

(12) DEMANDE INTERNATIONALE PUBLIÉE EN VERTU DU TRAITÉ DE COOPÉRATION EN MATIÈRE DE BREVETS (PCT)

(19) Organisation Mondiale de la
Propriété Intellectuelle
Bureau international



(10) Numéro de publication internationale
WO 2018/178545 A1

(43) Date de la publication internationale
04 octobre 2018 (04.10.2018)

WIPO | PCT

(51) Classification internationale des brevets :

G06F 21/55 (2013.01) H04L 29/08 (2006.01)
H04L 29/06 (2006.01) H04L 12/801 (2013.01)
H04W 12/12 (2009.01) H04L 12/851 (2013.01)

(21) Numéro de la demande internationale :

PCT/FR2018/050714

(22) Date de dépôt international :

23 mars 2018 (23.03.2018)

(25) Langue de dépôt :

français

(26) Langue de publication :

français

(30) Données relatives à la priorité :

1752576 28 mars 2017 (28.03.2017) FR

(71) Déposant : ORANGE [FR/FR] ; 78 rue Olivier de Serres,
75015 PARIS (FR).

(72) Inventeurs : CHAIGNON, Paul ; ORANGE GARDENS
- IMT/OLR/IPL/PAT, 44 Avenue de la République, CS
50010, 92326 CHATILLON (FR). CHOUAKI, Yacine ;
ORANGE GARDENS - IMT/OLR/IPL/PAT, 44 Avenue
de la République, CS 50010, 92326 CHATILLON (FR).
LAZRI, Kahina ; ORANGE GARDENS - IMT/OLR/
IPL/PAT, 44 Avenue de la République, CS 50010, 92326
CHATILLON (FR).

(74) Mandataire : RENARD, Beatrice ; ORANGE IMT/OLR/
IPL/PAT, 44 Avenue de la République, CS 50010, 92326
CHATILLON (FR).

(81) États désignés (sauf indication contraire, pour tout titre de
protection nationale disponible) : AE, AG, AL, AM, AO,

(54) Title: METHOD FOR ASSISTING IN THE DETECTION OF DENIAL-OF-SERVICE ATTACKS

(54) Titre : PROCÉDÉ D'AIDE À LA DÉTECTION D'ATTAQUES PAR DÉNIS DE SERVICES

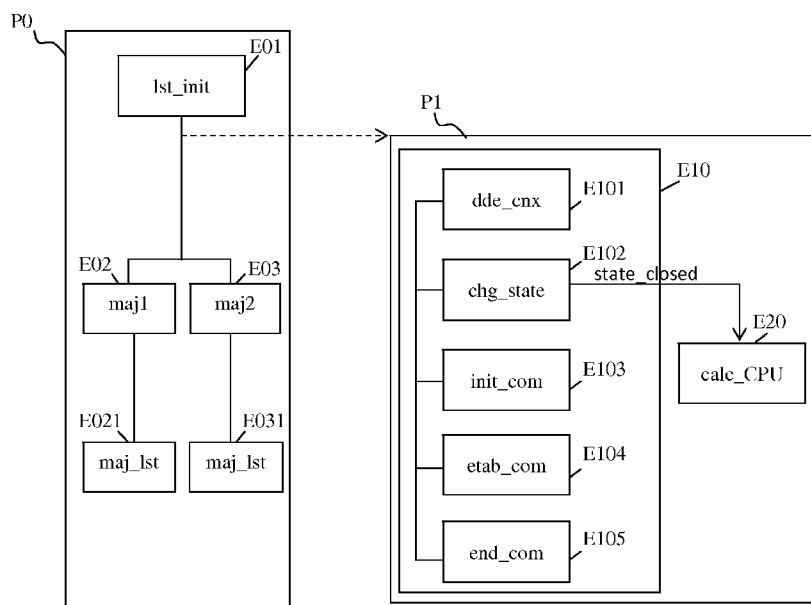


Figure 1

(57) Abstract: The invention relates to a method for assisting in the detection of application denial-of-service attacks in a system providing at least one application, said application using at least one service, the method comprising: - intercepting (E101) the execution of an operating system core function representative of the reception of a request to connect to the application coming from a client device, associating an identifier of said connection with the identifier of at least one process, said process being responsible for processing said connection on a first service, and obtaining a first time interval representative of a CPU time consumed by said process since its creation, - intercepting (E102) the execution of a second operating system core function representative of a closure of said connection, and obtaining a second time interval representative of a CPU time consumed by said process since its creation, - calculating (E20)

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) États désignés (*sauf indication contraire, pour tout titre de protection régionale disponible*) : ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, ST, SZ, TZ, UG, ZM, ZW), eurasién (AM, AZ, BY, KG, KZ, RU, TJ, TM), européen (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).

Publiée:

— avec rapport de recherche internationale (Art. 21(3))

a total CPU time consumed to process said connection on the basis of at least the second and the first CPU time interval consumed by said process.

(57) Abrégé : L'invention concerne un procédé d'aide à la détection d'attaques par dénis de service applicatifs dans un système proposant au moins une application, ladite application utilisant au moins un service, le procédé comprenant : - interception (E101) de l'exécution d'une fonction du noyau du système d'exploitation représentative de la réception d'une demande de connexion à l'application en provenance d'un équipement client, association d'un identifiant de ladite connexion à l'identifiant d'au moins un processus, ledit processus étant en charge du traitement de ladite connexion sur un premier service, et obtention d'un premier intervalle de temps représentatif d'un temps CPU consommé par ledit processus depuis sa création, - interception (E102) de l'exécution d'une deuxième fonction du noyau du système d'exploitation représentative d'une fermeture de ladite connexion, et obtention d'un deuxième intervalle de temps représentatif d'un temps CPU consommé par ledit processus depuis sa création, - calcul (E20) d'un temps CPU total consommé pour le traitement de ladite connexion à partir d'au moins le deuxième et le premier intervalle de temps CPU consommé par ledit processus.

Procédé d'aide à la détection d'attaques par dénis de services

La présente invention se rapporte au domaine général des télécommunications. Elle concerne plus particulièrement l'aide à la détection d'attaques par dénis de service applicatifs
5 qui visent à rendre un service indisponible à des utilisateurs légitimes par saturation des ressources d'un serveur ou d'un équipement réseau.

On connaît plusieurs types d'attaques par dénis de services (on parle habituellement de DoS ou de DDoS, pour « Distributed Denial of Service ») : les attaques par dénis de services réseaux et les attaques par dénis de services applicatifs.

10 Les attaques par dénis de services réseaux consistent en l'envoi d'un nombre important de paquets réseau à un service cible, par exemple un équipement réseau ou un serveur Web, de manière à mobiliser les ressources dans le traitement de ces paquets. Cela a pour conséquence de saturer les ressources du service cible et de rendre ces ressources non disponibles pour le traitement de requêtes provenant de clients légitimes. Les attaques par dénis de services réseaux
15 se caractérisent principalement par l'envoi d'une quantité significative de paquets réseau. La détection de telles attaques repose donc principalement sur une analyse quantitative des requêtes reçues par un équipement réseau.

La deuxième catégorie d'attaques par dénis de services correspond à des attaques applicatives. Celles-ci visent des vulnérabilités d'implémentation d'une application Web ou
20 directement des pages qui consomment beaucoup de ressources CPU (de l'anglais « Central Processing Unit ») et de mémoire pour se créer ou s'exécuter. Par exemple, une telle page peut permettre de faire des recherches dans une base de données de taille importante. De telles pages utilisent souvent des langages interprétés qui provoquent de forts besoins en consommation de ressources de type processeur et mémoire.

25 Une telle attaque est plus facile à réaliser qu'une attaque par dénis de service réseaux et est également plus subtile à détecter. En effet, l'attaque applicative ne requiert pas d'importantes ressources réseaux pour parvenir à une saturation du service ciblé : il suffit d'envoyer des requêtes légitimes à une page web consommatrice de ressource CPU à une fréquence que l'on peut qualifier de normale. Une « fréquence normale » correspond à quelques
30 requêtes par seconde. On comprend que les attaques par dénis de service applicatifs sont plus difficiles à identifier que des attaques par dénis de services réseaux.

Un des buts de l'invention est de remédier à des insuffisances/inconvénients de l'état de la technique et/ou d'y apporter des améliorations.

A cette fin, l'invention propose un procédé d'aide à la détection d'attaques par dénis de service applicatifs dans un système proposant au moins une application, ladite application utilisant au moins un service, le procédé comprenant :

- 5 - interception de l'exécution d'une fonction du noyau du système d'exploitation représentative de la réception d'une demande de connexion à l'application en provenance d'un équipement client, association d'un identifiant de ladite connexion à l'identifiant d'au moins un processus, ledit processus étant en charge du traitement de ladite connexion sur un premier service, et obtention d'un premier intervalle de temps représentatif d'un temps CPU consommé par ledit processus depuis sa création,
- 10 - interception de l'exécution d'une deuxième fonction du noyau du système d'exploitation représentative d'une fermeture de ladite connexion, et obtention d'un deuxième intervalle de temps représentatif d'un temps CPU consommé par ledit processus depuis sa création,
- 15 - calcul d'un temps CPU total consommé pour le traitement de ladite connexion à partir d'au moins le deuxième et le premier intervalle de temps CPU consommé par ledit processus.

Le procédé décrit ici permet d'associer à chaque équipement client qui demande l'exécution d'une application sur un serveur un profil de consommation précis, représentatif du temps CPU réellement consommé au niveau du serveur pour le traitement de l'application pour ce client. Le temps CPU correspond au temps que le processeur a consacré au traitement de la

20 requête initiée par l'équipement client et est mesuré comme une somme de mesures de temps d'exécution d'au moins un processus du serveur. Il est important de noter que dans un environnement multi tâches tel qu'un serveur susceptible d'héberger une pluralité d'applications, la détermination du temps CPU consommé par des processus dans le cadre d'une activité donnée, ici le traitement d'une connexion issue d'un équipement client, n'est pas

25 évidente. Il est en effet connu qu'un processus donné s'occupe généralement de plusieurs connexions. En outre, une connexion est généralement traitée par plusieurs processus.

Ainsi, le procédé permet d'associer à chaque client d'une application web un profil de consommation de temps CPU précis qu'il a engendré au niveau du serveur hébergeant l'application.

30 Le procédé repose sur l'interception d'un certain nombre de fonctions du noyau du système d'exploitation. En se plaçant ainsi au niveau du noyau du système d'exploitation, le procédé est agnostique et ne dépend pas de programmes utilisateur particuliers. Par exemple il ne nécessite pas qu'une application supervisée soit programmée de manière à intégrer sa propre supervision. Par ailleurs, on s'affranchit ainsi de spécificités relatives à des interfaces propres à

35 certaines applications.

Dans un exemple de réalisation, le procédé comprend, lorsque ladite application utilise au moins deux services et qu'un premier processus d'un premier des au moins deux services établit une communication avec un deuxième processus d'un deuxième des au moins deux services, un identifiant dudit premier processus étant associé à l'identifiant de ladite connexion :

5 - interception de l'exécution d'une fonction du noyau du système d'exploitation représentative d'une demande d'établissement d'une communication entre le premier processus et le deuxième processus,

 - interception de l'exécution d'une fonction du noyau du système d'exploitation représentative de l'établissement de la communication entre le premier et le deuxième
10 processus, association d'un identifiant du deuxième processus à l'identifiant de ladite connexion, et obtention d'un deuxième premier intervalle de temps représentatif du temps CPU consommé par le deuxième processus depuis sa création,

 - interception de l'exécution d'une fonction du noyau du système d'exploitation représentative de la fermeture de la communication entre le premier processus et le deuxième
15 processus, et obtention d'un deuxième intervalle de temps représentatif du temps CPU consommé par le deuxième processus depuis sa création,

 - calcul d'un deuxième temps CPU de traitement associé à ladite communication à partir du deuxième et du premier deuxième intervalle de temps,

 - prise en compte du deuxième temps CPU pour le calcul du temps CPU total
20 consommé pour le traitement de ladite connexion.

Le procédé s'applique à des applications qui font appel à une pluralité de services. Typiquement il s'applique à une application Web qui repose sur un service HTTP, un service PHP et un service SQL et pour laquelle un processus d'un premier service envoie des requêtes à un processus d'un deuxième service. Le procédé permet de chaîner les différents processus à un
25 identifiant de la connexion établie par l'équipement client, en l'espèce une adresse IP de l'équipement, et de récolter les informations relatives aux temps d'activité qu'un processus consacre au traitement de la connexion. Ce chaînage est complété notamment lors de l'établissement de communications inter processus au cours desquelles un premier processus, dit processus client, établit une communication avec un deuxième processus, dit processus
30 serveur, par le biais d'une communication inter processus tel qu'un socket.

Dans un exemple de réalisation, l'identifiant du deuxième processus est associé à l'identifiant de ladite connexion lorsqu'un premier identifiant de la communication entre le premier et le deuxième processus, ledit premier identifiant étant obtenu dans une structure de données associée au premier processus est identique à un deuxième identifiant de la

communication entre le premier et le deuxième processus, ledit deuxième identifiant étant obtenu dans une structure de données associée au deuxième processus.

Ainsi, une liste des processus qui contribuent au traitement de la connexion est construite au fur et à mesure que des processus actifs, i.e., contributeurs, sont identifiés.

5 Avantageusement, le procédé comprend, dans une étape de configuration initiale une fourniture d'une liste de processus à superviser pour ladite application, et

- interception de l'exécution d'une fonction du noyau du système d'exploitation représentative de la création d'un processus fils par un processus parent,

- lorsque le processus parent appartient à la liste des processus à superviser, obtention
10 de l'identifiant dudit processus fils et ajout dudit processus fils à la liste des processus à superviser.

Le procédé s'applique à une liste initiale de processus fournie par l'administrateur en charge de superviser l'exécution de l'application. Cette liste de processus comprend les processus à superviser dans le cadre de l'exécution de l'application. Fournir une telle liste
15 initiale permet de limiter les interceptions de fonctions du noyau du système d'exploitation à des processus qui sont effectivement associés à l'application Web. Cette liste est dynamique en ce sens que des processus qui sont créés dans le cadre de l'exécution de l'application sont intégrés à la liste de processus à superviser dès leur création. Le procédé s'applique à tout instant à l'ensemble des processus qui contribuent à l'exécution de l'application. Ainsi, le calcul
20 du temps CPU total est précis et tient compte de l'ensemble des processus exécutés dans le cadre de l'application.

Dans un exemple de réalisation, le procédé comprend :

- interception de l'exécution d'une fonction du noyau du système d'exploitation représentative de l'arrêt d'un processus existant,

- lorsque le processus existant appartient à la liste des processus à superviser, obtention
25 de l'identifiant dudit processus existant et suppression dudit processus existant dans la liste des processus à superviser.

De même, lorsque des processus qui font partie de la liste des processus supervisés sont arrêtés, ils sont supprimés de la liste. Ainsi, le procédé optimise le traitement qu'il met en œuvre
30 en se concentrant sur les processus susceptibles de consommer du temps CPU.

L'invention concerne également un serveur apte à héberger au moins une application, ladite application utilisant au moins un service, ledit serveur comprenant :

- des premiers moyens d'interception et d'obtention, agencés pour intercepter l'exécution d'une fonction du noyau du système d'exploitation représentative de la réception
35 d'une demande de connexion à l'application en provenance d'un équipement client, associer un

identifiant de ladite connexion à l'identifiant d'au moins un processus, ledit processus étant en charge du traitement de ladite connexion, et pour obtenir un premier intervalle de temps représentatif d'un temps CPU consommé par ledit processus depuis sa création,

5 - des deuxièmes moyens d'interception et d'obtention, agencés pour intercepter l'exécution d'une deuxième fonction du noyau du système d'exploitation représentative d'une fermeture de ladite connexion, et pour obtenir un deuxième intervalle de temps représentatif d'un temps CPU consommé par ledit processus depuis sa création,

10 - des moyens de calcul, agencés pour calculer un temps CPU total consommé pour le traitement de ladite connexion à partir d'au moins le deuxième et le premier intervalle de temps CPU consommé par ledit processus.

L'invention porte également sur un programme pour un serveur apte à héberger au moins une application utilisant au moins un service, ledit programme comprenant des instructions de code de programme destinées à commander l'exécution des étapes du d'aide à la détection d'attaques par dénis de service applicatifs, tel que décrit précédemment, lorsque le
15 programme est exécuté sur ledit serveur.

L'invention concerne aussi un support de données dans lequel est enregistré le programme décrit ci-dessus.

20 D'autres caractéristiques et avantages de la présente invention seront mieux compris de la description et des dessins annexés parmi lesquels :

- la figure 1 présente les étapes d'un procédé d'aide à la détection d'attaques par dénis de services applicatifs, selon un exemple de réalisation ;

25 - la figure 2 est une représentation schématique d'un serveur, apte à mettre en œuvre le procédé d'aide à la détection d'attaques par dénis de services applicatifs, selon un exemple de réalisation.

Les étapes d'un procédé d'aide à la détection d'attaques par dénis de service applicatifs, selon un exemple de réalisation, vont maintenant être décrites en relation avec la figure 1.

30 L'invention est décrite ici dans le cas de l'accès à partir de navigateurs Web d'équipements clients à une ou des applications Web hébergées par un équipement informatique, tel un serveur. L'accès à ces applications Web repose sur le protocole de communication HTTP (pour « HyperText Transfer Protocol ») et le protocole de transport TCP (pour « Transmission Control Protocol »). Beaucoup d'attaques visent de telles applications, accessibles à partir de navigateurs Web. Ces applications utilisent la plupart du temps le
35 protocole HTTP et le protocole TCP qui est le protocole de transport utilisé pour la majorité du

trafic Internet. Dans l'exemple de réalisation décrit ici, le système d'exploitation du serveur qui héberge les applications Web est Linux. L'invention est applicable à l'ensemble des systèmes d'exploitation de type Linux et Unix. De même l'invention n'est pas limitée à des applications Web et s'applique à toute application qui nécessite l'établissement d'une connexion TCP avec
5 le serveur.

Une application Web repose sur un ou plusieurs services pour s'exécuter : par exemple un serveur HTTP, tel Apache, adapté pour recevoir des requêtes émises par des navigateurs d'équipements clients, chercher une page et la renvoyer, un service de génération de contenus tel que PHP (pour « PHP: Hypertext Preprocessor »), adapté pour écrire dans une page Web un
10 affichage dynamique d'informations, et un système de gestion de base de données tel que MySQL, adapté pour stocker des données et permettre leur manipulation à travers un langage de requête associé. L'utilisation de ces trois services n'est nécessaire que lorsque l'on souhaite afficher des informations provenant d'une base de données. On admet qu'une telle configuration se présente assez fréquemment car elle répond globalement à la majorité des besoins.

15 Les différents services impliqués lors de l'exécution de l'application Web comprennent un service dit frontal (ou service « frontend »), agencé pour recevoir les requêtes en provenance des navigateurs des équipements clients et un ou plusieurs services en arrière-plan, ou backoffice (ou services « backend »), lorsqu'ils sont appelés par un autre service. Typiquement un service HTTP, qualifié de service frontend, qui reçoit des requêtes de navigateurs
20 d'équipements clients, fait appel à des services PHP et MySQL, qui sont qualifiés de services backend.

Les différents services impliqués lors de l'exécution d'une application Web s'exécutent sous forme de processus. Un processus est défini comme une instance de programme en cours d'exécution. Les processus communiquent afin de se transmettre des requêtes à traiter et des
25 réponses à ces requêtes au moyen de mécanismes de communication interprocessus (ou « IPC » en anglais pour « Inter Process Communications »). Ces mécanismes de communication font en général appel au noyau du système d'exploitation du serveur qui héberge l'application Web pour l'établissement de la connexion entre deux processus.

Le procédé d'aide à la détection d'attaques par dénis de services applicatifs décrit ici
30 consiste à associer à une connexion TCP au niveau du serveur, initiée par un client lors de l'accès à l'application Web via le navigateur du client, l'ensemble des processus qui sont impliqués par cette exécution, et à comptabiliser le temps CPU que l'ensemble de ces processus a consommé pour traiter cette connexion. Ce temps CPU correspond à un cumul de temps effectif passé par le processeur du serveur pour le traitement de la connexion TCP. Il est
35 important de souligner que la détermination du temps consacré par des processus à une

connexion TCP donnée n'est pas évidente puisqu'il est connu qu'un processus donné s'occupe généralement de plusieurs connexions. En outre, une connexion est généralement traitée par plusieurs processus.

Pour identifier les processus qui contribuent au traitement d'une connexion TCP, ou
5 connexion cliente, et comptabiliser le temps CPU associé, des sondes sont utilisées afin d'intercepter l'exécution et/ou le retour ou fin d'exécution de certaines fonctions du noyau du système d'exploitation et de déclencher des traitements particuliers sur ces interceptions. Plus précisément, un programme de supervision, apte à mettre en œuvre les étapes du procédé d'aide à la détection d'attaques par dénis de services applicatifs positionne des points d'interception
10 (ou « hook » en anglais) sur l'appel et/ou la sortie de certaines fonctions du noyau du système d'exploitation et déclenche un traitement spécifique lors d'une telle interception. Ainsi, à chaque fois que le noyau du système d'exploitation appelle ou sort d'une fonction supervisée, le programme de supervision prend la main pour mettre en œuvre une fonction d'interception prévue par le programme de supervision. La fonction d'interception est appelée avec des
15 informations de contexte d'exécution de la fonction du noyau dont l'exécution a été interceptée et qui comprennent des informations sur le processus courant, ou processus en cours d'exécution, lorsque la fonction du noyau a été appelée, les valeurs de registres CPU, etc. Les informations sur le processus courant sont comprises dans une structure de données, appelée « struct task_struct », allouée dynamiquement à chaque processus et qui peut être
20 obtenue à partir des informations de contexte d'exécution. A noter que les arguments passés à la fonction du noyau du système d'exploitation lorsqu'elle est appelée peuvent être obtenus à partir des valeurs des registres CPU.

Une fois obtenu le temps CPU accordé par chacun des processus au traitement d'une connexion cliente, il est calculé un temps CPU total effectif que le système d'exploitation a
25 consacré au traitement de cette connexion. Ce calcul du temps CPU total est mis en œuvre lors de la fermeture de la connexion TCP initiée par le client.

A noter que le programme de supervision qui met en œuvre le procédé d'aide à la détection d'attaques par dénis de services applicatifs s'exécute dans l'espace utilisateur, au même titre que l'application Web. Le programme de supervision est indépendant de
30 l'application Web ; il peut s'appliquer à toute application Web qui s'exécute sur le serveur. En ce sens, le procédé d'aide à la détection d'attaques par dénis de services applicatifs est qualifié d'agnostique en ce qui concerne les applications Web susceptibles de s'exécuter sur le serveur.

Dans une phase P0 de configuration initiale, dans une sous-étape E01 de fourniture d'une liste initiale de processus, il est fourni par un administrateur en charge de la supervision
35 des applications Web proposées sur le serveur une liste initiale de processus qui comprend les

identifiants de tous les processus des différents services associés à l'application Web pour laquelle la supervision doit être mise en œuvre. Un identifiant de processus, ou PID (pour « Process IDentifier), est un code unique attribué par le système d'exploitation à tout processus lors de son démarrage. Il permet d'identifier les processus dans toute commande qui s'applique

5 à un processus. L'étape E01 de fourniture de la liste initiale de processus est mise en œuvre à chaque fois que le programme de supervision est exécuté pour une application Web. Fournir une telle liste initiale permet de limiter les interceptions de fonctions du noyau du système d'exploitation à des processus qui sont effectivement associés à l'application Web. Ainsi, lors de toute interception de l'exécution d'une fonction du noyau du système d'exploitation, il est

10 d'abord vérifié que le processus courant associé à l'exécution de la fonction du noyau et dont l'identifiant est disponible dans des informations de contexte d'exécution de la fonction du noyau, fait partie des processus associés à l'application Web. Si le processus courant ne fait pas partie de la liste des processus relatifs à l'application Web, alors l'interception de la fonction du noyau, plus précisément le traitement associé à cette interception, est interrompue de manière à

15 poursuivre l'exécution normale du noyau. Ce cas de figure correspond à l'exécution d'un processus associé à une autre application qui s'exécute sur le même serveur que l'application Web. Interrompre la supervision dans ce cas permet de limiter l'impact du procédé sur cette autre application.

Dans un exemple de réalisation, la liste des processus est segmentée par service. Un

20 premier groupe de processus correspond par exemple aux processus du service frontend HTTP, un deuxième groupe de processus regroupe les processus d'un service backend tel que PHP, etc. Une telle segmentation permet de détailler la répartition par service de la consommation CPU relative à une connexion cliente donnée, et d'analyser plus finement l'origine de consommations anormales de temps CPU.

25 A noter cependant que la configuration initiale n'est pas forcément représentative de l'ensemble des processus impliqués lors de l'exécution de l'application Web. En effet, la liste initiale de processus peut évoluer au fil du temps : les services créent et arrêtent des processus pour s'adapter à une charge courante, par exemple en fonction du nombre de requêtes reçues. Ainsi il est nécessaire de tenir compte de créations et d'arrêts de processus lors de l'exécution

30 de l'application Web.

A cette fin dans une étape suivante E02 de première mise à jour dynamique de la liste des processus, l'exécution d'une première fonction du noyau, appelée à chaque création d'un nouveau processus est interceptée. Intercepter l'exécution de cette première fonction permet d'obtenir, via les informations de contexte d'exécution de la fonction du noyau au moment de

35 l'interception, l'identifiant du nouveau processus créé et l'identifiant du processus à l'origine de

la création de ce nouveau processus. Ce processus est habituellement appelé « processus père », et le nouveau processus « processus fils ». Dans le cas où le processus père fait partie des processus supervisés qui figurent dans la liste initiale de processus obtenue au cours de l'étape E01 de fourniture de la liste initiale, alors il est nécessaire de superviser également le processus fils qui fait partie de l'application Web. Dans une sous-étape E021 de mise à jour de la liste, mise en œuvre lorsque le processus père fait partie de la liste des processus supervisés, l'identifiant du processus fils, obtenu à partir des informations de contexte d'exécution associées à la fonction du noyau en cours d'exécution, est ajouté à la liste de processus à superviser.

10 Dans un exemple de réalisation, correspondant au cas où l'application Web s'exécute sur un serveur dont le système d'exploitation est Linux, le prototype de la fonction du noyau du système d'exploitation dont l'exécution est interceptée est :
`void wake_up_new_task(struct task_struct *p)`. On rappelle que le prototype d'une fonction est une description qui comprend le nom de la fonction, le type des arguments, 15 ou paramètres d'entrée de la fonction, et le type de la valeur renvoyée. La fonction du noyau `wake_up_new_task` est appelée à chaque fois qu'un nouveau processus est créé. Un nouveau processus est créé au moyen d'un appel système, par exemple « fork » ou « clone ». L'objet donné en argument de la fonction, en l'espèce la structure de données `struct task_struct *p`, contient les informations sur le nouveau processus. En interceptant 20 l'exécution de cette fonction, l'identifiant du nouveau processus est récupéré via l'objet donné en argument. L'identifiant du processus père est obtenu via les informations de contexte d'exécution de la fonction du noyau.

Lors d'une étape E03 de deuxième mise à jour dynamique, mise en œuvre lors de l'arrêt d'un des processus supervisés, il est intercepté l'exécution d'une fonction du noyau du système d'exploitation appelée à chaque fois qu'un processus est arrêté. Cette interception permet de 25 supprimer de la liste des processus à superviser celui qui a été arrêté. L'identifiant du processus arrêté figure dans les informations de contexte d'exécution de la fonction du noyau dont l'exécution est interceptée. Ainsi, l'identifiant du processus arrêté, s'il fait partie de la liste des processus associés à l'exécution de l'application Web est alors supprimé de la liste des 30 processus à superviser dans une sous-étape E031 de mise à jour de la liste. La liste de processus est ainsi mise à jour dynamiquement. Cela permet d'optimiser les traitements en se limitant aux processus actifs pour l'application web en cours d'exécution.

Dans l'exemple de réalisation décrit ici qui correspond au système d'exploitation Linux, le prototype de la fonction dont l'exécution est interceptée lors de l'arrêt d'un processus est : 35 `void do_exit(long code)`.

L'étape E02 de première mise à jour de la liste de processus, respectivement E03 de deuxième mise à jour, est exécutée dès qu'un processus associé à l'application Web est créé, respectivement arrêté. Ces étapes s'exécutent comme tâche de fond pendant toute l'exécution de l'application de supervision.

5 Dans une phase suivante P1 de traitement, mise en œuvre en tâche de fond pendant toute l'exécution du programme de supervision et de préférence après avoir fourni la liste initiale de processus au cours de l'étape E01 de la phase P0, l'exécution et/ou la fin d'exécution d'une pluralité de fonctions du noyau du système d'exploitation est interceptée dans une étape E10 d'interception. Des informations relatives à l'exécution de l'application Web sont alors
10 récupérées à partir des informations de contexte d'exécution de la fonction du noyau interceptée et/ou de la structure de données `struct task_struct` associée au processus pour lequel la fonction du noyau est exécutée. On rappelle que la structure de donnée `struct task_struct` d'un processus peut être obtenue à partir des informations de contexte d'exécution de la fonction du noyau dont l'exécution est interceptée. Pour des raisons de
15 lisibilité, l'étape E10 d'interception est décrite sous forme d'une pluralité de sous-étapes, chacune des sous-étapes étant dédiée à une fonction du noyau du système d'exploitation dont l'exécution est interceptée. A noter que pour la pluralité de sous-étapes de l'étape E10 décrites ci-dessous, il est vérifié que le processus courant pour lequel l'exécution de la fonction du noyau est interceptée fait partie de la liste des processus supervisés telle que fournie et enrichie
20 au cours des étapes E01 de fourniture de la liste initiale de processus, E02 et E03 de première et deuxième mise à jour, décrites précédemment.

Ainsi :

- dans une sous-étape E101 de l'étape d'interception E10, l'exécution d'une fonction de demande de connexion, mise en œuvre à chaque fois qu'un client demande l'ouverture d'une
25 connexion TCP à partir d'un équipement client, est interceptée. Dans l'exemple décrit ici, cette fonction est appelée lorsque le serveur HTTP, en écoute sur un port TCP destiné à recevoir les demandes de connexion reçoit une demande de connexion.

Dans le cas de Linux, le prototype de la fonction du noyau appelée est :
`struct sock *inet_csk_accept(struct sock *sk, int flags, int *err)`. Plus
30 précisément, le serveur HTTP qui reçoit une demande de connexion sur le port TCP en provenance d'un équipement client utilise l'appel système `accept` qui résulte en l'exécution de la fonction du noyau `inet_csk_accept`. A noter que cette fonction est bloquante dans le sens où elle ne retourne qu'une fois qu'une connexion a été établie avec le client. L'interception du retour de cette fonction permet d'établir une relation entre la connexion distante d'un client,
35 identifiée par une adresse IP associée à l'équipement du client à l'origine de la demande de

connexion courante, et le processus du serveur HTTP qui gère cette connexion courante, en l'espèce celui qui a appelé la fonction `accept`. Des informations sur la connexion TCP établie sont disponibles dans l'objet retourné par la fonction du noyau, en l'espèce l'objet `struct sock`. En particulier, l'objet retourné contient l'adresse IP du client, ce qui permet de

5 l'identifier de façon unique. Il est mémorisé dans un tableau associatif l'adresse IP du client en association avec au moins l'identifiant du processus qui gère la connexion courante pour le serveur HTTP. L'identifiant du processus est récupéré à partir des informations de contexte d'exécution de la fonction dont l'exécution a été interceptée. Par ailleurs, dans un exemple de réalisation, une première instance de la structure de données `struct task_struct` propre

10 au processus à l'origine de l'interception de l'exécution de la fonction du noyau est mémorisée en association avec l'identifiant du processus. La première instance de la structure de données est obtenue à partir des informations de contexte d'exécution de la fonction du noyau. La première instance de la structure de données `struct task_struct` comprend un intervalle de temps, noté Δt_1 et appelé « premier intervalle de temps » correspondant au temps

15 d'exécution CPU consommé par le processus depuis sa création. Cet intervalle est donc représentatif de l'activité passée du processus, mesurée en consommation CPU, avant qu'il ne soit en charge du traitement de la connexion TCP. Ce premier intervalle de temps est destiné à être utilisé dans le calcul du temps CPU dédié à la connexion TCP ;

- dans une sous-étape E102 de l'étape E10 d'interception, l'exécution d'une fonction

20 représentative d'une fermeture de connexion, mise en œuvre à chaque fois qu'une connexion TCP est fermée est interceptée. L'interception de l'exécution de cette fonction déclenche dans une étape E20 de calcul de temps CPU, le calcul de la consommation CPU totale dédié au traitement de la connexion TCP. L'étape E20 de calcul est décrite précisément par la suite.

Dans le cas de Linux, la fonction dont l'exécution est interceptée est représentative d'un

25 changement d'état d'une connexion. Un des paramètres de cette fonction fournit le nouvel état que prend la connexion. Ainsi, l'interception de cette fonction du noyau permet de détecter qu'une connexion se termine lorsque l'état retourné par cette fonction est « fermé » (ou « closed » en anglais). le prototype de la fonction du noyau appelée lorsque la connexion TCP change d'état est : `void tcp_set_state(struct sock *sk, int state)`. La variable

30 « state » comprend l'état de la connexion. Le paramètre `struct sock *sk` permet d'identifier la connexion TCP qui vient d'être fermée. L'identifiant du processus en charge de la connexion TCP, obtenu dans les informations de contexte d'exécution, permet d'identifier, via le tableau associatif l'adresse IP du client associée à l'identifiant du processus en charge de la connexion. Le calcul de la consommation CPU totale, mis en œuvre au cours de l'étape E20

35 permet ainsi d'associer une consommation CPU totale à la connexion TCP. Dans un exemple de

réalisation, une deuxième instance de la structure de données `struct task_struct` propre au processus à l'origine de l'exécution de la fonction du noyau interceptée est mémorisée en association avec l'identifiant du processus. La deuxième instance comprend un intervalle de temps, noté Δt_2 et appelé « deuxième intervalle de temps » correspondant au temps CPU
5 consommé par le processus depuis sa création. Ce deuxième intervalle de temps est représentatif de l'activité passée du processus, c'est-à-dire depuis sa création et jusqu'à ce que la connexion TCP soit fermée. Ce deuxième intervalle de temps est destiné à être utilisé dans le calcul du temps d'exécution CPU dédié à la connexion au cours de l'étape E20 ;

Les fonctions suivantes sont spécifiques à une configuration dans laquelle plusieurs
10 services interagissent pour fournir l'application Web. Typiquement, l'application Web implique un premier service HTTP, un deuxième service de création de pages Web dynamiques, tel que PHP et un troisième service de gestion de base de données, tel que MySQL. Dans ce cas, un processus d'un premier service peut communiquer avec un processus d'un deuxième service dans le cadre de l'exécution de l'application Web. Lorsque ce genre de communication est mis
15 en œuvre, les services peuvent utiliser des mécanismes de communication inter processus pour se transmettre des requêtes. Dans une communication inter processus, on distingue un processus client qui transmet une requête à traiter à un processus serveur destiné à recevoir et traiter la requête. Bien sûr, dans une communication inter processus un processus serveur peut également jouer le rôle d'un processus client si, après avoir reçu une requête d'un processus client, il
20 transmet lui-même une requête à un autre processus. C'est le cas par exemple lorsqu'un processus du service HTTP transmet une requête au service PHP qui lui-même fait appel à un service MySQL. Le service PHP joue successivement le rôle du serveur pour la requête en provenance du service HTTP et le rôle du client pour la requête qu'il transmet au service MySQL. Afin de mesurer le temps CPU total consommé pour traiter la connexion TCP, il est
25 donc nécessaire de tracer ces communications inter processus afin d'identifier tous les processus impliqués dans le traitement de la connexion et de prendre en compte le temps CPU qu'ils ont consommé pour contribuer au traitement de la connexion TCP.

Dans un exemple de réalisation, les communications inter processus sont mises en œuvre au moyen d'un ensemble normalisé de fonctions de communication, également appelé
30 interfaces de connexion. Le terme habituellement utilisé pour désigner cet ensemble de fonctions de communication est le terme anglais « socket ». Parmi les communications inter processus basées sur les sockets, on distingue les communications basées sur des sockets Unix et les communications basées sur des sockets TCP.

Dans le cas de communications inter processus reposant sur des sockets, les fonctions
35 du noyau dont on souhaite superviser l'exécution sont les suivantes :

- dans une sous-étape E103 de l'étape E10 d'interception, l'exécution d'une fonction du noyau appelée lorsqu'un premier processus d'un premier service initie une communication avec un deuxième processus d'un deuxième service est interceptée. La communication entre les deux processus est initiée au moyen d'un socket en faisant un appel système de type `connect`. Un tel appel système est destiné à débiter une communication sur un socket.

Dans un premier exemple de réalisation, basé sur les socket Unix, le prototype de la fonction du noyau d'initiation d'une communication dont l'exécution est interceptée est :

```
int unix_stream_connect(struct socket *sock, struct sockaddr *uaddr, int addr_len, int flags).
```

Au retour de cette fonction, la connexion entre les premier et deuxième processus est établie. Dans cette communication le premier processus est le processus client et le deuxième processus est le processus serveur. L'identifiant du premier processus, ou processus client, qui a initié la communication est obtenu dans les informations de contexte d'exécution de la fonction du noyau. Il est possible de récupérer un objet représentatif de la communication côté processus serveur. Cet objet, `struct sock`, est obtenu dans le champ `peer` de l'argument `struct socket *sock` de la fonction. Un lien entre le processus client et l'objet `struct sock *sock` est mémorisé jusqu'à obtenir l'identifiant du processus serveur lorsque le processus serveur accepte la communication inter processus.

Dans un deuxième exemple de réalisation, basé sur les sockets TCP, le prototype de la fonction du noyau d'initiation d'une communication est :

```
int tcp_connect(struct sock *sk).
```

Cette fonction est appelée lorsqu'un processus souhaite établir une connexion TCP en faisant un appel système de type `connect`. Le processus qui fait l'appel système joue le rôle du client dans la connexion à établir entre les deux processus. L'interception de l'exécution de cette fonction permet de récupérer un 4-uplet TCP destiné à identifier la connexion entre les deux processus. Ce 4-uplet comprend une adresse IP source, un port source, une adresse IP destination et un port destination. Les connexions entre les processus client et serveur s'effectuent sur l'interface locale, i.e. sur le même équipement. De ce fait, les adresses IP source et destination sont égales à une adresse désignée par « localhost » pour une connexion locale. Les deux ports permettent donc d'identifier de manière unique la connexion depuis les deux processus. Un lien entre l'identifiant du processus client, à l'origine de l'appel système `connect`, et les ports source et destination est mémorisé jusqu'à identifier le processus serveur.

A noter qu'au moment de l'initiation d'une communication entre un processus client et un processus serveur, que ce soit au moyen d'un socket Unix ou d'un socket TCP, on ne connaît pas encore l'identifiant du processus serveur ;

- dans une sous-étape E104 de l'étape E10 d'interception, l'exécution d'une fonction du noyau destinée à établir la communication entre deux processus de services différents, après initiation d'une communication inter processus, est interceptée. Cette fonction est appelée par le processus serveur lorsque celui-ci accepte des communications de processus client via des sockets. L'interception de l'exécution de cette fonction permet d'obtenir l'identifiant du processus serveur qui accepte la communication inter processus initiée par le processus client et permet ainsi de compléter le tableau associatif propre à la connexion TCP qui mémorise les identifiants des processus impliqués dans le traitement de la connexion TCP.

Dans un premier exemple de réalisation, basé sur les socket Unix, le prototype de la fonction du noyau d'établissement de la communication dont l'exécution est interceptée est :
`int unix_accept(struct socket *sock, struct socket *newsock, int flags)`. Cette fonction est appelée par un processus serveur qui souhaite accepter des communications de processus clients initiées via des sockets Unix. Cette fonction est bloquante dans le sens où elle ne retourne une valeur qu'une fois la connexion effectivement établie avec un processus client. L'interception du retour de cette fonction permet de récupérer l'objet `struct_sock` correspondant au côté serveur de la connexion. Plus précisément, cet objet est un champ de la structure `struct socket *newsock`, paramètre de `unix_accept`. Ainsi, il est possible de corréler l'identifiant du processus client avec l'identifiant du processus serveur à partir des informations sauvegardées lors de l'exécution de la fonction `unix_stream_connect` lorsque l'identifiant du socket obtenu dans le champ `peer` est identique à l'identifiant du socket qui figure dans l'objet `struct socket *newsock`.

Ainsi, lors de l'interception du retour de l'exécution de cette fonction, la communication entre le processus client et le processus serveur est établie et les identifiants du processus client et du processus serveur sont disponibles. L'identifiant du processus serveur est mémorisé dans le tableau associatif propre à la connexion TCP courante et qui comprend déjà l'identifiant du processus client.

Dans un deuxième exemple de réalisation, basé sur les socket TCP, le prototype de la fonction du noyau d'établissement de la communication est :
`struct sock *inet_csk_accept(struct sock *sk, int flags, int *err)` qui correspond à la fonction appelée dès lors que des services écoutent sur un port TCP pour recevoir des connexions. Cette fonction est bloquante et ne retourne qu'une fois qu'une connexion a été établie avec un processus client. A noter que cette fonction a été interceptée lors d'une demande de connexion d'un client et a permis d'associer un identifiant de processus à l'adresse IP du client. Afin de distinguer la connexion client à travers le réseau des connexions inter processus sur le système, le 4-uplet TCP est examiné. Il est vérifié si les adresses IP source

et destination sont égales à l'adresse « localhost », indiquant qu'il s'agit d'une connexion locale, c'est-à-dire une connexion inter processus sur le même système. En interceptant le retour de cette fonction, on récupère le 4-uplet à partir de la valeur de retour `struct sock *` de la fonction du noyau. L'égalité entre le port source et le port destination permet d'obtenir

5 l'identifiant du processus client à partir des données mémorisées lors de l'interception de l'exécution de la fonction `tcp_connect`. Ainsi l'identifiant du processus serveur est corrélé à l'identifiant du processus client, lui-même associé à l'identifiant de la connexion TCP dans le tableau associatif.

Dans un exemple de réalisation, mis en œuvre lors de l'interception de l'exécution de la

10 fonction du noyau d'établissement d'une communication inter processus au moyen d'un socket Unix ou TCP, une première instance de la structure de données `struct task_struct` associée au processus serveur est mémorisée, en association avec l'identifiant du processus serveur. Cette première instance comprend un premier intervalle de temps, noté Δt_1 , qui correspond au temps CPU consommé par le processus serveur depuis sa création. Cette valeur

15 est destinée à être utilisée pour calculer le temps CPU total consacré au traitement de la connexion TCP, plus précisément afin de calculer le temps CPU consommé par le processus serveur pour traiter la connexion TCP.

Ainsi, que ce soit au moyen de socket Unix ou de socket TCP, le procédé offre une solution pour identifier précisément l'ensemble de processus qui sont impliqués dans chacune

20 des connexions TCP qui émanent d'équipements clients. Un chaînage de processus est ainsi mémorisé dans le tableau associatif et ce chaînage de processus est adapté pour calculer précisément le temps CPU dédié à chacune des connexions TCP à partir des temps CPU que chacun des processus a consommé lors du traitement de la connexion TCP. Par ailleurs, le temps CPU peut être précisé et décliné pour chacun des services obtenu pour chacun des

25 services impliqués dans l'exécution de l'application Web dès lors que l'on segmente les informations relatives aux processus impliqués en fonction des services sous-jacents ;

- dans une sous-étape E105 de l'étape E10 d'interception, l'exécution d'une fonction du noyau représentative de la fermeture d'une connexion inter processus est interceptée.

Dans un premier exemple de réalisation, basé sur les socket Unix, le prototype de la

30 fonction représentative de la fermeture d'une communication inter processus est : `int unix_shutdown(struct socket *sock, int mode)`. L'exécution de cette fonction est interceptée afin de détecter la fin de la communication inter processus. Dans un exemple de réalisation, cela déclenche le calcul de la consommation CPU du processus serveur qui participe à la communication inter processus. L'identifiant du processus serveur est obtenu à

35 partir de l'argument `struct socket *sock` de la fonction `unix_shutdown`.

Dans un deuxième exemple de réalisation, basé sur les socket TCP, le prototype de la fonction représentative de la fermeture d'une communication inter processus est : `void tcp_set_state(struct sock *sk, int state)`. L'identifiant du processus serveur est obtenu à partir de l'argument `struct sock *sk`.

5 Dans un exemple de réalisation, mis en œuvre lors de l'interception de l'exécution de la fonction du noyau représentative de la fermeture de la communication inter processus au moyen d'un socket Unix ou TCP, une deuxième instance de la structure de données `struct task_struct` associée au processus serveur est mémorisée, en association avec l'identifiant du processus serveur. Cette deuxième instance comprend un deuxième intervalle de temps, noté
10 Δt_2 , qui correspond au temps CPU consommé par le processus serveur depuis sa création. Cette valeur est destinée à être utilisée pour calculer le temps CPU total consacré au traitement de la connexion TCP, plus précisément afin de calculer le temps CPU consommé par le processus serveur pour le traitement de la connexion TCP.

Comme précisé précédemment, l'exécution des sous-étapes E101 à E105 commence par
15 une vérification que le processus pour lequel la fonction du noyau dont l'exécution a été interceptée fait partie de la liste de processus supervisés. Pour un processus qui ne fait pas partie de la liste, l'exécution de la sous-étape est interrompue.

Dans l'étape suivante E20 de calcul du temps CPU, mise en œuvre lors de l'interception au cours de l'étape E102 de l'étape E10 d'interception du retour de l'exécution de la fonction du
20 noyau représentative d'une fermeture de la connexion courante, il est procédé au calcul du temps CPU total consommé par l'ensemble des processus impliqués dans le traitement de la connexion TCP. Ce temps correspond à la somme des temps CPU que chacun des processus impliqué dans l'exécution de l'application Web lancée depuis la connexion client a consommé lors du traitement de la connexion TCP.

25 Lors de l'exécution de l'application Web, il a été mémorisé dans le tableau associatif un identifiant d'une connexion TCP courante en association avec l'ensemble des identifiants de processus impliqués dans le traitement de la connexion. Pour chacun des processus impliqués, il a été mémorisé une première instance de la structure de données `struct task_struct` à un instant initial, représentatif d'un début d'activité du processus pour le traitement de la
30 connexion. Cette première instance comprend un premier intervalle de temps, Δt_1 , représentatif du temps CPU consommé par le processus depuis sa création jusqu'à l'instant initial. Il a également été mémorisé une deuxième instance de la structure de données `struct task_struct` à un second instant, représentatif d'une fin de contribution du processus au traitement de la connexion TCP. La deuxième instance comprend un deuxième intervalle de
35 temps, Δt_2 , représentatif du temps CPU consommé par le processus depuis sa création jusqu'au

second instant. Ainsi, le temps Δt , correspondant au temps CPU consommé par le processus pour traiter la connexion TCP est obtenu en calculant la différence entre le deuxième et le premier intervalle de temps. En d'autres termes, $\Delta t = \Delta t_2 - \Delta t_1$. En faisant la somme des temps CPU que chacun des processus a consommé pour traiter la connexion TCP on obtient le

5 temps CPU total du serveur consommé pour le traitement de la connexion.

Dans le cas où un seul service est impliqué dans l'exécution de l'application Web, les premier et deuxième intervalles de temps correspondent respectivement à l'intervalle de temps qui figure dans la structure de donnée `struct task_struct` lors de l'interception du retour de la fonction du noyau appelée pour demander l'ouverture d'une connexion TCP et à

10 l'intervalle de temps qui figure dans la structure de données `struct task_struct` lors de l'interception de l'exécution de la fonction de fermeture de la connexion.

Dans le cas où plusieurs services sont impliqués et que des communications inter processus sont mises en œuvre, les premier et deuxième intervalles pris en compte pour un processus backend correspondent respectivement à l'intervalle de temps qui figure dans la

15 structure de donnée `struct task_struct` du processus lors de l'interception de l'exécution de la fonction du noyau d'établissement de la communication consécutive à l'initiation d'une communication par un autre processus, et à l'intervalle de temps qui figure dans la structure de données `struct task_struct` du processus lors de l'interception de l'exécution de la fonction du noyau de fermeture de la communication.

20 Dans l'exemple de réalisation de l'étape E20 de calcul de temps CPU total décrit ici, le calcul du temps CPU total est mis en œuvre à la fermeture de la connexion TCP, à partir des structures de données `struct task_struct` mémorisées durant l'exécution de l'application Web. Dans un autre exemple de réalisation, le calcul est mis en œuvre au fil de l'eau. Ainsi, les temps CPU consommés par les processus qui participent au traitement de la

25 connexion TCP sont cumulés au fil du temps. Le calcul de la valeur du temps CPU total est finalisé lors de la fermeture de la connexion TCP. Cet exemple de réalisation permet de limiter le stockage des structures de données `struct task_struct`. L'espace mémoire nécessaire à la mise en œuvre du procédé d'aide à la détection d'attaques par dénis de services applicatifs est ainsi optimisé.

30 Dans un exemple de réalisation, un outil ou interface appelé eBPF (pour « extended Berkeley Packet Filter »), destiné à traiter des événements, tels que des appels aux fonctions interceptées directement dans le contexte du noyau du système d'exploitation est utilisé. Cependant cet outil ne permet d'accéder à la structure de données `struct task_struct` qui contient la durée d'activité d'un processus courant que dans des versions récentes du noyau

35 du système d'exploitation Linux. Dans un autre exemple de réalisation, mis en œuvre pour des

versions plus anciennes du noyau, l'exécution de certaines fonctions du noyau du système d'exploitation est interceptée de manière à récupérer la structure de données `struct task_struct` d'un processus qui fait partie de la liste des processus à superviser. Ces fonctions sont :

5 - `struct rq *finish_task_switch(struct task_struct *prev)` ;

cette fonction est appelée après qu'un nouveau processus ait été assigné par l'ordonnanceur (ou « scheduler » en anglais) au CPU. L'argument `struct task_struct *prev` représente le processus précédemment assigné au CPU et la structure de données `struct task_struct` comprend l'identifiant du nouveau processus. Intercepter l'exécution de cette fonction permet
10 de récupérer les structures de données `struct task_struct` correspondant aux processus supervisés au fur et à mesure que leur exécution est arrêtée. Lors de l'interception de l'exécution de cette fonction, l'adresse de la structure de données `struct task_struct` est mémorisée en association avec l'identifiant du processus à laquelle elle est associée. Cependant, l'interception de l'exécution de cette fonction implique qu'un processus soit arrêté avant de
15 pouvoir calculer sa consommation de temps CPU. Il y a donc, au lancement de l'outil de supervision, un laps de temps pendant lequel la consommation CPU associée à une connexion TCP ne peut être observée. Ce laps de temps s'applique aussi à tout nouveau processus jusqu'à ce qu'un autre processus soit pris en charge par le CPU. Pour pallier ce problème, l'exécution de la fonction du noyau suivante est interceptée, afin d'obtenir la structure de données `struct task_struct` d'un processus dès sa création :

20 - `void wake_up_new_task(struct task_struct *p)` ; cette fonction est appelée lorsqu'un nouveau processus est créé à l'aide d'appels système. L'argument `struct task_struct *p` fournit l'adresse en mémoire de l'objet `struct task_struct` associé au nouveau processus. Il est alors mémorisé une association entre l'identifiant du nouveau
25 processus et l'adresse en mémoire de l'objet `struct task_struct`.

Le procédé décrit ici repose sur des communications inter processus basées sur des sockets et ne tient pas compte de communications inter processus basées sur une mémoire partagée, (« shared memory Inter Process Communication » en anglais). Dans des communications basées sur une mémoire partagée, le noyau du système d'exploitation réserve
30 un espace mémoire que les processus utilisent ensuite pour communiquer, sans passer par le noyau. On considère que ce cas nécessite de définir un protocole très bas niveau au niveau de l'application Web et complexifie également pour un programme de supervision l'obtention d'informations sur les temps CPU des processus. Le procédé décrit ici s'applique à des applications web pour lesquelles les communications inter processus reposent sur une interface
35 standardisée, en l'espèce les communications inter processus basées sur les sockets.

Le procédé tel que décrit précédemment suppose qu'un processus ne traite qu'une seule connexion TCP à la fois. Il est cependant possible que des processus d'un serveur aient un fonctionnement asynchrone : chaque processus s'occupe alors de plusieurs connexions TCP en même temps. Ce mode de fonctionnement est parfois utilisé pour des processus frontend qui se contentent de transmettre des requêtes à des processus backend. Ainsi, plutôt que d'attendre qu'un processus backend termine le traitement d'une connexion TCP, le processus frontend continue de transmettre des requêtes liées à une connexion TCP à d'autres processus backend. Ainsi, un petit nombre de processus frontend est nécessaire pour dispatcher un grand nombre de requêtes à des processus backend. Dans un tel fonctionnement asynchrone, le procédé décrit ici néglige la consommation de temps CPU induite par les processus frontend. En effet, parce que les processus frontend se contentent de dispatcher les requêtes TCP au processus backend, on estime qu'ils consomment très peu de temps CPU pour traiter une connexion distante. Négliger ce temps n'impacte donc pas le temps total de consommation CPU de manière significative. Dans un exemple de réalisation, un traitement asynchrone des connexions par un serveur peut être détecté par l'outil dans une phase de pré-analyse. En effet, un processus traitant des connexions TCP de façon asynchrone reçoit de nouvelles connexions avant d'avoir fermé les précédentes. Lors de l'interception de l'exécution de fonctions du noyau, un processus asynchrone fait donc des appels système de type `accept` sans qu'il y ait forcément un appel système de type `close`. La phase de pré-analyse est agencée pour détecter qu'au moins deux appels système `accept` sont mis en œuvre, sans qu'un appel système de type `close` soit détecté.

Le procédé d'aide à la détection d'attaques par dénis de services applicatifs tel que décrit précédemment peut facilement s'intégrer à des procédés de détection d'attaques. De tels procédés reposent en général sur des métriques particulières que l'on ne détaillera pas ici. On peut citer cependant que le procédé d'aide à la détection permet d'identifier des connexions de certains équipements client pour lesquelles le traitement par le serveur est anormalement long par rapport à des requêtes équivalentes en provenance d'autres équipements client. Il offre également une aide à l'analyse en permettant d'identifier quels processus et/ou quel(s) service(s) sont à l'origine d'une consommation CPU anormale.

Pour des raisons de clarté, on a décrit ici le cas d'une connexion TCP établie à partir d'un équipement client. Bien sûr, le procédé est prévu pour gérer en parallèle une pluralité de connexions TCP entrantes et les moyens décrits permettent de gérer pour chacune des connexions les informations relatives au processus impliqués et de calculer pour chacune des connexions TCP le temps CPU consacré au traitement de cette connexion.

Un serveur 20, apte à mettre en œuvre les étapes du procédé d'aide à la détection d'attaques par dénis de services applicatifs, selon un exemple de réalisation, va maintenant être décrit en relation avec la figure 2.

Le serveur 20 est un équipement informatique tel un ordinateur. Il comprend :

5 - une unité de traitement ou processeur 201, ou CPU, destinée à charger des instructions en mémoire, à les exécuter, à effectuer des opérations ;

 - un ensemble de mémoires, dont une mémoire volatile 202, ou « RAM » (pour « Random Access Memory »), utilisée pour exécuter des instructions de code, stocker des variables, etc., et une mémoire de stockage 203 de type « EEPROM » (de l'anglais
10 « Electrically Erasable Programmable Read Only Memory »). En particulier, la mémoire de stockage 203 est agencée pour mémoriser un premier programme, comprenant des instructions de code pour mettre en œuvre une application, telle qu'une application Web, et un deuxième programme qui comprend des instructions de code pour mettre en œuvre les étapes du procédé d'aide à la détection d'attaques par dénis de services tel que décrit précédemment ;

15 Le serveur 20 comprend également :

 - un premier module 204 d'interception et d'obtention, agencé pour intercepter l'exécution d'une fonction du noyau du système d'exploitation représentative de la réception d'une demande de connexion à l'application en provenance d'un équipement client, et pour associer un identifiant de ladite connexion à l'identifiant d'un processus, ledit processus étant en
20 charge du traitement de ladite connexion. Le premier module 204 d'interception et d'obtention est également agencé pour obtenir un premier intervalle de temps représentatif d'un temps CPU consommé par ledit processus depuis sa création. Le module 204 d'interception et d'obtention est agencé pour mettre en œuvre l'étape E101 du procédé d'aide à la détection d'attaques par dénis de services applicatifs ;

25 - un deuxième module 205 d'interception et d'obtention, agencé pour intercepter l'exécution d'une deuxième fonction du noyau du système d'exploitation représentative d'une fermeture de ladite connexion. Le deuxième module 205 d'interception et d'obtention est également agencé pour obtenir un deuxième intervalle de temps représentatif d'un temps CPU consommé par ledit processus depuis sa création. Le deuxième module 205 d'interception et
30 d'obtention est également agencé pour obtenir un premier intervalle de temps représentatif d'un temps CPU consommé par ledit processus depuis sa création. Le deuxième module 205 d'interception et d'obtention est agencé pour mettre en œuvre l'étape E102 du procédé d'aide à la détection d'attaques par dénis de services applicatifs ;

 - un module de calcul 206, agencé pour calculer un temps CPU total consommé pour le
35 traitement de ladite connexion à partir d'au moins le deuxième et le premier intervalle de temps

CPU consommé par ledit processus. Le module de calcul 206 est agencé pour mettre en œuvre l'étape E20 du procédé décrit précédemment.

Le premier module 204 d'interception et d'obtention, le deuxième module 205 d'interception et d'obtention et le module de calcul 206 sont de préférence des modules
5 logiciels comprenant des instructions logicielles pour mettre en œuvre les étapes du procédé d'aide à la détection d'attaques par dénis de services applicatifs qui sont mises en œuvre par le serveur 20.

L'invention concerne donc aussi :

- un programme d'ordinateur comportant des instructions pour la mise en œuvre du
10 procédé d'aide à la détection d'attaques par dénis de services applicatifs tel que décrit précédemment lorsque ce programme est exécuté par un processeur du serveur, et

- un support d'enregistrement lisible sur lequel est enregistré le programme d'ordinateur décrit ci-dessus.

REVENDICATIONS

1. Procédé d'aide à la détection d'attaques par dénis de service applicatifs dans un système proposant au moins une application, ladite application utilisant au moins un service, le
5 procédé comprenant :

- interception (E101) de l'exécution d'une fonction du noyau du système d'exploitation représentative de la réception d'une demande de connexion à l'application en provenance d'un équipement client, association d'un identifiant de ladite connexion à l'identifiant d'au moins un processus, ledit processus étant en charge du traitement de ladite connexion sur un premier
10 service, et obtention d'un premier intervalle de temps représentatif d'un temps CPU consommé par ledit processus depuis sa création,

- interception (E102) de l'exécution d'une deuxième fonction du noyau du système d'exploitation représentative d'une fermeture de ladite connexion, et obtention d'un deuxième intervalle de temps représentatif d'un temps CPU consommé par ledit processus depuis sa
15 création,

- calcul (E20) d'un temps CPU total consommé pour le traitement de ladite connexion à partir d'au moins le deuxième et le premier intervalle de temps CPU consommé par ledit processus.

20 2. Procédé selon la revendication 1 comprenant, lorsque ladite application utilise au moins deux services et qu'un premier processus d'un premier des au moins deux services établit une communication avec un deuxième processus d'un deuxième des au moins deux services, un identifiant dudit premier processus étant associé à l'identifiant de ladite connexion :

- interception (E103) de l'exécution d'une fonction du noyau du système d'exploitation représentative d'une demande d'établissement d'une communication entre le premier processus
25 et le deuxième processus,

- interception (E104) de l'exécution d'une fonction du noyau du système d'exploitation représentative de l'établissement de la communication entre le premier et le deuxième processus, association d'un identifiant du deuxième processus à l'identifiant de ladite
30 connexion, et obtention d'un deuxième premier intervalle de temps représentatif du temps CPU consommé par le deuxième processus depuis sa création,

- interception (E105) de l'exécution d'une fonction du noyau du système d'exploitation représentative de la fermeture de la communication entre le premier processus et le deuxième processus, et obtention d'un deuxième intervalle de temps représentatif du temps CPU
35 consommé par le deuxième processus depuis sa création,

- calcul (E20) d'un deuxième temps CPU de traitement associé à ladite communication à partir du deuxième et du premier deuxième intervalle de temps,

- prise en compte du deuxième temps CPU pour le calcul du temps CPU total consommé pour le traitement de ladite connexion.

5

3. Procédé selon la revendication 2, dans lequel l'identifiant du deuxième processus est associé à l'identifiant de ladite connexion lorsqu'un premier identifiant de la communication entre le premier et le deuxième processus, ledit premier identifiant étant obtenu dans une structure de données associée au premier processus est identique à un deuxième identifiant de la communication entre le premier et le deuxième processus, ledit deuxième identifiant étant obtenu dans une structure de données associée au deuxième processus.

4. Procédé selon l'une des revendications précédentes, comprenant, dans une étape de configuration initiale une fourniture d'une liste de processus à superviser pour ladite application, et

15

- interception (E02) de l'exécution d'une fonction du noyau du système d'exploitation représentative de la création d'un processus fils par un processus parent,

- lorsque le processus parent appartient à la liste des processus à superviser, obtention de l'identifiant dudit processus fils et ajout dudit processus fils à la liste des processus à superviser.

20

5. Procédé selon la revendication précédente, comprenant :

- interception (E03) de l'exécution d'une fonction du noyau du système d'exploitation représentative de l'arrêt d'un processus existant,

- lorsque le processus existant appartient à la liste des processus à superviser, obtention de l'identifiant dudit processus existant et suppression dudit processus existant dans la liste des processus à superviser.

25

6. Serveur (20) apte à héberger au moins une application, ladite application utilisant au moins un service, ledit serveur comprenant :

30

- des premiers moyens (204) d'interception et d'obtention, agencés pour intercepter l'exécution d'une fonction du noyau du système d'exploitation représentative de la réception d'une demande de connexion à l'application en provenance d'un équipement client, associer un identifiant de ladite connexion à l'identifiant d'au moins un processus, ledit processus étant en

charge du traitement de ladite connexion, et pour obtenir un premier intervalle de temps représentatif d'un temps CPU consommé par ledit processus depuis sa création,

- des deuxièmes moyens (205) d'interception et d'obtention, agencés pour intercepter l'exécution d'une deuxième fonction du noyau du système d'exploitation représentative d'une
5 fermeture de ladite connexion, et pour obtenir un deuxième intervalle de temps représentatif d'un temps CPU consommé par ledit processus depuis sa création,

- des moyens de calcul (206), agencés pour calculer un temps CPU total consommé pour le traitement de ladite connexion à partir d'au moins le deuxième et le premier intervalle de temps CPU consommé par ledit processus.

10

7. Programme pour un serveur apte à héberger au moins une application utilisant au moins un service, ledit programme comprenant des instructions de code de programme destinées à commander l'exécution des étapes du d'aide à la détection d'attaques par dénis de service applicatifs, selon l'une des revendications 1 à 5, lorsque le programme est exécuté sur
15 ledit serveur.

8. Support de données dans lequel est enregistré le programme selon la revendication 7.

20

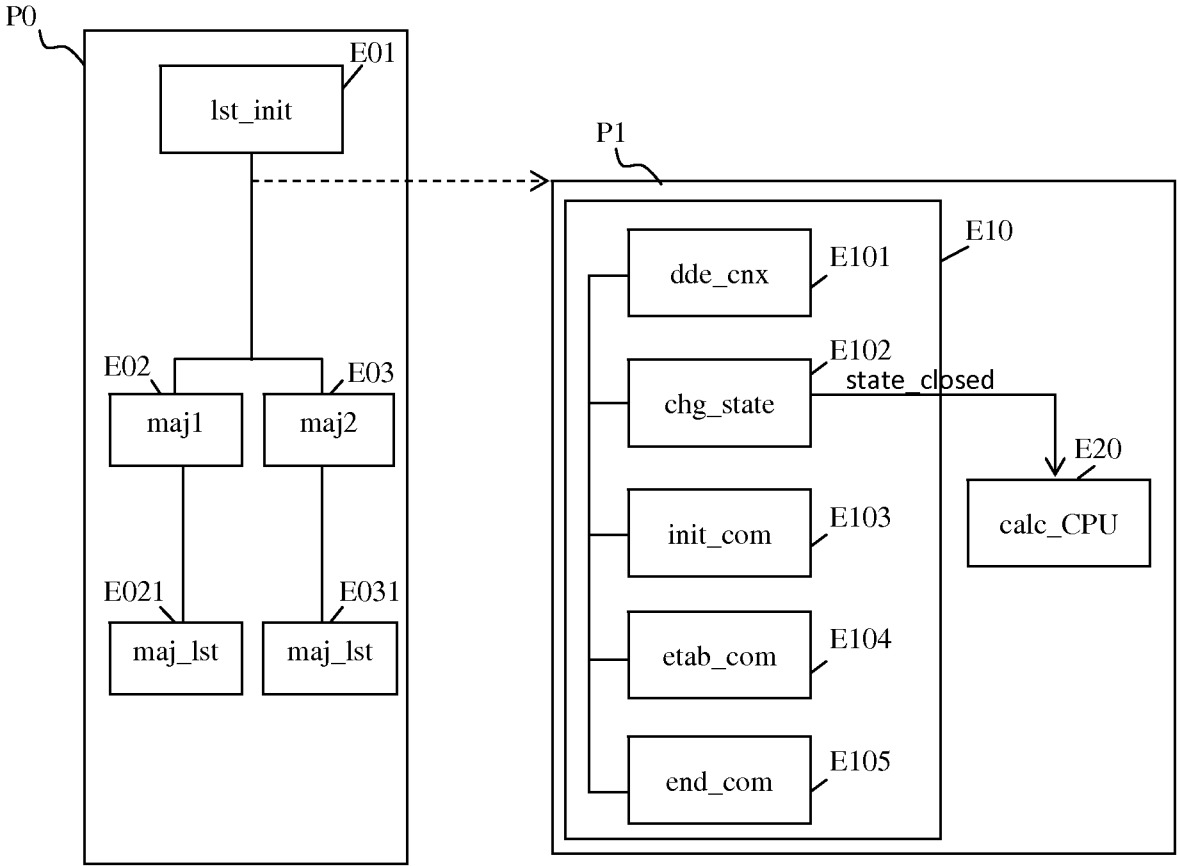


Figure 1

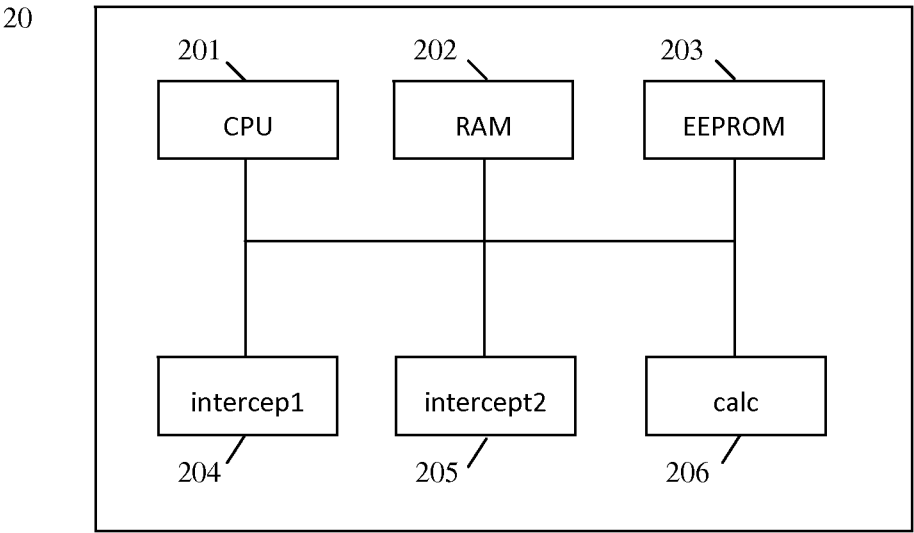


Figure 2

INTERNATIONAL SEARCH REPORT

International application No
PCT/FR2018/050714

A. CLASSIFICATION OF SUBJECT MATTER INV. G06F21/55 H04L29/06 H04W12/12 H04L29/08 H04L12/801 H04L12/851 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) H04L G06F H04W 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 <table border="1"> <thead> <tr> <th>Category*</th> <th>Citation of document, with indication, where appropriate, of the relevant passages</th> <th>Relevant to claim No.</th> </tr> </thead> <tbody> <tr> <td>X</td> <td>EP 1 788 752 A1 (ALCATEL LUCENT [FR]) 23 May 2007 (2007-05-23) paragraph [0020] - paragraph [0026] figure 1</td> <td>1-8</td> </tr> <tr> <td>A</td> <td>----- US 2014/304798 A1 (IYENGAR MEGHASHREE [IN] ET AL) 9 October 2014 (2014-10-09) figure 7B</td> <td>1-8</td> </tr> <tr> <td>A</td> <td>----- WO 2015/145210 A1 (TELEFONAKTIEBOLAGET ERICSSON L M) 1 October 2015 (2015-10-01) paragraph [0053] - paragraph [0056] figure 4 ----- -/-</td> <td>1-8</td> </tr> </tbody> </table>			Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.	X	EP 1 788 752 A1 (ALCATEL LUCENT [FR]) 23 May 2007 (2007-05-23) paragraph [0020] - paragraph [0026] figure 1	1-8	A	----- US 2014/304798 A1 (IYENGAR MEGHASHREE [IN] ET AL) 9 October 2014 (2014-10-09) figure 7B	1-8	A	----- WO 2015/145210 A1 (TELEFONAKTIEBOLAGET ERICSSON L M) 1 October 2015 (2015-10-01) paragraph [0053] - paragraph [0056] figure 4 ----- -/-	1-8
Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.												
X	EP 1 788 752 A1 (ALCATEL LUCENT [FR]) 23 May 2007 (2007-05-23) paragraph [0020] - paragraph [0026] figure 1	1-8												
A	----- US 2014/304798 A1 (IYENGAR MEGHASHREE [IN] ET AL) 9 October 2014 (2014-10-09) figure 7B	1-8												
A	----- WO 2015/145210 A1 (TELEFONAKTIEBOLAGET ERICSSON L M) 1 October 2015 (2015-10-01) paragraph [0053] - paragraph [0056] figure 4 ----- -/-	1-8												
☒ Further documents are listed in the continuation of Box C. ☒ 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 4 May 2018		Date of mailing of the international search report 15/05/2018												
Name and mailing address of the ISA/ European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Fax: (+31-70) 340-3016		Authorized officer Bae, Jun-Young												

INTERNATIONAL SEARCH REPORT

International application No

PCT/FR2018/050714

C(Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT		
Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	Vilhelm Gray: "linux - How do I get the total CPU usage of an application from /proc/pid/stat? - Stack Overflow", 24 May 2013 (2013-05-24), pages 1-4, XP055413055, Retrieved from the Internet: URL:https://stackoverflow.com/questions/16726779/how-do-i-get-the-total-cpu-usage-of-an-application-from-proc-pid-stat [retrieved on 2017-10-05] page 1 - page 2 -----	1-8

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No

PCT/FR2018/050714

Patent document cited in search report		Publication date		Patent family member(s)		Publication date
EP 1788752	A1	23-05-2007	CN	1972248 A		30-05-2007
			EP	1788752 A1		23-05-2007
			US	2007127491 A1		07-06-2007

US 2014304798	A1	09-10-2014	US	2014304798 A1		09-10-2014
			US	2015281272 A1		01-10-2015

WO 2015145210	A1	01-10-2015	EP	3123685 A1		01-02-2017
			US	2017118242 A1		27-04-2017
			WO	2015145210 A1		01-10-2015

RAPPORT DE RECHERCHE INTERNATIONALE

Demande internationale n°

PCT/FR2018/050714

A. CLASSEMENT DE L'OBJET DE LA DEMANDE INV. G06F21/55 H04L29/06 H04W12/12 H04L29/08 H04L12/801 H04L12/851 ADD.		
Selon la classification internationale des brevets (CIB) ou à la fois selon la classification nationale et la CIB		
B. DOMAINES SUR LESQUELS LA RECHERCHE A PORTE Documentation minimale consultée (système de classification suivi des symboles de classement) H04L G06F H04W		
Documentation consultée autre que la documentation minimale dans la mesure où ces documents relèvent des domaines sur lesquels a porté la recherche		
Base de données électronique consultée au cours de la recherche internationale (nom de la base de données, et si cela est réalisable, termes de recherche utilisés) EPO-Internal, WPI Data		
C. DOCUMENTS CONSIDERES COMME PERTINENTS		
Catégorie*	Identification des documents cités, avec, le cas échéant, l'indication des passages pertinents	no. des revendications visées
X	EP 1 788 752 A1 (ALCATEL LUCENT [FR]) 23 mai 2007 (2007-05-23) alinéa [0020] - alinéa [0026] figure 1	1-8
A	----- US 2014/304798 A1 (IYENGAR MEGHASHREE [IN] ET AL) 9 octobre 2014 (2014-10-09) figure 7B	1-8
A	----- WO 2015/145210 A1 (TELEFONAKTIEBOLAGET ERICSSON L M) 1 octobre 2015 (2015-10-01) alinéa [0053] - alinéa [0056] figure 4 ----- -/-	1-8
☒ Voir la suite du cadre C pour la fin de la liste des documents ☒ Les documents de familles de brevets sont indiqués en annexe		
* Catégories spéciales de documents cités:		
"A" document définissant l'état général de la technique, non considéré comme particulièrement pertinent "E" document antérieur, mais publié à la date de dépôt international ou après cette date "L" document pouvant jeter un doute sur une revendication de priorité ou cité pour déterminer la date de publication d'une autre citation ou pour une raison spéciale (telle qu'indiquée) "O" document se référant à une divulgation orale, à un usage, à une exposition ou tous autres moyens "P" document publié avant la date de dépôt international, mais postérieurement à la date de priorité revendiquée "T" document ultérieur publié après la date de dépôt international ou la date de priorité et n'appartenant pas à l'état de la technique pertinent, mais cité pour comprendre le principe ou la théorie constituant la base de l'invention "X" document particulièrement pertinent; l'invention revendiquée ne peut être considérée comme nouvelle ou comme impliquant une activité inventive par rapport au document considéré isolément "Y" document particulièrement pertinent; l'invention revendiquée ne peut être considérée comme impliquant une activité inventive lorsque le document est associé à un ou plusieurs autres documents de même nature, cette combinaison étant évidente pour une personne du métier "&" document qui fait partie de la même famille de brevets		
Date à laquelle la recherche internationale a été effectivement achevée 4 mai 2018		Date d'expédition du présent rapport de recherche internationale 15/05/2018
Nom et adresse postale de l'administration chargée de la recherche internationale Office Européen des Brevets, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Fax: (+31-70) 340-3016		Fonctionnaire autorisé Bae, Jun-Young

C(suite). DOCUMENTS CONSIDERES COMME PERTINENTS		
Catégorie*	Identification des documents cités, avec, le cas échéant, l'indication des passages pertinents	no. des revendications visées
A	Vilhelm Gray: "linux - How do I get the total CPU usage of an application from /proc/pid/stat? - Stack Overflow", 24 mai 2013 (2013-05-24), pages 1-4, XP055413055, Extrait de l'Internet: URL:https://stackoverflow.com/questions/16726779/how-do-i-get-the-total-cpu-usage-of-an-application-from-proc-pid-stat [extrait le 2017-10-05] page 1 - page 2 -----	1-8

RAPPORT DE RECHERCHE INTERNATIONALE

Renseignements relatifs aux membres de familles de brevets

Demande internationale n°

PCT/FR2018/050714

Document brevet cité au rapport de recherche		Date de publication	Membre(s) de la famille de brevet(s)	Date de publication
EP 1788752	A1	23-05-2007	CN 1972248 A	30-05-2007
			EP 1788752 A1	23-05-2007
			US 2007127491 A1	07-06-2007

US 2014304798	A1	09-10-2014	US 2014304798 A1	09-10-2014
			US 2015281272 A1	01-10-2015

WO 2015145210	A1	01-10-2015	EP 3123685 A1	01-02-2017
			US 2017118242 A1	27-04-2017
			WO 2015145210 A1	01-10-2015
