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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
02 January 2020 (02.01.2020)

WIPO | PCT

(10) International Publication Number
WO 2020/002203 A1

(51) International Patent Classification:
G06F 21/55 (2013.01) *G06F 21/56* (2013.01)

(21) International Application Number:
PCT/EP2019/066621

(22) International Filing Date:
24 June 2019 (24.06.2019)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
LU100844 25 June 2018 (25.06.2018) LU

(71) Applicant: **UNIVERSITÉ DU LUXEMBOURG** [LU/LU]; 2, avenue de l'Université, 4365 Esch-sur-Alzette (LU).

(72) Inventors: **GENÇ, Ziya, Alper**; 10A/426, avenue du Swing, 4367 Belvaux (LU). **LENZINI, Gabriele**; 50, route d'Abweiler, 3211 Bettembourg (LU). **RYAN, Peter, Yvain, Anthony**; 55, rue des Maraichers, 2124 Kirchberg-Luxembourg (LU).

(74) Agent: **WAGNER, Jean-Paul**; Corax IP 2, rue Wilson, 2732 Luxembourg (LU).

(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, 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, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, 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:** METHOD FOR PREVENTING RANSOMWARE ATTACKS ON COMPUTING SYSTEMS

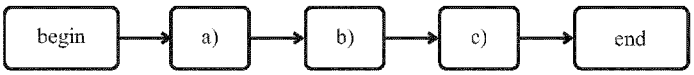

Fig. 1

(57) **Abstract:** The proposed invention concerns a method for preventing ransomware attacks on a computing system. By controlling the access to a calling interface through which cryptographic functions, such as the random number generator, can be accessed to generate strong encryption keys the method allows to efficiently terminate cryptographic ransomware attacks on the system before they can start doing any damage. If the access to the cryptographic functions, such as the random number generator, is not granted, the ransomware is unable to build a strong encryption key, and it is unable to deploy its intended effect.

WO 2020/002203 A1

METHOD FOR PREVENTING RANSOMWARE ATTACKS ON COMPUTING SYSTEMS

Technical field

The invention lies in the field of computer science and aims at enhancing the security of computing systems such as computing devices. In particular, the invention relates to the prevention of attacks on such systems that are carried out by means of malicious pieces software, known as malware, that make use of cryptographic resources to encrypt vital data stored on said computing systems, thereby rendering the affected computing systems unusable.

Background of the invention

Computing systems regularly come under attack of malicious software, which, when executed by a computing system, deteriorates or even entirely prohibits the functioning of the system. It is known that malicious software, also known as computer viruses or malware, spread among computing systems via data communication networks which connect the systems to one another, such as for example via the public Internet. Other sources of infection include storage media which comprise the malware. The malware is generally designed so that it stealthily executes on the computing system and performs actions that are not allowed and not supervised by the computing system's administrator or user.

It is known to use firewall devices and firewall software for preventing malware to gain access to the computing system via a networking interface thereof. By monitoring the data traffic on a networking interface, suspicious behaviour previously observed in the context of a malware attack, may therefore be identified and blocked from doing further damage.

Cryptographic ransomware as emerged as a particular efficient type of malware, in that it is hard to detect and to stop, once a computing system has been infected. Cryptographic ransomware encrypts files and data on the infected computing system's filesystem, so that they are not usable anymore. The files can however be recoverable at the condition to know a secret, the decryption key. The authors of the malware often offer a "recovery service" to victims against payment of a substantial monetary ransom. To make sure that the encrypted filed cannot be decrypted without paying the ransom, cryptographic ransomware use strong encryption, including:

- robust and known encryption algorithms, such as the Advanced Encryption Standard, AES, standardized by the National Institute of Standards and Technology of the United States of America, NIST;
- strong encryption keys, featuring an important length and a high degree of randomness.

A strong encryption key makes it practically impossible to decrypt the encrypted files. Established theoretical results proves that guessing or determining the key without any prior knowledge nowadays would take for any computing systems, present and future, an enormous amount of time far beyond one's lifetime

Ransomware are particularly dangerous both technically and psychologically. They attack large enterprises and individuals alike, thereby disrupting import and services and workflows, while eroding people's trust and sense of security in the communication technologies. Indeed, some known malware threaten to increase the ransom fee or to irrevocable destroy the file if the payment is not done, and some do not decrypt the files even though the ransom has been paid up. An economical plague ensues: the total costs due to cryptographic ransomware attacks has by some estimates exceed worth of 5 billion US Dollars in the year 2017, and it has been predicted to cost up to 6 trillion US Dollars by 2021.

Known solutions to the threat of cryptographic ransomware rely on the following three principles, each of which presents distinct disadvantages, and none of which has been capable of reliable preventing all known ransomware attacks.

1. The first known principle relies on keeping back-ups of files and data on a secure storage, so that after a ransomware attack, it is possible to restore a non-encrypted version of the affected files.

   However, most users and some administrators do not keep back-ups or do not check the sanity of their backup copies regularly, so that this solution is both expensive, as large amounts of storage are required, and ineffective. According to recent surveys, only about 42% of affected victims were able to recover all their data after a ransomware attack.

2. The second known principle relies on monitoring activities on the computing system, in search for suspicious ones. Once suspicious activities are detected, the corresponding processes are isolated or stopped to prevent further harm.

   Monitoring for suspicious activities is a resource expensive task and intense run-time monitoring consumes considerable computational resources. This may slow down a computer's performance for day-to-day working tasks. Moreover, since the models of ransomware on which such solutions rely are not perfect, they need to be regularly refined and updated in order to adapt to the evolution of the ransomware, which may change their behavioural patterns in time.

3. The third known principle relies on logging any encryption keys used in the computing systems. After an attack, the corresponding logs are searched in order to find the encryption key used by the ransomware, so that the affected files may be decrypted.

The key-logging approach has several drawbacks:

- it undermines the security of the system of ephemeral cryptographic keys that are used by hones protocols, e.g., Transport Layer Security, TLS, or Secure Shell, SSH. Their keys, once logged, may be stolen by malware from the log;
- it is an inefficient strategy. Finding the keys used by ransomware among the heap of keys stored is done by trial-and-error, which requires a large amount of time. Honest software, such as remote cloud-based storage, may indeed use thousands of encryption keys on a single device. To that, one has to add the time needed to decrypt the files once the appropriate key has been determined. This time may amount to several hours on a personal computer relying on average available computing resources;
- the approach may still fail is the ransomware uses third-party libraries or functions that escape the logging strategy;
- some known ransomware reboots the computing system as soon as the encryption key has been constructed. Upon reboot of the computing system, the file encryption process is immediately launched by the ransomware, so that the access to the key log is effectively prohibited.

Technical problem to be solved

It is an objective of the invention to present a method, which overcomes at least some of the disadvantages of the prior art. In particular, it is an objective of the invention to present a method for improved prevention of cryptographic ransomware to encrypt files on computing systems.

Summary of the invention

According to a first aspect of the invention, a method for preventing ransomware attacks on a computing system is provided. The computing system has access to computing resources including cryptographic resources, which comprise random number or pseudo-random number generation means, and which are accessible through calling interface means. The method comprises the following steps:

a) providing predetermined requirements describing a permissible call through said calling interface means in a memory element;

b) using monitoring means, monitoring the use of said calling interface means, comprising the obtention of data identifying a call;

c) blocking said call to said cryptographic resources, unless the data identifying said call complies with said predetermined requirements.

According to an aspect of the invention, a method for preventing ransomware attacks on a computing system is provided. The computing system has access to computing resources including a source of randomness , which comprises random number or pseudo-random number generation means, and which are accessible through calling interface means, The method is remarkable in that it comprises the following steps:

a) providing predetermined requirements describing a permissible call to said source of randomness comprising random number or pseudo-random number generation means through said calling interface means in a memory element;

b) using monitoring means, monitoring the use of said calling interface means, comprising the obtention of data identifying a call;

c) blocking said call to said source of randomness, unless the data identifying said call complies with said predetermined requirements.

Preferably, the data identifying said call may comprise data identifying the calling process. The data identifying the calling process may preferably include a process ID that identifies the process within the computing system's operating system. The data identifying the calling process may preferably comprise a higher-level description of the software program executed through said process.

The predetermined requirements may preferably comprise data identifying at least one calling process that is allowed to access said cryptographic means or source of randomness. The call may preferably be blocked unless the data identifying the calling process is comprised in the data identifying at least one allowed calling process.

The cryptographic means or source of randomness may preferably comprise a cryptographically secure random number generator.

Preferably, the calling interface means comprise an application programming interface providing access to said cryptographic means or source of randomness.

Blocking the call may preferably comprise terminating the corresponding calling process.

Preferably, the monitoring means and/or calling interface means may be part of the computing system's operating system. As such they may preferably be implemented by a computer program stored in a memory element to which a processing unit of the computing system has read/write access. The computer program may comprise instructions that are executable by the processing unit and which, when executed, provide the monitoring/interface function as described.

The cryptographic resources or source of randomness may further comprise a source of entropy.

According to a further aspect of the invention, a computer program is provided. The computer program comprises computer readable code means, which when run on a computer, cause the computer to carry out the method according to aspects of the invention.

According to another aspect of the invention a computer program product is provided. The computer program product comprises a computer-readable medium on which the computer program according to aspects of the invention is stored.

According to a final aspect of the invention, a computer configured for carrying out the method according to aspects of the invention is provided. The computer or computing device comprises a processing unit and at least one memory element to which the processing unit/processor has read/write access, for example through a data bus connection. The memory element stores computer readable instructions which, when read and executed by the processing unit, causes the computer to run the method in accordance with aspects of the invention. The computer further comprises cryptographic resources or a source of randomness, which are accessible through a calling interface, and which is preferably provided by the operating system of the computer, which is executed by said processing unit.

By using the method as proposed by aspects of the invention, it becomes possible to stop an attack on a computing system by cryptographic ransomware before the ransomware starts to compromise and encrypt any files. The method also stops ransomware programs that, before encrypting files, reboot the infected system: in this case the ransomware are stopped before they cause a reboot. The method may work as a process in the computing system's Operating System, it may be an independent software solution, or integrated with known anti-malware software solutions. In contrast to known solutions to ransomware threats, the proposed strategy is not based on back-ups, nor on observing, analysing or recognising a complex evolving behaviour within the computing system. Nor does the proposed strategy store information for a later recovery/decryption attempt of already encrypted files. Rather, the proposed strategy relies on intercepting requests addressed to cryptographic resources that are required to build a strong encryption key, as used among others by

cryptographic ransomware. The strategy checks whether a requester is legitimate to access the cryptographic resources and terminates any illegitimate requesters. By adopting this approach, the proposed strategy allows to stop an attack before the encryption key is produced, so that it obliviates any drawbacks caused by key-logging approaches. Importantly, no sensitive information about honest processes needs to be stored, which may compromise their security. The proposed strategy only requires little CPU power and only requires a small memory footprint as it does not rely on large amounts of storage, as used for example in back-up-based strategies. Furthermore, by only blocking requesters that have not been explicitly cleared, honest protocols and software may still use the cryptographic resources or sources of randomness of the computing system unhindered. Compared to solutions which monitor application behaviour and which require updates on the heuristic methods whenever new ransomware is discovered, the proposed strategy is stable in time, as the conditions that determine allowable requesters may be updated easily. A Microsoft Windows 7™ based implementation of the method according to aspects of the present invention using a simple white list as a set of predetermined conditions for authorizing access to the computing system's cryptographic resources (i.e., a cryptographically secure pseudo-random number generator) has been tested on several computing systems against 307 real ransomware. The method was successful in blocking 94% of the attempted attacks, including new generation ransomware like the NotPetya malware, which currently overcomes all known anti-ransomware solutions.

Brief description of the drawings

Several embodiments of the present invention are illustrated by way of figures, which do not limit the scope of the invention, wherein:

- figure 1 illustrates the main steps of a method in accordance with a preferred embodiment of the invention;
- figure 2 provides a workflow diagram illustrating a method in accordance with a preferred embodiment of the invention;
- figure 3 is a schematic illustration of a device in accordance with a preferred embodiment of the invention.

Detailed description of the invention

This section describes the invention in further detail based on preferred embodiments and on the figures.

It should be noted that features described for a specific embodiment described herein may be combined with the features of other embodiments unless the contrary is explicitly mentioned. Features commonly known in the art will not be explicitly mentioned for the sake of focusing on the features that are specific to the invention. For example, a computing system is evidently

powered by an electric supply, even though such supply is not explicitly referenced on the figures nor referred to in the description. Similarly, an Operating System provides a plethora of functions and provides APIs to numerous components of a computing system (networking interface, display, data bus, etc...), even though these aspects are not explicitly referred to in the description.

Cryptographic ransomware needs strong encryption keys in order to implement strong encryption. In order to generate such keys, the cryptographic ransomware needs access to random numbers. The safest way for a cryptographic ransomware to get random numbers is to ask for them through the Operating System of the computing system it has infected.

As randomness is also necessary and required for other security tasks, such as secure data communications (e.g., in electronic banking), all modern computer Operating systems have dedicated high entropy functions for this purpose, which are known as cryptographically secure pseudo-random number generators. The mathematics and algorithms on which such pseudo-random number generators rely have been extensively studied and published, and they are well known in the art. Their details will therefore not be explained in the context of the present invention. Operating Systems also provide other lower entropy sources that ransomware may use to extract entropy in order to generate encryption keys. These include hard disk sensors, Central Processing Unit, CPU, sensors, voltage sensors, keyboard, mouse/pointer devices and system clocks. But relying on these functions is likely to lead to weaker encryption keys. All sources of randomness available to the computing system are referred to as cryptographic resources.

In accordance with embodiments of the invention, an access control mechanism is provided. The access control mechanism enforces access requests to the cryptographic resources and blocks or terminates unauthorized attempts. The cryptographic resources, including cryptographically secure pseud-random number generators, are considered to be security-critical resources. On the Microsoft Windows™ Operating System for example, the Application Programming Interface, API, calls that are monitored by the access control mechanism in accordance with aspects of the invention, includes the CryptGenRandom, BCryptGenRandom or RtlGenRandom calls. Further, a certification mechanism is used to certify applications, and to authorize them to access the cryptographic resources subject to certain conditions.

When any process running on the computing device requests access to the cryptographic resources, the proposed method intercepts the process's identity, PID, and checks whether the process is authorized to access the cryptographic resources in the current setting. If the check fails, the process is terminated. A ransomware will fail the check and the Operating System will therefore not provide the ransomware with the random number it requires to build an encryption key.

Figure 1 illustrates the main steps of a method in accordance with a preferred embodiment of the invention. The method aims at preventing ransomware attacks on a computing system or, equivalently, on a computer. The computer runs an Operating System which allows the execution of software programs through the use of a software process. The execution of a software program is enabled by reading corresponding software instructions from a memory element of the computer, and executing these instructions by means of a processor, which orchestrates the execution using other components of the computer's hardware and/or Operating System. The computer has access to computing resources including cryptographic resources or a source of randomness, which comprise a random number or pseudo-random number generator. The cryptographic resources/source of randomness may be physically remote from the computer and accessible through a secured data communication channel. Advantageously however, the cryptographic resources are physically collocated with the computer. The cryptographic resources comprise for example a cryptographic chip which provides a data processor that implements cryptographic routines including generating cryptographic keys and/or encrypting and decrypting data. The cryptographic resources/source of randomness are accessible through a calling interface, such as an API function call, which is provided by means of a function embedded in the computer's Operating System. As illustrated, the method comprising the following steps:

a) providing predetermined requirements describing a permissible call to the source of randomness or entropy through said calling interface means in a memory element. The predetermined requirements comprise for example a list identifying certified and authorized software processes.

b) using monitoring means, monitoring the use of said calling interface means, comprising the obtention of data identifying a call. The monitoring means provide an implementation of the access control mechanism outlined here above.

c) blocking said call to said cryptographic random number or pseudo-random number generation means, unless the data identifying said call complies with said predetermined requirements. If the calling process has not been previously certified or authorized for accessing the corresponding cryptographic resources, it will not be on the list, and it is terminated.

Figure 2 provides a workflow view of the method in accordance with a preferred embodiment of the invention. Two processes A and B are executed in the Operating System of a computing device and each issue a request 115A, 115B to access the computing device's cryptographic resources 110. The cryptographic resources 110 can only be accessed through a call to the corresponding API 120. This call is intercepted by the monitoring means 140. The APIs can be intercepted/blocked and their parameters as well as their return values can be intercepted and/or

modified easily. The corresponding actions are provided for by all modern Operating Systems. For instance, in Microsoft Windows™ systems, intercepting the APIs can be achieved through several techniques including Dynamic Link Library, DLL, injection, Import Address Table, IAT, hooking, or kernel mode hooking. The method may also be implemented in the Operating System's kernel, so that the API could be bypassed entirely. In accordance with a preferred but non-limiting embodiment of the invention, a DLL injection technique is used, which uses the Detours library of Microsoft™ Research for that purpose. Intercepting a call to the cryptographic resources is handled by a DLL module, which is loaded into a target process on the computing system. This for example done using the AppInit DLLs technique. Once loaded, the DLL hooks CryptGenRandom function, that is, whenever CryptGenRandom is called by a process, the corresponding program flow is routed to the monitoring means 140. The monitoring means call GetModuleFileName to obtain the full path to the module of the caller process, which can point to a DLL or to an executable. The SHA256 digest of the binary file of the module is computed and compared against the predetermined conditions 130, such as a whitelist. If the result is positive, the monitoring means 140 grant access to the cryptographic resources 110 through API 120: CryptGenRandom is called with the intercepted parameters and the result is returned to the caller process. If access is not granted, ExitProcess is called instead, which causes the caller process to end.

Upon interception of the call or request 115A issued by process A, the monitoring means 140 obtain the identity of the caller process. This information is available within the Operating System. Next, the so obtained identity 116A is checked against a set of predetermined conditions 130, which includes for example a list of process identities which are authorized to access the cryptographic resources/source of randomness 110. The set or predetermined conditions 130 implements a system policy, and may advantageously be kept as simple as possible: verify whether the process has authorization to access the cryptographic resources 110 in the current conditions. Deciding the criteria that define whether an application is eligible to get authorization and delivering an authorization is preferably implemented as an offline process, which may be updated periodically. This can be implemented in various ways. For example, the computer system's administrator, or a user may whitelist an application. The process may also be more complex in that an application is only authorized after thorough checks. Alternatively, applications that have been digitally signed by a trusted authority may be allowed the cryptographic resources. In the illustrated scenario, process 116A is whitelisted and satisfies the predetermined conditions, as it corresponds to a process implementing an SSH communication protocol, so that the corresponding call is cleared 117A. Upon the positive clearing of the initial request from process A, the request is forwarded 118A to the

cryptographic resources 110, which in reply outputs the requested random number 119A, which is forwarded to process A.

Upon interception of the call or request 115B issued by process B, the monitoring means 140 obtain the identity of the caller process. This information is available within the Operating System. Next, the so obtained identity 116B is checked against a set of predetermined conditions 130, which includes for example a list of process identities which are authorized to access the cryptographic resources/source of randomness 110. In the illustrated scenario, process 116B is not listed as it corresponds to a cryptographic malware. It does not satisfy the predetermined conditions, so that the corresponding call is not cleared 117B. Upon the negative clearing of the initial request from process B, the request to the cryptographic resources 110 is not granted, and a termination signal 119B is transmitted to process B.

Figure 3 provides a schematic illustration of a computing system 100 for implementing the method as described through several embodiments. The computing system 100 comprises a central processing unit, CPU, 102 that is functionally connected via a data bus to have read/write access to volatile 106 and/or persistent 104 memory elements. These may include Hard Disk Drives, HDD, Solid State Drives, SSD, Random Access Memory, RAM, chips, removable drives or networked storage that is accessible through a non-illustrated networking interface and through a corresponding data communication channel. The CPU executes an Operating System, OS, within which it executes several programs in the form of processes. The CPU 102 is also functionally connected to cryptographic resources 110 including a cryptographically secure pseudo-random number generator 110. The generator 110 may be implemented as a dedicated hardware chip, or it may be a software routine executed by the CPU itself, as shown in Figure 3. Calls to the generator 110 are handled using a calling interface 120. Any call 115 by any process executed by the CPU is intercepted by a monitoring means 140, which is implemented as a software routine run by the CPU itself. A set of predetermined conditions 130, stored for example in one of the memory elements 104, 106, and retrieved therefrom, is consulted to check whether an intercepted call is authorized to access the pseudo-random number generator 110 or not. The corresponding call 115 is only forwarded to the pseudo-random number generator 110 if satisfies the predetermined conditions 130. Otherwise, the calling process is terminated. While the example of figure 3 shows the method as being implemented within the Operating System, it may as well be implemented as a standalone software program executed by the CPU within the Operating System. In such case, the instructions corresponding to the software program are read from the memory element 104, 106 and configure the CPU in such a way as to implement the method in accordance with aspects of the invention.

Based on the description that has been given and the accompanying figures, a person skilled in the art will be able to write a computer program implementing the functionalities described herein without undue burden and without requiring any degree of additional inventiveness.

It should be understood that the detailed description of specific preferred embodiments is given by way of illustration only, since various changes and modifications within the scope of the invention will be apparent to the person skilled in the art. The scope of protection is defined by the following set of claims.

Claims

1. A method for preventing ransomware attacks on a computing system (100), the computing system having access to computing resources including a source of randomness (110), which comprises random number or pseudo-random number generation means, and which are accessible through calling interface means (120), the method comprising the following steps:
   a) providing predetermined requirements (130) describing a permissible call to said source of randomness comprising random number or pseudo-random number generation means through said calling interface means (120) in a memory element;
   b) using monitoring means (140), monitoring the use of said calling interface means (120), comprising the obtention of data identifying a call (115, 115A, 115B);
   c) blocking said call (115B) to said source of randomness (110), unless the data identifying said call complies with said predetermined requirements (130).

2. The method according to claim 1, wherein the data identifying said call comprises data identifying the calling process.

3. The method according to claim 2, wherein the predetermined requirements comprise data identifying at least one calling process that is allowed to access said source of randomness, and wherein said call is blocked unless the data identifying the calling process is comprised in the data identifying at least one allowed calling process.

4. The method according to any of claims 1 to 3, wherein said source of randomness comprises a cryptographically secure random number generator.

5. The method according to any of claims 1 to 4, wherein said calling interface means comprise an application programming interface providing access to said source of randomness.

6. The method according to any of claims 1 to 5, wherein blocking said call comprises terminating the corresponding calling process.

7. The method according to any of claims 1 to 6, wherein said monitoring means are part of the computing system's operating system.

8. The method according to any of claims 1 to 7, wherein the source of randomness comprises a source of entropy.

9. A computer program comprising computer readable code means, which when run on a computer, cause the computer to carry out the method according to any of claims 1 to 8.

10. A computer program product comprising a computer-readable medium on which the computer program according to claim 9 is stored.

11. A computer configured for carrying out the method according to any of claims 1 to 10.

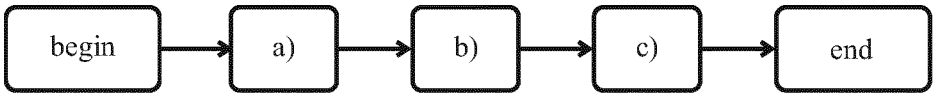

Fig. 1

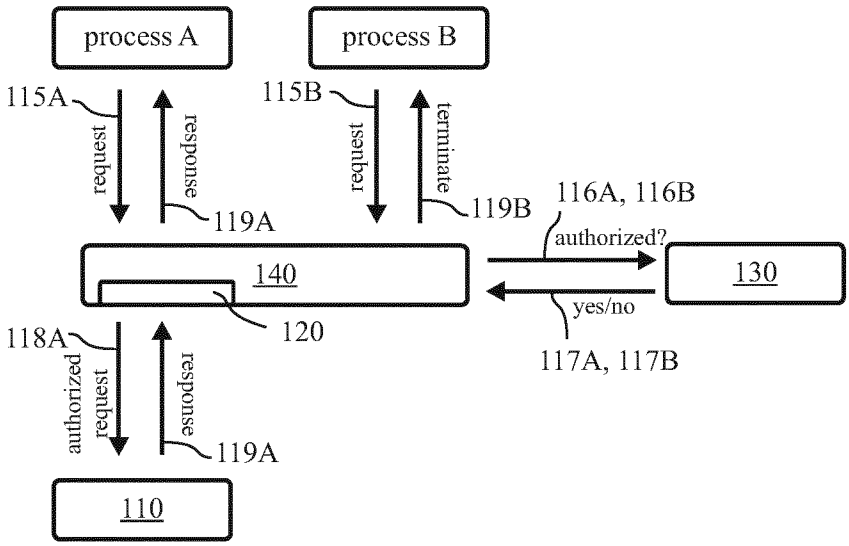

Fig. 2

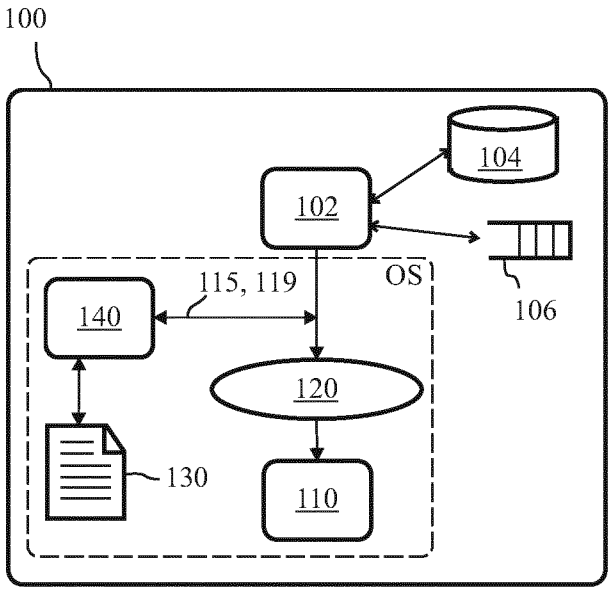

Fig. 3

INTERNATIONAL SEARCH REPORT

International application No
PCT/EP2019/066621

A. CLASSIFICATION OF SUBJECT MATTER
INV. G06F21/55 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.
A	Allan Liska ET AL: "Ransomware" In: "Ransomware", 30 November 2016 (2016-11-30), O'Reilly, U.S.A., XP055548888, ISBN: 978-1-4919-6788-1 pages 67-68, page 67, paragraph 2 - paragraph 3 -----	1-11
A	None: "Microsoft CryptoAPI", , 29 January 2018 (2018-01-29), XP055548900, Retrieved from the Internet: URL:https://web.archive.org/web/20180226073526/https://en.wikipedia.org/wiki/Microsoft_CryptoAPI [retrieved on 2019-01-29] page 1, paragraph 2 -----	1-11

-/--

[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
11 July 2019	22/07/2019

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	Meis, Marc

1

INTERNATIONAL SEARCH REPORT

International application No
PCT/EP2019/066621

C(Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	US 2014/140512 A1 (HADLEY TED A [US]) 22 May 2014 (2014-05-22) paragraph [0010] - paragraph [0037] -----	1-11
A	WO 2017/125935 A1 (MINERVA LABS LTD) 27 July 2017 (2017-07-27) page 8, line 23 - page 9, line 11 page 12, line 10 - line 13 -----	1-11

1

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No
PCT/EP2019/066621

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 2014140512 A1	22-05-2014	CN 103688269 A	26-03-2014
		CN 103733204 A	16-04-2014
		CN 103890852 A	25-06-2014
		EP 2734903 A1	28-05-2014
		EP 2734951 A1	28-05-2014
		EP 2735000 A1	28-05-2014
		US 2013024143 A1	24-01-2013
		US 2013024153 A1	24-01-2013
		US 2013024637 A1	24-01-2013
		US 2013024716 A1	24-01-2013
		US 2014130189 A1	08-05-2014
		US 2014140512 A1	22-05-2014
		US 2014149729 A1	29-05-2014
		US 2014156961 A1	05-06-2014
		US 2014164793 A1	12-06-2014
		US 2014165206 A1	12-06-2014
		US 2014189340 A1	03-07-2014
		US 2014223113 A1	07-08-2014
		WO 2012177295 A1	27-12-2012
		WO 2013012435 A1	24-01-2013
		WO 2013012436 A1	24-01-2013
		WO 2013012437 A1	24-01-2013
		WO 2013012444 A1	24-01-2013
		WO 2013012447 A1	24-01-2013
		WO 2013012449 A1	24-01-2013
		WO 2013012461 A1	24-01-2013
WO 2017125935 A1	27-07-2017	US 2018211038 A1	26-07-2018
		WO 2017125935 A1	27-07-2017