According to the appellant, features "2" and "3" had the synergistic effect that complex malicious behaviour involving a plurality of related processes could still be detected, even if some particular processes of the plurality of related processes had "died", understood by the board to mean that the relevant separate object contexts had been deleted, since the event information associated with the related processes persisted in the common context; see original description, page 8, lines 1 to 7. As no evidence has been produced for such a synergistic effect, and the board can see no reason why such an effect would occur, the board finds that no such synergistic effect exists.

5.1.7 It is common ground between the board and the appellant that the subject-matter of claim 1 also differs from the disclosure of D1 in the steps of:

a. creating for two or more related processes or threads a new separate common context and merging events stored in the separate object contexts of the two or more related processes or threads into the separate common context and

b. independently analyzing merged events of the related processes or threads stored in the separate common context using malware behavior

rules to identify one or more malicious software objects having complex malicious behavior patterns.

It is also common ground that features "a" and "b" are related to each other; the separate common context of "b" is defined in "a", making "a" a necessary precursor to "b".

In the oral proceedings it was common ground between the board and the appellant that the objective technical problem starting from D1 was to extend the malware detection system known from D1 to also cover complex malware, meaning multi-component malware that could not be detected by signature recognition.

5.1.8 The appellant has disputed the conclusion in the decision (see section 3.4) that the creation of a common context as a separate entity from the object context (difference feature "a") was an obvious measure to achieve the technical effect known from Dl, namely analysing the correlation of programs and/or the behaviour of programs. Firstly, the invention used a different data structure for organizing process events to the process tree structure used in D1; see [17-21]. Secondly, by storing events associated with related processes in two different types of context, the invention could perform a complex malware detection using the remaining running processes, even after the events associated with terminated processes had been deleted, since the events associated with the terminated processes persisted in the common context.

5.1.9 Regarding the differences in data structure between the invention and D1, the board notes that, other than specifying that events are stored in object contexts

and common contexts, claim 1 does not limit the data structure used to store events. Hence the object contexts set out in claim 1 cover the nodes of the process tree known from D1, since a node could be considered as an object context comprising references to "parents" and "children" in the tree structure. The board appreciates that the process tree of D1 is not suited to storing sets of object contexts - i.e. nodes - in separate data structures. If such sets were to correspond to paths or sub-trees in the process tree, then creating a separate data structure for the set would result in objects lacking a natural "place" in the process tree. A data structure combining nodes spread over the tree would likewise not have a natural place in the tree structure. However, the process tree of D1 does not discourage the skilled person from creating common object contexts as data structures separate from the process tree, a matter of usual design for the skilled person.

5.1.10 On the question of events associated with a terminated process persisting in the common context and being used to detect complex malware, the board notes that the appellant's argument is based on the paragraph bridging pages 7 and 8 of the description, referring to a complex malware attack starting with a first process embedding a thread in an existing process and then terminating. As claim 1 is not limited to object contexts being deleted, the board is not persuaded that this advantage always occurs.

5.1.11 D3 shows that it was known at the priority date to detect complex (multi-component) malware threats by monitoring the behaviour of groups of objects which, although individually innocuous, could in concert pose a complex malware threat.

5.1.12 In the oral proceedings the board put it to the appellant that the skilled person solving the above problem could form common contexts to collect information on possible complex malware by either referring to the nodes in the correlative tree known from D1 (see [17 to 21]) for storing information on single software objects or by copying information from said nodes into a newly created common object. As the latter solution, set out in difference feature "a", reduced the time required to access the common object data it would have been a usual matter of design for the skilled person. The analysis of a common object to detect complex malware and thus solve the objective technical problem, set out in difference "b", followed from feature "a" as a matter of usual design for the skilled person.

5.1.13 The appellant argued that storing information on events firstly in separate object contexts and secondly in the common contexts had the advantage of added flexibility. The board finds that this flexibility is not a surprising effect and that it results from one of the two possible ways of realising common object contexts starting from D1.

5.1.14 Hence the board concludes that the subject-matter of claim 1 of the main request does not involve an inventive step, Article 56 EPC, starting from D1.

5.2 The first auxiliary request

5.2.1 Compared to claim 1 of the main request, claim 1 sets out two additional features, which are not known from D1, namely:

c. "when a monitored process or thread terminates, discontinuing monitoring said process or thread and deleting a separate object context (203, 204) associated with said process or thread" and

d. [independently analyzing merged events] "involving a plurality of related processes or threads even when one of the plurality of related processes or threads has terminated and a separate object context of said terminated process or thread was deleted."

These amendments focus the claims on the embodiment disclosed in the paragraph bridging pages 7 and 8 referring to a complex malware attack starting with a first process embedding a thread in an existing process and then terminating.

5.2.2 The board finds that feature "c" relates to prudent memory management (otherwise termed "garbage collection"), a usual matter of design for the skilled person, in particular in the claimed situation in which the content of the separate context was copied to the common context and thus would not be lost for the analysis of complex malware.

5.2.3 Turning to feature "d", the skilled person starting from D1 and addressing the objective technical problem, set out above, would have added feature "d" without inventive step as part of detecting complex malware threats.

5.2.4 Consequently the additional features are unable to lend inventive step, Article 56 EPC, to claim 1.

5.3 The second auxiliary request

5.3.1 The features added to claim 1 with respect to the previous request, namely

[determining if objects are trusted or non-trusted] "by computing digital signatures of the objects and checking if the digital signatures are associated with a trusted object; and discontinuing monitoring of one or more processes or threads of a trusted object"

are based on page 9, lines 22 to 23, and page 10, second paragraph.

5.3.2 The appellant has argued that the added features reduce the amount of data which is stored, increasing efficiency and producing a synergistic effect in combination with the other difference features.

5.3.3 The board notes that D1 discloses a "Virus-Identifying Knowledge Base"; see [10] and [108-9]. The skilled person would understand "Identification" in this context as "signature recognition" and "Virus-identification" to mean that an object is non-trusted. The board is also not persuaded that a synergistic effect accrues regarding memory management, there being no disclosure to this effect in the original application. Moreover, even on the assumption that memory is saved by not monitoring trusted objects, separately or in combination with other processes, this comes at the price of not being able to detect complex malware comprising a trusted component. In general, this appears to imply a loss of accuracy of complex malware detection. The significance of discontinuing monitoring of trusted object cannot in general be judged. Therefore, the board concludes that

discontinuing monitoring of processes or threads of a trusted object is a trade-off between detection quality and the efficient use of system resources, a usual design choice for the skilled person.

5.3.4 Hence the board finds that the added features are unable to lend inventive step to claim 1, Article 56 EPC.

Order

For these reasons it is decided that:

The appeal is dismissed.

The Registrar: The Chairman:

I. Aperribay M. Müller

Decision electronically authenticated