

CLAIMS

1. A computer-implementable method for detecting malicious files in non-isolated environment, the method comprises:

a preparatory step, wherein including:

- ~~—~~creating a collection of files, the collection of files comprises at least one malicious executable file and at least one non-malicious executable file;
- analyzing at least one executable file from the collection of files, including wherein:
 - extracting data from both a binary form and/or a disassembled forms of the at least one executable file,
 - determining parameters based on a the extracted data—extracted—creating parameters, for further classifier training, wherein
 - determining parameters specific to malicious files and/or vice versa to non-malicious files by a statistical method; and
 - ~~wherein additionally~~ extracting the a first data flow graph and the a second data flow graph;
- constructing ~~the a~~ first feature vector and ~~the a~~ second feature vector based on the ~~created~~ parameters;
- ~~creating training~~ an ensemble of classifiers including from: the a first trained classifier based trained on the basis of the first feature vector, ~~at the~~ second trained classifier trained based on the basis of the second feature vector; ~~the a~~ third classifier trained on the basis of the first ~~control data~~ flow graph, ~~the a~~ fourth classifier trained on the basis of the second ~~control data~~ flow graph; and
- ~~wherein~~ predefining a decisive priority for each of the ensemble of classifiers; and

an operating step, wherein including:

- obtaining at least one executable file;
- detecting whether the at least one executable file is a malicious executable file using starting the ensemble of classifiers trained at the preparatory step ~~for detection of malicious executable files; and~~
- outputting the ~~analysis~~ result of the analyzing at least one executable file.

2. The method of claim 1, wherein the first feature vector comprises all numerical parameters ~~obtained at the analysis~~ from the parameters, and the second feature vector comprises all string parameters from the parameters ~~obtained at the analysis~~.

3. The method of claim 1, ~~wherein further comprising~~ extracting at least the following file properties/features from the at least one executable file:

the first data flow ~~control~~ graph,

- the second ~~data control~~-flow graph,
 Metadata,
~~an~~ Import Table,
~~an~~ Export Table, ~~and~~
~~a~~ Byte/Entropy Histogram.
4. The method of claim 1, wherein ~~extracting at least the following parameters from at least one~~for
~~the~~ binary ~~form of the at least one~~ executable file ~~the parameters include~~:
- ~~a~~ Byte n-grams,
 - information from the executable file fields,
 - ~~a~~ file section entropy,
 - metadata, ~~and~~
 - line length distribution histograms.
5. The method of claim ~~34~~, wherein the information ~~extracted~~ from the executable file fields ~~of at~~
~~least one binary executable file comprises at least~~includes:
- a number of sections in the file,
 - a size of headers, a code size,
 - an availability of digital signature,
 - an availability of imported dll and functions where these libraries have been used,
 - names of exported functions,
 - ~~a~~ data from a resource catalog, ~~and~~
 - an assembler version.
6. The method of claim ~~34~~, wherein the file section entropy ~~extracted from the fields of at least one~~
~~binary executable file~~ is calculated using a sliding-window method followed by a number of statistical
 measures to characterize the obtained entropy measure vector.
7. The method of claim ~~34~~, wherein the metadata is extracted from ~~the a~~ header of ~~at least one~~the
 binary form of the at least one executable file, wherein the metadata comprises includes:
- a number of sections in the file,
 - a size of sections,
 - an availability of a checksum,
 - a file size,
 - a creation time,
 - an operating system version, ~~and~~
 - an assembler version.
8. The method of claim 1, wherein ~~extracting at least the following parameters from~~for ~~the~~ at least
~~one~~ disassembled form of the at least one executable file the parameters include:

characters,
registers,
operation codes,
a system interface (Windows APIs) access rate,
sections, ~~and~~
metadata.

9. The method of claim 78, ~~further comprising wherein additionally~~ defining a frequency of occurrence of the characters extracted from ~~at least one the~~ disassembled form of the at least one executable file, wherein the characters ~~could be at least~~ include: “-“, “+“, “*“, ”]“, ”[“, ”?“, ”@”.
10. The method of claim 78, ~~wherein further comprising~~ additionally calculating a ratio between a number of known registers extracted from ~~at least one the~~ disassembled form of the at least one executable file and a number of unknown registers extracted from ~~at least one the~~ disassembled form of the at least one executable file.
11. The method of claim 78, wherein the sections extracted from at least one disassembled executable file are predefined and the sections ~~are~~ include: .text, .data, .bss, .rdata, .edata, .idata, .rsrc, .tls and .reloc.
12. The method of claim 78, wherein the metadata extracted from ~~at least one the~~ disassembled form of the at least one executable file ~~is~~ includes the file size and a number of code lines.
13. The method of claim 1, wherein the analysis result ~~could be~~ is binary or probabilistic.
14. The method of claim 1, wherein the decisive priority of each of the ensemble of classifiers is defined based on a test sample, wherein at least the following is used for estimation: accuracy, completeness, ~~or~~ F-measure.
15. A system for detecting malicious files in a non-isolated environment, the system comprises:
a data extraction module configured to implement the preparatory step in the method of any one of claims 1-14 ~~to extract data from at least one executable file~~;
a computer device configured to implement the operating step in the method ~~for detecting malicious files in non-isolated environment according to~~ of any one of claims 1-14; ~~and~~
an output module configured to output the result of the analyzing at least one executable file in the method of any one of claims 1-14.