European Journal of Computer Science and Information Technology
Vol.11, No.2, pp.70-99, 2023
Print ISSN: 2054-0957 (Print),
Online ISSN: 2054-0965 (Online)
Website: https://www.eajournals.org/
Publication of the European Centre for Research Training and Development -UK

Approche opérationnelle de la protection du système noyau sous Windows Server 2019 : Optimisation, QoS et Performance.

[Operational approach to kernel system protection under Windows Server 2019: Optimization, QoS and Performance].

[1]Dr YENDE RAPHAEL Grevisse, [2]LOWEMBO A TSHOTSHO Raymond Albert, [3]KENA MULUMBA Clovis, [4]KABANGU KANGA Chappelle, [5]ILUNGA KIFEMBE Corneille, [6]KABADA SESWA David-Jackson, [7]KAMBALE SIWAYITIRA Elisée, [8]ILUNGA KALALA Désiré.

[1]*Département d'Informatique de l'Université de Bas-Uélé (UNIBAS), Buta (RDC). Email : grevisse29@gmail.com.*
[2]*Département des Nouvelles Technologies et IT associé chez HCR/Kananga, Kananga, Kasaï Central (RDC).*
[3]*Département d'Informatique de l'Université Officielle de Mbuji-Mayi (UOM), Mbuji-Mayi, Kasaï Oriental (RDC).*
[4]*Département d'Informatique de l'Institut Supérieur des Arts et Métiers de Mbuji-Mayi (ISAM), Kasaï Oriental (RDC).*
[5]*Département d'Informatique de l'Institut Supérieur Technique d'Informatique Appliquée de Mbuji-Mayi, Kasaï Oriental (RDC).*
[6,7]*Département d'Informatique de l'Institut Supérieur de Commerce de Goma (ISC-Goma), Goma, Nord-Kivu, (RDC).*
[8]*Département d'Informatique de l'Université de Kananga (UNIKAN), Kananga, Kasaï Central (RDC).*

doi: https://doi.org/10.37745/ejcsit.2013/vol11n27099 Published June 4, 2023

Citation : Grevisse Y.R.,Raymond Albert L.A.T., Clovis K.M., Chappelle K.K., Corneille I.K., David-Jackson K.S., Elisée K.S., Désiré I.K.. (2023) Operational approach to kernl system protection under Windows Server 2019 : Optimization, QoS and Performance, *European Journal of Computer Science and Information Technology*, Vol.11, No.2, pp.70-99,

RESUME : *L'informatique est devenue de nos jours, le point culminant de toute activité humaine mais elle représente également la pire crainte qu'aucune épidémie n'a inspirée de nos jours. Et malgré cela, tout le monde concède que l'utilisation des ordinateurs (en particulier au travers du réseau Internet) occupe désormais la première place voire indispensable dans la vie quotidienne. Chacun d'entre nous utilise un ordinateur pour travailler, pour échanger des informations, pour faire des achats, etc. Malheureusement, les activités malveillantes visant les ordinateurs se multiplient régulièrement et essaient d'exploiter des vulnérabilités qui sont de plus en plus nombreuses en raison de la complexité toujours croissante. Eu égard, la présente recherche s'est fixée l'objectif de raccommoder l'adéquation (optimisation, dynamique et performance) des systèmes d'exploitation à leurs divers environnements de déploiement en émancipant des*

European Journal of Computer Science and Information Technology
Vol.11, No.2, pp.70-99, 2023
Print ISSN: 2054-0957 (Print),
Online ISSN: 2054-0965 (Online)
Website: https://www.eajournals.org/
Publication of the European Centre for Research Training and Development -UK

approches à priori, généralement mises en défaut dans leur capacité à surpasser les besoins futurs spécialement pour la correction des failles de sécurité, en se focalisant sur les fonctionnalités de l'environnement matériel.

MOTS-CLEFS : approche, opérationnel, déploiement, sécurité, système, noyau, windows server 2019, optimisation, dynamique, QoS, Performance, etc.

ABSTRACT *Computer Sciences has become the culmination of all human activity these days, but it is also the worst fear that no epidemic has inspired today. And despite this, everyone concedes that the use of computers (especially through the Internet) now occupies the first place, even essential, in everyday life. Each of us uses a computer to work, to exchange information, to make purchases, etc. Unfortunately, malicious activity targeting computers is steadily increasing and trying to exploit vulnerabilities that are growing in number with ever-increasing complexity. In view of this, the present research has set itself the objective of mending the adequacy (optimization, dynamics and performance) of operating systems to their various deployment environments by emancipating a priori approaches, generally lacking in their capacity. to surpass future needs especially for the correction of security vulnerabilities, focusing on the functionalities of the hardware environment.*

KEYWORDS*:* approach, operational, deployment, security, system, kernel, windows server 2019, optimization, dynamics, QoS, performance

INTRODUCTION

De nos jours, Nul n'ignore l'importance du système d'exploitation qu'il soit en mode isolé (*non connecté*) ou en mode réseau (*connecté*). Toutefois, nombre de personnes mésestiment que ces systèmes dit « *d'exploitation* » sont fragiles dans un environnement aussi en perpétuelle croissance et très turbulent. Divers problèmes apparaissent en cours de route comme les crises administratives et de gestion, vue les besoins de plus en plus grandissants d'informations par le biais des nouvelles technologies de l'information et de la communication [**6]**. Dans un contexte où les technologies de communication et le multimédia évoluent à grande vitesse, et/ou une part importante des systèmes informatiques ne remplissent pas leurs cahiers des charges lors de leurs livraisons, il est nécessaire de concevoir des systèmes évolutifs sous peine de les voir rapidement devenir obsolètes [**21]**. Malheureusement, la poursuite effrénée des nouvelles fonctionnalités est souvent engagée au détriment d'une solution plus adaptable et extensible, et donc plus pérenne [**11]**.

D'ores et déjà, l'on est certain que la démocratisation de l'univers informatique par l'arrivée massive des produits de plus en plus sophistiqués a bouleversé sans aucun doute le mode de fonctionnement traditionnel de toute activité humaine. Par ailleurs, au centre de cette révolution se trouve la multiplicité des logiciels sur une large gamme de matériels informatiques rendant ainsi l'évolution plus contingente comme une transformation des civilisations entières bien que le principe de base restant inchangé [**3]**. En effet, le parallèle

entre évolution biologique et informatique n'est pas totalement fortuit. L'évolution des systèmes d'exploitation, tout comme l'évolution biologique, se manifeste par une accumulation de petites modifications. De plus, selon Lehman [**20]**, un système d'exploitation utilisé dans un environnement réel doit nécessairement évoluer en termes de toutes descentes et de sûreté, faute de quoi, il devient de moins en moins indispensable dans cet environnement et tend à disparaître.

L'accomplissement de systèmes d'exploitation évolutifs est depuis longtemps une question radicale en informatique. La diffusion croissante des produits informatiques auprès du grand public exaspère les utilisateurs vis-à-vis de l'ambiguïté sécuritaire et condescendante de la grande quantité des informations disponibles. Par exemple, on constate qu'une distribution Windows ou Linux est avant tout préférée pour son système de diffusion et d'installation, et la fréquence de ses mises à jour. Nonobstant, de nombreux développeurs indifférent la componction de cette ambiguïté sécuritaire et condescendante et se contentent, comme seul moyen d'adapter et d'étendre les systèmes, de diffuser des versions modifiées des logiciels. Les utilisateurs sont alors en charge de l'application des mises à jour, ce qui impose des coûts excessifs, le répit fortuit et le raccommodage des systèmes en cours d'exécution. Cette pratique de propagation des renouvellements n'est pas simplement un désagrément pour les utilisateurs, mais trahit l'absence de prise en compte de la problématique de l'évolution des systèmes d'exploitation [**21]**.

En plus d'imposer des contraintes à l'utilisateur, la négligence déplorable des développeurs contribue à la prolifération des virus informatiques. En effet, contrairement aux développeurs, les pirates informatiques ont bien compris les enjeux posés par l'adaptabilité des systèmes. Ils s'en servent régulièrement pour propager massivement des virus exploitant des failles de sécurité connues sachant pertinemment que le délai entre la découverte d'une faille et sa correction éventuelle sur le parc informatique mondial leurs laisse tout loisir d'opérer à l'insu des utilisateurs et des développeurs [**22]**. Cette complexité fragilise les utilisateurs vis-à-vis de la sécurité et de tout éventuel mécanisme de protection. En effet, il devient quasiment impossible de les vérifier parfaitement, c'est-à-dire de garantir d'une part l'absence de bogues, et d'autre part l'absence de fonctions cachées. Par conséquent, des pirates profitent inlassablement de cette situation pour réussir à pénétrer dans ces systèmes, et d'en utiliser les ressources associées [**31]**. Pire, est de constater que ces pirates ont même la préséance de s'attaquer aux applications installées sur le système mais aussi au système d'exploitation lui-même et en particulier, à son noyau. Corrompre le noyau d'un système d'exploitation est particulièrement intéressant du point de vue d'un attaquant parce que cela signifie corrompre potentiellement tous les logiciels qui s'exécutent au- dessus de ce noyau. Cela peut même aller jusqu'à la prise de contrôle complète du système par l'attaquant [**31]**.

[**2]** Le noyau d'un système d'exploitation doit donc être fortement protégé. Mais, protéger un noyau de façon efficace par des mécanismes de sécurité est particulièrement délicat car il est extrêmement difficile de rendre ces mécanismes incontournables. Inversement à ce que pourrait laisser penser cet état de fait, la conception de systèmes d'exploitation extensibles a récemment fait l'objet d'importantes recherches ayant abouties à des solutions originales. Néanmoins, ces approches sont difficilement applicables aux applications système puisque la qualité des codes de ces systèmes est l'aboutissement de nombreux remaniements et corrections et

ces systèmes sont soumis à de fortes contraintes de performance [**5]**. Leur refactorisation impliquerait une dégradation de leur qualité et de leur performance pour un résultat dont l'adaptabilité reste à démontrer.

En effet, il est difficile voire impossible d'anticiper quelles seront les évolutions futures, les interfaces permettant l'extensibilité d'un système se révèlent souvent inadaptées si ce n'est inutiles face aux réels besoins d'évolution des utilisateurs et des entreprises [**2]**. À l'inverse, les travaux sur l'adaptation par la transformation à la volée des applications se confrontent mieux aux contraintes de performance des applications systèmes modernes [**8]**. En n'imposant pas de refactorisation constante, ces approches maintiennent la qualité des systèmes, tout en permettant l'adaptation fine et performante des applications. Ces travaux se bornent néanmoins à des outils de maniement des codes à la volée, dès lors, ils demeurent réservés à une utilisation expérimentée. Ainsi, par manque de support langage, les adaptations par transmutation de programme sont fréquemment complexes à concevoir et malaisément réutilisables [**10]**.

Vue sous cet angle, la présente recherche focalisera toute son application sur une sécurité opérationnelle en se basant sur les vulnérabilités existantes et fortuites pouvant se manifester pendant et après les manœuvres des administrateurs-systèmes. En admettant, l'hypothèse selon laquelle chaque système d'exploitation renferme une multitude des failles de sécurité bien plus nombreuses dans les pilotes des périphériques que dans le reste du noyau lui-même ou encore exécutés incontinent grâce à la participation des logiciels applicatifs possiblement implantés et intégrés dans le noyau sous un mode plus privilégié que les entités qu'ils protègent [**31]**. Ainsi, Pour protéger efficacement un noyau contre ces exécutions des codes malveillants, il semble évident de le faire depuis un mode plus privilégié que le noyau lui-même et de façon inéluctable (c'est-à-dire par le noyau lui-même, les applications et/ou les périphériques) [**12]**. La présente recherche préconise un objectif fonctionnel de concilier l'adéquation (optimisation, dynamique et performance) des systèmes d'exploitation à leurs divers environnements de déploiement en émancipant des approches à priori, généralement mises en défaut dans leur capacité à surpasser les besoins futurs spécialement pour la correction des failles de sécurité. Ainsi, pour y parvenir, il nous paraît alors nécessaire de focaliser notre saillante réflexion sur les fonctionnalités de l'environnement matériel, pour deux principales raisons : D'une part, le matériel constitue la base essentielle au fonctionnement de tout système informatique, c'est-à-dire, un mandataire dont tous les logiciels reposent intégralement sur leurs bases pour leur exécution. D'autre part, les composants matériels sont plus difficiles à corrompre que les composants logiciels. Leur corruption implique le plus souvent un accès physique au système informatique, dont le problème relève de la sécurité physique et organisationnelle.

IDEMARCHE METHODOLOGIQUE

[**32]** La présente recherche préconise l'approche holistique qui ne considère pas seulement la problématique sécuritaire du système noyau comme un simple énoncé scientifique des faits isolés, mais plutôt comme un ensemble du système qui mérite d'affronter le verdict de l'expérience des entreprises et des utilisateurs. Cette approche est soutenue promptement par la méthode analytique, exploratoire et systémique qui nous ont permis d'étudier rationnellement les grandeurs manipulables pouvant influer dans la mauvaise gestion des systèmes d'exploitation, plus singulièrement leur noyau, actuellement au centre des activités de toute entreprise

émergente en schématisant un ensemble d'éléments complexes en relation dans la gestion de la performance, de la dynamique et de l'optimisation afin d'aboutir à un cadre de modélisation rénovée, simplifiée et efficace ; et qui garantissent les mises à jours extensibles et accommodées. La technique de sondage et celle de l'observation ont été les moyens par excellence qui nous ont permis de collecter les informations indispensables concourant à l'élaboration de ladite recherche. Signalons également, que la technique de la documentation a également suppléé aux techniques précédemment supputées.

REGARD SUR LE SYSTEME D'EXPLOITATION

Contexte de l'étude

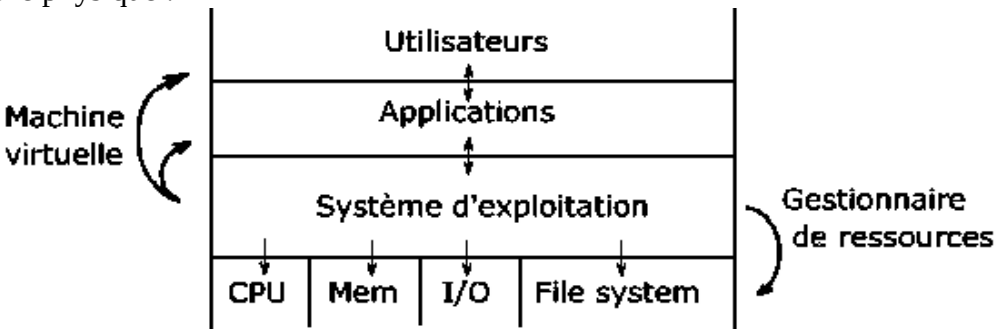
Au tout début de l'informatique, les logiciels n'étaient pas compatibles sur une large gamme de matériel. Avec le temps, les informaticiens ont inventé des techniques d'abstraction matérielle qui permettent à un programme de s'exécuter sur des ordinateurs avec des matériels très différents [**16]**. Ainsi, rendre les programmes indépendants du matériel a demandé de revoir l'organisation des logiciels, qui ont dû être découpés en plusieurs programmes séparés tels [**29]** :

- Les programmes systèmes gèrent la mémoire et les périphériques ;
- Les programmes applicatifs ou applications délèguent la gestion de la mémoire et des périphériques aux programmes systèmes ;
- Etc.

Pour simplifier, l'ensemble des programmes système porte le nom générique *de systèmes d'exploitation*. Un système d'exploitation (*OS : Operating System, en Anglais*) est un programme (logiciel) qui joue un rôle d'interface entre le matériel et l'utilisateur, en consentant à la présentation d'une machine virtuelle et le partage (Gestionnaire) des ressources [**2]** :

- **Machine virtuelle** - *signifie transformer un assemblage de chips et de circuits en un appareil plus utilisable, c'est-à-dire de travailler avec un outil moderne qui offre une abstraction simple au niveau des entrées/sorties, de l'utilisation de la mémoire, de la gestion des fichiers, de la protection et du contrôle des erreurs, de l'interaction des programmes entre eux et de leur contrôle. Ce rôle permet d'éviter au programmeur de connaître les détails électroniques de tel ou tels chips et de permettre à l'utilisateur de sauvegarder ses fichiers sans se soucier du type de modulation de fréquence utilisée pour stocker les informations.*
- **Gestionnaire des ressources** – *Ici, le système d'exploitation exécute sa fonction de répartiteur (partage) des ressources. Il se fait entre les programmes appelés plus justement les processus. Ce rôle de policier du système d'exploitation permet d'éviter les conflits d'utilisation de la mémoire, des périphériques d'entrées/sorties, des interfaces réseau... etc. On peut facilement imaginer ce qui arriverait si trois programmes essayaient d'imprimer en même temps sans qu'un certain ordre ne soit respecté. Ce travail d'alternance assuré par le système d'exploitation permet de mettre de l'ordre dans un chaos potentiel.*

En voici ci-dessous une abstraction illustrative du système d'exploitation présentée comme machine virtuelle et marche physique :



Schématisation du déploiement d'un système d'exploitation

De plus, lorsque l'ordinateur est utilisé par plusieurs usagers (presque tout le temps), le partage de la mémoire et surtout sa protection demeure une priorité absolue. En tout temps, un bon système d'exploitation connaît l'utilisateur d'une ressource, ses droits d'accès, son niveau de priorité [**16]**.

Principes de fonctionnement d'un système d'exploitation

Le système d'exploitation [**2]** est le programme fondamental des logiciels système. Il contrôle les ressources de l'ordinateur et fournit la base essentielle sur laquelle sont échafaudés les logiciels application. Le système d'exploitation opère sur un ordinateur constitué d'un ou plusieurs processeurs, d'une mémoire principale, d'horloges, de disques, de périphériques E/S, d'interfaces de connexion à des réseaux... etc. C'est la complexité toujours croissante du matériel qui a entraîné au cours des années, un développement rapide et spectaculaire des systèmes d'exploitation. Aujourd'hui, on parle de virtualité à tous égards ce qui permet aux usagers et aux programmeurs (de plus en plus) d'utiliser l'ordinateur avec un niveau d'abstraction élevé.

Remarquons que tout système d'exploitation est subséquemment fragmenté en six principales couches assemblées en deux grandes catégories (les couches bases : dont la fonction principale est d'assurer l'application et exécutions des diverses tâches sollicitées et les couches hautes : dont la fonction est de contrôler le processus de chaque exécution) pour permettre un meilleur contrôle de l'ensemble de l'ordinateur. Ces couches sont : ***Application, Interpréteur des commandes, noyau, langage machine, Microprogrammation et le dispositif physique*** [**21].** Le principe d'un système d'exploitation se démarque de deux points de vue [21] :

- **La vue descendante virtuelle** – A ce niveau, le système fournit à son utilisateur une vue uniforme des entrées/sorties, une mémoire virtuelle partageable, une gestion des fichiers et répertoires, une gestion des droits d'accès, de sécurité et des traitements des erreurs, une gestion de processus et finalement une gestion de communication interprocessus. On parle tout simplement de la « *machine virtuelle* ».

- **La vue ascendante -** A ce niveau, le système d'exploitation assure un fonctionnement adéquat des ressources, le respect des délais, l'identification de l'usager d'une ressource, le contrôle d'accès aux ressources, l'interruption d'utilisation des ressources, et l'évitement des conflits dans un ordinateur ou réseau. On parle du « *gestionnaire des ressources* ».

Ainsi, de façon plus établie, le fonctionnement d'un système d'exploitation se condense par quatre principes généraux à savoir [**28]** :

- **Le processus** – ceci représente une instance d'un programme en cours d'exécution, ce qui peut être en entrée des données, pour l'opération effectuée ou pour enregistrer les données en sortie.
- **Les appels systèmes** – on parle des « *appels systèmes* » lorsqu'un programme d'application fait appel à une procédure spéciale fournie par le système d'exploitation. L'ensemble de ces procédures est aussi appelé API : Application Programming Interface. C'est par exemple : le besoin d'une information (données) sur le disque dur.
- **Le système des fichiers** – représente l'organisation en arborescence et permet de classer les informations persistantes (durée de vie supérieure à celle du processus). C'est grâce à ce principe, que les opérations classiques telles que la création, suppression, ouverture, fermeture, positionnement, lecture et écriture deviennent faisables et réalisables par l'utilisateur.
- **La structure générale** – c'est la modélisation interne d'un système d'exploitation par une représentation en couches fonctionnelles basses et hautes qui réorganisent et rationnalisent toutes les opérations effectuées en trois principaux niveaux procéduraux. C'est le cas par exemple du premier niveau « *la procédure principale* » qui s'exécute lors de chaque appel système sollicité. Le deuxième niveau « *procédure de services* », dédié au traitement de chaque appel système en s'appuyant sur les instructions des utilisateurs. Le troisième niveau « *procédure des utilisateurs* » assiste les procédures des services en s'appuyant sur les formulations des requêtes des divers utilisateurs.

Organisation interne du système d'exploitation

L'organisation d'un système d'exploitation est l'abstraction des différentes composantes du système, leurs fonctions et leur mode d'interaction. Du point de vue interne, le système d'exploitation est formellement sectionné en 4 grandes composantes [**2]** :

1. **Call (Appels système)**[1] – constituent les différentes interfaces entre les systèmes d'exploitation et les programmes d'application qui peuvent s'exécuter soit en mode non-privilégié (Mode utilisateur) et/ou

[1] Tous les programmes systèmes sont des routines d'interruptions, fournies par l'OS ou les pilotes, qui permettent d'exploiter les périphériques. Sur les PC actuels, où le BIOS fournit des routines de base, l'OS doit modifier le vecteur d'interruption avec les adresses de ses routines. On dit qu'il détourne l'interruption. Les applications peuvent appeler à la demande ces routines via une interruption logicielle : ils effectuent ce qu'on appelle un ***appel système***. Tout OS fournit un ensemble d'appels systèmes de base, qui servent à manipuler la mémoire, gérer des fichiers, etc. Par exemple, linux fournit les appels systèmes open, Read, write et close pour manipuler des fichiers, les appels brk, sbrk, pour allouer et désallouer de la mémoire, etc. Évidemment, ceux-ci ne sont pas les seuls : linux fournit environ 380 appels systèmes distincts. Ceux-ci sont souvent encapsulés dans des librairies et peuvent s'utiliser comme de simples fonctions. L'appel système se charge automatiquement de basculer le

soit en mode privilégié (Mode noyau). C'est également grâce à cette composante que le système d'exploitation admet les états tels que : Contexte du programme appelant ; changement de contexte ; Contexte du noyau ; changement de contexte ; Contexte du programme appelant … Le coût d'un appel système est nettement plus élevé qu'un simple appel de fonction intra-processus : alors qu'un appel de fonction ne suppose que quelques instructions primitives (chargement et exécution d'une zone mémoire), le coût d'un appel système se compte en milliers ou dizaines de milliers d'instructions primitives, générant à la fois une charge et des délais d'exécution supplémentaires. Pour ces raisons, les fonctions qui sont utilisées de manière intense sont déplacées dans l'espace noyau. Les programmes utilisateurs font alors un nombre restreint d'appels système de haut niveau. Les nombreuses interactions de bas niveau générées par ces appels système sont effectuées dans l'espace noyau. Cela concerne notamment les pilotes de périphériques. Les entrées/sorties font également l'objet d'un traitement par l'ordonnanceur.

2. **Trap (Bibliothèque)** – c'est un sous-élément interdépendant de « call » qui permet à tout programme d'application de dérouter son exécution pour exécuter le code et les demandes du système central ; autrement dit, c'est une exécution de routine d'un système d'exploitation permettant généralement l'utilisation des ressources (matérielles ou logicielles) non accessibles dans l'espace utilisateur lorsque ce dernier est dissocier de l'espace noyau pour empêcher un éventuel plantage d'un programme d'application … Ce mécanisme doit être implémenté de façon protocolaire car les programmes ne peuvent connaitre leur exécution que lors de leur compilation de l'adresse à laquelle, ils devront se brancher pour appeler les fonctionnalités d'un système d'exploitation. C'est grâce à cette composante que le « call » ou les « appels système » peuvent exécuter des fonctions appelées depuis un programme de l'espace utilisateur ; dont l'exécution (le traitement) est effectuée dans l'espace noyau ; et dont le retour est effectué dans le programme appelant dans l'espace utilisateur.

3. **Mode Noyau** – Le **noyau**[2] aussi appelé **« Kernel,** en anglais **»** est une composante au cœur d'un ordinateur de système d'exploitation, avec un contrôle complet sur tout le système. Sur la plupart des systèmes, il est l'un des premiers programmes chargés sur le démarrage (après le *bootloader*). Il gère le reste de démarrage ainsi que d'entrée / sortie demandes de logiciel, les traduire en traitement des données des instructions pour l'unité centrale de traitement. Il gère la mémoire et des périphériques comme les claviers, les moniteurs, les imprimantes et les hauts - parleurs… De manière brève, le noyau est une instruction centrale d'un système d'exploitation où repose toutes les opérations et autorisations reliant le programme d'application au matériel d'un ordinateur.

4. **Mode utilisateur** - En revanche, c'est un espace où s'exécute tout ce qu'un utilisateur veut formuler comme requête au moyen d'une interface graphique ou autre. C'est par exemple : un texte écrit dans un éditeur, les programmes en cours d'exécution dans une interface graphique, etc. Cette séparation empêche

processeur dans l'espace noyau. Cette commutation n'est cependant pas gratuite, de même que l'interruption qui lui est associée. Ainsi, les appels systèmes sont généralement considérés comme lents, très lents. Divers processeurs incorporent des techniques pour rendre ces commutations plus rapides, via des instructions spécialisées (SYSCALL/SYSRET et SYSENTER/SYSEXIT d'AMD et Intel), ou d'autres techniques (call gate de l'Intel, Supervisor Call instruction des IBM 360, etc.).

[2] Le code critique du noyau est généralement chargé dans une zone séparée de la mémoire, qui est protégé contre l'accès par des programmes d'application ou d'autres parties, moins critiques du système d'exploitation. Le noyau exécute ses tâches, telles que l'exécution des processus, la gestion des périphériques matériels tels que le disque dur et la gestion des interruptions, dans ce protégé l'espace noyau.

les données utilisateur et les données du noyau d'interférer les uns avec les autres et provoquer l'instabilité et la lenteur, ainsi que la prévention le mauvais fonctionnement des programmes d'application de plantage l'ensemble du système d'exploitation.

Aperçu sur le noyau des systèmes d'exploitation

Un **noyau[3] de système d'exploitation**, ou simplement **noyau**, ou « ***kernel*** » en anglais, est une des parties fondamentales des systèmes d'exploitation. Comme précédemment dit, il gère les ressources de l'ordinateur et permet aux différents composants matériels et logiciels de communiquer entre eux [**8]**. En tant que partie du système d'exploitation, le noyau fournit des mécanismes d'abstraction du matériel, notamment de la mémoire, des processeurs, et des échanges d'informations entre logiciels et périphériques matériels. Le noyau autorise aussi diverses abstractions logicielles et facilite la communication entre les processus [**10]**.

Le noyau d'un système d'exploitation est lui-même un logiciel, mais ne peut cependant utiliser tous les mécanismes d'abstraction qu'il fournit aux autres logiciels [**4]**. Son rôle central impose par ailleurs des performances élevées. Cela fait du noyau la partie la plus critique d'un système d'exploitation et rend sa conception et sa programmation particulièrement délicates. Plusieurs techniques sont mises en œuvre pour simplifier la programmation des noyaux tout en garantissant de bonnes performances ... En informatique, le noyau d'un système d'exploitation est le logiciel qui assure [**22]** :

- La communication entre les logiciels et le matériel ;
- La gestion des divers logiciels d'une machine (*lancement des programmes, ordonnancement…)* ;
- La gestion du matériel (*mémoire, processeur, périphérique, stockage…).*

La majorité des systèmes d'exploitation est construite autour de la notion de noyau. L'existence d'un noyau, c'est-à-dire d'un programme unique responsable de la communication entre le matériel et le logiciel, résulte de compromis complexes portant sur des questions de performance, de sécurité et d'architecture des processeurs [**8]**. L'existence d'un noyau présuppose une partition virtuelle de la mémoire vive physique en deux régions disjointes, l'une étant réservée au noyau (l'espace noyau) et l'autre aux applications (l'espace utilisateur). Cette division, fondamentale, de l'espace mémoire en un espace noyau et un espace utilisateur contribue beaucoup à donner la forme et le contenu des actuels systèmes généralistes (GNU/Linux, Windows, MacOs X, etc.), le noyau ayant de grands pouvoirs sur l'utilisation des ressources matérielles, en particulier de la mémoire [**16]**. Elle structure également le travail des développeurs : le développement de code dans l'espace noyau est *a priori* plus délicat que dans l'espace utilisateur car la mémoire n'y est pas protégée. Ceci

[3] Les premiers concepteurs informatiques avaient l'habitude de décrire les différentes couches logicielles d'un système par une analogie : celle du noyau de la noix. En anglais, le mot « *kernel* » désigne le cerneau, la partie comestible de la noix, c'est-à-dire son coeur. À l'inverse, la coque (partie non comestible de la noix) est une enveloppe très dure qui entoure la partie comestible. En anglais cette coque est appelée « *Shell* ». Cette analogie permettait de comprendre que l'accès à la partie « comestible et incidemment cachée de la noix implique l'ouverture de la coque avec un instrument spécifique ». La dualité « cerneau / coque » illustre bien le rapport entre le *kernel* et le *shell* et constitue un paradigme aisément extensible à l'ensemble de l'informatique. Celui-ci peut être aussi compris comme une succession d'étapes permettant de diffuser entre chaque niveau l'ensemble des apports de celui qui le précède pour servir une information de plus en plus enrichies mais également, à chaque niveau, partagée de façon différente : « si la coque est nécessaire, seule la manière dont on peut la briser est d'une grande importance ».

implique que des erreurs de programmation, altérant éventuellement les instructions du noyau lui-même, sont potentiellement beaucoup plus difficiles à détecter que dans l'espace utilisateur où de telles altérations sont rendues impossibles par le mécanisme de protection [**9]**.

Rappelons que le noyau offre ses fonctions (l'accès aux ressources qu'il gère) au travers *des appels système*. Il transmet ou interprète les informations du matériel via des *interruptions*. C'est ce que l'on appelle les entrées et sorties. Signalons également que diverses abstractions de la notion d'*application* sont fournies par le noyau aux développeurs [**15]**. La plus courante est celle de processus (ou tâche). Le noyau du système d'exploitation n'est en lui-même pas une tâche, mais un ensemble de fonctions pouvant être appelées par les différents processus pour effectuer des opérations requérant un certain niveau de privilèges. Le noyau prend alors en général le relais du processus pour rendre le service demandé et lui rend le contrôle lorsque les actions correspondantes ont été réalisées [**21]**. Il peut arriver cependant que le processus puisse poursuivre une partie de son exécution sans attendre que le service ait été complètement réalisé. Des mécanismes de synchronisation sont alors nécessaires entre le noyau et le processus pour permettre à celui-ci, une fois qu'il est arrivé au point où il a besoin que le service ait été rendu, d'attendre que le noyau lui notifie l'exécution effective du service en question [**18]**. Par contre, Un processeur est capable d'exécuter un seul processus, un système multiprocesseur est capable de gérer autant de processus qu'il a de processeurs. Pour pallier cet inconvénient majeur, les noyaux multitâches permettent l'exécution de plusieurs processus sur un processeur, en partageant le temps du processeur entre les processus. Lorsque plusieurs tâches doivent être exécutées de manière parallèle, un noyau multitâche s'appuie sur les notions de *commutation de contexte*, *ordonnancement* et *temps partagé*. Les entrées et les sorties font l'objet d'un traitement spécifique par l'*ordonnanceur* [**20]**.

Remarque : Il existe de nombreux noyaux aux fonctionnalités restreintes tels que *les micro-noyaux, les systèmes sans noyau* (*MS-DOS, CP/M*) ou *les exo-noyaux*. Ces systèmes sont généralement adaptés à des applications très ciblées mais posent des problèmes variés (*de sécurité avec MS-DOS, de performances avec HURD ou QNX*). La plupart d'entre eux sont actuellement inadaptés pour une utilisation généraliste, dans des serveurs ou des ordinateurs personnels [**24]**.

Fonctionnement du Noyau

Les noyaux ont comme fonctions de base d'assurer le chargement et l'exécution des processus, de gérer les entrées/sorties et de proposer une interface entre l'espace noyau et les programmes de l'espace utilisateur. À de rares exceptions, les noyaux ne sont pas limités à leurs fonctionnalités de base. On trouve généralement dans les noyaux les fonctions des micronoyaux : un gestionnaire de mémoire et un ordonnanceur, ainsi que des fonctions de communication interprocessus [**30]**. En dehors des fonctions précédemment listées, de nombreux noyaux fournissent également des fonctions moins fondamentales telles que : la gestion des systèmes de fichiers ; Plusieurs ordonnanceurs spécialisés (Batch, temps réel, entrées/sorties, etc.) ; Des notions de processus étendues telles que les processus légers ; Des supports réseaux (*TCP/IP, PPP, Pare-feu, etc*.) ; Des services réseau (*NFS, etc*.) … [**25]** Enfin, la plupart des noyaux fournissent également des modèles de pilotes et des pilotes pour le matériel.

En dehors des fonctionnalités de base, l'ensemble des fonctions des points suivants (y compris les pilotes matériels, les fonctions réseaux et systèmes de fichiers ou les services) n'est pas nécessairement fourni par un noyau de système d'exploitation. Ces fonctions du système d'exploitation peuvent être implantées tant dans l'espace utilisateur que dans le noyau lui-même. Leur implantation dans le noyau est faite dans l'unique but d'augmenter les performances [**24]**. En effet, suivant la conception du noyau, la même fonction appelée depuis l'espace utilisateur ou l'espace noyau a un coût temporel notoirement différent. Si cet appel de fonction est fréquent, il peut s'avérer utile d'intégrer ces fonctions au noyau pour augmenter les performances. Ces techniques sont utilisées pour pallier des défauts de noyaux tels que les latences élevées.

Autant que possible, il est préférable d'écrire un logiciel hors du noyau, dans l'espace utilisateur. En effet, l'écriture en espace noyau suppose l'absence de mécanismes tels que la protection de la mémoire. Il est donc plus complexe d'écrire un logiciel fonctionnant dans l'espace noyau plutôt que dans l'espace utilisateur, *les bugs et failles de sécurité* étant plus dangereux [**31]**. Le noyau utilise également les techniques ci-après pour gouverner en toute sécurité et fiabilité l'ensemble de l'ordinateur :

Ordonnanceur

L'ordonnanceur d'un système d'exploitation n'a de sens qu'en système multitâche. Il gère l'ordre dans lequel les instructions de *différentes* tâches sont exécutées et est responsable de la sauvegarde et de la restauration du contexte des tâches (ce contexte est constitué des registres processeurs), appelée également commutation de contexte. La plupart des ordonnanceurs modernes permettent d'indiquer sur quel processeur sont exécutées les tâches [**24]**. Certains permettent également de migrer des tâches sur d'autres machines d'une grappe de calcul. L'algorithme d'ordonnancement détermine quelle tâche doit s'exécuter en priorité et sur quel processeur. Cet algorithme doit permettre d'utiliser efficacement les ressources de la machine. L'ordonnancement peut être de type « coopératif » **[24]** : les tâches doivent être écrites de manière à coopérer les unes avec les autres et ainsi accepter leur suspension pour l'exécution d'une autre tâche. L'ordonnancement peut être également de type *préemptif* : l'ordonnanceur a la responsabilité de l'interruption des tâches et du choix de la prochaine à exécuter. Certains noyaux sont eux-mêmes préemptifs : l'ordonnanceur peut interrompre le noyau lui-même pour faire place à une activité (typiquement, toujours dans le noyau) de priorité plus élevée [**24]**.

Gestionnaire de mémoire

Le gestionnaire de mémoire est le sous-ensemble du système d'exploitation qui permet de gérer la mémoire de l'ordinateur. Sa tâche la plus basique est d'allouer de la mémoire à des processus lorsqu'ils en ont besoin. Cette mémoire allouée est par défaut propre au processus qui en fait la demande. Sur les noyaux récents, [**12]** le gestionnaire de mémoire masque la localisation physique de la mémoire (en mémoire vive ou sur disque dur, dans l'espace de *mémoire paginée*) et présente au programme une *mémoire globale* uniforme dite mémoire virtuelle. Ainsi, tout processus croit manipuler une mémoire "logique" qui a les propriétés suivantes [**7]** :

- La mémoire peut être étendue jusqu'aux capacités théoriques de la machine ;
- La mémoire est privée (protégée), un processus ne peut pas accéder à la mémoire d'un autre processus (sauf allocations et autorisations spécifiques).

L'intérêt de ne pas indiquer au processus l'emplacement physique des données est de permettre au gestionnaire de mémoire de placer et déplacer à sa convenance les données en mémoire, sans affecter les processus. Ces données peuvent notamment être fragmentées dans la mémoire vive lorsqu'un processus demande un bloc de mémoire d'une taille supérieure au plus grand bloc physique libre. Le contenu de la mémoire peut aussi être migré. Cette migration est faite sur les différents supports mémoires tels que dans la mémoire physique (plus ou moins proche du processeur), dans la mémoire paginée, dans la mémoire accessible par réseaux (*grappe de calcul*) [**6]**.

La virtualisation de la mémoire permet aussi une gestion optimiste des ressources : la mémoire allouée mais pas encore utilisée peut-être virtuellement allouée à plusieurs processus. Les programmes dans l'espace utilisateur disposent de pouvoirs restreints sur la mémoire : ils doivent demander au noyau de la mémoire. Le noyau fait appel à son gestionnaire de mémoire pour allouer (ou non) la mémoire au processus qui la demande. Si un processus tente d'utiliser des zones de mémoire ne lui appartenant pas, il est évincé automatiquement [**23]**. Le mécanisme d'éviction repose sur un mécanisme du processeur, nommément une *unité de gestion de la mémoire*, ou MMU, qui signale au noyau l'existence d'un accès fautif. C'est le noyau lui-même qui prend la décision de suspendre ou détruire immédiatement le processus fautif [**16]**.

Gestion du matériel

La gestion du matériel se fait par l'intermédiaire de pilotes des périphériques. Les pilotes sont des petits logiciels légers dédiés à un matériel donné qui permettent de faire communiquer ce matériel. En raison du très grand nombre d'accès à certains matériels (disques durs par exemple), certains pilotes sont très sollicités. Ils sont généralement inclus dans l'espace noyau et communiquent avec l'espace utilisateur via les appels système [**9]**.

En effet, comme cela a été vu dans le précédent paragraphe, un appel système est coûteux : il nécessite au moins deux changements de contexte. Afin de réduire le nombre des appels système effectués pour accéder à un périphérique, les interactions basiques avec le périphérique sont faites dans l'espace noyau [**15]**. Les programmes utilisent ces périphériques au travers d'un nombre restreint d'appels système. Cependant, indépendamment de l'architecture, de nombreux périphériques lents (certains appareils photographiques numériques, outils sur liaison série, etc.) sont et peuvent être pilotés depuis l'espace utilisateur, le noyau intervenant au minimum. Il existe des couches d'abstraction de matériel qui présentent la même interface à l'espace utilisateur et simplifient ainsi le travail des développeurs d'applications. Dans les systèmes de type UNIX [**8]**, l'abstraction utilisée est le système de fichiers : les primitives *open, close, Read* et *write* sont présentées à l'espace utilisateur pour manipuler toutes sortes de périphériques. On parle dans ce cas de système de fichiers *synthétique* [**10]**.

Classification des noyaux

Il existe toutes sortes de noyaux, plus ou moins spécialisés. Des noyaux spécifiques à une architecture, souvent *monotâches*, d'autres *généralistes* et souvent *multitâches et multi-utilisateurs*. L'ensemble de ces noyaux peut être divisé en deux approches opposées d'architectures logicielles : *les noyaux monolithiques*[4] et *les micro-noyaux*[5] [**27]**. En conséquence, on distinguera les noyaux tels que :

Noyaux monolithiques non modulaires

Certains systèmes d'exploitation, comme d'anciennes versions de Linux, certains BSD ou certains vieux Unix ont un noyau monolithique. C'est-à-dire que l'ensemble des fonctions du système et des pilotes sont regroupés dans un seul bloc de code et un seul bloc binaire généré à la compilation. De par la simplicité de leur concept mais également de leur excellente vitesse d'exécution, les noyaux monolithiques ont été les premiers à être développés et mis en œuvre [**28]**. Cependant, au fur et à mesure de leurs développements, le code de ces noyaux monolithiques a augmenté en taille et il s'est avéré difficile de les maintenir. Le support par les architectures monolithiques des chargements à chaud ou dynamiques implique une augmentation du nombre de pilotes matériels compilés dans le noyau, et par suite, une augmentation de la taille de l'empreinte mémoire des noyaux [**9]**.

Celle-ci devient rapidement inacceptable. Les multiples dépendances créées entre les différentes fonctions du noyau empêchaient la relecture et la compréhension du code. L'évolution du code s'est faite en parallèle à l'évolution du matériel, et des problèmes de portage ont alors été mis en évidence sur les noyaux monolithiques [**9]**. En réalité les problèmes de la portabilité de code se sont révélés avec le temps indépendants de la problématique de la technologie des noyaux. Pour preuve, NetBSD est un noyau monolithique et est porté sur un très grand nombre d'architectures, alors que des noyaux tels que GNU Mach ou NT utilisent des micronoyaux censés faciliter le portage mais n'existent que pour quelques architectures [**9]**.

Noyaux monolithiques modulaires

Pour résoudre les problèmes évoqués ci-dessus, les noyaux monolithiques sont devenus modulaires. Dans ce type de noyau, seules les parties fondamentales du système sont regroupées dans un bloc de code unique (monolithique). Les autres fonctions, comme les pilotes matériels, sont regroupées en différents modules qui peuvent être séparés tant du point de vue du code que du point de vue binaire [**9]**. La très grande majorité des

[4] On considère généralement les noyaux monolithiques, de conception ancienne, comme obsolètes car difficiles à maintenir et moins « propres ». Le noyau Linux était déjà qualifié d'obsolète par Andrew Tanenbaum, dès sa création en 1991. Il ne croyait pas, à l'époque, pouvoir faire un noyau monolithique multiplateforme et modulaire

[5] La mise en place de micro-noyaux, qui consiste à déplacer l'essentiel des fonctions du noyau vers l'espace utilisateur, est très intéressante en théorie mais s'avère difficile en pratique. Ainsi les performances du noyau Linux (monolithique) sont supérieures à celles de ses concurrents (noyaux généralistes à micro-noyaux), sans compter qu'il fut finalement porté sur de très nombreuses plates-formes et qu'il est modulaire depuis 1995. Pour ces raisons de performance, les systèmes généralistes basés sur une technologie à micro-noyau, tels que Windows et Mac OS X, n'ont pas un « vrai » micro-noyau enrichi. Ils utilisent un micro-noyau hybride : certaines fonctionnalités qui devraient exister sous forme de mini-serveurs se retrouvent intégrées dans leur micro-noyau, utilisant le même espace d'adressage. Pour Mac OS X, cela forme XNU : le noyau monolithique BSD fonctionne en tant que service de Mach et ce dernier inclut du code BSD dans son propre espace d'adressage afin de réduire les latences. Ainsi, les deux approches d'architectures de noyaux, les micro-noyaux et les noyaux monolithiques, considérées comme diamétralement différentes en termes de conception, se rejoignent quasiment en pratique par les micro-noyaux hybrides et les noyaux monolithiques modulaires.

systèmes actuels utilise cette technologie : Linux, la plupart des *BSD ou Solaris*. Par exemple avec le noyau Linux, certaines parties peuvent être non compilées ou compilées en tant que modules chargeables directement dans le noyau [**15]**.

La modularité du noyau permet le chargement à la demande de fonctionnalités et augmente les possibilités de configuration. Ainsi les systèmes de fichiers peuvent être chargés de manière indépendante, un pilote de périphérique changé, etc. *Les distributions Linux*, par exemple, tirent profit des modules chargeables lors de l'installation **8]**. L'ensemble *des pilotes* matériels sont compilés en tant que modules. Le noyau peut alors supporter l'immense variété de matériel trouvé dans les compatibles PC. Après l'installation, lors du démarrage du système, seuls les pilotes correspondants au matériel *effectivement* présent dans la machine sont chargés en mémoire vive. La mémoire est économisée [**10]**.

Les noyaux monolithiques modulaires conservent les principaux atouts des noyaux monolithiques purs dont ils sont issus. Ainsi, la facilité de conception et de développement est globalement maintenue et la vitesse d'exécution reste excellente [**8]**. L'utilisation de modules implique le découpage du code source du noyau en blocs indépendants. Ces blocs améliorent l'organisation et la clarté du code source et en facilitent également la maintenance. Les noyaux monolithiques modulaires conservent également un important défaut des noyaux monolithiques purs : une erreur dans un module met en danger la stabilité de tout le système. Les tests et certifications de ces composants doivent être plus poussés. D'un point de vue théorique, le grand nombre de lignes de code exécutées en mode noyau engendre des problèmes de portabilité. La pratique contredit largement la théorie et les noyaux modulaires sont aujourd'hui les plus portés [**2]**.

Les systèmes à micronoyaux

Les limitations des noyaux monolithiques ont amené à une approche radicalement différente de la notion de noyau : les systèmes à micronoyaux. Les systèmes à micronoyaux cherchent à minimiser les fonctionnalités dépendantes du noyau en plaçant la plus grande partie des services du système d'exploitation à l'extérieur de ce noyau, c'est-à-dire dans l'espace utilisateur. Ces fonctionnalités sont alors fournies par de petits serveurs indépendants possédant souvent leur propre espace d'adressage. Un petit nombre de fonctions fondamentales est conservé dans un noyau minimaliste appelé « *micro-noyau* » [**9]**. L'ensemble des fonctionnalités habituellement proposées par les noyaux monolithiques est alors assuré par les services déplacés en espace utilisateur et par ce micro-noyau. Cet ensemble logiciel est appelé « *micro- noyau enrichi* ». Ce principe a de grands avantages théoriques : en éloignant les services « à risque » des parties critiques du système d'exploitation regroupées dans le noyau, il permet de gagner en robustesse et en fiabilité, tout en facilitant la maintenance et l'évolutivité. En revanche, les mécanismes de communication (*IPC – InterProcess Communication*), qui deviennent fondamentaux pour assurer le passage de messages entre les serveurs, sont très lourds et peuvent limiter les performances [**22]**. Les avantages théoriques des systèmes à micro-noyaux sont :

- La conséquence de l'utilisation du mode protégé par les services qui accompagnent le micro-noyau. En effet, en plaçant les services dans l'espace utilisateur, ceux-ci bénéficient de la protection de la

mémoire. La stabilité de l'ensemble en est améliorée : une erreur d'un service en mode protégé a peu de conséquences sur la stabilité de l'ensemble de la machine.

- De plus, en réduisant les possibilités pour les services de pouvoir intervenir directement sur le matériel, la sécurité du système est renforcée. Le système gagne également en possibilités de configuration. Ainsi, seuls les services utiles doivent être réellement lancés au démarrage. Les interdépendances entre les différents services sont faibles. L'ajout ou le retrait d'un service ne perturbe pas l'ensemble du système. La complexité de l'ensemble est réduite.
- Le développement d'un système à micro-noyau se trouve également simplifié en tirant parti à la fois de la protection de la mémoire et de la faible interdépendance entre les services. Les erreurs provoquées par les applications en mode utilisateur sont traitées plus simplement que dans le mode noyau et ne mettent pas en péril la stabilité globale du système. L'intervention sur une fonctionnalité défectueuse consiste à arrêter l'ancien service puis à lancer le nouveau, sans devoir redémarrer toute la machine.
- Ils sont beaucoup plus compacts que les noyaux monolithiques avec 6 millions de lignes de code pour le noyau Linux V2.6.0 contre en général moins de 50 000 lignes pour les micro-noyaux. La maintenance du code exécuté en mode noyau est donc simplifiée. Le nombre réduit de lignes de code peut augmenter la portabilité du système.

Cependant, ces nouveaux noyaux ne tarderont pas à présenter à la face du monde leurs plus grandes difficultés vis-à-vis de l'utilisation de nombreux services dans l'espace utilisateur telles que **[8]** :

- La plupart des services sont à l'extérieur du noyau et génèrent un très grand nombre d'appels système ;
- Les interfaces de communication entre les services (IPC) sont complexes et trop lourdes en temps de traitement.
- Le grand nombre d'appels système et la communication sous-jacente sont un défaut inhérent à la conception des micro-noyaux. Ce qui pourra être résolu, par exemple en supprimant toute vérification des permissions, laissant ce soin aux serveurs externes.

Ces modifications radicales ont permis d'obtenir de bonnes performances mais elles ne doivent pas faire oublier qu'un micro-noyau doit être accompagné d'un grand nombre de services pour fournir des fonctionnalités équivalentes à celles des noyaux monolithiques. De plus, la grande liberté dont disposent les services au niveau de la sécurité et de la gestion de la mémoire accroît la difficulté et le temps de leur développement (*ils doivent fournir leurs propres interfaces*) [**21]**.

Noyaux hybrides

La dénomination « *noyaux hybrides* » désigne principalement des noyaux qui reprennent des concepts à la fois des noyaux monolithiques et des micro-noyaux, pour combiner les avantages des deux. Lorsque, au début des années 1990, les développeurs et concepteurs se sont aperçus des faiblesses des premiers micro-noyaux, [**9]** certains réintégrèrent diverses fonctionnalités non fondamentales dans le noyau, pour gagner en

performance. Les micro-noyaux « purs » semblaient condamnés à l'échec. Alors que la philosophie générale des systèmes à micro-noyaux est maintenue (seules les fonctions fondamentales sont dans l'espace noyau), certaines fonctions non critiques, mais très génératrices d'appels système, sont réintégrées dans l'espace noyau. Ce compromis permet d'améliorer considérablement les performances en conservant de nombreuses propriétés des systèmes à micro-noyaux[6] [**8]**.

Exo-noyaux

Étymologiquement, « *exo* » signifie en grec « *hors de* ». Un exo-noyau est donc un système d'exploitation fonctionnant en espace utilisateur (en « *user-space* », au lieu du « *kernel-space* » dans le cas des autres noyaux). Les fonctions et services du système d'exploitation sont assurés par de petits modules qui, selon les approches techniques, sont des bibliothèques [**8]**.

Méta-noyaux

Un « méta-noyau » est un ensemble de logiciels qui vise à appliquer la notion de noyau informatique au niveau d'un réseau informatique, en créant une unique couche de gestion des périphériques au niveau d'un réseau. De cette manière, les logiciels peuvent être déployés et utilisés sur le réseau informatique comme s'il s'agissait d'une machine unique, et l'ensemble des logiciels fonctionnant sur cette plate-forme peuvent se partager les ressources de manière intégrée, comme elle le ferait sur un noyau simple. Un méta système doit également permettre la personnalisation, la gestion des permissions ainsi que l'utilisation d'informations dépendant de la localisation. Cette notion rejoint les notions de Grappe de calcul, de machine virtuelle, de serveur d'applications et de CORBA [**18]**.

Noyaux temps réel

Les noyaux temps réel sont fonctionnellement spécialisés. Ce sont des noyaux généralement assez légers qui ont pour fonction de base stricte de garantir les temps d'exécution des tâches. Il n'y a pas à proprement parler de notion de rapidité de traitement ou de réactivité dans les noyaux temps réel, cette notion est plutôt implicite à la garantie des temps d'exécution en comparaison aux critères temporels de l'application industrielle (*la réactivité d'un système de freinage ABS n'a pas les mêmes critères temporels que le remplissage d'une cuve de pétrole*) [**15]**. Très utilisés dans le monde de l'électronique embarquée, ils sont conçus pour tourner sur des plates-formes matérielles limitées en taille, puissance ou autonomie [**19]**.

Les noyaux temps réel[7] peuvent adopter en théorie n'importe quelle architecture précédemment listée. Ils fournissent souvent deux interfaces séparées, l'une spécialisée dans le temps réel et l'autre générique [**17].**

[6] Un exemple de ce type de noyau hybride est le noyau XNU de MacOs X. Il est basé sur le micro-noyau *Mach 3.0*, mais qui inclut du code du noyau monolithique BSD au sein de l'espace noyau. Cette dénomination est également utilisée pour désigner d'autres types de noyaux, notamment les noyaux monolithiques sur micro-noyaux (temps réel ou non) tels que *L4Linux* (Linux sur L4), *MkLinux* (le noyau Linux sur Mach), Adeos, RTLinux et RTAI … Plus rarement, on peut rencontrer le terme « noyau hybride » pour remplacer improprement « *noyau monolithique modulaire* » ou « micro-noyau enrichi ».

[7] Par exemple, on peut avoir un micro-noyau temps réel allouant des ressources à un noyau non temps réel tel que Linux (RTLinux, RTAI, Xenomai) ou Windows RTX. L'environnement GNU (responsable de Windows) peut alors être exécuté à l'identique sur le noyau pour lequel il a été conçu, alors que les applications temps réel peuvent faire directement appel au micro-noyau temps réel pour garantir leurs délais d'exécutions. VxWorks est un noyau propriétaire temps réel très implanté dans l'industrie bien que les systèmes à base de noyau

Les applications temps réel font alors appel à la partie temps réel du noyau. Une des architectures souvent retenue est un noyau hybride qui s'appuie sur la combinaison d'un micro-noyau temps réel spécialisé, allouant du temps d'exécution à un noyau de système d'exploitation non spécialisé. Le système d'exploitation non spécialisé fonctionne en tant que service du micro-noyau temps réel. Cette solution permet d'assurer le fonctionnement temps réel des applications, tout en maintenant la compatibilité avec des environnements préexistants [**3]**.

RESULTATS ET DISCUSSIONS

Exigences de sécurité pour les systèmes d'exploitation

Aujourd'hui, si la technologie continue d'ouvrir la porte à de nouvelles formes d'aménagements, elle donne lieu à de nombreuses réflexions dans le domaine de la sécurité. En effet, en repoussant au fur et à mesure les frontières de ce qui semblait difficile à mettre en œuvre en termes de dessin des contours des organisations, les exigences de sécurité continuent de changer [**31]**. Les principes qui étaient ceux appliqués aux bredouillements de l'Internet ne sont plus les mêmes, et aujourd'hui nous pouvons constater des changements très forts voire parfois un revirement total sur les principes guidant la mise en œuvre de politiques sécuritaires [**3]**.

Si pendant longtemps, la sécurisation du noyau des systèmes d'exploitation constituait un processus indépendant des programmes et des applications qui étaient exécutées, on envisage de plus en plus aujourd'hui l'intégration de ce processus au sein des activités métiers des organisations. En conséquence, le contrôle d'accès aux ressources se voit de plus en plus impacté par un lien très étroit. Les mutations dans la plupart des systèmes d'exploitation sont telles que l'on doit envisager l'organisation du contrôle d'accès en tenant compte de ces nouveaux paradigmes. [**4]** Dans ce contexte, la détermination des exigences de sécurité devient un élément fondamental pour l'organisation des accès sur des ressources, des applications, des programmes, voire la mise en œuvre de processus en mode privilégié et non privilégié du système noyau afin d'assurer et de garantir les propriétés de sécurité qui leur sont associées. Dans le contexte du présent du travail, il s'agit essentiellement des deux exigences : *La gestion des identités et des accès et la gestion des autorisations* [**11].**

La gestion des identités et des accès

La gestion des identités et des accès engage de façon intégrante la sécurité des systèmes d'exploitation [**1].** La gestion des identités prend en considération la problématique de création, modification et destruction des comptes et privilèges associés à des utilisateurs. Elle permet également de s'assurer de l'authentification des utilisateurs, de garantir l'accès aux ressources et services en fonction de critères polymorphes. De plus, elle doit permettre de fournir un historique complet des accès tout en garantissant la protection des libertés individuelles, et de traiter les consentements. Cet aperçu répond matériellement à deux préalables [**11]** :

La conception d'identité

Linux se déploient énormément et aient un succès grandissant via RTAI et Xenomai (RTLinux étant breveté).

Sur le plan humain, l'approche des identités dans le domaine de la psychologie sociale est profondément paradoxale. La notion d'identités fait état d'une articulation où se mêle similitude, différence, unicité et communauté. L'identité résulte d'interactions complexes qui se nouent entre un individu et d'autres appartenant à un groupe. L'identité se crée en considérant entre autres des déterminants sociaux, des positions intimes, la perception de soi et le regard d'autrui. Le concept d'identité n'est donc pas simple à établir et sa mise en place se justifie par la nécessité de répondre à une problématique donnée. Ainsi la délivrance d'une carte d'identité professionnelle dans une entreprise, permet à un employé (à minima) d'affirmer son appartenance à cette entreprise. La délivrance d'une carte de séjour pour un étudiant étranger permet de justifier de la régularité de sa présence sur un territoire. Un passeport permet de justifier un lien avec le pays qui en a assuré la délivrance [**9]**.

Dans le monde numérique toute entité (sujet, objet) peut de manière semblable se voir associer une ou plusieurs identités dans un ou plusieurs référentiels. On associe aux utilisateurs des comptes (*user account*) sur un serveur de fichiers ou encore des comptes de messagerie électronique. Pendant longtemps, l'identité numérique a été considérée comme l'équivalent de l'identité classique d'un être humain ou d'une personne morale. Une identité numérique se compose d'un ensemble de données descriptives (caractéristiques ou attributs) que l'on peut désigner par un identifiant. Cet identifiant est une valeur unique dans l'espace des valeurs servant à l'identification d'une entité dans le référentiel concerné (messagerie électronique, serveurs de fichiers). [**13]** Lorsque l'identité est associée à une personne physique, un ou plusieurs profils peuvent être établis à partir de sous-ensembles des attributs (*à partir de la totalité*). L'identité numérique est une représentation « informatique » d'une entité [**14]**.

Comme dans le domaine classique, une identité numérique n'a de sens que si elle a une utilité, c'est-à-dire une entité est amenée à interagir avec un ou plusieurs systèmes pour la réalisation d'un ensemble d'activités. Ainsi un usager d'une entreprise accède à des serveurs de messagerie, des serveurs de fichiers, certains administrent des systèmes d'exploitation, d'autres accèdent à une base de données … Généralement un lien est établi entre un élément d'identité, ou un profil généré à partir d'une identité pour répondre à un besoin fonctionnel dans le cadre de l'accès à une ressource [**14]**.

Rappelons tout de même, que toute identité présente un cycle de vie. La création d'une identité intervient lorsque l'on souhaite qu'une entité puisse exister dans un référentiel donné. Dans la création d'une identité, plusieurs attributs sont renseignés [**1]**. Certains de ces attributs sont statiques, alors que d'autres sont dynamiques. Par exemple le nom d'un utilisateur humain est statique, alors que les pseudos qu'il utilise peuvent changer dans le temps. La modification ou maintenance des attributs fait partie du cycle de vie. Une identité peut être détruite lorsqu'une entité n'a plus de raison d'exister dans le référentiel où elle figurait. Le cycle de vie d'une identité est classique dans le sens où il comporte une phase de création, de maintenance et de destruction. [**3]** La durée de l'existence d'une identité est variable. La création d'un compte au sein d'un système informatique peut être réalisée pour une durée de plusieurs mois ou de plusieurs années. La création d'un processus associée à l'exécution d'un programme n'aura pour durée de vie que le temps de l'exécution d'un programme.

European Journal of Computer Science and Information Technology
Vol.11, No.2, pp.70-99, 2023
Print ISSN: 2054-0957 (Print),
Online ISSN: 2054-0965 (Online)
Website: https://www.eajournals.org/
Publication of the European Centre for Research Training and Development -UK

La gestion des accès

La gestion des accès fait référence au contrôle et à la sécurisation des accès aux ressources ou services, et donc à la protection des informations contre tout accès non-autorisé. Le contrôle d'accès recouvre donc un sens particulier car il garantit la confidentialité des informations et sa signification recouvre trois processus nommés « **Authentification – Autorisation - Audit** »[8] [**5]**.

En sécurité informatique, le *« contrôle d'accès* » se réfère à l'utilisation de mécanismes pour permettre aux entités authentifiées d'exécuter des actions en fonction de leur niveau d'autorisation et d'empêcher les entités de réaliser des actions non autorisées. **Le processus d'authentification** est une étape préliminaire permettant de vérifier et de valider l'identité numérique d'une entité qui veut accéder à une ressource ou un service [**8]**. L'absence d'un tel processus ne peut que menacer les ressources sur lesquelles portent les accès. Une ressource offerte en libre accès, l'est en particulier à des utilisateurs malicieux, qui peuvent alors profiter de cette situation pour détourner l'utilisation prévue de la ressource. Un résultat erroné d'une tentative de vérification de l'identité peut également conduire des utilisateurs malicieux à s'introduire auprès d'un système ou d'exploiter sous une forme non légitime les ressources concernées [**23]**.

Ce processus doit permettre de vérifier qu'un utilisateur qui réclame l'accès à une ressource est bien celui qu'il prétend être. La solution la plus courante est celle qui consiste à gérer les accès sur la base d'un couple (nom de compte d'utilisateur, mot de passe). Une autre solution est d'utiliser des certificats électroniques X.509 comme moyen d'authentification. Suivant le domaine d'application métier, ce processus de vérification peut être réalisé de manière récurrente. Ainsi, dans le domaine bancaire, l'accès à un certain nombre de serveurs n'est possible qu'après avoir décliné son identité, plusieurs fois, sur des serveurs différents et dans un temps imparti [**11]**. Ce principe est équivalent à celui utilisé dans les aéroports internationaux où la production de son passeport est exigée à de nombreux endroits pour atteindre un avion ou pour pénétrer le territoire d'un pays.

Le processus d'autorisation [**5]** est celui qui permet de déterminer et de décider si une entité dont l'identité a été préalablement vérifiée est habilitée ou non à réaliser une ou plusieurs activités ou actions sur une ressource ou un service. Ce processus permet donc de refuser à certaines entités (non autorisées) l'accès aux ressources ou services sensibles ou non. La traçabilité d'un certain nombre d'événements est obtenue en

[8] Dans le domaine de la gestion des accès, ces trois processus complémentaires viennent constituer le standard d'or de la sécurité. Aujourd'hui, l'organisation des systèmes d'information est telle que les systèmes de gestion des accès ne portent plus sur des ressources uniques ou sur des organisations unitaires. La complexité est telle que le contrôle d'accès doit être établi à partir de caractéristiques propres à la ressource ou au service, en considérant des éléments de localisation géographique, temporels, contextuels… Ainsi le contrôle d'accès peut porter sur des applications spécifiques, avec franchissement de pare-feu, exploitation de VPN, d'unités de stockage, des bases d'informations maintenues soit au sein de sa propre organisation soit au sein d'une organisation partenaire… La complexité est aujourd'hui si forte que les travaux de recherche portent sur la prise en compte de la complémentarité des aspects en termes d'organisation, de formalisation, de sémantique, d'architecture… pour offrir un système de gestion des accès qui soit le plus adapté aux exigences les plus ouvertes possibles. Cette complexité vient de la diversité des profils d'usagers, des ressources, des équipements, des relations établies « Man to Man », « Machine to Machine », « Man to Machine ». Si à l'origine, la gestion des accès était réglée sous une forme simple et directe (ex : accès à un shell sur une machine), aujourd'hui il apparait de plus en plus comme le résultat d'évaluations de situations composites : par exemple l'accès à une donnée peut engendrer des accréditations différentes si l'on est dans l'enceinte d'une entreprise ou si l'on est à l'extérieur de celle-ci. Les solutions de gestion des accès doivent évoluer pour être en adéquation avec l'évolution des exigences naissant de l'adoption des technologies au sein des organisations.

réalisant une journalisation de ces événements. Ainsi, la création de comptes, la modification et la suppression donnent lieu à historisation pour déterminer les utilisateurs d'un système. La journalisation des « *logs* » donne la possibilité de rechercher à quel moment le titulaire d'un compte a cherché à atteindre une ressource (voire la modifier). Au-delà de la vérification d'identité et de la possibilité ou non d'atteindre une ressource. **Les éléments d'audit et de comptabilité** (*Accounting*) [**5]** permettent de retracer et/ou de suivre qui a eu accès et à quel niveau sur une ressource donnée. Il s'agit là d'une information capitale dans le contexte de la sécurité puisque ce processus permet de réaliser des analyses comportementales d'usagers sur des ressources. Dans certains cas de figure, cela permet de repérer des comportements malveillants et/ou des failles de sécurité qui peuvent affecter certains produits ou services.

La gestion des autorisations

La gestion des autorisations a pour fonction d'évaluer la capacité de décider d'admettre ou non des sujets (utilisateur, programme en cours d'exécution, calculateur, etc.) à réaliser des activités/actions sur un ou plusieurs objets/ressources (fichier, dossier, bâtiment, application informatique spécifique, base de données, serveur etc.). Cette gestion est basée sur deux mécanismes de protection [**5]** :

Les autorisations AAA

Ce mécanisme de protection consiste à garantir trois fonctionnalités de contrôle d'accès qui sont l'authentification, l'autorisation et l'audit (AAA), (voire précédemment). Ce mécanisme vise à répondre aux besoins relevés en termes de gestion des accès (qui a le droit de faire quoi, comment et dans quelles circonstances). Ici, les interactions des autorisations AAA peuvent suivre les modèles de déploiement tels que :

- **Le modèle Agent -** L'utilisateur formule sa requête de demande d'accès à un équipement de service auprès d'une partie tierce (serveur d'autorisation AAA). Celle-ci est l'agent situé entre l'utilisateur et l'équipement de service. Le serveur AAA applique la politique qui a été préalablement établie pour la demande en cours. Si l'accès est autorisé, le serveur transfère la requête de service ; sinon, il renvoie un message d'interdiction à l'utilisateur ayant formulé la requête. L'équipement de service retourne le résultat de la requête au serveur AAA qui à son tour renvoie la réponse à l'utilisateur.
- **Le modèle Push -** En premier lieu, l'utilisateur demande l'autorisation de sa requête au serveur AAA (par exemple, via serveur d'autorisation). Si le serveur AAA autorise la requête du sujet, il envoie à l'utilisateur un message sécurisé (jeton ou certificat) qui agit comme une preuve de droit (assertion d'autorisation). Généralement une telle assertion a un temps de validité qui lui est associée. L'assertion peut ensuite être utilisée par l'utilisateur pour demander un service spécifique en communiquant avec l'équipement de service. L'équipement de service applique la décision qui sera d'accepter ou de refuser l'assertion d'autorisation. Un tel système est mis en œuvre dans la distribution de tickets comme Kerberos.
- **Le modèle Pull -** En premier, l'utilisateur contacte l'équipement de service directement avec une requête. Pour accorder ou refuser la requête de l'utilisateur, l'équipement de service contacte l'autorité

d'autorisation (serveur AAA). Ensuite le serveur AAA prend une décision d'autorisation et renvoie le résultat via un message à l'équipement de service. L'équipement de service accordera ou non le service à l'utilisateur en analysant le contenu du message contenant le résultat de la décision.

Les autorisations à base de Politiques

Les autorisations à base de politiques sont une autre solution pour contourner la difficulté inhérente aux solutions qui consistaient entre autres à construire des ACLs (Access Control List). Tout comme les autorisations AAA décrites précédemment [**11]**, cette architecture présente l'avantage d'externaliser le processus de décisions de contrôle d'accès en dehors de l'application gérée. De plus, les langages de politique ont une capacité d'expressivité plus importante que les ACLs [**7]**. Cette architecture rend la gestion plus dynamique et évolutive, car une autorisation est explicitement décrite par deux fonctionnalités complémentaires [**5]** :

- Une fonction de prise de décisions d'autorisation réalisée après consultation des politiques et ;
- Une fonction d'application des décisions.

Ces deux actions sont accomplies par deux entités distinctes nommées respectivement :

- **Un PDP** (*Policy Decision Point*) - est une entité logique qui prend des décisions d'autorisation en considérant les informations suivantes : Un contexte de requête qui inclut des informations sur l'entité qui demande l'accès, l'opération à effectuer (consultation, modification, etc.) et la ressource à protéger ; Et la politique d'autorisation qui définit les autorisations.

- **Un PEP** (*Policy Enforcement Point*) est une entité logique qui applique la décision d'autorisation prise par le PDP. C'est le PEP qui réalise techniquement l'attribution ou le refus d'une demande d'accès à une ressource.

ATTAQUES SUR LE NOYAU D'UN SYSTEME D'EXPLOITATION

La sécurité des systèmes d'exploitation s'inscrit dans le cadre plus large de la sûreté de son fonctionnement réelle ou potentielle, c'est-à-dire une aptitude à délivrer un service des dépendances acceptées implicitement et explicitement (*Prévention des fautes, Tolérances aux fautes, Elimination des fautes, et Prévision des fautes*). Toute fonction de sécurité se base sur cinq prérogatives (*Confidentialité, Intégrité, disponibilité, Non Répudiation et Authentification*) [**12]**. Par conséquent, tout acte sur un système visant à nuire à ces propriétés peut être qualifié de *malveillant ou d'attaque*. Une attaque ou acte *de malveillance* sur un système est une combinaison d'actions qui ciblent ce système et dont l'*intention* est de nuire au moins à l'une des propriétés de disponibilité, d'intégrité ou de confidentialité de ce système, ou d'un programme logiciel ayant une quelconque relation avec lui [**31]**. Rappelons que toute attaque logique sur un système informatique peut être classée d'après les quatre étapes ci-dessous :

1. **Intrusion dans le système informatique** - Une intrusion dans un système informatique est le fait pour un système externe d'y *pénétrer* alors que sa présence n'y est pas autorisée. Afin de réaliser une intrusion, divers choix peuvent se présenter pour l'attaquant. Par exemple, il peut usurper l'identité d'un utilisateur légitime du système (*vol d'identifiant et de mot de passe, attaque par dictionnaire, etc.*), ou encore exploiter une faille du système logiciel qui lui permet d'accéder aux services du système informatique. Ces différentes possibilités d'intrusion pour l'attaquant dans le système informatique sont directement liées aux contraintes associées à ce système [**31]**. Cette étape est le point de départ d'une attaque qui installe un *sniffer* sur un système informatique complexe. Ici, l'attaquant souhaite tout simplement récupérer par exemple des mots de passe d'utilisateurs légitimes du système. Alors que le travail du *sniffer* ne requiert aucune intrusion sur le système (il effectue une écoute passive), son installation nécessite une intrusion préalable.

2. **Acquisition de privilèges** – Cette deuxième étape du processus d'attaque d'un système est associée à l'état dans lequel se trouve un attaquant lorsque les privilèges dont il dispose sur un système informatique sont insuffisants pour la réalisation de son objectif. Ainsi, dans cet état, l'attaquant tente d'acquérir les privilèges qui lui sont nécessaires au travers de failles existantes dans le système informatique dans lequel il a pénétré [**31]**. Par exemple, un attaquant ayant usurpé l'identité d'un utilisateur légitime du système peut avoir besoin des privilèges d'un administrateur du système pour réaliser son objectif. Il peut également nécessiter des privilèges encore supérieurs comme ceux associés à un noyau de système d'exploitation.

3. **Altération du système logiciel** - Ce Cette troisième étape du processus d'attaque d'un système représente l'état dans lequel se retrouve l'attaquant lorsque le système dans lequel il a pénétré ne dispose pas de la structure adéquate à la réalisation de son objectif, mais que l'attaquant dispose des privilèges nécessaires pour y remédier. Il s'agit alors pour l'attaquant, de modifier la structure et/ou l'état du système logiciel afin que ce dernier soit apte à la réalisation de son objectif de l'attaquant [**31]**. Une altération du système logiciel par un attaquant peut être vu comme l'ajout (ou encore l'installation) d'un ou de plusieurs sous-systèmes malveillants au système logiciel, soit de l'altération d'informations qui impactent la sécurité du système. À noter que tout système peut être qualifié de malveillant dès lors que l'utilisation qui en est faite a pour objectif de réaliser un acte malveillant. Certains systèmes sont cependant intrinsèquement malveillants et peuvent être classés suivant leur nature et leur comportement. Il s'agit des infections informatiques.

4. **Infections informatiques** - Une infection informatique [**10]** est le résultat de l'installation dans un système d'information, à l'insu du ou des utilisateurs, d'un programme à caractère offensif, en vue de porter atteinte à la sécurité de ce système. Voici les infections informatiques, les plus usuelles :

- **Bombe logique** - Une *bombe logique* est une infection qui attend un évènement (date, action, données particulières, etc.) appelé en général « gâchette » pour exécuter sa fonction offensive.
- **Cheval de Troie** - Un *Cheval de Troie* est directement issue de la mythologie grecque, plus précisément de la guerre de Troie. Un cheval de Troie, depuis lors désigne un artifice quelconque qui entraîne une cible à inviter un ennemi dans un endroit sécurisé. Dans le contexte de la sécurité

informatique, un cheval de Troie, désigne un système réalisant une fonction a priori souhaitée par un utilisateur mais qui réalise une fonction pour l'attaquant. Il vise de par sa nature à leurrer sa victime.

- **Porte dérobée** - Une *porte dérobée* (*Backdoor/Trapdoor*) est un moyen de contourner les mécanismes de contrôle d'accès. Il s'agit d'une faille du système de sécurité qui provient soit d'une faute de conception (*volontaire ou accidentelle*), soit d'une altération du système durant sa phase opérationnelle [**7]**.

- **Spyware** - Un *spyware* est une infection dont l'objectif est de divulguer à un attaquant des informations issues du système infecté [**31]**.

- **Adware** - Un *adware* est une infection dont l'objectif est d'envoyer à l'utilisateur du système infecté de la publicité ciblée. Pour cela, il espionne les habitudes de l'utilisateur [**17]**.

- **Programmes autoreproducteurs ou Virus** - Un *virus* est une séquence de symboles qui, interprétée dans un environnement donné (adéquat), modifie d'autres séquences de symboles dans cet environnement, de manière à y inclure une copie de lui-même, cette copie ayant éventuellement évolué [**31]**.

- **Vers** - Un vers (*worm*) est un programme *autonome* qui se propage sur les réseaux, se reproduit et s'exécute à l'insu des utilisateurs normaux [**31]**.

- **Rootkit** - Un *rootkit* est un système parasite permettant à un attaquant de maintenir dans le temps un contrôle frauduleux sur un système informatique. C'est donc un ensemble des techniques mises en œuvre par un ou plusieurs logiciels dont le but est d'obtenir et de pérenniser un accès généralement non autorisé dans un noyau du système d'exploitation de la manière la plus fictive possible [**31]**.

FAILLES GENERALES DU NOYAU SYSTEME

Pour chacune des étapes essentielles des attaques logiques sur le noyau présentées ci-dessus. Nous pouvons facilement remarquer que le dénominateur commun est la perte de l'intégrité du noyau. La perte d'intégrité d'un noyau en cours d'exécution provient de plusieurs sources [**17]** :

- Soit du composant matériel qui le supporte (c'est-à-dire de la structure ou de l'état du composant matériel qui contient le noyau), qui est la mémoire principale ;

- Soit des composants matériels dont il dépend pour son exécution (*le processeur, le chipset, et BIOS*) ;

- Soit des composants matériels avec qui il communique (qui sont les périphériques) ;

- Soit les périphériques comme source possible à la perte d'intégrité du noyau car le noyau est censé permettre à l'utilisateur d'accéder aux services offerts par les périphériques. Si la perte de leur intégrité provoque une impossibilité pour le noyau d'assurer à l'utilisateur l'accès à ces services, alors l'intégrité du noyau est également affectée.

Failles liées à l'accès de la mémoire du noyau

La première façon d'accéder à la mémoire du noyau se fait au travers de la CPU. Cet accès fait intervenir automatiquement dans un premier temps la MMU (*Memory Management Unit - lié aux mécanismes de segmentation9 et de pagination*10) de la CPU et dans un second temps le MCH (*Memory Controller Hub* – lié au contrôle des registres) [**18]**. En conséquence, une modification inappropriée de la mémoire du noyau peut donc provenir d''une fonctionnalité du système qui fournit directement le moyen de modifier n'importe quelle région de l'espace de mémoire du noyau. Il s'agit soit :

- D'une fonctionnalité logicielle, telle que, dans le cas de Windows, le chargeur de modules, ou encore les périphériques virtuels chargés directement lors du démarrage boot ;

- D'une fonctionnalité matérielle, telle que le mode SMM (*System Management Mode*) du CPU ;

- D'une fonctionnalité du système vulnérable qui fournit le moyen de corrompre l'espace "noyau" au travers de l'exploitation de sa vulnérabilité, par exemple : le débordement d'un tampon, l'utilisation d'une chaîne de format malveillante, l'utilisation de données incorrectes (déréférencement de pointeur du noyau ou de valeur nulle ou périmées, etc.

Notons, cependant que l'accès à la mémoire du noyau peut se produire également dans le cache du CPU, si les portions auxquelles on accède y sont disponibles. Ainsi des lectures ou des écritures à ces portions de mémoire peuvent être accomplie sans l'intervention de la MCH, mais aussi une autre façon d'accéder à la mémoire du noyau se fait via les périphériques au travers d'un bus d'entrée/sortie (E/S) supportant le DMA (*Direct Memory Access*) et engage par conséquent le MCH [**15]**.

Failles liées à l'accès au support de contrôle

La mémoire du support de contrôle est composée des registres et de la mémoire interne de la CPU (dont ses différents niveaux de cache), des registres du MCH, et enfin de certaines régions du BIOS (*Basic Input and Output System*). L'exécution du noyau repose en effet sur certaines fonctionnalités du BIOS [**12]**. Mis à part celles qui sont employées lors de l'initialisation du noyau, deux autres sont fréquemment utilisées au cours de l'exécution du noyau. Il s'agit d'une part des tables ACPI (*Advanced Configuration and Power Interface*) qui définissent notamment des méthodes de gestion de l'énergie et de la configuration de la plateforme matérielle (ces méthodes sont employées par le système d'exploitation). D'autre part, le BIOS contient également la routine de traitement de l'interruption SMI (*System Management Interrupt)* qui s'occupe de la gestion de certaines spécificités matérielles de la plateforme. Elle fait alors partie du support de contrôle. Notons que le BIOS est copié dans la mémoire principale lorsque la machine est démarrée [**7]**. Ainsi cette

partie de la mémoire du support de contrôle peut facilement être altérée et paralyser le fonctionnement du noyau d'un système d'exploitation.

Failles liées à l'accès aux périphériques

La CPU est le composant matériel principal qui accède aux périphériques d'un ordinateur. Il est cependant possible pour un périphérique d'accéder à certaines régions de mémoire d'autres périphériques. Nous commençons par présenter le cas de la CPU, avant de brièvement exposer le cas des accès entre périphériques [**11]**. La CPU accède aux périphériques pour d'une part les configurer et d'autre part pour employer leurs fonctions (*ces accès sont généralement regroupés sous la désignation d'accès d'entrée/sortie*). Trois principales méthodes d'accès sont fournies qui dépendent du périphérique auquel on accède [**16]** :

- Le mécanisme MMIO (*Memory-Mapped I/O*), qui accomplit la projection des registres dans l'espace d'adressage physique ;
- Le mécanisme PIO (*Programmed I/O*), qui accomplit la projection des registres dans un espace d'adressage séparé de 16 bits ;
- Le mécanisme PCI (*Peripheral Component Interconnect*), qui est employé pour accéder aux registres de configuration PCI, situés dans un troisième espace d'adressage et qui emploie le mécanisme PIO.

Notons également, que la routine SMI qui s'exécute dans le mode SMM de la CPU peut effectuer également des accès d'entrée/sortie à l'insu du noyau [**8]**. Ainsi, cette routine prévoit qu'un périphérique branché sur le même pont qu'un autre périphérique puisse accéder à certaines de ses régions de mémoire (*à l'exclusion de la mémoire dédiée à l'espace de configuration noyau*). Ce qui peut représenter un éventuel danger pour l'altération du noyau [**3]**.

CRITERES D'EVALUATION DE SECURITE

Dans le cadre d'un déploiement efficace d'un système informatique, il est important de mesurer la sécurité d'un système d'exploitation, pour savoir si on a obtenu un niveau de sécurité satisfaisant, pour identifier les points les plus critiques à surveiller ou à corriger, et définitivement pour estimer s'il est rentable de mettre en œuvre telle ou telle défense supplémentaire. Pour cela, nous avons préconisé de prime abord mettre en place un certain nombre des critères communs pour servir d'éléments de jugement. Les critères communs englobent deux parties bien séparées : *fonctionnalité* et *assurance*. Ils définissent une *cible d'évaluation* (*Target of Evaluation ou TOE*) [**6]** qui désigne le système ou le produit à évaluer, et la *cible de sécurité* (*Security Target ou ST*) qui contient l'ensemble des exigences de sécurité et de spécifications à utiliser comme base pour l'évaluation d'une cible d'évaluation identifiée. La cible de sécurité pour une cible d'évaluation représente la base d'entente entre développeurs et évaluateurs. Une cible de sécurité peut contenir les exigences d'un ou de plusieurs *profils de protection* (*Protection Profiles ou PP*) prédéfinis. Un profil de protection définit un ensemble d'exigences de sécurité et d'objectifs, indépendants d'une quelconque implémentation, pour une catégorie de cible d'évaluation.

L'intérêt de ces profils de protection est double : un développeur peut inclure dans une cible de sécurité un ou plusieurs profils de protection ; un client désirant utiliser un système peut également demander à ce que son système corresponde à un profil de protection particulier [**4]**, ceci lui évitant de donner une liste exhaustive de fonctionnalités et assurances qu'il exige du système. Une partie importante des critères communs est donc consacrée à la présentation détaillée de profils de protection prédéfinis. A ces critères communs, s'ajoutent autant trois indispensables méthodes dites « *méthodes d'évaluation* » :

- La première méthode d'évaluation est l'*analyse de risques* [**1]**, qui consiste à identifier les menaces auxquelles devra faire face le système d'exploitation (quels types d'attaquants, mais aussi quels phénomènes physiques : incendie, inondation, etc.), identifier les *vulnérabilités* du système face à ces menaces, et estimer les conséquences (surtout financières) qui résulteraient de la réalisation des menaces et de l'exploitation des vulnérabilités. Les risques sont alors évalués à partir d'une évaluation de la fréquence de réalisation des menaces et des conséquences.
- La deuxième méthode d'évaluation est basée *sur des critères d'évaluation*. Les critères portent à la fois sur des fonctionnalités de sécurité (authentification, autorisation, etc.) et sur des niveaux d'assurance, définis en fonction des méthodes de développement et de vérification. On s'appuie sur ces critères communs afin d'apprécier et de traiter les risques relatifs à la sécurité des systèmes d'information [**15]**.

- Enfin, la troisième méthode consiste à *utiliser des indicateurs quantitatifs de la sécurité*, permettant aux administrateurs de surveiller au jour le jour le niveau de sécurité et de prendre des mesures correctives en cas de baisse de ces indicateurs. Ceci permet d'évaluer la sécurité opérationnelle, c'est-à-dire correspondant à la façon d'utiliser les systèmes et ses mécanismes de protections, plutôt qu'à une vision statique comme celle donnée par l'analyse de risques ou les critères d'évaluation [**9]**.

RESOLUTIONS DES PROBLEMES DU SYSTEME NOYAU

Proposez des pistes de solutions pour la gamme des systèmes d'exploitation Windows est sans doute la tâche la plus complexe pour cette recherche, cependant, il sied de rappeler ici que, nous focaliserons toute notre attention sur les fonctionnalités du noyau qui fournissent directement le moyen d'écrire dans n'importe quelle région mémoire de l'espace noyau (tel que : le chargeur de modules noyau ou bien les périphériques virtuels), qui sont couramment employées par de nombreux maliciels pour s'injecter dans l'espace noyau. [16**]** Ces fonctionnalités doivent évidemment être contrôlées. Par exemple, les périphériques virtuels peuvent être désactivés e/ou filtrés afin d'autoriser uniquement les accès aux entrées/sorties projetées en mémoire (*comme le font les noyaux Linux actuels s'ils sont correctement configurés*). Aussi, afin de détecter les modules noyau malveillants, une solution est d'établir une vérification des modules avant chargement via l'utilisation de signatures cryptographiques. Cependant, de cette façon nous n'empêchons pas l'exploitation de bogues qui peuvent se trouver à l'intérieur des modules signés. De surcroît, nous devons garantir que la façon d'ajouter des modules est incontournable et ne peut être altérée [22**]**.

L'autre vecteur d'accès employé par les maliciels afin d'altérer la mémoire du noyau est l'exploitation de failles au sein de fonctionnalités du noyau qui n'ont pas pour vocation la modification de l'espace noyau. Évidemment, à la différence du vecteur d'accès précédent, celui-ci ne peut être contrôlé par les mêmes techniques. De plus, ce type de vecteur d'accès est d'autant plus facile à trouver dans le noyau que le nombre de modules (qui peuvent être potentiellement bogués) y étant ajoutés est important. En fait, la grande majorité des failles du noyau proviennent des pilotes de périphériques. La solution de sécurité serait de se focaliser sur des approches génériques de protection du noyau vis-à-vis d'actions malveillantes. Cependant, ces mécanismes sont actuellement implémentés au même niveau de privilège que le noyau et ainsi ne peuvent empêcher que l'entrée dans l'espace noyau de données malveillantes. Ils ne peuvent pas être efficaces si du code malveillant est déjà présent dans le noyau [**20]**.

En ce qui concerne les régions de code peuvent être rendues exécutables et non modifiables. De même, afin de se protéger des régions de données qui peuvent être rendues lisibles, modifiables et non exécutables. Cela peut être effectué par la modification des attributs des entrées des tables de pages. Un simple *patch* du noyau répond légitimement à cette problématique, et place dans des pages séparées le code et les données du noyau. Cependant, une action malveillante pourrait désactiver cette protection en changeant tout d'abord les attributs de pages d'une région mémoire de données qui contiendrait du code malveillant et ensuite exécuter cette région [**18]**. Ainsi, la modification des attributs de pages doit être impossible afin d'empêcher qu'une région de données deviennent une région de code. Nous pourrions empêcher que les tables de pages soient modifiées en spécifiant que les pages qui les contiennent ne soient pas modifiables. Mais alors, il ne serait plus possible pour le noyau d'ajouter de nouveaux *mappings* de mémoire (pour l'ajout de modules par exemple) étant donné que les pages contenant les tables de pages ne seraient plus modifiables. L'unique solution serait alors de créer de nouvelles tables de pages et de charger le registre avec l'adresse physique qui les référence. Mais cela peut aussi bien être effectué par un maliciel qui s'exécute au même niveau que le noyau. Finalement, afin de se protéger en partie contre cette vulnérabilité imminente, les solutions génériques de protection contre l'exploitation de débordements de tampon sont envisageables [**10]**, car elles protègent l'espace noyau vis-à-vis des modifications malveillantes extérieures au noyau. Ainsi, elles protègent contre le détournement du flux d'exécution vers une adresse spécifique en mémoire.

D'autres éventuelles complications recouvrent l'exécution du noyau d'un code spécifique dans une région de l'espace utilisateur qui provoque l'altération des variables de l'état d'exécution. Il est fondamental de protéger les données de contrôle de flux (par exemple : protéger les informations de contrôle de flux dans la pile, empêcher les pointeurs noyau d'être altérés, etc.), mais ce n'est pas suffisant [**28]**. Les variables de l'état d'exécution sont nombreuses, et il n'existe malheureusement pas une solution générique pour protéger le noyau contre des modifications inappropriées de ces variables. Mais des solutions pour certains types de variable existent. C'est par exemple, Au sujet des données de contrôle de flux, il est faisable de détourner le flux d'exécution vis-à-vis des débordements dans la pile, afin de détecter des débordements éventuels de tampon. Ce mécanisme sert également à empêcher l'exploitation des vulnérabilités associées, comme par exemple la libération d'une région de mémoire à deux reprises sans qu'il n'y ait eu de réallocation entre (vulnérabilité de type *double-free*).

CONCLUSION

Dans la présente recherche, il a été question de présenter un point de vue holistique concernant un cadre de déploiement et de sécurité du système d'exploitation de la gamme Windows (plus particulièrement son système noyau). Nous avons eu un réel plaisir de revisiter les quelques concepts relatifs au système d'exploitation en expliquant compendieusement les principes de fonctionnement et organisation internes de tout système d'exploitation. Un aperçu très précis a été effectué quant au fonctionnement et classification du système noyau. Finalement, la recherche s'est posée la limitation de discuter ses résultats en s'appuyant sur la diversité des notions y relatives telle que : des exigences de la sécurité informatique basée sur la gestion des identités, d'accès et des autorisations d'un système informatique ; les éventuelles attaques sur le système noyau ; les failles des systèmes noyaux ; les critères d'évaluation d'une sécurité appropriée et la proposition des quelques pistes des solutions aux problèmes de sécurité.

REMERCIEMENTS

L'équipe du présent projet de recherche tient à exprimer sa profonde gratitude envers tous ceux qui ont permis de parachever cette nouvelle prospection, en l'occurrence les différentes entreprises et internautes congolais. L'équipe tient également à remercier spécialement le responsable de ce projet, **Professeur Docteur YENDE Raphael Grevisse**, dont la contribution s'est avérée plus que satisfaisante lors de la rédaction et de l'interprétation des aboutissements de la présente recherche. Sans oublier l'effort collectif, nous félicitons tous les membres de ce projet.

CONFLITS D'INTERETS

Les auteurs de la présente recherche scientifique ne déclarent aucun conflit d'intérêts.

REFERENCES

1] A. Reed, « *The Definitive Guide to Identity Management* », Ebook, 2004
2] Andrew Tanenbaum, « *Systèmes d'exploitation* », Pearson, 2e éd., 2003,
3] ANSSI, « *Expression des Besoins et Identification des Objectifs de Sécurité* ». Rapport technique, 2004
4] B ARNETT John M, « An *analysis of the effects of personal computer use by Corporate Middle Management", Thèse* de doctorat, GGU California 1985.
5] B. Aboba, P. Calhoun, S. Glass, « *Criteria for Evaluating AAA Protocols for Network Access* », RFC 2989, IETF, Nov. 2000
6] BAILEY, James E. & PEARSON Sammy W, *"Development of a tool for measuring and analyzing computer user satisfaction"* Management Science, May 1983 Vol.29 No.5
7] Clemens Szyperski, "*Component Software : Beyond Object-Oriented Programming*. Addison-

Wesley, December 1997.
8] Daniel P. Bovet, Marco Cesati, « *Le Noyau Linux* », O'Reilly, 3e éd., août 2006,
9] Eric Lacombe, « thèse de doctorat sur Sécurité des noyaux de systèmes d'exploitation », 2010, p.16
10] F. Bimbard and L. George, "*Feasibility conditions with kernel overheads for mixed preemptive and non-preemptive periodic tasks with fp/fifo scheduling on an event driven system*", California, USA, 2006
11] F. Chong, « *gestion de l'identité et de l'accès* », Site web :http://msdn.microsoft.com/fr-fr/library/aa480030.aspx, publié 2004.
12] H. Cervantes, & R. S. Hall, « *Autonomous Adaptation to Dynamic Availability Using a Service-Oriented Component Model* », 26th ICSE, 2004
13] J. O. Toelen, *« Identity and Access Management* », Master thesis, 2008
14] J. Vacca, « *Computer and Information Security* », 2009
15] J.M. Autebert, « *Théorie des langages et des automates* », MASSON, 1994
16] James L. Peterson, Abraham S., Peter B. Galvin, "*Operating system concepts*", Addison-Wesley, 1990,
17] Kerberos, Site web: http://www.ietf.org/rfc/rfc4120.txt, 2011
18] L. Butti et J. Tinnes, « *Recherche de vulnérabilités dans les drivers 802.11 par techniques de fuzzing* », Rennes, France, juin 2007
19] L. George, « *Conditions de faisabilité pour l'ordonnancement temps réel préemptif et non préemptif* ». Ecole d'été sur le Temps Réel, ETR'2005,
20] Laszio Belady and Manny Lehman., "*A model of large program development, IBM Journal*", 1976
21] Laurent Bloch, *« Les systèmes d'exploitation des ordinateurs : histoire, fonctionnement, enjeux »*, Vuibert, 2003
22] M. Blanc, « *Sécurité des systèmes d'exploitation répartis : architecture décentralisée de métapolitique pour l'administration du contrôle d'accès obligatoire* », Thèse soutenue en 2006.
23] M. Kamel, « *Patrons organisationnels pour la sécurisation des Dispositions virtuelles* », PhD Thesis, 2008
24] N. Navet and B. Gaujal, "*Ordonnancement temps réel et minimisation de la consommation d'énergie. Chapter 4, Traité I2C Systèmes Temps Réel »,* volume 2, Hermès Science, June 2006.
25] RAYMOND Louis, « *Validité des systèmes d'information dans les PME, analyse et perspectives* " Institut de recherches politiques. Les presses de l'Université Laval. 1987
26] RAYMOND Louis, RIVARD S. & B. François, « *L'informatisation des PME - douze cas types* " P.U.L.1988.
27] Richard J. Moore, "*Dynamic probes and generalized kernel hooks interface for Linux. In Proceedings of the 4th Annual Linux Showcase and Conference"*, Atlanta, USA, October 2000
28] Robert Baron, Richard and Michael Young, "*An operating system environment for large-scale multiprocessor applications, IEEE Software", July 1985*
29] ROBEY, D, "Computer *Informations Systems and organization structure",* Communication of the ACM, Vol.24 No. 10 1981
30] X. Allamigeon et C. Hymans, « *Analyse Statique par Interprétation Abstraite : Application à la Détection de Dépassement de Tampon* », Rennes, France, juin 2007.
31] YENDE R. Grevisse, « *Sécurité informatique et cryptographie* », HAL, 2018 p.42

[32] YENDE R. Grevisse., "*Possible dangers of electromagnetic waves from cell phones and relay antennas on human health in DRC: Effects, Risks, Health and Protection",* EJMER, 10 (1), 26-45, 2023

[33] YENDE R. Grevisse et KASEKA K.V, *« Divergence possible des processus de Data mining et Knowledge Discovery in Databases* », European Journal of Natural and Social Sciences, EJNSS-NOVUS, 01(10), Jan 2023