



République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

**Université des Sciences et de la Technologie Houari Boumediene**

Faculté d'Electronique et d'Informatique

Département Informatique

Mémoire de Licence

Filière: Informatique

Spécialité:

**Informatique Académique**

---

**Thème**

**Réalisation d'un contrôleur d'intégrité des fichiers système UNIX**

---

**Sujet Proposé par :**

**Mme ALOUANE Lynda**

**Soutenu le : 05/06/2018**

**- ALIM Yanis**

**Présenté par :**

**- MELLAL Houdaifa**

**Devant le jury composé de:**

**Mme ZEBBANE Bahia**

**Mme MOSTEGHANEMI Hadia**

**Mme BOUCHENE Sabrina**

**Présidente**

**Membre**

**Membre**

Binôme n° : 110 / 2018

# Remerciements

Nous tenons à remercier tout d'abord notre promoteur Mme ALIOUANE lynda, d'avoir accepté de nous encadrer durant ce projet, de nous avoir beaucoup aidé et soutenu, pour sa disponibilité, sa patience, ses conseils, et surtout son encouragement durant les moments difficiles.

Nous remercions Mme ZEBBANE Bahia d'avoir accepté de présider le jury de soutenance de ce projet.

Nous adressons toute notre gratitude et reconnaissance à Mme MOSTEGHANEMI Hadia et Mme BOUCHENE Sabrina, qui ont accepté de faire partie de ce jury.

Nous désirons encore exprimer notre très vive reconnaissance aux enseignants de l'USTHB qui nous ont aidés et encouragés, de près ou de loin.

Je remercie toute ma famille, ma mère et mon père pour leurs conseils, leur soutien inconditionnel, leur contribution et leur patience, mon frère Lounes qui rend ma vie heureuse, mes sœurs Lydia et Katia, en leur souhaitant une vie pleine de bonheur. Je remercie Mr. BELLAL Arezki Mustapha qui m'a guidé pour être ce que je suis, pour ses conseils et son inquiétude pour mes études. Je remercie MELLAL Houdaifa mon binôme pour sa disponibilité et les beaux moments passés au cours de notre projet, tous mes amis Alla, Nazim, Soheib, Djalil, Haithem et mon frère Islam.

**Y.ALIM**

Je remercie mes très chers parents qui ont toujours été là pour moi, « Vous avez tout sacrifié pour vos enfants n'épargnant ni santé ni efforts. Vous m'avez donné un magnifique modèle de labeur et de persévérance. Je vous suis redevable d'une éducation dont je suis fier ». Je remercie mes sœurs pour leur encouragement. Je remercie mon cousin Zaki qui m'a beaucoup aidé. Je tiens à remercier mon binôme Yanis pour l'effort qu'il a fait, sa disponibilité et tous les moments qu'on a passés, je remercie mes chers amis Alla, Haitem et Nazim pour leur amitié, leur soutien inconditionnel et leur encouragement.

**H.MELLAL**

# Sommaire

<b>Liste des figures</b>	<b>6</b>
<b>Introduction générale</b>	<b>7</b>
<b>Chapitre I : Sécurité des Systèmes d'Information.</b>	<b>9</b>
1) Introduction	9
2) Définition de la Sécurité des Systèmes d'Informations (SSI)	9
3) L'importance de la sécurité de l'information	9
4) Les composants de la sécurité de l'information	10
5) Les critères de la sécurité	10
5.1) Les critères fondamentaux	10
5.1.1) Disponibilité	10
5.1.2) Intégrité	11
5.1.3) Confidentialité	11
5.2) Les critères complémentaires	11
5.2.1) Non répudiation	11
5.2.2) Identification	12
5.2.3) Authentification	12
6) Les dangers qui menacent les systèmes d'information	12
6.1) Les virus et programmes malveillants	13
6.2) La fuite d'information	13
6.4) Le piratage (hacking)	13
6.5) Le sabotage	14
7) La Cryptographie	14
7.1) définition	14
7.2) Le chiffrement	15
7.3) Fonction de hachage	15
7.4) Algorithmes de hachage	16
8) Des solutions pour protéger le système d'information	17
8.1) La politique de mot de passe	17
8.2) Identification	18
8.3) La sécurité du réseau local	18

<b>Chapitre II : Contrôleur d'Intégrité.</b>	<b>20</b>
1) Introduction	20
2) Définition d'un contrôleur d'intégrité	20
3) L'utilité d'un contrôleur d'intégrité	20
4) Les composants d'un contrôleur d'intégrité	21
4.1) Algorithme de hachage	21
4.2) Les attributs des fichiers	21
4.3) La base de donnée	23
5) Quelques contrôleurs d'intégrité	24
5.1) Simple script (Shell)	24
5.2) Tripwire (C++)	25
5.3) Advanced Intrusion Detection Environment «AIDE» (C)	25
5.4) Another File Integrity Checker «AFICK» (PERL)	25
5.5) Comparaison entre les trois contrôleurs:	26
6) Conclusion :	27
<b>Chapitre III: Super Integrity Checker (SIC)</b>	<b>28</b>
1) Introduction	28
2) Les outils utilisés	28
2.1) Le langage Python	28
2.1.1) Présentation	28
2.1.2) Les modules utilisés	29
2.2) L'Algorithme de Hachage	31
2.3) Les attributs des fichiers	31
2.4) La base de donnée PostgreSQL	33
2.4.1) Présentation	33
2.4.2) Comparaison avec MySQL et SQLite	33
2.4.3) Protection de la base de donnée	34
3) Les fonctionnements principaux de SIC	37
3.2) Choisir un mode de scan	39
3.2.1) Manual	39
3.2.2) Automatique	39
3.2.3) Temps réel	40
3.3) Choisir un type de scan	41
3.3.1) Scan Total (Long Scan)	41
3.3.3) Scan Personnalisé (Specific Scan)	42
3.4) La mise à jour de la BDD	43
4) Les fonctionnements secondaires	44

5) L'interface graphique	46
5.1) Page d'accueil	46
5.2) Analyse	46
5.3) Fichiers de log	47
5.5) À Propos	48
6) Conclusion	49
<b>Conclusion générale</b>	<b>50</b>
<b>BIBLIOGRAPHIE</b>	<b>52</b>

## Liste des figures

Figure 1.1 : Critères de la sécurité.....	10
Figure 1.2 : Science générique de la cryptographie.....	13
Figure 1.3 : Fonction de hachage itérative.....	14
Figure 2.1 : Les attributs d'un fichier sous Unix.....	21
Figure 3.1 : Les différents modules utilisés par SIC.....	27
Figure 3.2 : Les arguments possibles pour SIC.....	28
Figure 3.3 : Les algorithmes offerts par SIC.....	29
Figure 3.4 : Les attributs utilisés par SIC.....	30
Figure 3.5 : Serveur PostgreSQL utilisé par SIC.....	32
Figure 3.6 : Génération de clé secrète par SIC.....	33
Figure 3.7 : Demande de la clé secrète par SIC.....	33
Figure 3.8 : Sécuriser la connection à sic_db.....	33
Figure 3.9 : Demande d'adresse mail par SIC.....	34
Figure 3.10 : Message d'alerte de la part de SIC.....	34
Figure 3.11 : Architecture fonctionnelle du sic.....	35
Figure 3.12 : Les modes de scan disponibles.....	36
Figure 3.13 : Lancement d'un scan manuel.....	36
Figure 3.14 : Configuration d'un scan automatique (cronjob).....	37
Figure 3.15 : Les types de scan version graphique.....	38
Figure 3.16 : Les types de scan version console.....	38
Figure 3.17 : Les fichiers inclus dans un scan total.....	38
Figure 3.18 : Les fichiers inclus dans un scan rapide.....	39
Figure 3.19 : Lancement d'un scan personnalisé.....	39
Figure 3.20 : Détection de changement de permission.....	39
Figure 3.21 : Email du résultat de scan.....	40
Figure 3.22 : Lancement d'une comparaison d'architecture.....	41
Figure 3.23 : La fenêtre d'accueil.....	42
Figure 3.24 : La fenêtre d'analyse.....	43
Figure 3.25 : La fenêtre des fichiers log.....	43
Figure 3.26 : La fenêtre des paramètres.....	44
Figure 3.27: La fenêtre À propos.....	45

# Introduction générale

“Everything is a file” *ou bien* “Tout est fichier”, une description d’une des propriétés des systèmes Unix-like. Un document texte est évidemment un fichier, tout comme un document OpenOffice, une image, une vidéo ou un MP3. Mais qu’en est-il des répertoires ? Il s’agit aussi de fichiers d’un type particulier, qui contiennent des informations sur d’autres fichiers. Les lecteurs de disques sont de gros fichiers. Les connexions réseaux sont des fichiers, même les processus en cours d’exécution sont des fichiers. Bref, sous Unix, tout est fichier.

Les systèmes d'exploitation Unix-like sont en général considérés très bien protégés mais pas totalement immunisés. En effet, Une des vulnérabilités est générée par le fait que de nombreux utilisateurs imaginent qu’un système Unix-like n'est pas sensible aux virus, ou alors ils configurent leur ordinateur de façon non sécurisée, que ce soit par ignorance ou par maladresse, ce qui facilite l'accès à un pirate.

Comme tout est fichier, une fois le pirate a l'accès à la machine, il doit interagir avec des fichiers (binaire, service, config,... etc.). Quelles que soit ses actions, il essaiera de masquer son passage en supprimant les traces dans les journaux d'activités. Par ailleurs, il installe un certain nombre d'outils lui permettant de créer une porte dérobée (backdoors), afin de pouvoir revenir ultérieurement. Sa présence sur une machine peut être trahie par un certain nombre de commandes d'administration permettant d'afficher la liste des processus en cours qui sont jugés insuffisantes. Il existe ainsi des logiciels (appelés rootkits) chargés d'écraser la plupart des outils du système et de les remplacer par des commandes équivalentes masquant la présence du pirate. Il est donc aisé de comprendre qu'en l'absence de détérioration, il peut être très difficile pour un utilisateur de s'apercevoir qu'une machine a été compromise.

Afin de s'assurer de l'intégrité d'un système Unix-like, il faut assurer l'intégrité de ses fichiers, il est donc nécessaire de détecter les compromissions en amont. C'est ainsi l'objectif poursuivi par les contrôleurs d'intégrité.

Notre objectif est de réaliser un contrôleur d'intégrité qui aidera les utilisateurs à détecter toute sorte d'intrusion. En se basant sur des solutions déjà existantes, notre contrôleur doit satisfaire ce que les autres offrent comme service. En plus, on vise à ajouter des fonctionnalités qui n'existaient pas déjà et qui offrent plus d'efficacité à la

détection des changements anormales et plus de précision dans les valeurs changées et le temps de changement.

Nous allons partager notre mémoire en trois chapitres. Le premier chapitre contiendra l'état de l'art et une vue globale sur l'importance de la sécurité des systèmes informatisés. Le deuxième chapitre sera une étude d'une solution pour protéger le système d'information qui est le contrôleur d'intégrité, une analyse du fonctionnement principal de cette solution sera faite, suivi par une comparaison entre des solutions déjà existantes. Dans le dernier chapitre, on décrit notre contrôleur : Super Integrity Checker (SIC), on parlera des fonctionnalités principales et secondaires et les avantages que SIC possède par rapport aux solutions déjà implémentées.



# **Chapitre I : Sécurité des Systèmes d'Information.**

## **1) Introduction**

Avec l'évolution de la technologie et des sciences informatiques et leurs expansion dans tous les domaines, un nouveau risque est apparu en menaçant la sécurité des biens et des informations personnels (secret ou public) par plusieurs types d'attaques malveillantes. Cela a obligé les informaticiens à chercher une solution adéquate afin de développer une défense solide et une protection contre ces tentatives malicieuses nommée par la suite "Sécurité des Systèmes d'Information (SSI)".

## **2) Définition de la Sécurité des Systèmes d'Informations (SSI)**

La sécurité des systèmes d'information (SSI) ou plus simplement sécurité informatique, est l'ensemble des moyens techniques, organisationnels, juridiques et humains nécessaires à la mise en place des moyens visant à empêcher l'utilisation non-autorisée. [1]

## **3) L'importance de la sécurité de l'information**

L'information est aujourd'hui la base de l'entreprise. C'est ce qui fait à la fois sa force et son existence. Fichiers, bases de données, méthodes de travail et de fabrication, fiches des salariés et informations industrielles sont autant d'informations qui composent la structure et la base d'une entreprise. Il s'agit de son capital intellectuel, ou plutôt capital informationnel. Toute perte d'information peut porter un coup fatal à une entreprise ou même à une nation.

Si ces informations venaient à être perdues, volées ou à tomber dans les mains d'une autre entreprise, la donnée n'aurait plus de raison d'exister car elle ne serait plus exclusive. L'information a aujourd'hui de la valeur de par son côté unique et exclusif pour une entreprise. Il est donc dans l'intérêt de l'entreprise de protéger son patrimoine informationnel.

## 4) Les composants de la sécurité de l'information

Pour définir la sécurité de l'information, il faut étudier ses deux composants :

- **L'information**
- **La sécurité**

Le RFC 4949 nommé "Internet Security Glossary version 2" sortie en août 2007, version mise à jour du RFC 2828 définit ces deux éléments comme suit : [4]

**Information system** (I) Un ensemble organisé de ressources, de procédures informatique et de télécommunications (incluant équipements et services) qui permettent de créer, collecter, enregistrer, traiter, stocker, retrouver, afficher, contrôler ou de disposer d'informations afin d'accomplir une action spécifique.

**Information security** (INFOSEC) (N) Mesures qui permettent d'implémenter et d'assurer la sécurité des systèmes d'information incluant le système informatique ("computers systems") et les systèmes de communication.

## 5) Les critères de la sécurité

Il existe plusieurs critères pour assurer la sécurité de système d'information qui sont:

### 5.1) Les critères fondamentaux

Les critères fondamentaux (Figure 1.1) de la sécurité ou les **D.I.C** (Disponibilité, Intégrité, Confidentialité) sont trois termes permettant de fixer les lignes directrices dont se constitue la sécurité de l'information :

### 5.1.1) Disponibilité

La disponibilité d'une ressource est relative à la période de temps pendant laquelle le service offert est opérationnel. Le volume potentiel de travail susceptible d'être pris en charge durant la période de disponibilité d'un service, détermine la capacité d'une ressource à être utilisée (serveur ou réseau par exemple).[2]

### 5.1.2) Intégrité

Le critère d'intégrité des ressources physiques et logiques (équipements, données, traitements, transactions, services) est relatif au fait qu'elles n'ont pas été détruites (altération totale) ou modifiées (altération partielle) à l'insu de leurs propriétaires d'une manière intentionnelle ou accidentelle. Une fonction de sécurité appliquée à une ressource pour contribuer à préserver son intégrité, permettra de la protéger plus ou moins efficacement contre une menace de corruption ou de destruction. [2]

### 5.1.3) Confidentialité

Il s'agit de garantir le secret du document numérique transmis ou archivé. Ce service de sécurité consiste à s'assurer que seules les personnes autorisées peuvent prendre connaissance des données échangées. [5]

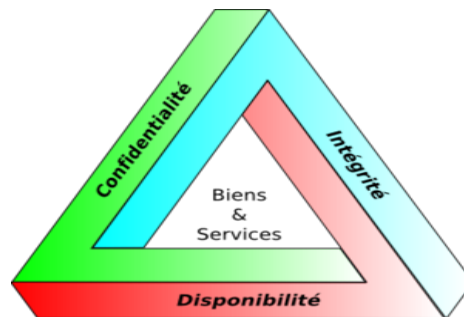


Figure 1.1 : Critères de la sécurité.

## 5.2) Les critères complémentaires

Dans le but d'améliorer la sécurité des systèmes, d'autres termes viennent compléter les fondations, cela vient du fait que les systèmes informatiques permettent aujourd'hui de combler des exigences supplémentaires et que ce sont les outils les plus performants pour gérer l'information :

### 5.2.1) Non répudiation

La non-répudiation est le fait de ne pouvoir nier ou rejeter qu'un événement (action, transaction) a eu lieu. À ce critère de sécurité peuvent être associées les notions d'imputabilité, de traçabilité ou encore parfois d'auditabilité [2].

- **L'imputabilité** : se définit par l'attribution d'une action (un événement) à une entité déterminée (ressource, personne). Elle peut être réalisée par un ensemble de mesures garantissant l'enregistrement fiable d'informations pertinentes par rapport à une entité et à un événement. [2]
- **La traçabilité** : permet de suivre la trace numérique laissée par la réalisation d'un événement (message électronique, transaction commerciale, transfert de données...). Cette fonction comprend l'enregistrement des événements, de la date de leur réalisation et leur imputation. Elle permet, par exemple, de retrouver l'adresse IP d'un système à partir duquel des données ont été envoyées. [2]
- **L'auditabilité** se définit par la capacité d'un système à garantir la présence d'informations nécessaires à une analyse ultérieure d'un événement (courant ou exceptionnel) effectuée dans le cadre de procédures de contrôle spécifiques et d'audit. Cet audit peut être mis en œuvre pour diagnostiquer ou vérifier l'état de la sécurité d'un système ou encore pour déterminer s'il y a eu ou non violation de la politique de sécurité et, éventuellement quelles sont les ressources compromises. [2]

### 5.2.2) Identification

Elle permet de connaître l'identité d'une entité. Le vérificateur vérifie l'information révélée par l'entité contre celles de toutes les entités connues. L'identification doit être unique. [3]

### 5.2.3) Authentification

C'est l'action qui consiste à prouver son identité. Ce service est généralement rendu par l'utilisation d'un "échange d'authentification" qui implique un certain dialogue entre les tiers communicants. Ce dialogue est appelé protocole d'authentification. [5]

## **6) Les dangers qui menacent les systèmes d'information**

Il existe plusieurs types de risques qui menacent les systèmes d'information, ci-dessous on liste les principaux dangers fréquemment trouvés :

### **6.1) Les virus et programmes malveillants**

Le terme « malware » (fusion de « malicious » pour malveillant et de « software » pour programme/logiciel) est désormais utilisé pour désigner tout programme malveillant présent sur un ordinateur ou un appareil mobile. Ces programmes sont installés à l'insu des utilisateurs et peuvent générer de nombreux effets indésirables, comme la paralysie des performances informatiques, l'exploitation des données personnelles du système, la suppression des données, voir même le dysfonctionnement du matériel contrôlé par ordinateur.

Les virus informatiques ont acquis ce nom en raison de leur capacité à « infecter » plusieurs fichiers sur un ordinateur. Ils se propagent sur les autres machines lorsque des fichiers infectés sont envoyés par e-mail ou lorsque des utilisateurs les transportent sur des supports physiques tels que des clés USB. [6]

### **6.2) La fuite d'information**

La fuite d'information est traditionnellement comprise comme accidentelle. Elle résulte de la négligence des employés qui dans le cadre de leur travail ou en dehors commettent des erreurs non intentionnelles de divulgation d'information. Caractéristique de l'erreur humaine, elle s'exprime par l'oubli d'un cellulaire ou d'un ordinateur portable dans un aéroport, par la discussion entre collègues au restaurant du coin de la rue, par l'envoi d'un message électronique à un mauvais destinataire. Il n'existe pas de définition connue, car elle est plutôt définie par opposition au vol de donnée en interne qui se caractérise par la malveillance, par opposition à l'acte de vol délibéré. On peut noter ici que, contrairement aux deux prochaines formes de perte, celle-ci n'est pas criminelle bien qu'elle peut avoir les mêmes conséquences si les données sont récupérées et utilisées illégalement. [7]

### **6.3) Le phishing**

Le phishing est une technique frauduleuse utilisée par les pirates informatiques pour récupérer des informations sensibles, personnelles et/ou confidentielles (coordonnées bancaires, vol d'identité...) appartenant à des internautes. Pour cela, ils reproduisent

parfaitement le design d'un site commercial légitime, d'un fournisseur d'accès à Internet, d'une banque, etc. [8]

## **6.4) Le piratage (hacking)**

Le piratage est un ensemble de techniques informatiques, visant à attaquer un réseau, un site, etc. Ces attaques sont diverses. On y retrouve :

- L'envoi de "bombes" logicielles.
- La recherche de trous de sécurité.
- Le détournement d'identité.
- La surcharge provoquée d'un système d'information.
- Changement des droits des utilisateurs d'un ordinateur.
- La provocation d'erreurs non gérées.

Les attaques peuvent être locales (sur le même ordinateur, voir sur le même réseau) ou distantes (sur internet, par télécommunication).[9]

## **6.5) Le sabotage**

Le sabotage informatique peut se définir comme du vandalisme informatique. On peut par exemple parler de sabotage informatique lorsque quelqu'un met délibérément un virus en circulation mais aussi lorsque quelqu'un détruit les données clients d'un concurrent même sans en tirer d'avantage financier. [10]

# **7) La Cryptographie**

## **7.1) définition**

La cryptographie est l'étude des méthodes permettant de transcrire des données intelligibles, en des données inintelligibles, par l'application de transformations mathématiques dont l'effet est réversible.

Ces transformations, basées le plus souvent sur l'arithmétique modulaire, désignent un processus appelé chiffrement (noté E), qui donne un texte chiffré C ou cryptogramme, à partir d'un texte en clair M. On a donc que:

$$E(M) = C$$

Inversement, le déchiffrement (noté D), est le processus qui permet de reconstruire le texte en clair à partir du texte chiffré. On a alors que:

$$D(C) = D(E(M)) = M$$

La figure 1.2 est un schéma qui montre le fonctionnement de la cryptographie. [11]



Figure 1.2: fonctionnement *de la cryptographie*.

## 7.2) Le chiffrement

Le chiffrement consiste à transformer une donnée (texte, message,...etc) afin de la rendre incompréhensible par une personne autre que celui qui a créé le message et celui qui en est le destinataire. La fonction permettant de retrouver le texte clair à partir du texte chiffré porte le nom de déchiffrement.

Les transformations de la cryptographie traditionnelle sont basées sur les caractères, en les remplaçant par d'autres caractères ou en transposant les caractères ou en faisant les 2 opérations plusieurs fois. [11]

## 7.3) Fonction de hachage

Une fonction de hachage ou fonction de condensation est une fonction qui convertit un message de longueur quelconque en une chaîne de taille inférieure et fixe, appelée empreinte ou condensé (ou digest en anglais) du message initial.

Une fonction de hachage à sens unique est une fonction de avec laquelle il est aisé de calculer l'empreinte d'un message donné, mais il est difficile d'engendrer des messages qui ont une empreinte donnée, et donc de déduire le message initial à partir de l'empreinte. On demande généralement en plus à une telle fonction d'être sans collision, c'est-à-dire qu'il soit impossible de trouver deux messages ayant la même empreinte. En fait, on utilise souvent le terme fonction de hachage pour désigner une fonction de hachage à sens unique sans collision.

La plupart des fonctions de hachage sont construites par itération d'une fonction de compression : le message  $M$  est décomposé en  $n$  blocs  $m_1, \dots, m_n$ , puis une fonction de compression  $f$  est appliquée à chaque bloc et au résultat de la compression du bloc précédent ; l'empreinte notée  $h(M)$  est le résultat de la dernière compression. (figure 1.3) [5]

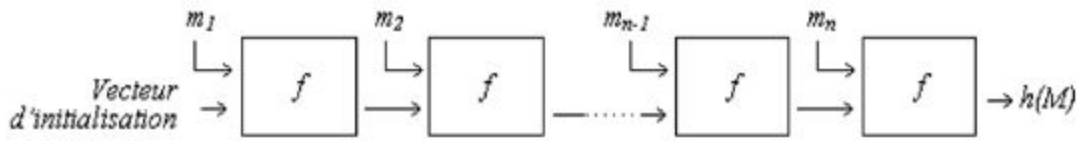


Figure 1.3 : *Fonction de hachage itérative*

## 7.4) Algorithmes de hachage

- **MD5** : Produit par Ronald Rivest, c'est le dernier d'une série (MD2, MD4). Cet algorithme produit un condensé de 128 bits. Il était il y a encore quelques temps l'algorithme de hachage le plus largement répandu. La cryptanalyse et l'attaque par force brute (2004) l'ont affaibli. Ses spécifications sont disponibles sur Internet dans le RFC 1321. [12]
- **SHA-1** (*Secure Hash Algorithm*) : c'est la norme du gouvernement Américain pour le hachage. SHA-1 est une amélioration de SHA qui produit une empreinte de 160 bits à partir d'un message de longueur maximale de  $2^{64}$  bits. Tout comme MD5, SHA-1 travaille sur des blocs de 512 bits. [5]
- **SHA-2** : est une famille de fonctions de hachage qui ont été conçues par la NSA, sur le modèle des fonctions SHA-1 et SHA-0, elles-mêmes fortement inspirées de la fonction MD4 de Ron Rivest . Telle que décrite par le NIST, elle comporte les fonctions, SHA-256 et SHA-512 dont les algorithmes sont similaires mais opèrent sur des tailles de mot différentes (32 bits pour SHA-256 et 64 bits pour SHA-512), Le dernier suffixe indique le nombre de bits de l'empreinte. [13]
- **SHA3** : est une fonction de hachage cryptographique conçue par Guido Bertoni, Joan Daemen, Michaël Peeters et Gilles Van Assche à partir de la fonction RadioGatún. SHA-3 est issu de la NIST hash function compétition qui a élu l'algorithme SHA3 le 2 octobre 2012. Elle n'est pas destinée à remplacer SHA-2, qui n'a à l'heure actuelle pas été compromise par une attaque significative, mais à fournir une autre solution à la suite des possibilités d'attaques contre les standards MD5, SHA-0 et SHA-1.[14]



Algorithme	La taille en sortie	La taille interne	La taille de bloc	La taille max de message	Rondes	Opérations	Bits de sécurité	Exemple performance (MiB/s)	Date de réalisation
MD5	128	128 ( 4 x 32 )	512	Illimité	64	AND XOR ROT OR ADD(mod $2^{32}$ )	<64 (collusion trouvée)	335	1992
SHA-0	160	160 ( 5 x 32 )	512	$2^{64} - 1$	80		<80 (collusion trouvée)	-	1993
SHA-1	160	160 ( 5 x 32 )	512	$2^{64} - 1$	80		<63 (collusion trouvée)	192	1996
SHA-2 (256)	256	256 ( 8 x 32 )	512	$2^{64} - 1$	64	AND XOR ROT OR ADD(mod $2^{32}$ ) Shr	128	139	2001
SHA-2 (512)	512	512 ( 8 x 64 )	1024	$2^{128} - 1$	80	AND XOR ROT OR ADD(mod $2^{64}$ ) Shr	256	154	2001
SHA-3 (256)	256	1600 (5x5x 64)	1088	illimité	24	AND XOR ROT NOT	128	-	2015
SHA-3(512)	512	1600 (5x5x 64)	567	illimité	24		256	-	2015

Tableau 1.1 : Comparaison entre les algorithmes de hachage. [16]

## 8) Des solutions pour protéger le système d'information

### 8.1) La politique de mot de passe

L'accès à un poste de travail informatique ou à un fichier par identifiant et mot de passe est la première des protections. Le mot de passe doit être individuel, difficile à deviner et rester secret. Il ne doit donc être écrit sur aucun support. La DSI ou le responsable informatique devra mettre en place une politique de gestion des mots de passe rigoureuse : un mot de passe doit comporter au minimum 8 caractères incluant chiffres, lettres et caractères spéciaux et doit être renouvelé fréquemment (par exemple tous les 3 mois). Le système doit contraindre l'utilisateur à choisir un mot de passe différent des trois qu'il a utilisés précédemment. Généralement attribué par l'administrateur du système, le mot de passe doit être modifié obligatoirement par l'utilisateur dès la

première connexion. Enfin, les administrateurs des systèmes et du réseau doivent veiller à modifier les mots de passe qu'ils utilisent eux-mêmes. [15]

## **8.2) Identification**

L'accès aux données personnelles traitées dans un fichier doit être limité aux seules personnes qui peuvent légitimement y avoir accès pour l'exécution des missions qui leur sont confiées. De cette analyse, dépend « le profil d'habilitation » de l'agent ou du salarié concerné. Pour chaque mouvement ou nouvelle affectation d'un salarié à un poste, le supérieur hiérarchique concerné doit identifier le ou les fichiers auxquels celui-ci a besoin d'accéder et faire procéder à la mise à jour de ses droits d'accès. Une vérification périodique des profils des applications et des droits d'accès aux répertoires sur les serveurs est donc nécessaire afin de s'assurer de l'adéquation des droits offerts et de la réalité des fonctions occupées par chacun. [15]

## **8.3) La sécurité du réseau local**

Un système d'information doit être sécurisé vis-à-vis des attaques extérieures. Un premier niveau de protection doit être assuré par des dispositifs de sécurité logique spécifiques tels que des routeurs filtrants (ACL), pare-feu, sonde anti intrusions, etc. Une protection fiable contre les virus et logiciels espions suppose une veille constante pour mettre à jour ces outils, tant sur le serveur que sur les postes des agents. La messagerie électronique doit évidemment faire l'objet d'une vigilance particulière. Les connexions entre les sites parfois distants d'une entreprise ou d'une collectivité locale doivent s'effectuer de manière sécurisée, par l'intermédiaire des liaisons privées ou des canaux sécurisés par technique de « tunneling » ou VPN (réseau privé virtuel). Il est également indispensable de sécuriser les réseaux sans fil compte tenu de la possibilité

d'intercepter à distance les informations qui y circulent : utilisation de clés de chiffrement, contrôle des adresses physiques des postes clients autorisés, etc. Enfin, les accès distants au système d'information par les postes nomades doivent faire préalablement l'objet d'une authentification de l'utilisateur et du poste. Les accès par internet aux outils d'administration électronique nécessitent également des mesures de sécurité fortes, notamment par l'utilisation de protocoles IPsec, SSL/TLS ou encore HTTPS. [15]

## **9) Conclusion**

Pour assurer la sécurité des systèmes d'informations, il faut construire une base de sécurité solide en combinant les critères essentiels de la sécurité et quelques critères complémentaires.

Nous nous intéressons dans le prochain chapitre au deuxième critère essentiel de la sécurité des systèmes d'information "l'intégrité" en étudiant un outil qui permet d'assurer l'intégrité des fichiers nommé contrôleur d'intégrité.

# **Chapitre II : Contrôleur d'Intégrité.**

## **1) Introduction**

L'étude d'une solution quelconque nécessite une connaissance de son fonctionnement, ses composants, ses services et une comparaison entre différents produits déjà réalisés. Dans ce chapitre, nous allons définir un contrôleur d'intégrité, son utilité, ses composants et à la fin décrire quelques contrôleurs existants.

## **2) Définition d'un contrôleur d'intégrité**

Contrôleur d'intégrité est un programme qui performe la vérification de la validité des fichiers système d'un système d'exploitation ou d'une application en utilisant une méthode de vérification entre l'état actuel du fichier et une son bon état stocké dans une base. Cette comparaison implique souvent un calcul d'une somme de contrôle cryptographique du fichier dans son état original et de la comparer avec la somme de contrôle calculée de l'état actuel du fichier. D'autres attributs de fichier peuvent également être utilisés pour surveiller l'intégrité.

Généralement, l'exécution d'un contrôle d'intégrité des fichiers est automatisée à l'aide d'autres applications externes ou des processus internes. Une telle surveillance peut être effectuée de manière aléatoire, à un intervalle d'interrogation défini, ou en temps réel. [17]

## **3) L'utilité d'un contrôleur d'intégrité**

Avoir une «défense en profondeur» est un élément essentiel dans toute architecture de sécurité. La défense en profondeur signifie que plusieurs niveaux de défense ont été établis et chaque niveau fournit une défense de secours dans le cas où les niveaux les plus haut sont défaits.

Le contrôleur d'intégrité est une partie importante d'une défense en profondeur, il sert à déclencher l'alarme si les couches extérieures de la défense ont été pénétrées. il n'empêche pas une attaque, mais il détecte les modifications non autorisées aux fichiers système de l'hôte, il détecte aussi une intrusion réussite.

Les pirates malveillants ont développé de nombreuses techniques pour cacher leur activité sur une machine compromise. Si celle-ci est sans contrôleur d'intégrité, une activité malicieuse de l'attaquant sur la machine ne peut jamais être détectée. En cas de compromission, un contrôleur d'intégrité peut aider à identifier quels fichiers ont été endommagés pendant l'attaque, aidant ainsi au nettoyage et réparation. [17]

## 4) Les composants d'un contrôleur d'intégrité

### 4.1) Algorithme de hachage

L'utilité d'un hachage unidirectionnel réside dans l'unicité de la valeur de sortie, si un bit de la donnée d'entrée change implique que la donnée de sortie change. En addition, les données d'entrée d'origine ne peuvent pas être dérivées de la valeur de sortie.

De ce fait, un hache aidera le contrôleur d'intégrité à identifier si un fichier a été modifié, il est considéré comme l'empreinte digitale des fichiers.

### 4.2) Les attributs des fichiers

En plus de générer des haches pour vérifier le contenu des fichiers système, Plusieurs contrôleurs d'intégrité gardent trace des nombreux attributs utilisés par la machine hôte. En général, dans les systèmes d'exploitation populaires (UNIX, WINDOWS, MAC) ces attributs sont les suivants :

**a-Index de noeux** : ou inode (contraction de l'anglais index et node) est une structure de données contenant des informations à propos d'un fichier ou répertoire stocké dans certains systèmes de fichiers (notamment de type Linux/Unix). À chaque fichier correspond un numéro d'inode (i-number) dans le système de fichiers dans lequel il réside, unique au périphérique sur lequel il est situé.

Chaque fichier a un seul inode, même s'il peut avoir plusieurs noms (chacun de ceux-ci fait référence au même inode). Chaque nom est appelé link.

**b-Type** : permet de connaître si le fichier est un :

1.     **-** : regular file
2.     **d** : directory
3.     **c** : character device file
4.     **b** : block device file
5.     **s** : local socket file
6.     **p** : named pipe

7. **l** : symbolic link

**c-Propriétaire** : ou bien Owner en anglais, il indique le propriétaire de fichier, et il est unique.

**d-Groupe** : c'est souvent utilisé sur les systèmes UNIX, il permet de regrouper plusieurs utilisateurs sur un seul groupe pour attribuer par la suite les mêmes droits.

**e-Les permissions** : constituent un système simple de définition des droits d'accès aux ressources, représentés par des fichiers disponibles sur un système informatique. Elles restent le moyen le plus utilisé pour définir les droits des utilisateurs sur les systèmes de type UNIX.

Unix applique les droits d'accès en définissant trois catégories d'utilisateurs, et en donnant à chaque classe une combinaison de trois types d'accès. Les classes d'utilisateurs sont:

-Propriétaire : Le propriétaire du fichier. C'est le compte d'utilisateur qui a été utilisé pour créer le fichier.

-Groupe : Il s'agit du groupe Unix associé au fichier. Tous les utilisateurs qui composent ce groupe auront accès à ce fichier.

-Autre : Cette catégorie représente toute personne qui n'est pas propriétaire ou dans le groupe.

-root : est un compte spécial Unix, souvent appelé "super utilisateur". L'accès au fichier pour le compte root n'est pas explicitement spécifié dans les autorisations de fichier car il a un accès complet et sans restriction à n'importe quel fichier.

Les trois types d'accès accordés à chaque classe d'utilisateur sont:

1. Lire : Cet accès accord aux classes d'utilisateurs la possibilité de lire le contenu d'un fichier.

2. Écrire : l'accès en écriture accorde à une classe d'utilisateurs la possibilité de modifier le fichier.

3. Exécuter : C'est le type d'accès requis pour exécuter un programme ou un script.

**f-Horodatages** : trois dates sont associées pour chaque fichier :

1- Date d'accès : dernière date où les données du fichier sont lues.

2- Date de modification : dernière date où le contenu de fichier est modifié.

3- Date de changement : dernière date où les attributs du fichier sont changés.

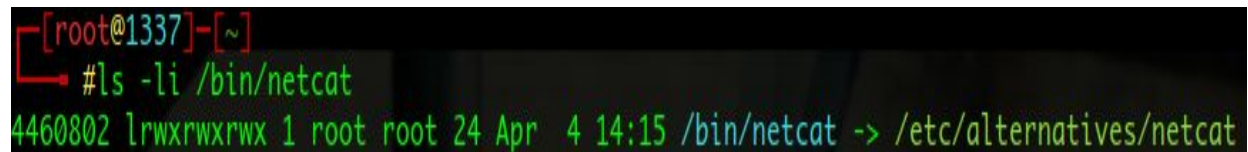
**g-Taille** : la taille du fichier en octets.

**h-Nombre de référence** : un seul fichier peut être référencé par plusieurs noms en utilisant le mécanisme des liens.

Cet attribut compte le nombre de fichiers lien vers ce fichier.

**i-Nom du lien** : si un fichier est un lien vers un autre, alors il aura un attribut qui est le nom du fichier où il pointe.

**Exemple** : la commande « ls » avec l'argument « li » et « nom\_fichier » sert à lister les attributs du fichier avec leurs index :



```
[root@1337]-[~]  
#ls -li /bin/netcat  
4460802 lrwxrwxrwx 1 root root 24 Apr  4 14:15 /bin/netcat -> /etc/alternatives/netcat
```

Figure 2.1 : Les attributs d'un fichier sous Unix.

Dans cet exemple:

- Le numéro d'inode est 4460802
- Les permissions de fichiers sont 'l rwx rwx rwx', ce qui se traduit par :
  - o Le fichier est un fichier de type lien : ('l --- --- ---')
  - o Le propriétaire a un accès en lecture, écriture et exécution: ('- rwx --- ---')
  - o Le groupe a un accès en lecture, écriture et exécution : ('- --- rwx ---')
  - o Les autres ont un accès en lecture, écriture et exécution : ('- --- --- rwx')
- Le nombre de référence sur le fichier est 1 (c'est-à-dire qu'il n'y a aucun autre fichier lié à celui-ci)
- Le propriétaire du fichier est root
- Le groupe propriétaire du fichier est root
- La taille du fichier est 24 octets
- La date et le temps de la dernière modification est : 4 avril à 14:15
- Le fichier est un lien vers un autre fichier qui a le chemin /etc/alternatives/netcat

### 4.3) La base de donnée

La création d'une base de données qui contient les résultats des haches et les attributs des fichiers système est une caractéristique commune entre tous les contrôleurs d'intégrité. Cette base de données est une cible pour chaque pirate car s'il réussit à la modifier, il peut agir et modifier ce qu'il veut sur le système sans qu'il soit détecté. Il existe trois techniques pour protéger le fichier d'information de base de données :

**1- Créer la base de donnée en mode Read-Only** : l'utilisateur n'est pas autorisé à faire autre chose que la lecture, cette méthode est faite en stockant le fichier d'information de la bdd dans un matériel de stockage (USB,CD-DVD ROM,Disque Dur),

ce qui permet de donner l'accès et le droit de modification seulement aux gens qui possèdent le support physique, c'est une méthode qui fonctionne parfaitement dans le cas d'un petit nombre d'ordinateurs, ou bien dans le cas des ordinateurs qui nécessitent rarement des modifications. Sinon, si la base de données nécessite des mises à jour fréquentes, cette méthode pourra devenir consommable de temps.

**2- Hacher la base de données** : cette technique consiste à garder un hachage de la base de données dans le but de détecter tout changement de la base de données.

**3- Stocker la base de données dans un serveur** : cette méthode consiste à maintenir une base de données par distance dans un serveur dédié, elle est plus sécurisée car même si le pirate est en contrôle discret de la machine, il ne pourra pas modifier la base initiale qui est stockée dans le serveur distant.

Certains contrôleurs d'intégrité combinent entre la deuxième et la troisième technique (hacher la base de données et aussi la garder dans un serveur distant) qui est considéré comme la meilleure technique. [17]

## 5) Quelques contrôleurs d'intégrité

Il existe de nombreux programmes qui tentent de vérifier l'intégrité des fichiers des systèmes. Ils diffèrent de simples scripts shell jusqu'à des programmes très sophistiqués et des systèmes client-serveur complexes avec des interfaces utilisateurs graphiques et des bases de données relationnelles.

### 5.1) Simple script (Shell)

Probablement, le premier contrôleur d'intégrité a été un script similaire à celui-ci :

```
#!/bin/sh
ls -lr / > /tmp/current-files
diff -qr /etc/baseline-files /tmp/current-files
```

- `#!/bin/sh` : est l'en-tête du script qui indique le langage qui est le Shell.
- `ls -lr / > /tmp/current-files` : liste récursivement le répertoire racine « / » et redirige le résultat vers le fichier « `/tmp/current-files` »
- `diff -qr /etc/baseline-files /tmp/current-files` : extraire la différence entre le fichier « `/etc/baseline-files` » qui est la liste récursive du répertoire racine considéré comme la base de données initiale et le fichier « `/tmp/current-files` » qui est la liste récursive du répertoire racine en état actuel.



Bien que ce script peut probablement détecter des changements (ajout/suppression) dans plusieurs répertoires mais ce n'est pas suffisant car des attaques contre le contenu et les attributs des fichiers sont possibles. [18]

## **5.2) Tripwire (C++)**

Tripwire a été initialement créé comme un projet d'étudiant à la fin des années 1980 par Gene Kim et Eugene Spafford. Sa sortie initiale était en 1992 et en 1997 comme Tripwire Inc. Il utilise une base de données et plusieurs messages digest pour la protéger, utilisant MD5, SHA1 et d'autres algorithmes de hachages.

Tripwire est capable de détecter les modifications apportées à plusieurs propriétés de fichier, telles que les horaires d'accès et de modification, l'ID de l'utilisateur du propriétaire du fichier, les blocs alloués, le nombre de liens, l'augmentation de la taille du fichier et le numéro d'inode. En outre, Tripwire permet le hachage de la base de données et il est capable d'envoyer des rapports d'intégrité par e-mail.

La version commerciale de Tripwire comprend des fonctionnalités supplémentaires, telles que la gestion centralisée. [19]

## **5.3) Advanced Intrusion Detection Environment «AIDE» (C)**

AIDE était initialement développé en tant que logiciel gratuit similaire à Tripwire sous Licence Publique Générale GNU (GPL). Les principaux développeurs sont Rami Lehti et Pablo Virolainen, tous deux associés à l'Université de technologie de Tampere ainsi que Richard van den Berg, un consultant indépendant en sécurité.

AIDE prend à un «instant donnée» l'état du système, enregistre tout ce qui est lié aux fichiers système.

Lorsque l'administrateur souhaite exécuter un test d'intégrité, l'administrateur place la base de données précédemment générée en un lieu accessible et entre une commande AIDE afin de comparer la base de données avec l'état réel du système. Toute modification qui se serait produite sur l'ordinateur entre la création de l'instantané et le test sera détectée par AIDE et sera signalée à l'administrateur. AIDE peut être configuré pour s'exécuter de façon planifiée et signaler quotidiennement les changements grâce aux technologies d'ordonnancement comme le cronjob : processus exécuté en arrière-plan dans des temps spécifiés par l'utilisateur. [20]

## **5.4) Another File Integrity Checker «AFICK» (PERL)**

Afick est un projet open source créé par Eric Gerbier. Il est conçu pour être efficace et portable, et il prend en charge plusieurs plates-formes de système d'exploitation, telles que : Windows XP et 2000, Linux Debian et Fedora, et d'autres systèmes d'exploitation basés sur POSIX.

Afick est similaire à Tripwire et AIDE en manière de réaction en cas de détection d'un changement. Cependant, il ne supporte que MD5 et SHA1 pour le calcul du hache. Les propriétés vérifiées par Afick sont comparables à Tripwire et AIDE. Cependant, Afick fournit une interface utilisateur graphique par opposition aux deux autres. [21]

### 5.5) Comparaison entre les trois contrôleurs:

Un tableau comparatif est nécessaire pour voir la différence entre ces trois contrôleurs d'intégrité :

Critères	AFICK	Tripwire	AIDE
Date de réalisation	2013	2011	2012
Efficacité	6	4	8
Temps de réponse	4	5	7
Prérequis	3	9	6
Facilité d'utilisation	5	12	9
Sécurité	4	5	7
Total (moins de points est meilleur)	21	35	37

Tableau 2.1 : Comparaison entre Tripwire, AIDE et AFICK. [22]

Tripwire est le plus rapide dans les scans et efficace pour la génération d'une base de données optimale et aussi à la protection des fichiers liées aux scans mais il est le plus complexe dans l'installation, la configuration et les mises à jour. Il est non portable (Unix-like système d'exploitation seulement).

AIDE est plus facile à installer et plus portable que Tripwire, contrairement à Tripwire, il est lent et consomme plus de CPU.

Malgré qu'AFICK est le plus lent à scanner les fichiers, il est le plus efficace dans ses scans avec une bonne protection de la base de donnée. Il est aussi le plus facile à manipuler et il prend en charge plusieurs plates-formes de système d'exploitation (Windows, UNIX, POSIX). Afick est le moins consommable de CPU et aussi l'unique à avoir une détection en temps réel ce qui le caractérise comme le meilleur entre ces trois contrôleurs d'intégrité.

## **6) Conclusion :**

Dans le but d'augmenter la sécurité de système d'exploitation et dans le cadre de combattre les tentatives malicieuses, un contrôleur d'intégrité semble une solution très intéressante avec les services qu'il peut offrir.

Dans le chapitre suivant, on va présenter notre propre solution en utilisant le meilleur ensemble d'outils possible et en profitant des solutions existantes dans le but d'assurer un bon contrôle des fichiers.

# **Chapitre III: Super Integrity Checker (SIC)**

## **1) Introduction**

Pour réaliser un bon contrôleur d'intégrité, il faut faire le bon choix au niveau des algorithmes de hachages, le type et la protection de la base de données et les attributs essentiels des fichiers. Aussi, une augmentation de l'efficacité et du temps de réponse est nécessaire.

Nous avons réalisé un contrôleur d'intégrité nommé Super Integrity Checker (SIC) qu'est un contrôleur d'intégrité de tout type de fichiers (fichiers systèmes spécialement) destiné aux systèmes d'exploitation Unix-like.

SIC existe en deux versions, une version console pour les administrateurs et une version graphique pour les utilisateurs simples. Au début, on expliquera ce qui se passe en arrière-plan (version console), puis on donnera la version graphique.

En parallèle, nous allons voir en détails tous les outils utilisés, les fonctionnements principaux ainsi que les fonctionnements secondaires de notre contrôleur.

## **2) Les outils utilisés**

La réalisation de notre logiciel s'est faite en choisissant les meilleurs outils pour atteindre une solution qui répond bien aux exigences de la sécurité. Dans ce qui suit, nous allons présenter les différents outils utilisés

### **2.1) Le langage Python**

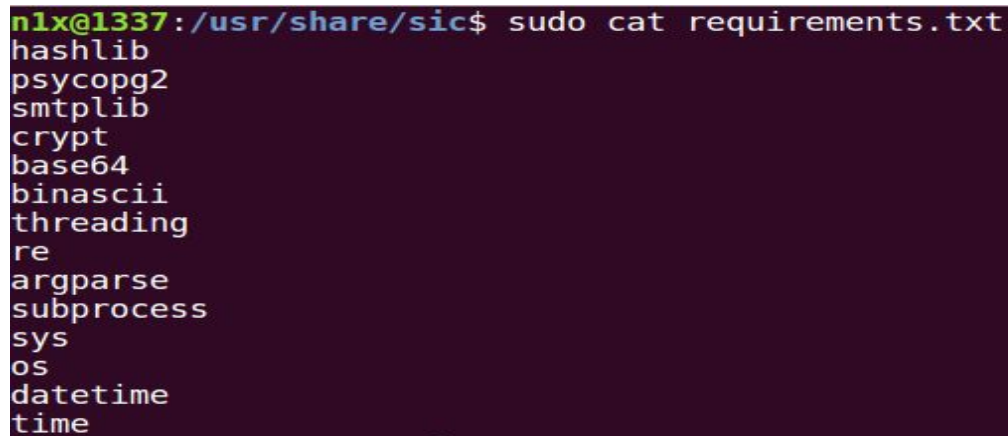
#### **2.1.1) Présentation**

Python est un langage portable, dynamique, extensible, qui permet une approche modulaire et orientée objet de la programmation. Il est considéré comme un langage de haut niveau généraliste et peut être utilisé pour la création de scripts destinés à l'administration système ou à la manipulation de fichiers textuels. Sa syntaxe est très simple, ce qui donne un code très lisible et ses performances sont très honorables pour un langage dit script, il raccourcit le cycle de développement par rapport aux langages compilés et permet un prototypage rapide des projets. Python contient une infinité de bibliothèques (modules) pour plusieurs domaines, ce qui rend le code final très petit par rapport aux autres langages [1].

Résultat, Nous avons choisis Python comme langage principale.

### 2.1.2) Les modules utilisés

Les modules en python sont des bibliothèques à importer pour utiliser leurs propre fonctions et/ou variables. La figure 3.1 montre les modules utilisés pour la réalisation du Super Integrity Checker.

A screenshot of a terminal window with a dark background. The prompt is 'n1x@1337:/usr/share/sic\$'. The command 'sudo cat requirements.txt' has been executed, and the output lists the following Python modules: hashlib, psycpg2, smtplib, crypt, base64, binascii, threading, re, argparse, subprocess, sys, os, datetime, and time.

```
n1x@1337:/usr/share/sic$ sudo cat requirements.txt
hashlib
psycpg2
smtplib
crypt
base64
binascii
threading
re
argparse
subprocess
sys
os
datetime
time
```

Figure 3.1 : Les différents modules utilisés par SIC.

**a- hashlib** : offre les différentes fonctions de hachage (MD5, SHA1-5, HMAC, etc...).

Utilisé pour hacher les fichiers et leurs attributs.

**b- psycpg2** : le pilote de la base de données PostgreSQL.

Utilisé pour se connecter à la base de donnée de SIC.

**c- smtplib** : le pilote de messagerie.

Utilisé pour envoyer des rapports de scans et des alertes.

**d- crypt** : offre les différentes fonctions de cryptage.

Utilisé pour sécuriser l'authentification de l'utilisateur à la base de données.

**e- base64** : sert à encoder un texte vers l'encodage base64.

Utilisé pour encoder l'email de l'utilisateur pour des raisons de sécurité.

**f- binascii** : convertisseur du binaire vers l'ascii.

Utilisé pour encoder l'email de l'utilisateur pour des raisons de sécurité.

**g- threading** : offre la fonction de multithreading.

Utilisé pour augmenter le temps de réponse.

**h- re** : offre le contrôle des expressions régulières.

Utilisé pour filtrer les fichiers à ne pas inclure dans les scans.

**i- argparse** : offre le contrôle des arguments passés à la commande.

Utilisé pour ajouter/gérer les arguments passé à la commande (Figure 3.2).

**j- subprocess** : offre la possibilité de créer des processus fils.

Utilisé pour créer des processus qui se chargent de la messagerie.

**k- sys** : offre l'accessibilité aux fonctions du système.

Utilisé pour gérer le nombre des arguments d'entrées et la valeur de l'argument de sortie.

**l- os** : donne accès aux routines du système.

Utilisé pour gérer le déplacement entre les répertoires système.

**m- datetime** : convertisseur d'une date en secondes vers une date lisible par l'utilisateur.

Utilisé pour afficher les temps des scans.

**n- time** : Offre la manipulation des fonctions de temps.

Utilisé pour afficher le temps de modifications des fichiers.

```
n1x@1337:~$ sic
usage: sic [-h] [-u] [-c] [-t {SHORT|LONG|SPECIFIC}]
          [-algo {SHA1|SHA256|SHA512}] [-dir [dir [dir ...]]] [-cron] [-real]
          [-arch] [-r] [-v]

Super Integrity Checker

Possible arguments:
-u, --update           Update the baseline database
-c, --check            Launch a check
-t {SHORT|LONG|SPECIFIC}, --type {SHORT|LONG|SPECIFIC}
                        Set the check type
-algo {SHA1|SHA256|SHA512}, --hash-algorithm {SHA1|SHA256|SHA512}
                        Set a hash-algorithm type
-dir [dir [dir ...]], --directories [dir [dir ...]]
                        Add specific directories
-cron, --cronjob       Add a cronjob for the check operation
-real, --real-time     Enable real-time detection
-arch, --architecture-difference
                        Check for architecture difference (added/deleted)
                        files
-r, --recommended     Show recommended options
-v, --verbose          Verbose output

Authors: <alimyanis8496@gmail.com> <Hodaifa.mellal@gmail.com>
```

Figure 3.2 : Les arguments possibles pour SIC.

Le manuel ci-dessus aide l'utilisateur à comprendre les différentes fonctions offertes par SIC :

-h ou --help : voir la page d'aide.

-u ou --update : mettre à jour la base de donnée initiale.

-c ou --check : lancer un scan.

-t ou --type : préciser le type de scan.

-algo ou --hash-algorithm : préciser l'algorithme de hachage à utiliser.

-dir ou --directories : spécifier des répertoires.

-cron ou --cronjob : configurer un cronjob qui lancera les scans automatiques.

-arch ou --architecture : lancer une comparaison d'architectures.

-r ou --recommended : afficher les options recommandées.

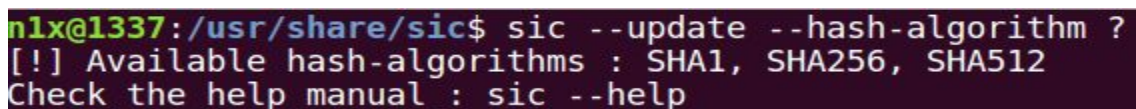
-v ou --verbose : afficher des détails sur ce qui se passe en arrière-plan.

## 2.2) L'Algorithme de Hachage

Après l'étude de différents algorithmes de hachage dans le premier chapitre et en prenant en compte le temps consommé pour générer un hache, nous avons choisi d'utiliser SHA256 comme un algorithme par défaut pour les raisons suivantes :

- MD5 : l'algorithme le moins sécurisé et il a aussi une forte possibilité de collision.
- SHA1: l'algorithme le plus rapide dans la famille des SHA, mais il a une possibilité de collision.
- SHA512 : l'algorithme le plus sécurisé mais le plus long aussi dans sa famille.

Ce qui nous a poussés à choisir l'algorithme SHA256 qui combine la sécurité et la rapidité. SHA256 permet de générer la base de données initiale. Néanmoins, on a ajouté la fonctionnalité de mettre à jour la base de donnée en utilisant un autre algorithme de hachage défini par l'utilisateur (Figure 3.3).



```
n1x@1337:/usr/share/sic$ sic --update --hash-algorithm ?  
[!] Available hash-algorithms : SHA1, SHA256, SHA512  
Check the help manual : sic --help
```

Figure 3.3 : Les algorithmes offerts par SIC.

## 2.3) Les attributs des fichiers

Après l'étude des différents attributs existants dans le deuxième chapitre et en prenant en compte nos besoins pour détecter la moindre tentative de modification d'un fichier système ou un fichier considéré important par l'utilisateur, on a choisi de garder trace de certains attributs dans la base de donnée (état initial) de SIC, puis à chaque scan, générer une nouvelle base de donnée (état actuel) et faire une comparaison entre ces attributs pour détecter les changements. Ces attributs sont :

- a- Chemin absolu** : Utilisé comme un premier identifiant de fichier lors d'extraction de son état de la base de données initiale.
- b- Hache** : Utilisé pour vérifier que le contenu d'un fichier n'a pas changé.
- c- Index de nœud** : Utilisé pour vérifier que le fichier n'a pas été remplacé par un autre.
- d- Permissions** : Utilisé pour vérifier que les permissions n'ont pas été changées pour des utilisations non autorisés.
- e- Utilisateur propriétaire** : Utilisé pour vérifier que l'utilisateur propriétaire de fichier

est toujours le même.

**f- Groupe propriétaire** : Utilisé pour vérifier que le groupe propriétaire de fichier est toujours le même.

**g- Nombre de référence** : Utilisé pour vérifier que le nombre de fichiers liens est toujours le même, sinon un fichier a été ajouté ou supprimé.

**h- Table** : Utilisé comme un second identifiant de fichier lors d'extraction de son état de la base de données initiale.

**i- Combinaison de quatre** : Cet attribut est créé spécialement car il existe deux types de vérification :

*1- Vérification complète* : Pour chaque fichier, SIC doit vérifier tous les attributs ci-dessus et donner importance à chaque valeur d'attribut changée.

Ce type est utilisé avec les fichiers exécutables (binaires), les différents services (apache, ssh, http, network-manager...etc) et les fichiers de configurations.

*2- Vérification de quatre* : Pour chaque fichier, SIC doit vérifier l'index de nœuds, les permissions, l'utilisateur propriétaire et le groupe propriétaire. Pour certains fichiers, le contenu change souvent, donc la vérification du hache causera une fausse alerte. Pour ce type de fichier, un changement dans ces quatre attributs est considéré comme un changement malveillant. Par exemple : pour les fichiers de démarrage (/boot), si un pirate arrive à ajouter la permission d'écriture dans ce répertoire, il pourra ajouter un service qui donnera la main au pirate d'accéder à la machine victime à chaque démarrage (ce qu'on appelle gagner de la persistance).

```
def FileAttributes(hash,name,table_name) :  
  
    file_path = str(name) #Le chemin absolu  
    file_hash = str(hash) #Le hash  
    file_inode = str(os.stat(name).st_ino) #Le numéro de noeud  
    file_permission = str(os.stat(name).st_mode) #Les permission  
    file_uid = str(os.stat(name).st_uid) #L'utilisateur propriétaire  
    file_gid = str(os.stat(name).st_gid) #Le groupe propriétaire  
    file_reference = str(os.stat(name).st_nlink) #Nombre de lien  
    file_table = str(table_name) #Nom de la table  
    check_some = file_inode+file_permission+file_uid+file_gid #Utiliser pour le deuxieme type de vérification  
    check_some = hashlib.sha256(check_some.encode()).hexdigest()  
    row=()  
    row=(file_path,file_hash,check_some,file_inode,file_permission,file_uid,file_gid,file_reference,file_table)  
  
    return row
```

Figure 3.4 : Les attributs utilisés par SIC.



## **2.4) La base de donnée PostgreSQL**

### **2.4.1) Présentation**

PostgreSQL est un système de gestion de base de données relationnelle et objet (SGBDRO). Ce système est concurrent d'autres systèmes de gestion de base de données, qu'ils soient libres (comme MariaDB et Firebird), ou propriétaires (comme Oracle, MySQL, Sybase, DB2, Informix et Microsoft SQL Server). Comme les projets libres Apache et Linux, PostgreSQL n'est pas contrôlé par une seule entreprise, mais fondé sur une communauté mondiale de développeurs et d'entreprises.[2]

### **2.4.2) Comparaison avec MySQL et SQLite**

Le système de gestion de base de données PostgreSQL est le plus anciens SGBD parmi les trois. Il a été développé en 1989 alors que MySQL en 1995 et SQLite en 2000. Néanmoins, le SGBD MySQL est le plus populaire actuellement.

PostgreSQL et MySQL sont des bases de données serveur, c'est à dire le pilote du serveur doit être installé chez l'utilisateur afin qu'il puisse se connecter à la base de donnée, par contre, SQLite ne nécessite pas d'installation avant l'utilisation.

SQLite est le plus rapide au niveau du temps d'exécution par rapport aux deux autres vu qu'il n'a pas besoin d'une connexion vers le serveur. Par contre, il ne supporte pas le multithreading (plusieurs connexions au même temps), ce qui est possible dans les deux autres SGBD. Cette propriété permet de réduire le temps d'exécution en cas de traitement de plusieurs données, ce qui nous élimine SQLite de notre choix.

Pour le temps d'exécution, MySQL et PostgreSQL sont presque égaux, avec un léger avantage pour MySQL, mais le PostgreSQL a la capacité de traiter de grandes quantités de données, ce qui est difficile avec MySQL.

Pour conclure, notre choix est d'utiliser le SGBDRO : PostgreSQL.

Le tableau ci-dessous (Tableau 3.1) résume les différences des SGBD dans le cadre de besoins de notre application.

SGBD	PostGreSql	MySql	Sqlite
Réalisation	1989	1995	2000
Type de SGBD	Serveur	Serveur	Publique
Multithreading	oui	oui	Non
Temps d'exécution	3	2	1
Traitement de grandes quantités de donnée (Big-DATA)	oui	non	Non
Objective	oui	non	Non

Tableau 3.1 : *Comparaison entre les différents SGBD.*

### 2.4.3) Protection de la base de donnée

Comme a été mentionné dans le deuxième chapitre, il y en a trois techniques pour protéger la base de données d'un contrôleur d'intégrité :

- 1- *Créer la base de données en mode lecture-seule.*
- 2- *Hacher la base de données.*
- 3- *Stocker la base de données dans un serveur.*

Notre contrôleur SIC est différent des autres contrôleurs d'intégrité qui ont combiné la deuxième et la troisième méthode. Dans SIC, on amélioré la troisième technique en utilisant d'autre techniques.

Lors d'installation, SIC installera un serveur PostgreSQL dans la machine hôte et créera un utilisateur sic\_user et une base de donnée sic\_db (Figure 3.5), contrairement à Tripwire qui utilise un serveur centralisé, SIC utilise un serveur local, ce qui permet de faire des scans même si la machine hôte n'a pas accès à internet ce qui n'est pas possible avec Tripwire.

```
n1x@1337:~$ sudo -u postgres -i
postgres@1337:~$ psql
psql (9.5.12)
Type "help" for help.

postgres=# \du
                                List of roles
Role name |                               Attributes                               | Member of
-----+-----+-----+-----+-----+-----
postgres | Superuser, Create role, Create DB, Replication, Bypass RLS | {}
sic_user  | Create DB                                                    | {}

postgres=# \l
                                List of databases
Name      | Owner   | Encoding | Collate |  Ctype  | Access privileges
-----+-----+-----+-----+-----+-----
postgres | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 |
sic_db   | sic_user | UTF8     | en_US.UTF-8 | en_US.UTF-8 |
```

Figure 3.5 : Serveur PostgreSQL utilisé par SIC.

La première technique est de donner une clé à l'utilisateur après que l'installation a été faite (Figure 3.6). Cette clé n'est générée qu'une seule fois, elle doit être gardée dans un endroit sécurisé car l'utilisateur aura besoin d'elle pour mettre à jour la base de donnée en cas où il a changé des fichiers qui existent déjà dans la base de donnée initiale sinon des fausses alertes seront générés (Figure 3.7).

```
Use this key to update the baseline database :
Aa93G0ZqHe39YSndP4HLnMRqA0Y1iRPkhDRP6kI8d5wDFP0qpeB
This key is generated once, keep it safe.
Press Enter..._
```

Figure 3.6 : Génération de clé secrète par SIC.

```
n1x@1337:/usr/share/sic$ sic --update --hash-algorithm SHA512
Enter your key : _
```

Figure 3.7 : Demande de la clé secrète par SIC.

La deuxième consiste à sécuriser les informations d'identification à la base de donnée de SIC (sic\_db) en cryptant le mot de passe (pwd dans Figure 3.8) à l'aide du module crypt qui utilise l'algorithme de cryptage symétrique Data Encryption Standard (DES) et en ajoutant un sel (salt) ce qui rend le mot de passe bien sécurisé, dans le cas où le pirate accède au code source, il ne pourra pas obtenir le mot de passe pour se connecter à la base de donnée .

```

x = '44523057353534505f433146'
y = binascii.unhexlify(x)
z = str(y,'ascii')
pwd = crypt.crypt(z[::-1],"0x")

def connect_db() :

    try :
        conn = psycopg2.connect(database="sic_db", user="sic_user", password=pwd, host="127.0.0.1", port="5432")
    except Exception as e :
        print("[!] ",e)
    else :
        print("Connected to sic_db successfully !")
    finally :
        conn.close()

```

Figure 3.8 : Sécuriser la connection à sic\_db.

La dernière technique est d'envoyer un email d'alerte à chaque fois que la base de données est mise à jour. Quand l'installation est terminée, l'utilisateur doit introduire son adresse mail (Figure 3.9). Si le pirate a eu la possibilité d'infecter la machine hôte, alors sa prochaine étape sera de mettre à jour la base de donnée initiale pour que la victime ne sera pas notifiée avec des alertes quand le pirate changera, supprimera ou ajoutera des fichiers pour des buts malveillants. Dans ce cas, l'utilisateur recevra un message dans sa boîte mail qui mentionne que la base de données a été mise à jour en précisant la date et l'heure pour qu'il sache si c'est lui qui a lancé cette action où le pirate (Figure 3.10).

```

For security reasons this tool requires a mail for :
    - Sending scan results
    - Sending alert when the database is updated
Please, enter your mail :
_

```

Figure 3.9 : Demande d'adresse mail par SIC

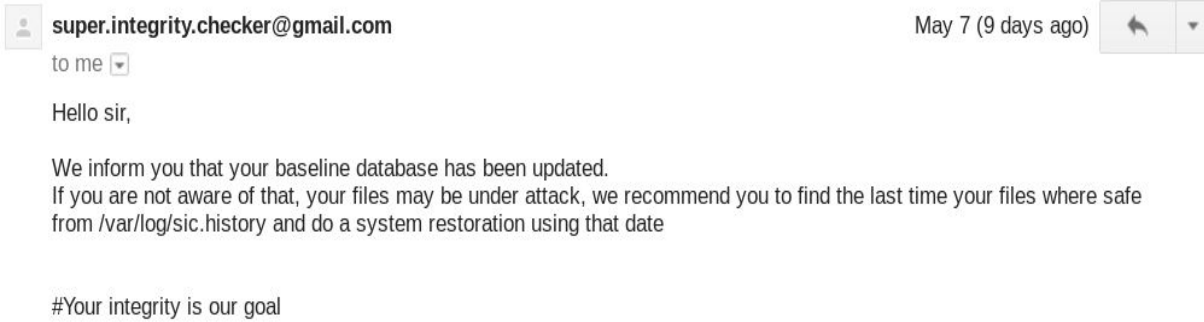


Figure 3.10 : *Message d'alerte de la part de SIC.*

### 3) Les fonctionnements principaux de SIC

Pour pouvoir lancer SIC, une installation doit être faite. Le programme qui se charge d'installer SIC est écrit en **Shell**, il préparera l'environnement où SIC pourra fonctionner correctement, il installera dans la machine hôte : le langage Python, les modules de python utilisés par SIC, un serveur local PostgreSQL puis il le configurera.

Une fois l'installation est terminée, SIC fonctionnera selon trois étapes principales (Figure 3.11). La première étape consiste à initialiser la base de données avec les informations des fichiers actuels pour garder trace. La deuxième et la troisième consistent à choisir le mode et le type de scan. Chaque étape est décrite indépendamment dans ce qui suit.

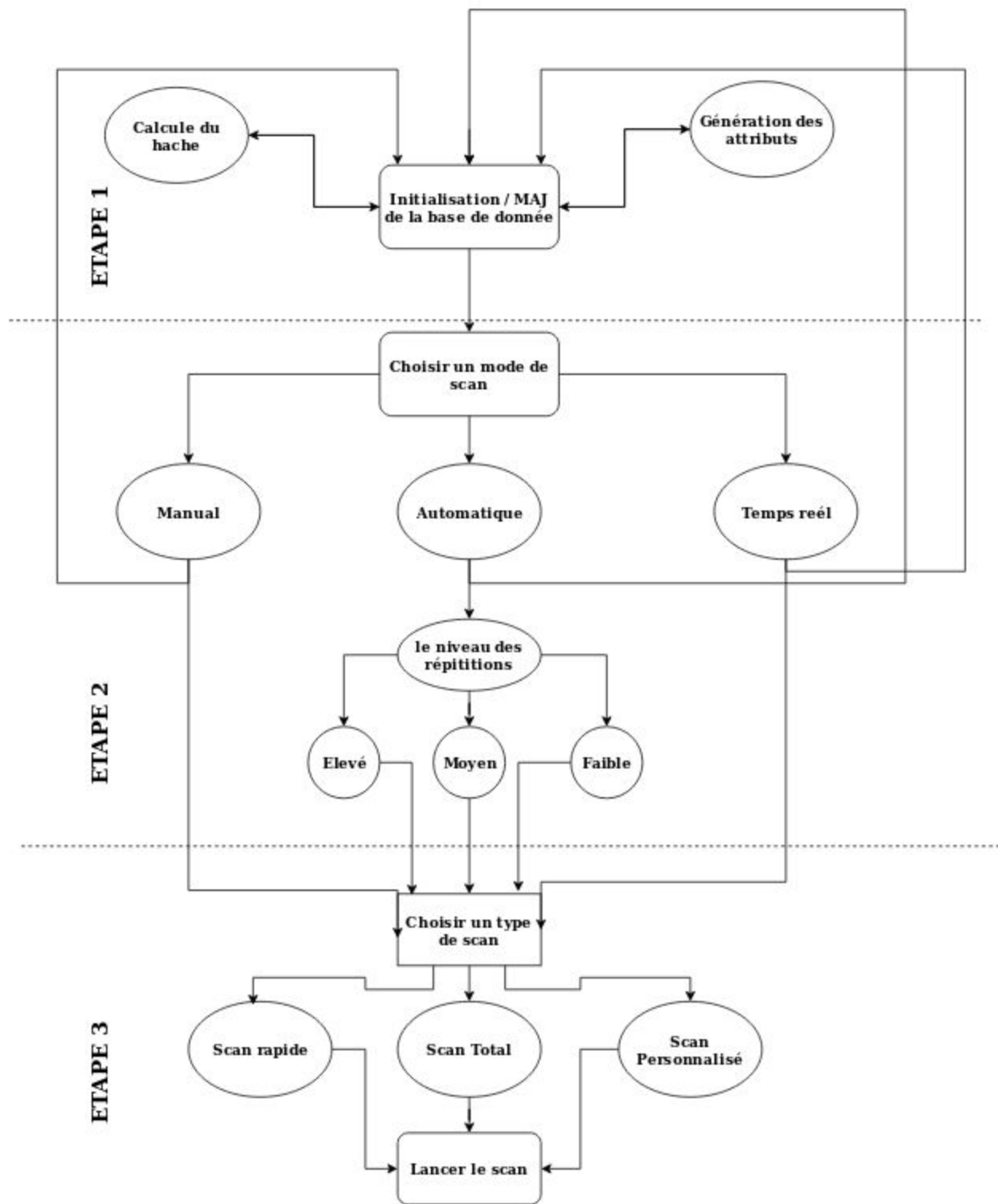


Figure 3.11 : Architecture fonctionnelle du sic.

### 3.1) L'initialisation de la BDD

Dans le but de garder trace de l'état actuel du système, un stockage des informations sur tous les fichiers systèmes (ou d'autres ajoutés par l'utilisateur) dans une base de donnée initiale est nécessaire. Ces informations sont les attributs de fichier vus précédemment, le hache du fichier et le hache des quatre attributs qui sont nécessaire pour le deuxième type de vérification. Cette base de donnée sera un repère de comparaison pour SIC.

### 3.2) Choisir un mode de scan

Après l'initialisation de la base de données, l'utilisateur doit choisir un mode de scan à suivre parmi les trois modes suivants (Figure 3.12) :



Figure 3.12 : Les modes de scan disponibles.

#### 3.2.1) Manual

Ce type est utilisé par les utilisateurs (les administrateurs généralement) qui veulent avoir un contrôle total des scans, ils doivent introduire l'argument (-c ou --check) pour mentionner qu'un scan manuel va être lancé par l'utilisateur puis ajouter l'argument (-t ou --type) pour préciser le type du scan (Figure 3.13), on n'en parlera par la suite des types de scan.

```
n1x@1337:~$ sic --check --type LONG --verbose
[!] Warning details will be outputed...
```

Figure 3.13 : Lancement d'un scan manuel.

#### 3.2.2) Automatique

Avec ce type, SIC lancera automatiquement des scans dans des temps spécifiés par

l'utilisateur, puis il configura ce qu'on appelle un cronjob (une tâche à exécuter par le processeur dans le temps spécifié) qui sera ajouté dans le crontab (un tableau qui contient des cronjobs). Néanmoins, SIC laisse la liberté pour choisir les temps des scans automatiques dans la version console (Figure 3.14). On a implémenté dans la version graphique trois niveaux pour les utilisateurs qui n'ont pas des connaissances dans la configuration des cronjobs :

- Niveau élevé : 1 scan le matin + 1 scan le soir.
- Niveau moyen : 1 scan par jour.
- Niveau faible : 1 scan par 3 jours.

```
n1x@1337:~$ sic --cronjob
CRONJOB-MANUAL:

    You need to set five fields, which are separated with white spaces :
        Minute Hour   Day_of_mounth   Mounth   Day_of_week

Minute : (0-59)
Hour   : (0-23)
Day_of_mounth : (1-31)
Mounth : (1-12)
Day_of_week : (Sunday-Saturday)

You can use meta-characters : '*' ',' to create more complex time schedule

EXAMPLES:

* * * * * : means every minute of every hours of every day of the mounth for every day of the week
0 16 1,10,22 * * : means at 16:00 on the 1st,10th,22th day of every mounth

Set your cronjob time : _
```

Figure 3.14 : Configuration d'un scan automatique (cronjob).

### 3.2.3) Temps réel

Dans ce mode de scan, un processus daemon (désigne un processus de long-term qui se lance au démarrage et s'exécute plutôt en arrière-plan) est créé par SIC.

Ce processus aura la tâche de suivre chaque opération de mise à jour (ajout/modification/suppression) à n'importe quel fichier (système ou spécifié par l'utilisateur). Ce mode consomme plus de CPU que les deux autres modes, mais il est considéré comme le plus sécurisé.

Pour choisir ce mode, il suffit d'ajouter l'argument (-real ou --real-time) pour la version console (Figure 3.2). Sinon pour la version graphique, il faut sélectionner l'option temps



réel dans paramètres de scan (Figure 3.).

### 3.3) Choisir un type de scan

Le lancement de scan se fera automatiquement dans le cas où le mode choisi précédemment est un mode automatique ou bien temps réel, par contre si le mode est manuel, alors l'utilisateur doit spécifier le type du scan.

SIC offre trois types de scan qui sont comme suit (Figure 3.15 et Figure 3.16) :



Figure 3.15 : Les types de scan version graphique.

```
n1x@1337:~$ sic --check --type ?  
[!] Available scan types : LONG, SHORT, SPECIFIC.  
Check the help manual : sic --help.
```

Figure 3.16 : Les types de scan version console.

#### 3.3.1) Scan Total (Long Scan)

Ce scan génère la totalité des fichiers système (Figure 3.17) et les fichiers ajoutés par l'utilisateur, c'est le scan le plus sûr et permet de détecter tous les changements effectués dans n'importe quel fichier ainsi il détecte l'ajout ou la suppression d'un fichier. Mais, ce type de scan peut prendre beaucoup de temps par rapport autres scans.

```
n1x@1337:/usr/share/sic$ cat AllFiles.list
/etc
/boot
/var
/lib64
/lib/modules
/lib/systemd
/usr/bin
/usr/sbin
/usr/local/bin
/usr/local/sbin
/bin
/sbin
```

Figure 3.17 : Les fichiers inclus dans un scan total.

### 3.3.2) Scan Rapide (Short Scan)

Ce type de scan est le plus utilisé par les administrateurs car les fichiers inclus dans ce scan sont les fichiers les plus visés par les hackers (fichier binaire *ou* exécutable), un changement dans un de ces fichiers donnera au hacker le contrôle total de la machine victime. Ce scan détectera toute modification dans ces fichiers (Figure 3.18), il est moins consommable en temps et CPU que le scan total.

```
n1x@1337:/usr/share/sic$ cat BinFiles.list
/bin
/sbin
/usr/bin
/usr/sbin
/usr/local/bin
/usr/local/sbin
```

Figure 3.18 : Les fichiers inclus dans un scan rapide.

### 3.3.3) Scan Personnalisé (Specific Scan)

Ce scan est utilisé quand l'utilisateur suspect des fichiers, il spécifie juste ces fichiers et le sera plus rapide que les deux autres et consomme moins de CPU (Figure 3.19).

```
n1x@1337:~$ sic --check --type SPECIFIC
You need to specify the directorie(s) to check using the -dir option.
n1x@1337:~$ sic --check --type SPECIFIC -dir /bin /sbin
[!] Warning details will be outputed...
```

Figure 3.19 : Lancement d'un scan personnalisé.

A la fin de chaque scan de ces scans cités ci-dessus, le résultat final est écrit dans le fichier `/var/log/sic.log`, ce fichier contiendra des informations détaillées sur chaque fichier modifié (Figure 3.20). Aussi, le fichier `/var/log/sic.history` gardera l'historique des résultats des scans lancés depuis que SIC a été installé dans la machine hôte.

En plus de ça, un message est envoyé à l'email de l'utilisateur en lui joignant le résultat du scan (Figure 3.21).

```
n1x@1337:~$ chmod 777 /home/n1x/Test/f.txt
n1x@1337:~$ sic --check --type SPECIFIC -dir /home/n1x/Test &>2
n1x@1337:~$ cat /var/log/sic.log

/home/n1x/Test/f.txt :
| file's permission has been changed at : 2018-05-18 02:14:21.980788

Total files scanned : 4
Total warnings found : 1
```

Figure 3.20 : *Détection de changement de permission.*



Figure 3.21 : *Email du résultat de scan.*

### 3.4) La mise à jour de la BDD

Sous UNIX, les mises à jour de système sont souvent effectuées. Ils provoquent des changements dans le contenu de certains fichiers, ajoutent et suppriment certains d'autres.

SIC détecte ces changements comme des modifications non autorisé et lance des fausses alertes à l'utilisateur. Pour cela, SIC effectue des mises à jour également à sa base de donnée initiale à chaque fois une mise à jour de système est faite.

Comme c'est mentionné dans la figure 3.11, la mise à jour n'est qu'une redirection vers l'initialisation de la base de donnée (génération de nouveaux haches, récupération des nouvelles valeurs des attributs des fichiers... etc). La mise à jour peut être faite manuellement dans le cas où l'utilisateur à lui-même a modifié des fichiers, mais dans

ce cas il doit introduire la clé secrète pour être sûr que c'est l'utilisateur qui est entrainé de faire la mise à jour et aussi un email d'alerte que la base de donnée est mis à jour est envoyé, comme vu précédemment (Figure 3.7 et Figure 3.10).

#### 4) Les fonctionnements secondaires

Tout contrôleur d'intégrité doit satisfaire les fonctionnements principaux cités ci-dessus. Les services offerts par les trois solutions étudiés dans le chapitre 2 sont :

- Tripwire : Offre le service de messagerie pour envoyer des notifications.
- AIDE : Offre le service de scan automatique à l'aide du cronjob.
- AFICK : Offre la facilité d'utilisation à l'aide d'une interface graphique.

Ces services sont disponibles dans SIC. En plus de ça, SIC offre d'autres services qui sont :

- Le mode de scan : Temps réel.
- Des techniques modernes pour protéger la base de données comme vu dans la partie *Protection de la base de donnée 2.4.3*.
- La mise à jour spécifique de la base de données :

Chez les autres contrôleurs, si une mise à jour est lancée, alors elle va mettre à jour tous les fichiers. SIC a ajouté la possibilité de mettre à jours juste des fichiers spécifiques, dans le cas où l'utilisateur sait quels fichiers sont modifiés, ils peut mettre à jour juste ces fichiers, comme il peut aussi lancer une mise à jour complète avec ou sans l'ajout de nouveaux fichiers.

- **Comparaison d'architecture (arborescence) des fichiers :**

Souvent l'attaquant évite la modification d'un fichier, il préfère ajouter un fichier et/ou supprimer un fichier, cela l'aidera à être moins détectable. Pour cela, SIC a implémenté sa propre méthode pour détecter tout fichier supprimé ou ajouté en stockant l'arborescence des fichiers dans une base de donnée puis à chaque fois un scan est lancé, une arborescence est générée et comparée avec celle de la base de donnée (Figure 3.22).

```
n1x@1337:~$ sudo rm /var/www/html/*.png
n1x@1337:~$ sic --architecture
[!] File has been deleted from your file-system : /var/www/html/a.png
[!] File has been deleted from your file-system : /var/www/html/b.png
[!] File has been deleted from your file-system : /var/www/html/d.png
[!] File has been deleted from your file-system : /var/www/html/f.png
[!] File has been deleted from your file-system : /var/www/html/e.png
[!] File has been deleted from your file-system : /var/www/html/c.png
```

Figure 3.22 : *Lancement d'une comparaison d'architecture.*

## 5) L'interface graphique

L'interface graphique a été implémentée dans le but de faciliter l'utilisation du logiciel SIC en proposant tous les fonctionnements précédents, ainsi pour pouvoir gérer et vérifier les différents paramètres ou bien consulter les logs (résultats).

La version graphique est composé de cinq différentes fenêtres, chaque fenêtre possède une fonctionnalité différente, dans ce qui suit nous allons présenter toutes ces fenêtres avec leurs fonctionnalités.

### 5.1) Page d'accueil

C'est la fenêtre d'accueil (Figure 3.23), elle permet à l'utilisateur de se déplacer rapidement aux 4 autres fenêtres en utilisant un des quatres buttons, elle affiche également certains paramètres et donne un avis aux utilisateurs sur l'état actuel de la machine.



Figure 3.23 : La fenêtre d'accueil.

### 5.2) Analyse

Cette fenêtre (Figure 3.24) contient les différents type de scans vus précédemment, l'appuie sur le bouton démarrer provoque le début de scan choisi et les autres buttons seront désactivés, cela veut dire qu'on peut faire qu'un seul scan à la fois.

A la fin de scan, les boutons reviennent en fonction et l'utilisateur est dirigé vers la fenêtre Fichiers de log pour voir le résultat du scan.

Dans cette version de SIC, si le nombre total des fichiers scannés est différent de celui qui existe dans la base de donnée, alors une comparaison d'architecture est lancée automatiquement contrairement à la version console (Figure 3.22) où l'utilisateur doit lancer cette comparaison manuellement.



Figure 3.24 : La fenêtre d'analyse.

### 5.3) Fichiers de log

La fenêtre Fichiers de log (Figure 3.25) est faite pour afficher le résultat des scans (les fichiers modifiés, quels attributs sont modifiés, temps de modification).

L'utilisateur peut aussi consulter l'historique des scans lancé dès l'installation de SIC.

Au faite, ces deux fonctionnalités sont équivalent à la lecture des deux fichiers vus précédemment : `/var/log/sic.log` et `/var/log/sic.history`.

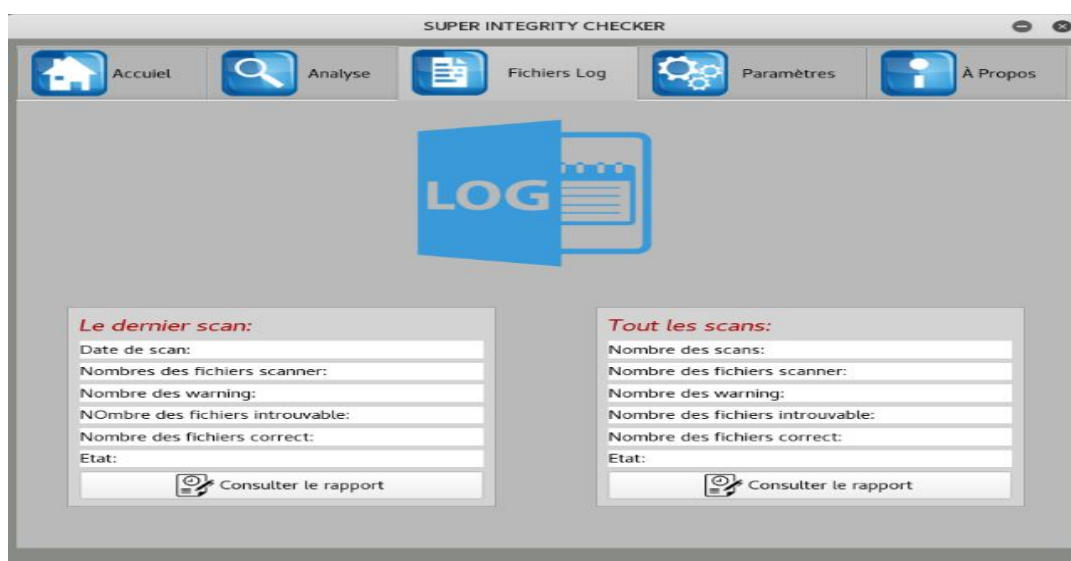


Figure 3.25 : La fenêtre des fichiers log.

## 5.4) Paramètres

La fenêtre Paramètres (Figure 3.26) permet à l'utilisateur de contrôler (modifier, retirer) les paramètres suivant :

- Introduire l'email de l'utilisateur.
- Activer/désactiver la mise à jour automatique de la base de donnée.
- Choisir un mode de scan.
- Choisir un algorithme de hachage.
- Réinitialiser les paramètres par défaut.

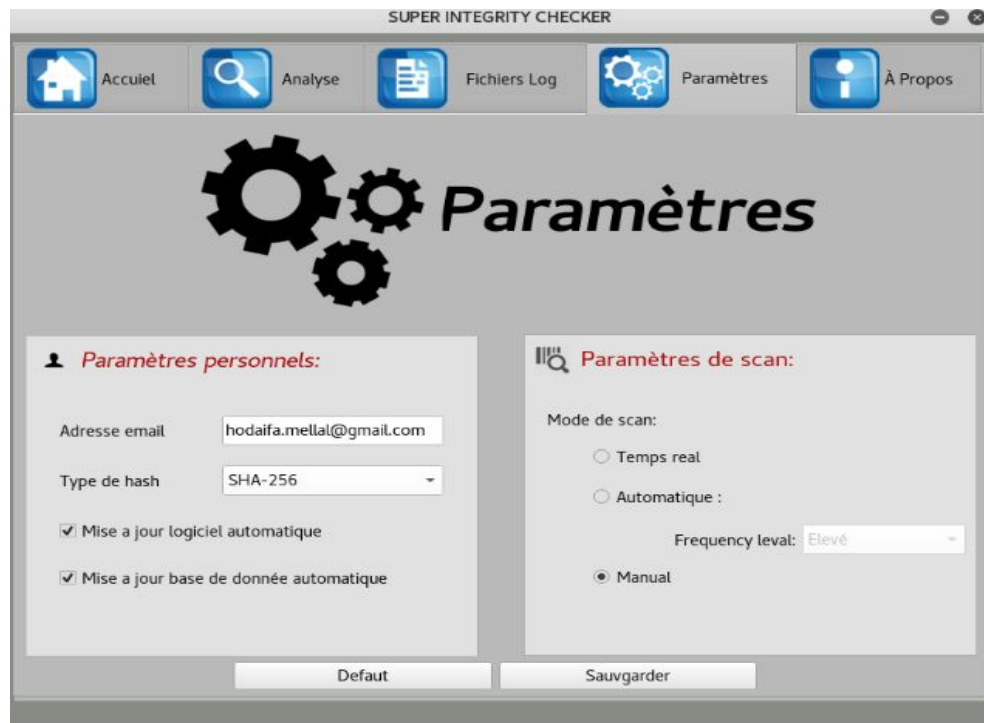


Figure 3.26 : La fenêtre des paramètres.

## 5.5) À Propos

La fenêtre A Propos (Figure 3.27) est une présentation du SIC qui contient une bref explication de ses fonctionnalités. Des informations sur les contacts disponibles sont joints dans cette fenêtre.





Figure 3.27: La fenêtre A Propos

## 6) Conclusion

Après la réalisation de Super Integrity Checker, il a été mis en tests. Les résultats de ces tests ont démontré qu'il n'est pas seulement capable de fournir les fonctionnalités principales qui doivent exister mais aussi les fonctionnalités secondaires qui n'existent pas chez les autres contrôleurs. En plus le temps de réponse est meilleur que celui d'AIDE et Tripwire malgré que le langage utilisé est un langage interprété, cela est due au multithreading. SIC possède plus de précision qu'AFICK mais un peu moins rapide par rapport à AFICK.

On conclut que Super Integrity Checker est une solution concurrente aux solutions existantes et satisfaisante à la détection des intrusions et la sécurisation de l'intégrité des fichiers UNIX.

# Conclusion générale

Le travail réalisé se positionne dans le domaine de la sécurité des systèmes d'information dont les concepts et les principes ont été déjà présentés. En effet le contrôle d'intégrité est un aspect très important de la sécurité qui est même plus utile que les autres composants sur le système d'exploitation UNIX vu l'importance de maintenir l'état des fichiers.

La réalisation de notre logiciel Super Integrity Checker (SIC) nous a permis d'améliorer et d'approfondir nos connaissances ainsi que nos compétences dans plusieurs domaines. D'abord, l'analyse des fichiers systèmes nous a permis de mieux connaître l'architecture du système d'exploitation UNIX en étudiant les différents rôles de chaque type de fichier système afin de pouvoir filtrer ces fichiers dans le but de détecter les changements non-autorisés qui peuvent réagir sur la configuration du système. Ensuite, le hachage des différents fichiers pour pouvoir comparer leur contenu nous a permis d'enrichir nos connaissances dans le domaine de la cryptographie en étudiant les différents algorithmes avec la comparaison et le choix de l'algorithme le plus cohérent pour notre logiciel. Finalement, SIC a été réalisé à l'aide du langage de programmation Python, Ceci nous a permis de se familiariser avec l'utilisation des différentes bibliothèques qui permettent de gérer les fichiers systèmes, générer des haches, utiliser les différents systèmes de gestion de base de données, l'envoi des mails ainsi que l'implémentation d'une interface graphique, ce qui a développé nos compétences en programmation.

Il existe déjà plusieurs contrôleurs d'intégrité dont on a parlé, mais la réalisation de SIC était dans le but de proposer plus de services et de fonctionnalités aux utilisateurs. D'abord, tous les outils existants détectent les changements de contenus des fichiers, mais par contre le temps de réaction et le nombre important des fichiers modifiés détectés diffèrent d'un contrôleur à un autre. SIC a montré des résultats supérieurs à tous les autres contrôleurs à cause du traitement multithreading qui lui a permis d'être plus rapide malgré qu'il traite beaucoup plus de fichiers. Ainsi, il propose à l'utilisateur d'ajouter des fichiers ou des répertoires autres que les fichiers systèmes. Ensuite, SIC utilise une protection moderne de la base de données, celui qui rend la probabilité de détecter des modifications non autorisées dans les niveaux très bas, ainsi que la réaction en temps réel qui détecte le changement sur place sans laisser aucune chance ou même une tentative dissimulation. Finalement, la facilité de l'utilisation et la double version sont aussi des caractéristiques de SIC. Deux versions sont disponibles l'une est celle de l'interface graphique qui facilite l'utilisation en proposant tout ce que l'utilisateur

peut faire juste en appuyant sur des boutons, et la deuxième est la version console avec un guide très riche qui explique toutes les commandes avec le détail demandé pour que l'utilisateur soit capable de l'utiliser facilement. Finalement, on peut dire que Super Integrity Checker est une solution concurrente aux solutions existantes et satisfaisante à la détection des intrusions et la sécurisation de l'intégrité des fichiers UNIX.

A cause de la limitation du temps et l'obligation de présenter notre logiciel dans les plus brefs délais, nous avons réalisé une première version de SIC qui sera par la suite enrichie par de nouvelles versions sous la forme des mises à jours. Plusieurs services seront implémentés sur notre logiciel dans les jours qui arrivent. Nous citons premièrement le stockage des fichiers essentiels du système dans des périodes aléatoires afin de pouvoir proposer une restauration dans le cas où ces derniers sont modifiés ou détruits. Deuxièmement, et dans le cadre d'améliorer la sécurité, nous allons implémenter une politique qui permet de bloquer tout changement non autorisé et avertir l'utilisateur afin de lui laisser le choix de suspendre ou d'autoriser le changement. Finalement, des versions destinées au système d'exploitation Windows et MAC OS seront implémentées avec des améliorations au niveau de l'interface graphique en ajoutant des autres options d'affichages et de consultation, ainsi que d'autres outils de configuration.

# BIBLIOGRAPHIE

## Référence du Chapitre I :

- [1] : [https://fr.wikipedia.org/wiki/Sécurité\\_des\\_systèmes\\_d'information](https://fr.wikipedia.org/wiki/Sécurité_des_systèmes_d'information)
- [2] : Solange Ghernaoui, Livre Sécurité Informatique et Réseau, 4eme édition, Dunod, Paris, 2013.
- [3] : Louis Salvail, Cours de l'Authentification et Identification, Cours 8, Université de Montréal.
- [4] : [RFC 4949 "Internet Security Glossary version 2", août 2007.](#)
- [5] : Belkaid Mohammed, Thèse de doctorat, Contribution à l'authentification souple d'images digitales par des techniques de marquage numérique Application aux images médicales, université de Mentouri de Constantine, Octobre 2008.
- [6]: <https://www.kaspersky.fr/resource-center/threats/computer-viruses-and-malware-facts-and-faqs>
- [7] : [Audrey Asseman, Cours Vol et Fuite de données, Université de Montréal, Année 2009.](#)
- [8] : <https://assistance.sfr.fr/internet-et-box/securite/tout-savoir-phishing.html>
- [9] : SecuriteInfo.com, Le grand live de SecuriteInfo.com, le 19 février 2004.
- [10]: [https://www.belgium.be/fr/justice/securite/criminalite/criminalite\\_informatique/sabotage\\_informatique](https://www.belgium.be/fr/justice/securite/criminalite/criminalite_informatique/sabotage_informatique)
- [11] : Christina Boura, Thèse de doctorat de l'UPMC, Analyse de Fonctions de Hachage et Cryptographie, 7 décembre 2012.
- [12] : Renaud Dumont, Université de Liège Faculté des Sciences Appliquées - Cryptographie et Sécurité informatique.
- [13] : <https://fr.wikipedia.org/wiki/SHA-2>
- [14] : <https://fr.wikipedia.org/wiki/SHA-3>
- [15] : <https://www.cnil.fr/fr/10-conseils-pour-la-securite-de-votre-systeme-dinformation>.
- [16] : [https://en.wikipedia.org/wiki/Hash\\_function](https://en.wikipedia.org/wiki/Hash_function).

## Référence du Chapitre II :

- [17] : [https://en.wikipedia.org/wiki/File\\_integrity\\_monitoring](https://en.wikipedia.org/wiki/File_integrity_monitoring)
- [18] : <https://stackoverflow.com/questions/13639276/check-file-integrity-for-unix-linux>
- [19] : Code source de Tripwire : <https://github.com/Tripwire/tripwire-open-source>
- [20] : Code source de AIDE : <http://aide.sourceforge.net/>

**[21]** : Code source d'AFICK : <http://afick.sourceforge.net/>

**[22]** : Lawrence Grim, IDS : File integrity Checking, PhD Dissertation at University of Maryland, 2014.

**Référence du Chapitre III :**

**[23]** : Durat Eric Landy, Cours 1, module de programmation script, Université de Lille 1, 2015.

**[24]** : <https://fr.wikipedia.org/wiki/PostgreSQL>

# Résumé:

*La mise en place d'un logiciel client serveur qui contrôle l'intégrité des fichiers systèmes sous le système d'exploitation UNIX nécessite une étude détaillée des outils utilisés dans sa réalisation ainsi que l'architecture du système.*

*La solution proposée combine un ensemble des outils selon la compatibilité avec le système d'exploitation et les conditions de l'utilisation afin de pouvoir maintenir l'état des fichiers systèmes. Ainsi que d'autres avantages proposés dans le but de faciliter l'utilisation et d'offrir plus de services aux utilisateurs.*

**Mot-clés:** Sécurité, Intégrité, Logiciel, Cryptographie, SGBD, Unix.

# Abstract

*The implementation of server client software that verify the integrity of system files under the UNIX operating system requires a detailed study of the tools used in its implementation as well as the architecture of the system.*

*The proposed solution combines a set of tools including OS compatibility and usage conditions to maintain the state of the system files. As well as other benefits proposed for the purpose of facilitating the use and offering more services to the users.*

**Keywords:** Security, Integrity, Software, Cryptography, DBMS, Unix

