UNIVERSITÉ DU QUÉBEC À MONTRÉAL

DÉTECTION ET EXPLOITATION DES COPIES PARTIELLES AFIN DE FACILITER L'ANALYSE FORENSIQUE D'UN SYSTÈME DE FICHIERS

MÉMOIRE
PRÉSENTÉ
COMME EXIGENCE PARTIELLE
DE LA MAÎTRISE EN INFORMATIQUE

PAR

HABIB LOUAFI

JUILLET 2006

REMERCIEMENTS

J'exprime ma gratitude à M. Guy Bégin pour m'avoir encadré tout au long de ce travail. Je le remercie également pour sa disponibilité, ses conseils précieux et ses encouragements.

Ce travail a été réalisé dans le laboratoire d'enseignement de la maîtrise en informatique.

J'exprime mes remerciements à toutes les personnes qui, de loin ou de près, m'ont aidé à venir à terme de ce travail.

Je ne manque pas également de remercier ma femme pour m'avoir aidé et supporté dans les moments de stress sans oublier mes amis pour leur soutien personnel.

Enfin, un merci à tous ceux qui m'ont appris l'honnêteté du cœur, avec une pensée toute spéciale pour mes parents.

TABLE DES MATIÈRES

LISTE DES FIGURES

LISTE DES TABLEAUX

RÉSUMÉ

La forensique informatique, connue sous le terme anglais « *computer forensic* », est le domaine de recherche qui s'intéresse aux enquêtes après incidents. Elle représente l'ensemble des principes scientifiques et des méthodes techniques appliquées à l'investigation criminelle pour prouver l'existence d'un crime et aider la justice à déterminer l'identité de l'auteur et son mode opératoire. Les investigateurs, dans leurs enquêtes, utilisent des outils matériels ou logiciels. Les outils logiciels sont des outils d'acquisition qui servent à copier d'importantes quantités de données à partir de l'ordinateur saisi, et des outils d'analyse de preuves.

En examinant les différents outils de la forensique informatique nous avons constaté l'absence et la nécessité d'un outil logiciel pour localiser les copies légèrement modifiées d'un fichier texte (*copies partielles*). En forensique informatique, la recherche de copies partielles d'un fichier texte contenant des informations douteuses peut aider grandement un investigateur à localiser la version originale de ce fichier à partir de la copie disponible, sachant que cette copie a, peut-être, été légèrement modifiée.

Après une recherche approfondie, nous avons constaté qu'une solution possible à ce problème vient combler un vide entre les outils classiques de recherche de copies identiques de fichiers utilisant les fonction de hachage (MD5, SHA-1, etc.) et les outils de comparaison de fichiers ligne par ligne ou encore caractère par caractère.

Notre travail a consisté à concevoir, développer et présenter à la communauté de la forensique informatique un outil logiciel qui sert à localiser les copies partielles d'un fichier texte dans un système de fichiers. Pour ce faire, nous avons adapté certains algorithmes d'alignement, utilisés en bioinformatique, pour comparer deux fichiers textes. Nous avons défini un seuil d'acceptation servant à distinguer les copies intéressantes (pour l'investigation) des autres fichiers. L'outil que nous avons développé utilise deux algorithmes d'alignement de séquences (global et local) conçus au départ pour trouver les similarités entre les séquences protéique ou nucléotidiques ainsi qu'un troisième algorithme d'alignement conçu pour résoudre un problème lié à la détection d'intrusion (semi-global). Nous avons aussi défini un alignement automatique : en se basant sur la différence de tailles entre les fichiers comparés, l'outil choisira automatiquement le type d'alignement à utiliser, à savoir, global ou local.

Nous avons testé cet outil sur un ensemble de fichiers textes sélectionnés au hasard et les résultats montrent que cette approche a vraiment permis de trouver une solution efficace au problème de détection de copies partielles d'un fichier texte. D'autre part, nous avons comparé notre outil avec les outils de comparaison de fichiers disponibles et nous avons constaté qu'aucun de ces outils ne donne de résultats comparables.

Mots clés : Forensique informatique, copie partielle, comparaison de fichiers, alignement de séquences, bioinformatique.

INTRODUCTION

Motivation

Les ordinateurs personnels qui sont de plus en plus à la portée de tout le monde, sont devenus également un outil puissant qui peut être utilisé pour perpétrer des crimes. Les criminels peuvent par exemple coordonner leurs activités en utilisant Internet, bien sûr en cryptant leurs communications. Et voila l'Internet qui devient un autre moyen pour commettre des activités criminelles. Les exemples sont nombreux, on peut citer l'espionnage industriel, la création et la distribution de virus, etc.

De nombreux travaux ont été effectués dans le but de développer des outils pour sécuriser les systèmes informatiques afin que de tels crimes informatiques (*cybercrimes*) ne se produisent pas. Mais, il est évident que le développement et le déploiement d'un système de sécurité fiable ne sont pas des choses évidentes; ils demandent un travail à la fois colossal et délicat et qui doit constamment être revu et corrigé. D'autre part, il existe un autre type de crimes utilisant l'informatique (*computer-facilitated crimes* ou *technocrimes*) qui n'ont aucun lien avec la sécurité des systèmes. Par exemple, la publication de recettes pour fabriquer des bombes, des courriels contenant des lettres de menaces, etc. D'une manière générale, quand il s'agit de crimes liés à l'informatique (*high-tech crimes*), que ce soit des crimes informatiques ou utilisant l'informatique, la recherche de preuves informatiques est un service auquel les entreprises font appel pour déterminer d'un côté les dégâts et leur ampleur, et de l'autre l'identité du criminel responsable et son mode opératoire. Dans le cas d'attaques, internes ou externes, la recherche de preuves peut aussi aider à déterminer les vulnérabilités de système qui sont à l'origine des attaques afin d'y remédier.

La forensique informatique, connue sous le terme anglais « *computer forensic* », est le domaine de recherche qui s'intéresse aux enquêtes après incidents. C'est un domaine à la

fois légal, scientifique et technique. Les enquêteurs, dans leur recherche de preuves, utilisent des outils logiciels ou matériels. Les outils logiciels sont des outils d'acquisition qui servent à copier d'importantes quantités de données à partir de l'ordinateur suspect, et aussi, des outils d'évaluation et d'analyse de preuves qui sont de plus en plus complexes et d'une portée limitée.

Objectif du présent travail

Le but du présent travail consiste à concevoir, développer et présenter à la communauté de la forensique informatique un outil logiciel qui sert à localiser les copies partielles d'un fichier texte dans un système de fichiers. Après une recherche approfondie, nous avons constaté qu'un tel outil permettrait de combler un vide très important entre les techniques classiques de recherche de copies utilisant le hachage[1] (MD5[2], SHA-1[3],...) et les techniques de comparaison de fichiers ligne par ligne ou caractère par caractère (fonction diff d'UNIX entre autres). Les techniques existantes ne donnent pas les résultats escomptés lorsqu'on s'intéresse à des copies légèrement modifiées (copies partielles).

Pour bien situer notre travail, nous allons dans un premier temps présenter en détail en quoi consistent les enquêtes après incidents, connues sous le terme scientifique de « forensique informatique » et présenter toutes les facettes de l'enquête allant de l'aspect juridique à l'aspect purement technique en présentant les différentes méthodes de recherche de preuves.

Le chapitre 2 sera consacré aux différents outils logiciels utilisés par les investigateurs en forensique informatique. Ces outils peuvent être classés en deux catégories : les outils d'acquisition et les outils de recherche de preuves.

---

[1] Un processus utilisant des algorithmes mathématiques sur des données pour générer une valeur numérique représentative de ces données.
[2] *Message Digest 5* : Algorithme de hachage (voir RFC1321)
[3] *Secure Hash Algorithm* : Algorithme de hachage (voir RFC 3174)

Dans le chapitre 3, nous présenterons les différentes techniques d'alignements de séquences génomiques, utilisées en bioinformatique, qui seront utilisées par la suite pour développer notre outil.

Dans le chapitre 4, nous allons présenter en détail l'outil que nous avons développé (*fCompare*) : sa conception, son fonctionnement et ses limites.

Nous avons réservé le chapitre 5 aux différents tests qui ont permis de valider notre approche.

Enfin, le dernier chapitre sera consacré à la conclusion et aux travaux futurs qui peuvent être réalisé en se basant sur notre outil et sur l'approche que nous avons adoptée.

CHAPITRE I

LA FORENSIQUE INFORMATIQUE

1.1 Définitions

Les sciences forensiques se définissent comme l'ensemble des principes scientifiques et des méthodes techniques appliqués à l'investigation criminelle pour prouver l'existence d'un crime et aider la justice à déterminer l'identité de l'auteur et son mode opératoire. Le terme « forensique », qui s'utilise notamment en médecine et en théologie, vient du latin « forum » qui veut dire place publique, lieu de jugement dans l'antiquité. Il appartient au domaine de la justice, et est à la fois légal, scientifique et technique. Ce nom existe dans la plupart des langues européennes mais son utilisation en français est récente (Schauer, 2001).

L'investigation forensique dans le domaine de l'informatique, mieux connue sous le terme anglais « *computer forensic* », regroupe l'ensemble des principes et méthodes utilisés pour les enquêtes après incidents qui s'appliquent à des supports informatiques.

Nous avons aussi trouvé d'autres définitions que nous préférons présenter dans leurs formulations originales, en anglais :

> "*The use of scientifically derived and proven methods toward the preservation, collection, validation, identification, analysis, interpretation, documentation and presentation of digital evidence from digital sources for the purpose of facilitating or*

furthering the reconstruction of events found to be criminal, or helping to anticipate unauthorized actions shown to be disruptive to planned operations.[4] "

"*Computer forensics involves the presentation, identification, extraction, documentation and interpretation of computer data.*[5] "

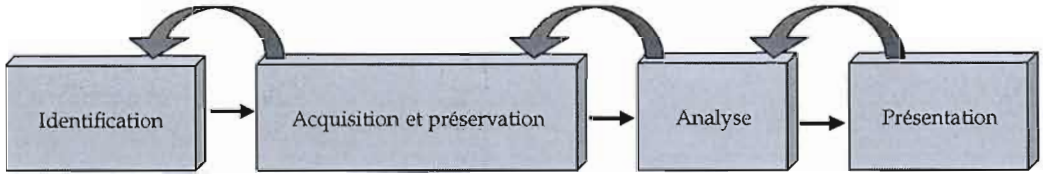
"*Computer forensic science is the science of acquiring, preserving, retrieving, and presenting data that has been processed electronically and stored on computer media.*[6] "

1.2 Le processus d'enquête

La forensique informatique est un processus composé de plusieurs phases qui se déroulent en séquence afin de fournir des preuves légalement acceptables. A partir des définitions citées plus haut, nous pouvons distinguer quatre étapes à suivre dans un processus forensique, à savoir :

- Identification
- Acquisition et préservation
- Analyse
- Présentation

Figure 1.1 Processus d'enquête forensique



[4] Palmer, Gary. 2001. *A Road Map for Digital Forensic Research*. En ligne. <http://www.dfrws.org/2001/dfrws-rm-final.pdf>. Consulté le 15 février 2006.
[5] Kruse, W. et J. Heiser. 2002. *Computer Forensics: Incident Response Essentials*. Boston: Addison-Wesley, 398 p.
[6] Noblett, M. G., M. M. Pollitt et L. A. Presley. 2000. *Recovering and Examining Computer Forensic Evidence*. En ligne. <http://www.fbi.gov/hq/lab/fsc/backissu/oct2000/computer.htm>. Consulté le 15 février 2006.

1.2.1 Identification

Il s'agit d'identifier et de repérer où sont présentes les preuves, où et comment elles sont stockées, quels systèmes d'exploitation sont utilisés. On doit établir une correspondance entre les informations recherchées et les sources de données. Il s'agit de répondre aux questions suivantes : quelles informations cherche-t-on ? Où peut-on les obtenir ? Dans quel ordre devrait se faire la recherche ? Les actions nécessaires doivent être déterminées avant de commencer l'acquisition des données. Puisqu'il faut prévoir les obstacles qui peuvent être rencontrés dans les phases de présentation et d'analyse, il faut dès cette phase établir un plan d'acquisition.

1.2.2 Acquisition et préservation

L'acquisition consiste à mettre en œuvre le plan d'acquisition créé lors de la phase d'identification. Le but est d'obtenir des copies forensiques[7] de toutes les données qui vont être utilisées dans la phase d'analyse. Aussi, il est important de préserver l'intégrité des preuves numériques pour s'assurer que la chaîne de continuité des preuves (*Chain of custody*) n'est pas brisée. Les données doivent être copiées sur des supports stables tel que des CD-ROMs. Les changements qui peuvent être apportés aux preuves doivent être bien documentés en incluant l'état initial des données, les changements apportés et les raisons de ces changements afin de pouvoir prouver l'intégrité des données dans une cour de justice.

1.2.3 Analyse

C'est lors de la phase d'analyse que les données collectées dans la phase précédente deviennent des preuves numériques (c'est-à-dire des informations stockées sous un format binaire qui peuvent être utilisées dans une cour de justice). Les composantes clés de l'analyses sont : le filtrage, la transformation, la corrélation, l'agrégation, et la génération de méta-données. L'avantage de copier les données sur des supports stables (CD-ROM par exemple) est de pouvoir manipuler ensuite ces supports et les données qui s'y trouvent sans risque. La façon dont l'investigateur interagit avec les données collectées a une grande

---

[7] Une reproduction exacte, bit par bit, de l'information contenue dans un support de stockage, dont la validité et l'intégrité ont été vérifiées en utilisant un algorithme connu.

influence sur la qualité des données présentées comme preuves numériques. Lors de cette phase, il faut dégager un ensemble de preuves numériques suffisant pour couvrir les besoins définis lors de la phase d'identification.

1.2.4 Présentation

Lors de la phase de présentation, les preuves doivent être présentées sous une forme légalement acceptable et compréhensible. Il s'agit de créer un rapport présentant les preuves numériques obtenues. Ce rapport doit détailler toutes les actions faites durant les différentes phases d'investigation. Les preuves doivent être présentées avec tous les détails nécessaires pour qu'un examinateur indépendant puisse reproduire et valider chaque pièce de preuve. Un autre point important est que les preuves seront présentées à des gens qui n'ont pas forcément les compétences informatiques permettant de comprendre des preuves dans leur état original. C'est pourquoi le rapport doit présenter les preuves d'une manière simplifiée tout en montrant les liens avec la version originale.

1.3 Règles de la forensique informatique

L'investigateur, tout au long de son processus à la recherche de preuves, doit respecter certaines règles fondamentales.

1.3.1 Manipulation minimale de l'original

C'est la règle la plus importante dans la forensique informatique. Il faut, dans la mesure du possible, faire des copies des données contenant des preuves et manipuler ensuite ces copies plutôt que les originaux. La copie doit être une reproduction exacte de l'original, qui doit être authentifiée en calculant une valeur d'authentification d'intégrité (CRC, MD5) de toutes les données avant et après la copie et en comparant les résultats. Dans le cas contraire, l'intégrité de la preuve pourrait être mise en cause.

1.3.2 Comptabiliser tous les changements

Il est évident que, dans certaines circonstances, il est inévitable de faire des changements aux données. Par exemple, démarrer ou arrêter la machine peut causer des

changements dans le contenu de la mémoire, ou dans les fichiers temporaires, etc. Dans ces conditions, tous les changements doivent être documentés.

1.3.3 Ne pas dépasser ses connaissances

Dans les circonstances où l'investigation nécessite des compétences qui dépassent le niveau de connaissances de l'investigateur, ce dernier doit rechercher de l'assistance auprès d'autres ressources plus compétentes tel qu'un investigateur spécialisé, ou si le temps le permet, en suivant une formation additionnelle pour améliorer ses connaissances. Il est fortement conseillé de ne pas continuer d'examiner les données, au-delà de ses connaissances, car on risque d'endommager les preuves.

1.3.4 Respecter les règles de preuves

Afin de s'assurer que les preuves collectées soient légalement acceptées, l'investigateur, lorsqu'il examine ou manipule les preuves, doit suivre et respecter certaines règles propres aux preuves qui sont décrites dans la section suivante.

1.4 Règles de preuves

Avant de commencer la collecte des preuves, il est important de s'assurer que le processus utilisé pour collecter ces preuves suive un procédé légal en respectant la juridiction de l'état ou du pays où les preuves vont être présentées. L'investigateur doit prendre en considération que les pratiques légales dans une juridiction peuvent ne pas être les même dans une autre. Matthew Braid, dans son article (Matthew, 2001), a donné une liste de cinq règles relatives aux preuves à respecter.

- Admissibilité
- Authenticité
- Complétude
- Fiabilité
- Crédibilité

1.4.1 Admissibilité

L'admissibilité est la règle la plus fondamentale, car une preuve qui ne peut pas être admise n'a aucun intérêt. Ne pas respecter cette règle est équivalent à chercher des preuves au mauvais endroit. Il faut s'assurer que la preuve collectée soit admissible, autrement dit, acceptable par une cour de justice.

1.4.2 Authenticité

Si l'investigateur ne fait pas ressortir le lien d'une façon claire entre la preuve et l'incident, il ne pourra pas utiliser sa preuve pour prouver quoi que ce soit. Il faut montrer que la preuve est reliée à l'incident et de quelle manière.

1.4.3 Complétude

Il ne faut pas que les preuves collectées ne montrent qu'une seule facette de l'incident. Il est utile de collecter non seulement les preuves qui incriminent directement un suspect mais également, pour que l'examen soit complet, de considérer et évaluer toutes les preuves disponibles afin de voir si elles peuvent être en contradiction ou diminuer la fiabilité des preuves directes. D'un autre côté, il est indispensable de collecter aussi les preuves qui éliminent les alternatives. Si, par exemple, on peut voir qu'un attaquant a pu se connecter à une date donnée, il faut voir aussi qui a pu se connecter pendant la même période et démonter pourquoi ces autres sujets ne peuvent pas être le criminel. Cette technique est connue en anglais sous le terme « *Exculpatory Evidence* ».

1.4.4 Fiabilité

Pour que les preuves présentées soient fiables, il faut que les procédures de collecte et d'analyse de preuves ne créent pas de doute sur l'authenticité et la véracité des preuves.

1.4.5 Crédibilité

Les preuves présentées doivent être claires, faciles à comprendre par le jury. Par exemple, il est insensé de présenter le contenu binaire d'un processus en mémoire si le jury n'a aucune idée de ce que cela représente réellement. Il sera plutôt intéressant de les présenter

dans un format compréhensible par le jury et montrer la relation qui existe entre ce format et la version originale (binaire).

1.5 La chaîne de continuité des preuves (Chain of custody)

Il est important que l'investigateur puisse tracer tous les éléments de preuves depuis la scène du crime jusqu'à la cour de justice, ce qui peut être appelé la préservation de la chaîne de continuité des preuves (*Chain of custody*) ou préservation de la continuité des preuves. L'investigateur doit prouver que tel élément de la preuve était à tel endroit et dans telles conditions. Ceci s'applique à la fois aux équipements de la scène du crime et aussi aux informations qui y ont été trouvées. Si cette chaîne est brisée, l'intégrité des preuves peut être compromise et, par conséquent, la crédibilité le sera aussi.

1.6 La gestion des preuves

Afin de préserver la continuité des preuves et assurer leur intégrité et leur crédibilité, il est important d'adopter une politique rigoureuse et de suivre des procédures préétablies pour mieux gérer ces preuves. La gestion des preuves comprend les points suivants :

- Déterminer quelle preuve était dans quelle pièce d'équipement.
- Documenter dans quel endroit a été trouvée cette pièce d'équipement.
- Noter les noms de toutes les personnes qui ont manipulé ces preuves.
- Stocker les preuves dans un lieu sûr et en limiter l'accès.
- Documenter tous les processus utilisés dans l'extraction de l'information.
- S'assurer que tous les processus utilisés sont reproductibles et qu'ils donnent le même résultat.

1.7 Qualité des preuves

Afin que les preuves collectées soient d'une qualité acceptable, il faut s'assurer que seules les personnes qualifiées puisse faire les analyses forensiques et qu'elles se conforment

aux standards de la profession, par exemple : AS/NZS 7799.2:2003 (*Standard On-line Select*, 2003) et ISO/IEC 17799:2005 (*Standard On-line Select*, 2005).

1.8 Les procédures standard

Un des aspects les plus importants dans l'investigation forensique est l'établissement de procédures de réponse aux incidents. Sans ces procédures, le risque de compromettre les preuves est considérable. Une bonne procédure de réponse aux incidents peut être subdivisée en onze étapes (Mandia et Prosise, 2001) que nous détaillerons par la suite :

- Planification et préparation
- Détection d'incident
- Réponse initiale
- Formulation de la stratégie de réponse
- Sauvegarde forensique
- Investigation
- Implémentation des mesures de sécurité
- Contrôle réseau (*Network monitoring*)
- Récupération (*Recovery*)
- Présentation (*Reporting*)
- Le suivi

1.8.1 Planification et préparation

Comme les incidents n'arrivent jamais aux moments convenables, il est important de se préparer pour ces événements avant qu'ils se produisent. Une façon simple qui permet de s'assurer que toutes les preuves ont été recueillies et documentées consiste à créer une liste de

contrôle (*check-list*), qui sera utilisée comme plan de réponse initial. Il faut prévoir, dans cette liste de contrôle, des informations relatives à la chaîne de continuité des preuves et à l'intégrité des données. Parmi les informations que peut contenir une liste de contrôle, on trouve :

- Date et heure du rapport d'investigation
- Nom de la personne rapportant l'incident et la personne contact.
- Nature de l'incident
- Personnes ayant détecté l'incident
- Systèmes affectés ou compromis
    - Équipements
    - Nom et version du système d'exploitation
    - Emplacement physique
    - Informations réseaux (adresse IP, adresse MAC, nom AppleTalk, numéro de téléphone de connexion, etc.)
    - Noms des contacts du personnel support
    - Fonctions commerciales et importance du système affecté.
- Actions déjà réalisées depuis la découverte du crime jusqu'au début de l'enquête
- Autres impacts possibles de cet incident.
- Autres considérations, tel que l'aspect légal, réglementaire de l'incident

1.8.1.1 Les outils nécessaires pour le plan de réponse

Après avoir rempli la liste de contrôle, il faut se procurer les outils adéquats et l'équipement nécessaire pour mettre en œuvre le plan de réponse. Les équipements et les outils peuvent être classés en deux catégories : réutilisables et consommables (voir l'appendice A pour des listes d'équipements et outils possibles).

1.8.2 Évaluation des preuves

Les preuves numériques doivent être évaluées soigneusement, tout en respectant le contexte de chaque cas, afin de déterminer la méthodologie à suivre dans l'investigation.

Il n'y a pas de méthodologie universelle à suivre, mais chaque cas a sa propre méthodologie qu'il faut déterminer après la phase d'évaluation. Il faut se conformer au mandat de recherche (ou tout autre autorisation légale) pour respecter le champ de l'autorité légale et s'assurer que la requête est complète et que la chaîne de continuité des preuves est préservée. Il faut aussi faire une évaluation en profondeur pour déterminer les détails du cas, la nature des équipements et des logiciels, les preuves potentielles recherchées et les circonstances entourant le processus d'acquisition de preuves. Durant l'évaluation, l'investigateur déterminera ce que l'investigation du cas peut ou ne peut révéler, en considérant les points suivants :

- Voir si d'autres processus forensiques non informatique doivent d'être appliqués sur les preuves (analyse d'ADN, empreintes digitales, traces, etc.)

- Discuter la possibilité d'explorer d'autres voies d'investigation pour obtenir d'autres preuves numériques (par ex., envoyer une ordonnance de préservation à un ISP[8], identifier des supports de stockage distants, obtenir des copies de courriels, etc.).

- Considérer le rapport des composantes périphériques avec l'investigation (les équipements qui ne sont pas des ordinateurs tels que : scanner, imprimante, carte de crédit, caméra numérique, etc.).

---

[8] ISP : *Internet Service Provider* : fournisseur de service Internet.

- Déterminer les types de preuves potentielles recherchées (photographies, documents, base de données, etc.).

- Déterminer d'autres informations liées au cas étudié (alias, comptes courriels, adresses courriel, ISP utilisé, configuration réseaux et utilisateurs, fichiers système log, mots de passe, noms d'utilisateurs, etc.). Ces informations peuvent être obtenues en demandant à l'administrateur système, aux utilisateurs et aux employés.

- Vérifier le niveau de compétence des utilisateurs impliqués. Des utilisateurs compétents peuvent dissimuler des preuves en utilisant des techniques plus sophistiquées (chiffrement, stéganographie[9], etc.).

- Déterminer l'ordre dans lequel les preuves doivent être examinées.

- Déterminer si l'investigation nécessite d'autres personnes.

- Déterminer les équipements nécessaires à l'investigation.

Remarque : L'évaluation du cas peut révéler la présence d'autres preuves pertinentes pour d'autres activités criminelles.

1.8.2.1 Considérations locales

Pour s'assurer de la sécurité des personnes faisant l'investigation, il faut sécuriser la scène du crime avant et durant la recherche de preuves. Parmi les tâches à effectuer localement, on trouve :

- Identifier le nombre et le type d'ordinateurs

- Déterminer si un réseau est présent

- Interviewer l'administrateur système et les utilisateurs.

---

[9] Stéganographie : La science de communication basée sur le principe de cacher la communication; cacher un fichier dans un autre.

- Identifier et documenter tous les supports de stockage utilisés y compris les supports amovibles. Documenter leurs emplacements originaux.
- Identifier les supports de stockage non présents sur le site ou les ordinateurs distants.
- Identifier le propriétaire (ou le détenteur des licences) des logiciels.
- Évaluer les conditions générales du site (scène du crime) et leur rapport avec le crime.

1.8.2.2 Procéder à l'évaluation locale

On commence par évaluer les preuves pour déterminer dans un premier temps à quel endroit l'examen de ces preuves doit se dérouler, tel qu'un lieu de travail forensique spécialisé ou un laboratoire. Les points à prendre en considération sont :

- Le temps nécessaire pour récupérer les preuves localement.
- La méthodologie et le personnel concerné par l'évaluation des preuves.
- L'impact de la durée de recherche de preuves sur les affaires.
- Les équipements ainsi que les ressources, supports de stockage, formations et expertise nécessaires pour l'examen local.

1.8.2.3 Considération légales

Du point de vue légal, il serait peut-être nécessaire de procéder à d'autres actions afin d'élargir l'autorité de recherche sur d'autres champs. Si des preuves ont été localisées à l'extérieur du champ légal de recherche, il faut déterminer quel processus légal doit être suivi pour continuer la recherche (mandat, formulaire de consentement, etc.).

1.8.3 Acquisition des preuves

Les preuves numériques sont par nature fragiles et leur mauvaise manipulation peut entraîner leur modification ou même leur destruction. Pour ces raisons, des précautions particulières doivent être suivies afin d'en préserver l'intégrité et la validité. Afin d'arriver à

ces objectifs, il faut suivre un plan détaillé et rigoureux lors de l'acquisition des preuves (voir appendice B).

1.8.4 Extraction des preuves

D'une façon synthétique, il y a deux étapes à suivre pour extraire les preuves : préparation et extraction.

- Préparation : Il faut préparer un (ou plusieurs) répertoire, selon le cas, pour y copier les données extraites.

- Extraction : Il existe deux sortes d'extractions : physique et logique. L'extraction physique identifie et récupère les données du disque tout entier sans tenir compte de la structure des fichiers. L'extraction logique identifie et récupère les fichiers et les données en se basant sur la nature du système d'exploitation installé sur la machine, le système de fichiers ou les applications.

1.8.4.1 Extraction physique

Durant cette phase, plusieurs méthodes peuvent être appliquées : recherche pas mot clé, sculpture de fichier (*file carving*), extraction de la table des partitions et extraction des espaces non alloués du disque.

- Recherche par mot clé : elle peut être très utile dans le sens où elle permet à l'investigateur d'extraire des données qui ne peuvent pas être extraites en se basant sur le système d'exploitation et le système de fichiers.

- Sculpture de fichier (*file carving*) : cette technique appliquée sur le disque peut aider l'investigateur à récupérer des fichiers utilisables qui ne peuvent pas être extraits en se basant sur le système d'exploitation et le système de fichiers (par exemple : récupération des fichiers après formatage de haut niveau car avec ce type de formatage seulement les adresses pour accéder aux fichiers sont détruites et pas les données.).

- Table des partitions : en examinant la table des partitions, l'investigateur peut déterminer la taille utilisée du disque et la comparer avec sa taille réelle (récupérée lors du démarrage contrôlé (voir appendice B)) et il peut conclure si tout le disque est pris en compte ou s'il reste encore des parties non considérées.

1.8.4.2 Extraction logique

Cette extraction est basée sur le système d'exploitation et le système de fichiers présents sur le disque. Elle peut inclure l'extraction des données à partir des espaces actifs tels que les fichiers, les fichiers supprimés, les données de remplissage des fichiers (*file slack*[10]) et aussi à partir des espaces non alloués. Les étapes à suivre comprennent :

- Extraction de l'information sur le système de fichiers pour déterminer la structure des répertoires, les attributs des fichiers, les noms de fichiers, la date et l'heure de création des fichiers, les tailles des fichiers, les emplacements des fichiers.

- Filtrage des données afin d'identifier et éliminer certains fichiers par comparaison avec les valeurs de hachage calculées sur ces données avant l'acquisition.

- Extraction des fichiers pertinents pour l'analyse en se basant sur le nom et l'extension du fichier, son entête, son contenu et son emplacement sur le disque.

- Récupération des fichiers supprimés.

- Extraction des fichiers protégés par mot de passe, les fichiers encryptés et les fichiers comprimés.

- Extraction des données de remplissage (*file slack*).

- Extraction des espaces non alloués.

---

[10] File slack : l'espace entre la fin logique d'un fichier et la fin de la dernière unité d'allocation de ce même fichier.

1.8.5 Analyse des preuves

L'analyse est le processus d'interprétation des données extraites afin de déterminer leur signification et leur pertinence pour le cas en cours d'étude. Comme mentionné plus haut, il n'existe pas de recette toute faite que l'investigateur peut suivre. Un exemple d'analyse peut inclure les points suivants :

- Analyse par recoupement temporel (*timeframe*)
- Analyse de données cachées (*data hiding*)
- Analyse d'applications et fichiers
- Analyse d'appartenance (*ownership and possession*)
- Combinaison des différentes analyses.

1.8.5.1 Analyse par recoupement temporel (timeframe)

Cette analyse sert à déterminer quand tel ou tel événement s'est produit dans le système afin de voir qui a pu utiliser le système pendant la même période de temps. En d'autres termes, il s'agit de faire l'association entre l'événement et la personne utilisant l'ordinateur pendant la période où l'événement s'est produit. Il faut aussi tenir compte des différences qui peuvent exister entre la date et l'heure du système d'exploitation et celles du BIOS[11]. Deux méthodes peuvent être utilisées :

- Revoir la date et l'heure dans les méta-données du système de fichiers (dernière modification, dernier accès, création, changement d'état du fichier) pour faire le lien entre les fichiers pertinents et le recoupement temporel de l'investigation. Par exemple, utiliser le temps et la date de modification du fichier et déterminer quand le contenu du fichier a été modifié.

---

[11] Il y a deux dates : date du système d'exploitation (Windows par exemple) et date fournie par l'ordinateur avant le démarrage du système d'exploitation. L'utilisateur peut changer l'une ou l'autre selon son niveau d'accès à l'ordinateur.

- Revoir les fichiers de journalisation (*log*) qui peuvent être présents. Ces fichiers peuvent contenir des informations sur les erreurs générées par le système ou l'application, les informations d'installations, de sécurité, etc. Par exemple, l'analyse du fichier log de la sécurité peut nous révéler quand un utilisateur (nom et mot de passe) a pu se connecter au système.

1.8.5.2 Analyse des données cachées (data hiding)

Des données peuvent avoir été cachées dans le système et l'analyse basée sur la recherche de ce type de données peut être très utile. Elle peut révéler les intentions du criminel, ses connaissances et même les outils qu'il a utilisés. Certaines méthodes peuvent être utilisées pour ce faire :

- Faire la corrélation entre l'entête du fichier et son extension pour chercher d'éventuelles discordances, qui peuvent indiquer que des données ont été cachées intentionnellement.

- Accéder aux fichiers protégés par mot de passe, chiffrés et comprimés pour voir si des données ont été cachées aux utilisateurs non autorisés.

- Utiliser les techniques de la stéganographie pour voir si des données ont été dissimulées.

- Accéder aux régions invisibles du disque (HPA : *Host Protected Area*[12]) pour voir si des données ont été cachées dans cette zone.

1.8.5.3 Analyse d'applications et fichiers

Les applications ou les fichiers identifiés peuvent contenir des informations pertinentes pour l'investigation. Ils peuvent donner une idée sur la capacité du système et le niveau de

---

[12] *Host Protected Area* : une zone qui peut être définie sur un lecteur IDE suivant les spécifications définies par ATA4 et ultérieur. Si l'adresse maximale atteinte est inférieure à l'adresse maximale native, alors la « HPA » est présente.

connaissance de l'utilisateur. Parmi les étapes qu'il faut entreprendre pour analyser les fichiers et les applications, on trouve :

- Revoir les noms de fichiers pour voir leur rapport avec les applications.
- Examiner le contenu des fichiers suspects.
- Identifier la version et le type du système d'exploitation.
- Faire la corrélation entre les fichiers et les applications installées.
- Considérer les relations qui peuvent exister entre certains fichiers (par ex. : historique Internet – fichiers dans le cache mémoire, courriels – fichiers attachés).
- Identifier les fichiers dont le type est inconnu et déterminer leur importance pour l'investigation.
- Examiner l'espace de stockage par défaut de l'utilisateur pour identifier les fichiers qui ont été stockés dans cet espace et les fichiers qui ont été stockés dans d'autres espaces.
- Examiner la configuration de l'utilisateur.
- Analyser les fichiers contenant des méta-données. Ces fichiers peuvent être visualisés en utilisant les applications qui les ont créées. Par exemple, un document Word peut contenir les méta-données suivantes : auteur, dernière date de modification, etc.

1.8.5.4 Analyse d'appartenance (ownership and possession)

Dans certains cas, il est intéressant de savoir qui a créé, modifié ou accédé aux fichiers. Il est aussi important de savoir à qui appartenaient ces fichiers ou qui les possédait au moment où le crime s'est produit. Un fichier appartenant à une personne (*ownership*) peut être utilisé par une autre personne (*possession*), par exemple un fichier reçu, volé ou piraté. On peut se baser sur les différentes analyses décrites ci-dessus pour déterminer la propriété ou la possession du fichier.

- Faire le lien entre le sujet, l'ordinateur et la période de temps (date et heure), pour déterminer à qui appartenaient ces fichiers (propriété) et qui les possédaient durant la même période (possession).

- Le nom de fichier lui-même peut nous donner des indications sur son contenu.

- La présence de données cachées peut vouloir dire qu'il y avait une intention délibérée d'éviter que les données soient détectées.

- La découverte du mot de passe des fichiers protégés et des fichiers chiffrés peut nous informer sur l'identité du propriétaire du fichier ou de celui qui le possédait.

- Le contenu du fichier lui-même peut contenir des données qui permettent d'indiquer son propriétaire ou celui qui le possédait.

1.8.5.5 Combiner les différentes analyses

Bien entendu, les résultats obtenus en faisant l'une ou l'autre des analyses ci-dessus séparément ne sont pas suffisants. Il faut considérer tous les éléments obtenus par toutes les analyses pour avoir une image globale de l'incident et ainsi pouvoir tirer des conclusions.

1.8.6 Documentation et présentation

L'investigateur a la responsabilité de documenter les résultats de l'analyse ainsi que les étapes qui ont été suivies en menant l'enquête. Il s'agit de faire un rapport complet, précis et compréhensible, car il est destiné non seulement aux gens du métier mais aussi aux juristes qui ne sont pas forcément des informaticiens. Tout au long de son enquête, l'investigateur, doit prendre des notes qui vont l'aider lors de la rédaction de son rapport. L'appendice A décrit les informations que peut contenir une liste de notes ainsi que les points important dans un rapport d'investigation.

CHAPITRE II

QUELQUES OUTILS LOGICIELS DE LA FORENSIQUE INFORMATIQUE

2.1 Introduction

En parcourant la littérature relative au sujet, on trouve plusieurs outils (logiciels) utilisables en forensiques informatique. Chaque auteur présente, à sa manière, les outils qu'il juge pertinents pour l'analyse forensique. Comme il s'agit d'une discipline encore jeune, la majorité des outils qui ont été développés dans un but académique ou scientifique contiennent des bogues. Les logiciels commerciaux sont généralement plus fiables.

Dans ce chapitre, nous présentons quelques outils forensiques à code source libre (*open source*), en commençant par le premier outil, le « père » de tous les outils à code source libre de la forensique informatique : TCT « *The Coroner's Toolkit* » (Farmer et Venema, 1999).

2.2 TCT (The Coroner's Toolkit)

En 2000, Dan Farmer et Venema ont lancé la première version du TCT. C'est une collection d'outils UNIX qui peuvent être exécutés à partir de la ligne de commande. Ces outils servent à collecter des données à partir d'un système en exécution et permettent de faire l'analyse des systèmes de fichiers FFS (*Fast File System*) et EXT2FS (Linux). Ces outils sont destinés dans un premier lieu aux administrateurs systèmes, aux investigateurs de sécurité et aux équipes d'intervention en sécurité. TCT est composé principalement de quatre groupes de fonctions :

- Grave-Robber
- Unrm & Lazarus

- Mactime
- Autres outils.

2.2.1 Grave-Robber

L'outil « *Grave-Robber* » est un utilisé pour faire l'acquisition des données à partir du disque de la machine victime. Il peut faire l'acquisition d'importantes quantités de données s'il est exécuté sur le système tout entier, en le pointant sur la racine du système de fichiers. Il faut noter aussi que s'il est lancé à partir d'un compte avec moins de privilèges, il risque ne pas collecter toutes les données. Enfin après avoir collecté toutes les données, il crée un haché avec la fonction de hachage MD5 de toutes les données collectées. La sortie du haché est redirigée vers le fichier : « *data/hostname/MD5_all* ». Avec ce haché MD5, on peut vérifier l'intégrité des données collectées (comme mentionné au chapitre 1, section 1.8.3).

2.2.2 Unrm et Lazarus

Unrm et *Lazarus* sont deux outils distincts. Le premier sert à récupérer des données à partir des espaces non alloués du disque alors que le deuxième sert à donner un sens à ces données, en les réorganisant dans des fichiers selon leur contenu.

Unrm est un programme qui sert à récupérer les données à partir des espaces non alloués du disque. Si par exemple, nous avons un disque de 10 Go dont seulement 2 Go sont utilisés, alors *Unrm* va générer 8 Go de données. Puisque le système de fichiers UNIX essaie d'allouer à chaque fichier un bloc différent, il faut prévoir tout un disque à part pour sauvegarder les données récupérées par *unrrm*.

Lazarus est programme qui essaie de récupérer les fichiers supprimés ou les données à partir d'un flux de données (généralement à partir des espaces non alloués d'un système de fichiers UNIX, mais il peut être utilisé avec n'importe quel type de données, tel que mémoire, fichiers swap, etc.). Pour récupérer le contenu de tout un disque ou une partie, on peut utiliser la commande UNIX *dd*, qui récupère les espaces alloués et les espaces non alloués. Mais, si on ne s'intéresse qu'aux espaces non alloués, il vaut mieux utiliser *unrm*.

Lazarus prend en entrée l'une ou l'autre des sorties des commandes *dd* ou *unrm* et essaie de faire une analyse par bloc de données afin de déterminer la structure et les types de fichiers. Il a été testé avec les systèmes de fichiers UFS, EXT2, NTFS, et FAT32, mais il peut être utilisé avec n'importe quel système de fichiers.

Les sorties de *lazarus* prennent deux formes : les blocs de données et une chaîne de correspondance pour ces blocs. Les blocs sont sauvegardés dans un répertoire à part (par défaut « blocks ») dans des fichiers. Les blocs qui se suivent et qui sont du même type sont placés dans le même fichier. A la rencontre d'un nouveau bloc, s'il est du même type que le bloc précédent, il est ajouté au fichier déjà créé et si le bloc est de type différent, *lazarus* crée un nouveau fichier et y insère le bloc, et ainsi de suite. Les noms de fichiers suivent la norme suivante : `numéro_de_bloc.type.txt` ou `numéro_de_bloc.type.{jpg, gif, etc.}` s'il s'agit d'une image. Si le fichier contient plusieurs blocs, alors c'est le numéro du premier bloc qui est utilisé. Comme ces fichiers vont être visualisé dans un fureteur, on a ajouté l'extension « .txt » aux noms de fichiers pour qu'ils ne soient pas interprétés directement par le fureteur.

Exemple : le nom de fichier pour les données d'un courriel : 100.m.txt

La chaîne de correspondance générée est une suite de caractères ASCII qui correspond aux différents types de données trouvés (ex : « c » : programme C, « h » : fichier HTML, etc.). Le premier caractère de la série de blocs appartenant au même fichier est mis en majuscule.

Exemple : les caractères relatifs aux bloc d'un courriel : Mmmmm

Pour que cette sortie soit d'une taille raisonnable, une compression logarithmique est appliquée. Le premier caractère représente un bloc de données ou moins, le second caractère représente 1-2 blocs, le troisième représente 1-4 blocs, etc.

Exemple : la chaîne correspondant au courriel précédent compte cinq caractères, donc le courriel devrait comporter 1+2+4+8 plus le dernier bloc (1-16), donc au total de 16-31 blocs de données. On remarque que le dernier caractère peut représenter de 1 à 16 blocs.

En cliquant sur le premier caractère (en majuscule) d'une suite de blocs, une fenêtre s'ouvre avec le contenu de ces blocs.

Il existe un autre type d'analyse qui consiste à analyser les données bit par bit plutôt que par bloc, ce qui rend l'analyse plus lente. Cette technique est intéressante pour les données non basées sur les structures de bloc (tel que le contenu de la mémoire ou les fragments de données).

2.2.3 Mactime

Mactime est l'outil le plus important de l'ensemble. Il permet de décortiquer le système de fichiers en se basant sur les trois informations connues sous le nom *Mactime* (Modification, Access, Change). Une fois cet outil lancé sur un système de fichiers, il génère un fichier ASCII contenant des axes chronologiques (*timeline*) où chaque axe correspond à la date et l'heure de modification, d'accès et de changement d'état du fichier. Cette représentation est très utile lorsqu'on veut déterminer quel fichier a été modifié ou accédé récemment.

- mtime : dernière date où le fichier a été modifié.
- atime : dernière date où on a accédé au fichier.
- ctime : dernière date où le statut du fichier a été modifié (par exemple : *chown*, *chmod*, etc.).

2.2.4 Autres outils

Tous les autres outils peuvent être lancés par l'intermédiaire de (*Grave-Robber*), comme ils peuvent être lancés directement. Ces outils ont été pris en charge par la suite par le TSK (*the sleuth kit*) que nous verrons en détail un peu plus loin.

2.3 TCT-UTILS

Les outils du TCT présentent deux limitations. La première est que ces outils agissent au niveau bloc et inode[13] (méta-donnée) seulement, aucune notion de fichier ou de répertoire n'est utilisée pendant l'analyse des données. La deuxième limitation est la dépendance à la plate-forme: le système qui fait l'analyse doit être du même type que le système analysé.

Afin de pallier à la première limitation, Brian Carrier a développé le TCT-UTILs (Carrier, 2001) qui est un ensemble d'outils ajoutés au TCT et qui permettent de lister les noms des structures de fichiers et de répertoires et d'établir des correspondances à différents niveaux du système de fichiers (fichier-inode, inode-bloc). Ces outils ont aussi été pris en charge et intégrés au TSK.

Il existe d'autres outils qui ont été développés en se basant sur TCT. Knut Eckstein a porté TCT et TCT-UTILs sur HP-UX (Eckstein, 2002). Rob Lee a développé l'outil « Mac-Daddy », une variante de « Grave-Robber » qui fait seulement l'analyse chronologique (*timeline*)...

2.4 TSK (The Sleuth Kit)

Avec STK (Carrier, 2003), Brian Carrier a fusionné les outils d'analyse de systèmes de fichiers et les outils du TCT et du TCT-UTILs en un seul ensemble et il a ajouté d'autres fonctionnalités. Parmi les améliorations majeures apportées à l'analyse, notons la fin de la dépendance à la plate-forme ; par exemple l'investigateur peut analyser un système Solaris sur n'importe quel système UNIX sur lequel TSK a été compilé. La deuxième amélioration est que les systèmes de fichiers FAT et NTFS sont supportés. Nous allons explorer les outils du TSK selon leurs champs d'application. Il faut noter que tous ces outils prennent en entrée un disque ou une image[14] du système de fichiers du disque.

---

[13] Inode : une structure de données contenant les informations concernant les fichiers dans un système de fichier UNIX. Chaque fichier possède un inode.

[14] Une image est une représentation numérique fidèle d'un ensemble de données stockées dans un support de stockage (disque dur, etc.). Elle peut contenir d'autres méta-données (valeur de hachage, CRC, etc.) pour garantir leur authenticité.

2.4.1 Les outils agissant au niveau système de fichiers (File system layer tools)

Ces outils manipulent les données générales du système de fichiers tel que la topologie du système de fichiers (*layout*), les structures d'allocation des fichiers et les blocs de démarrage.

- **fsstat** : affiche les détails du système de fichiers et quelques statistiques. Les sorties de cette commande sont spécifiques au système de fichiers analysé. Pour un système de fichiers FAT, c'est la table d'allocation des fichiers qui est affichée, alors que pour les systèmes de fichiers utilisant les groupes comme le UFS et EXT2FS, c'est la topologie de chaque groupe qui est affichée (voir figure 2.1).

Figure 2.1 Exemple de sortie « fsstat » pour un système de fichiers LINUX.

```
# fsstat images/hda1.dd
FILE SYSTEM INFORMATION
--------------------------------------------
File System Type: EXT3FS
Group: 0:
  Inode Range: 1 - 15392
  Block Range: 0 - 32767
   Super Block: 0 - 0
   Group Descriptor Table: 1 - 1
   Data bitmap: 2 - 2
   Inode bitmap: 3 - 3
   Inode Table: 4 - 484
   Data Blocks: 485 - 32767
Group: 1:
  Inode Range: 15393 - 30784
  Block Range: 32768 - 65535
   Super Block: 32768 - 32768
   Group Descriptor Table: 32769 - 32769
   Data bitmap: 32770 - 32770
   Inode bitmap: 32771 - 32771
   Inode Table: 32772 - 33252
   Data Blocks: 33253 - 65535
Group: 2:
  Inode Range: 30785 - 46176
  Block Range: 65536 - 98303
   Data bitmap: 65536 - 65536
   Inode bitmap: 65537 - 65537
   Inode Table: 65540 - 66020
   Data Blocks: 65538 - 65539, 66021 - 98303
<...>
```

2.4.2 Les outils agissant au niveau des noms de fichiers (File name layer tools)

Ces outils traitent les structures des noms de fichiers, qui sont typiquement localisées dans le répertoire racine.

- **ffind** : cherche les noms des fichiers ou de répertoires qui utilisent un inode en particulier. Par défaut, il retourne les noms de fichiers qu'il trouve. Mais, dans certains systèmes de fichiers comme Linux ou OpenBSD, il peut aussi retourner les noms de fichiers supprimés.

- **fls** : liste les noms des fichiers et de répertoires d'une image du disque. Il peut aussi lister les noms de fichiers récemment supprimés appartenant au répertoire qui utilise un inode donné.

2.4.3 Les outils agissant au niveau des méta-données (Meta data layer tools)

Ces outils du système de fichiers traitent les structures de méta-données qui stockent les détails concernant les fichiers; par exemple, les entrées du répertoire dans un système de fichiers FAT, les entrées MFT (*Master File Table*) dans un système de fichiers NTFS et les inodes dans un système ExtX et UFS.

- **icat** : extrait les unités de données d'un fichier qui est spécifié par son adresse de méta-données (au lieu du nom de fichier). Il copie le contenu du fichier spécifié par son inode vers la sortie standard.

- **ifind** : cherche la structure des méta-données vers laquelle pointe un nom de fichier donné ou attribuée à une unité de données (bloc, *cluster*[15], etc.).

- **ils** : liste les structures des méta-données et leurs contenus.

---

[15] Le *cluster* est la plus petite unité de stockage d'un système de fichiers (utilisé sur une partition d'un disque dur). La taille des *clusters* dépend de la taille des partitions et de leur format (FAT16, FAT32, NTFS).

- **istat** : affiche les statistiques et les détails concernant une structure particulière de méta-données dans un format facile à lire.

2.4.4 Les outils agissant au niveau des unités de données (Data unit layer tools)

Ces outils traitent les unités de données dans lesquelles le contenu des fichiers est stocké. Par exemple : groupes de secteurs dans les systèmes de fichiers FAT et NTFS, et blocs et fragments dans les systèmes de fichiers ExtX et UFS.

- **dcat** : extrait le contenu d'une ou plusieurs unités de données.

- **dls** : liste les détails relatifs aux unités de données et peut extraire les espaces non alloués dans un système de fichiers (par défaut). C'est la version qui a remplacé l'outil *unrm* du TCT.

- **dstat** : affiche les statistiques relatives à une unité de donnée dans un format facile à lire.

- **dcalc** : étant donnée une image récupérée à partir des espaces non alloués (à partir de *dls*), l'outil calcule son adresse dans l'image originale. Il fait la correspondance entre l'image originale et l'image des espaces non alloués récupérée par *dls*. Très utile quand des preuves ont été trouvées dans des espaces non alloués.

2.4.5 Les outils du journal du système de fichiers (File system journal tools)

Ces outils du système de fichiers traitent les fichiers de journalisation que certains systèmes de fichiers possèdent (par exemple : Ext3 et NTFS). Le journal enregistre les mises à jour apportées aux méta-données. Ceci peut aider à récupérer des données récemment supprimées.

- **jcat :** affiche le contenu d'un bloc en particulier du journal du système de fichiers. Le bloc ici est spécifié par son adresse de bloc dans le journal, à ne pas confondre avec l'adresse de bloc dans le système de fichiers.

- **jls :** liste les entrées du journal du système de fichiers.

2.4.6 Les outils de gestion des supports de stockage (Media management tools)

Ces outils prennent en entrée une image du disque (ou autre support de stockage) et analysent la structure de ses partitions. Ce peut être, par exemple, les partitions DOS, les étiquettes disque de disques BSD, la table des volumes de contenus de Sun (VTOC) et les partitions Macintosh. Ces outils peuvent être utilisés pour trouver les données qui seraient cachées entre les partitions et ainsi identifier le décalage (*offset*) du système de fichiers. Ce décalage peut être utilisé par la majorité des outils vus précédemment.

- **mmls** : permet de visualiser la topologie du disque (*layout*), en incluant les espaces non alloués. La sortie identifie le type des partitions et leurs tailles ce qui facilite la tâche de la commande « dd » pour extraire la partition. Les sorties sont triées par secteur. La figure 2.2 montre les résultats de la commande mmls pour une partition DOS et la figure 2.3 montre les résultats de la même commande pour partition MAC.

Figure 2.2 Exemple de sortie « mmls » pour une partition DOS

```
Output of running 'mmls' on a DOS partition

% mmls -t dos disk.dd
DOS Partition Table
Units are in 512-byte sectors

     Slot    Start        End          Length       Description
00:  -----   0000000000   0000000000   0000000001   Primary Table (#0)
01:  -----   0000000001   0000000062   0000000062   Unallocated
02:  00:00   0000000063   0002056319   0002056257   Win95 FAT32 (0x0B)
03:  00:01   0002056320   0008209214   0006152895   OpenBSD (0xA6)
04:  00:02   0008209215   0019999727   0011790513   FreeBSD (0xA5)
```

Figure 2.3 Exemple de sortie « mmls » pour une partition MAC

```
Output from running 'mmls' on a Mac system:

# mmls -t mac mac-disk.dd
MAC Partition Map
Units are in 512-byte sectors

     Slot    Start        End          Length       Description
00:  -----   0000000000   0000000000   0000000001   Unallocated
01:  00      0000000001   0000000063   0000000063   Apple_partition_map
02:  -----   0000000001   0000000010   0000000010   Table
03:  -----   0000000011   0000000063   0000000053   Unallocated
04:  01      0000000064   0000000117   0000000054   Apple_Driver43
05:  02      0000000118   0000000191   0000000074   Apple_Driver43
06:  03      0000000192   0000000245   0000000054   Apple_Driver_ATA
07:  04      0000000246   0000000319   0000000074   Apple_Driver_ATA
08:  05      0000000320   0000000519   0000000200   Apple_FWDriver
09:  06      0000000520   0000001031   0000000512   Apple_Driver_IOKit
10:  07      0000001032   0000001543   0000000512   Apple_Patches
11:  08      0000001544   0039070059   0039068516   Apple_HFS
12:  09      0039070060   0039070079   0000000020   Apple_Free
```

2.4.7 Les outils de fichiers images (Image file tools)

Cet ensemble regroupe les outils relatifs au format de l'image montée du disque. Cette image peut être divisée (*Split image*) ou compressée.

- **img_stat** : affiche les détails du format de l'image. Le format généré est spécifique au type de l'image.

2.4.8 Les outils agissant au niveau du disque (Disk tools)

Ces outils peuvent être utilisé pour détecter et supprimer un HPA (*Host Protected Area*) dans un disque ATA. Un HPA peut être utilisé pour cacher des données qui ne seront pas copiées lors de l'acquisition des données. En fait, le HPA est une zone que le constructeur du disque peut utiliser pour y stoker des données. Ces données peuvent être supprimées par les utilisateurs sans que le fonctionnement du disque soit altéré. De même, l'utilisateur peut utiliser cet espace pour stocker ses propres donnés en utilisant des commandes ATA de bas niveau. Actuellement ces outils ne fonctionnent que sous Linux.

- **disk_stat :** détecte la présence d'un HPA et donne le nombre de secteurs réels du disque comme illustré dans la figure 2.4.

- **disk-sreset** : cet outil supprime temporairement le HPA s'il existe. La taille du disque est temporairement redimensionnée pour pouvoir faire l'acquisition du disque au complet. Après le redémarrage du système, le HPA est restauré.

Figure 2.4 Exemple de sortie « disk_stat »

```
# diskstat /dev/hdb
Maximum Disk Sector: 120103199
Maximum User Sector: 11999

** HPA Detected (Sectors 12000 - 120103199) **
```

2.4.9 Autres outils

- **hfind** : utilise un algorithme de tri binaire pour chercher des hachés dans des bases de données de hachés tel que : NIST-NSRL (*National Institute of Standards and Technology*, S.d.), HashKeeper (*HashKeeper*, S.d.) ... ou même à partir de bases de données personnelles.

- **mactime** : prend ses données à partir de **fls** et **ils** pour créer des axes chronologiques relatifs aux activités des fichiers. Cet outil du TCT a été intégré dans TSK.

- **sorter** : trie les fichiers selon leurs types, fait la vérification de l'extension des fichiers avec leurs types et fait la recherche dans les bases de données des hachés. Donc, il peut distinguer entre les fichiers authentiques et les fichiers altérés. Cet outil fait appel à d'autres outils du STK.

- **sigfind** : recherche dans un fichier une valeur binaire qui représente la signature du système de fichiers. La recherche peut être spécifiée par la taille de bloc de recherche ou par un décalage où doit commencer la recherche. Il analyse le fichier contenant l'image du disque par bloc de données et affiche les blocs contenant la signature.

Exemple : Dans les secteurs de démarrage des systèmes FAT et NTFS, on trouve la valeur hexadécimale (signature) 0x55aa dans les deux derniers octets d'un secteur de 512 octets. La commande de la figure 2.5 liste les blocs où la signature 55AA est présente. La

recherche se fait par bloc de 512 octets et commence à partir du décalage 510. La figure 2.5 montre le résultat de la commande *sigfind*.

Figure 2.5 Exemple de sortie « sigfind »

```
# sigfind -o 510 -b 512 55AA fat32.dd
Block size: 512  Offset: 510  Signature: 55AA
Block: 0 (-)
Block: 1 (+1)
Block: 2 (+1)
Block: 6 (+4)
Block: 7 (+1)
Block: 8 (+1)
Block: 12 (+4)
[...]
```

L'ordre dans lequel apparaissent les blocs contenant la signature (0, 1, 2, 6, 7, et 8) indique qu'il s'agit d'un système de fichiers FAT32.

2.5 Autopsy Forensic Browser (AFB)

Étant donné que les outils du TCT et du TCTUTILS sont des outils qu'on ne peut lancer qu'à partir de la ligne de commande, Brian Carrier a aussi développé le logiciel (*Autopsy Forensic Browser*) (Carrier, 2003) qui est une application web (HTML) qui exécute les commandes du TSK et prépare les résultats pour visualisation dans un fureteur compatible HTML. AFB et TSK permettent d'analyser les systèmes de fichiers et les volumes de disques de façon plus conviviale. Comme AFB est une application compatible HTML, on peut se connecter à cette application à partir de n'importe quelle plate-forme en utilisant un fureteur compatible HTML.

C'est une nette amélioration surtout lorsqu'on veut manipuler une quantité importante de données. Les auteurs du TCT, TCTUTILS et TSK ont préféré présenter les outils sans interface graphique parce que dans certaines situations, on préfère utiliser la ligne de commande plutôt que l'interface graphique pour exécuter les commandes sur une partie précise des données. L'exemple le plus simple est de relancer la commande après un échec. Dans ce cas, on ne veut pas refaire tout le travail, on préfère continuer à partir du point où la commande a échoué, donc utiliser la ligne de commande.

2.5.1 Modes d'analyse avec AFB

- **Analyse hors ligne (*Dead analysis*)** : Dans ce genre d'analyse, AFB et le STK sont lancés à partir d'un environnement sécurisé tel qu'un laboratoire forensique par exemple.

- **Analyse en ligne (*Live analysis*)** : Dans ce cas le système de la machine victime est analysé alors qu'il est en pleine exécution. AFB et STK sont lancés à partir d'un CD dans un environnement non fiable. Cette analyse est souvent utilisée lors d'une intervention initiale afin de confirmer l'incident. Une fois l'incident confirmé, on peut faire des analyses hors ligne.

2.5.2 Techniques de recherche de preuves avec AFB

- **Liste des fichiers :** Il s'agit d'analyser les fichiers et les répertoires, y compris les noms de fichiers supprimés.

- **Contenu de fichier :** le contenu des fichiers peut être visualisé en flux de données (non formaté), hexadécimal ou en chaînes ASCII. Les données sont interprétées et nettoyées pour éviter qu'elles ne puissent endommager le système d'analyse. Aucun langage de script n'est utilisé du côté client pour ne pas permettre à ces scripts de s'exécuter, car ils peuvent altérer ou détruire des preuves.

- **Bases de données de hachés :** Fait la recherche de fichiers connus dans une base de données de hachés pour distinguer les fichiers authentiques des fichiers altérés. AFB utilise le NSRL comme il peut utiliser des bases de données personnelles.

- **Tri des fichier selon leur type :** trie les fichiers selon leurs signature interne (entête du fichier) pour identifier les fichiers qui ont des types connus (doc, C, HTML, etc.). AFB peut aussi extraire les images graphiques. L'extension du fichier est comparée avec son type pour repérer les fichiers dont l'extension a été modifiée. De telles modifications peuvent vouloir dire que des données ont été cachées délibérément.

- **Axes chronologiques d'activités des fichiers :** AFB peut créer des axes chronologiques dans lesquels chaque entrée contient les attributs MAC (modifié, accédé, changé) pour les fichiers présents sur le disque et les fichiers supprimés.

- **Recherche par mot clé :** la recherche par mot clé dans le système de fichiers peut être réalisée à l'aide de chaînes de caractères ASCII et d'expressions régulières comme avec l'utilitaire UNIX « *grep* ». La recherche peut être faite sur tout le système de fichiers ou simplement sur la partie de l'espace non allouée. Un fichier index peut être créé pour accélérer les recherches. Les chaînes souvent recherchées peuvent être pré-configurées pour automatiser les recherches futures.

- **Analyse des méta-données :** Les structures de données des méta-données contiennent des détails concernant les fichiers et les répertoires. AFB peut afficher tous les détails concernant la structure des méta-données du système de fichiers. Cette analyse peut aider à récupérer les données supprimées. L'outil fait la recherche dans tous les répertoires pour trouver le chemin complet des fichiers auxquels réfère la structure de méta-données.

- **Analyse des unités de données :** les unités de données stockent le contenu des fichiers. Avec AFB, on peut afficher le contenu de n'importe quelle unité de donnée sous plusieurs formats : ASCII, hexadécimal et chaînes. En utilisant les types de fichiers, AFB peut chercher dans les structures des méta-données pour identifier quel fichier a occupé telle ou telle unité de données.

- **Détail d'image :** les détails du système de fichiers peuvent être visualisés en incluant la topologie du disque (*on-disk layout*) et les dates et heures d'activités du disque. Ces informations peuvent être utiles durant la récupération des données.

2.6 Nécessité d'un outil pour localiser les copies partielles d'un fichier texte

Nous avons choisi de présenter les outils les plus importants, qui sont bien sûr à code source libre, pour leur intérêt scientifique. Ces outils sont à la base de toute une gamme d'outils, notamment leurs adaptations pour d'autres plate-formes (Windows, entre autres).

Nous avons aussi vu qu'afin de valider l'intégrité de certains fichiers systèmes, le « SleuthKit » (Carrier, 2003) propose l'outil « *hfind* » qui calcule les hachés de ces fichiers et les compare aux hachés calculés avec des versions des mêmes fichiers considérés authentiques en consultant des bases de données dédiées (NIST-NSRL ou HashKeeper). Le même principe peut être appliqué pour localiser les copies identiques d'un fichier texte. Mais, l'investigateur, afin valider certaines hypothèses, peut dans certains cas vouloir localiser des copies d'un fichier donné qui ne sont pas nécessairement des copies identiques. Il suffit d'imaginer, par exemple, que l'investigateur, en analysant l'ordinateur d'une personne, a identifié un fichier texte contenant des transactions financières douteuses provenant d'une autre personne (sous la forme d'un fichier texte attaché à un courriel par exemple). Il est possible que le fichier des transactions ait évolué avec le temps depuis qu'il a été reçu. L'investigateur va entamer les démarches légales pour lancer une nouvelle enquête visant cette deuxième personne. Après avoir obtenu le mandat nécessaire, il procède à la saisie de l'ordinateur de ce deuxième suspect et va essayer de localiser la copie originale des transactions. En procédant avec la démarche classique qui consiste à calculer les hachés des fichiers présents sur le disque, il ne trouvera rien, car la copie originale a évolué avec le temps (par l'ajout d'autres transactions par exemple). Cette technique utilisant les hachés n'est efficace que pour localiser des copies identiques à 100 %. De la même manière, si l'investigateur tente d'utiliser des outils qui comparent les fichiers ligne par ligne ou caractère par caractère (la commande *diff* sous UNIX, par exemple), la comparaison échouera également car une simple ligne insérée quelque part dans la copie créera un décalage dans les lignes.

Nous avons identifié quelques outils non forensiques qui peuvent être utilisé afin de comparer deux fichiers et établir les ressemblances ou les différences entre eux. Malheureusement, On ne peut pas les utiliser pour détecter les copies partielles comme nous allons le voir, en détail, plus loin dans le chapitre 5, section 5.4.1

La question qui se pose est donc la suivante : comment peut-on localiser efficacement les copies d'un fichier qui sont légèrement modifiées ? Nous appelons ces copies des « copies partielles ». Cette nouvelle notion que nous introduisons est basée sur des notions bioinformatiques que nous verrons au prochain chapitre.

Au chapitre 4, nous présentons un nouvel outil visant à aider l'investigateur à localiser ces copies partielles. Cet outil entend combler un vide important entre les deux types de comparaisons existants, soit le hachage et la comparaison ligne par ligne. Cet outil est basé sur une approche bioinformatique qui permet de définir et d'évaluer une mesure de ressemblance entre deux fichiers textes. Mais avant d'expliquer en quoi consiste cet outil, nous nous penchons au prochain chapitre sur les techniques d'alignement de séquences génomiques utilisées en bioinformatique.

CHAPITRE III

TECHNIQUES D'ALIGNEMENTS DE SÉQUENCES

3.1 Introduction

Les grandes découvertes dans le domaine de la biologie en générale, et de la théorie de la sélection naturelle en particulier sont basées sur l'observation et l'analyse comparative de différentes espèces. De nos jours, les bioinformaticiens utilisent le même type d'analyse pour comparer les séquences biologiques telles que les protéines, l'ADN, etc., mais à l'échelle moléculaire. L'objectif de ces différentes analyses est d'identifier les éventuelles similarités entres les séquences afin de déterminer, entre autres, les liens de parenté (phylogénie) qui peuvent exister entres elles. Depuis les années 70, les quantités d'informations relatives à ces séquences se sont considérablement accrues et des banques de données sont apparues qui peuvent être consultées via le Web. Les problèmes de recherche de séquences dans ces banques de données ainsi que l'analyse de ces informations sont devenus un problème de taille, d'où la nécessité et l'apparition de plusieurs algorithmes d'alignement, de plus en plus sophistiqués (Needleman et Wunch, 1970; Smith et Waterman, 1981; Goad et Manehisa, 1982; Meyrs, 1986; Wu, Manber, Myers et Miller, 1989). Le but de ces algorithmes consiste à aligner les séquences génomiques entre elles afin d'y trouver les sites similaires. Dans ce chapitre, nous allons voir les principes de base de ces algorithmes, leur fonctionnement et leur évolution.

3.2 Distance d'édition et algorithmes d'alignement

D'une manière générale, le but des algorithmes d'alignement consiste à trouver les similarités ou les différences entre deux chaînes de symboles et d'établir une correspondance

entre ces symboles. Les solutions à ce problème trouvent leur place dans plusieurs domaines d'application, à savoir les systèmes de correction d'orthographe, les outils de comparaison de fichiers et bien sûr les études sur l'évolution des espèces. Ces algorithmes peuvent être utilisés aussi pour trouver la plus longue sous-séquence commune entre deux chaînes de caractères (*LCS : Longest Commun Subsequence*) que nous verrons en détail dans la section suivante. Le calcul d'alignement possède un problème dual qui consiste à trouver le script minimum pour transformer une chaîne de caractères en une autre en utilisant certaines opérations auxquelles on a associé des coûts. Ce script décrit une séquence d'opérations de suppression, d'insertion de caractère et/ou remplacement d'un caractère par un autre.

3.2.1 Notions préliminaires

Soient $A=a_1a_2a_3...a_M$ et $B=b_1b_2b_3...b_N$ deux séquences de longueurs M et N respectivement. Une séquence $C=c_1c_2c_3...c_L$ est appelée une sous-séquence de A si C peut être obtenue à partir de A en faisant quelques suppressions de caractères (pas nécessairement consécutives). C est appelée une sous-séquence commune à A et B si C est une sous-séquence à la fois de A et de B. C est appelée la plus longue sous-séquence commune à A et B si la longueur de C est maximale parmi toutes les sous-séquences de A et B.

Un script d'édition est une liste d'opérations d'insertion et de suppression qui transforme, par exemple, A en B. La suppression indique quel caractère de la séquence A il faut supprimer et l'insertion indique quel caractère de la séquence B il faut insérer dans A. Le plus petit script d'édition SES (*shortest edit script*) est un script d'édition dont la longueur est minimale parmi tous les scripts possibles pour passer de la séquence A à la séquence B.

Exemple (Wu, Manber, Myers et Miller, 1989) **:**

Soient A = *'a c b d e a c b e d '* et B = *'a c e b d a b b a b e d'*

La plus longue sous-séquence commune est : C = *' a c b d a b e d '*

Pour trouver le plus petit script d'édition (SES) pour passer de A à B, on procède comme il est décrit dans le tableau 3.1.

Tableau 3.1 Exemple d'un plus petit script d'édition (SES)

	Opération	Les séquences *A* et *B*
$A_1 = B_1$	Ne rien faire	A = *'a c b d e a c b e d '*, *B = 'a c e b d a b b a b e d'*
$A_2 = B_2$	Ne rien faire	A = *'a c b d e a c b e d '*, *B = 'a c e b d a b b a b e d'*
$A_3 \neq B_3$	Insérer B_3 : (e)	A = *'a c e b d e a c b e d '*, *B = 'a c e b d a b b a b e d'*
$A_4 = B_4$	Ne rien faire	A = *'a c e b d e a c b e d '*, *B = 'a c e b d a b b a b e d'*
$A_5 = B_5$	Ne rien faire	A = *'a c e b d e a c b e d '*, *B = 'a c e b d a b b a b e d'*
$A_6 \neq B_6$	Supprimer A_5 : (e)	A = *'a c e b d a c b e d '*, *B = 'a c e b d a b b a b e d'*
$A_6 = B_6$	Ne rien faire	A = *'a c e b d a c b e d '*, *B = 'a c e b d a b b a b e d'*
$A_7 \neq B_7$	Supprimer A_7 : (c)	A = *'a c e b d a b e d '*, *B = 'a c e b d a b b a b e d'*
$A_7 = B_7$	Ne rien faire	A = *'a c e b d a b e d '*, *B = 'a c e b d a b b a b e d'*
$A_8 \neq B_8$	Insérer B_7 : (b)	A = *'a c e b d a b b a b e d '*, *B = 'a c e b d a b b a b e d'*
$A_9 \neq B_9$	Insérer B_8 : (a)	A = *'a c e b d a b b a b e d '*, *B = 'a c e b d a b b a b e d'*
$A_{10} \neq B_{10}$	Insérer B_9 : (b)	A = *'a c e b d a b b a b e d '*, *B = 'a c e b d a b b a b e d'*

Donc, le plus petit script d'édition pour passer de *A* à *B* est :

« insérer B_3 », « supprimer A_5 », « supprimer A_7 », « insérer B_7 », « insérer B_8 », « insérer B_9 »

où a_i représente le $i^{ième}$ caractère dans la séquence A et b_i représente le $i^{ième}$ caractère dans la séquence B.

3.2.2 Distance d'édition

En associant des coûts aux différentes opérations d'édition, on pourra alors parler de distance d'édition entre deux séquences. En plus des deux opérations citées plus haut, deux autres opérations sont considérées : le « remplacement » d'un caractère par un autre et le « *match* » qui veut dire les deux caractères en question sont identiques et par conséquent on ne fait ni insertion, ni suppression, ni remplacement. La distance d'édition est le coût minimal pour la somme de toutes les opérations qui permettent de passer d'une chaîne à une autre.

3.2.3 Propriétés de la distance d'édition

Notons par $d(A, B)$ la distance d'édition entre deux chaînes de caractères A et B. La distance d'édition vérifie les propriétés suivantes (Giegerich et Wheeler, 1986) :

- $d(A, A) = 0$,
- $d(A, \varepsilon) = N$ avec la longueur de A : $(|A|) = N$ et ε représente la chaîne vide
- $d(A, B) = d(B, A)$,
- $d(A, B) + d(B, C) \geq d(A, C)$ (inégalité triangulaire)

Ces propriétés justifient l'appellation « distance ».

3.2.4 Alignement de séquences

L'alignement de séquences est une généralisation du calcul de la LCS et de la distance d'édition. C'est un outil très puissant qui sert à calculer et visualiser les similarités entre deux séquences. Le problème de la LCS peut être vu comme le problème d'aligner deux chaînes de caractères afin de maximiser le nombre de correspondances (matchs) entre les caractères. Cet alignement se fait par insertions de *gaps*[16] (noté « - ») dans les deux chaînes de caractères afin

[16] Dans la littérature bioinformatique, on utilise le mot anglais « gap » pour désigner un saut inséré dans la séquence en question.

d'introduire des décalages (*shift*) entre certains caractères pour faciliter l'alignement. Remarquons qu'il peut y avoir plus d'une combinaison de décalages et d'opérations d'insertion et de suppression qui conduise au même résultat. Grâce aux coûts associés aux différentes opérations, on peut éliminer certaines combinaisons qui donnent des coûts d'édition élevés. Malgré tout, il peut tout de même y avoir plusieurs alignements différents qui donnent le même coût minimal.

Dans l'exemple précédent, si on ne considère que les opérations d'insertion et de suppression et en associant le même coût à ces opérations, on aura un alignement optimal suivant :

```
A = 'ac-bdeac---bed'
B = 'acebd-a-bbabed'
```

Supposons que les coûts des différentes opérations soient attribués tel qu'indiqué au tableau 3.2.

Tableau 3.2 Exemple de coûts associés aux opérations

Opération	Coût
Insertion	1
Suppression	1
Remplacement	2
Match	0

La distance d'édition est ici : $d(A, B) = 6$.

On peut voir que la distance d'édition peut servir de métrique pour mesurer le degré de similarité entre deux chaînes de caractères alors que l'alignement nous donne une représentation visuelle de cette similarité.

Il faut noter aussi qu'il existe plusieurs variantes d'algorithmes d'alignement, chaque variante étant appliquée à un contexte particulier.

3.3 Types d'alignement

Bien que le calcul de l'alignement optimal soit un bon outil pour trouver les similarités entre deux séquences, certains détails doivent êtres clarifiés. Il s'agit de voir si les deux séquences présentent des similarités globales ou locales. En d'autres termes, est-ce que les similarités sont présentes sur toute la longueur des deux séquences (ce qui amène à rechercher un alignement global) ou simplement sur une partie (ce qui conduit à un alignement local) ?

3.3.1 Alignement global

Dans un alignement global, tout caractère d'une séquence doit être aligné soit avec un caractère de l'autre séquence, soit avec un *gap*. Needelman et Wunch (Needleman et Wunch, 1970) ont proposé une solution efficace pour trouver un alignement optimal entre deux séquences. Leur algorithme calcule l'alignement de deux séquences qui sont susceptibles de présenter une similarité globale, il est donc un bon outil pour comparer les séquences qui sont de longueurs voisines. La figure 3.1 montre un exemple d'alignement global.

Figure 3.1 Exemple d'alignement global

```
--AGATCCGGATGGT--GTGACATGCGAT--AAG--AGGCGTT
GTCCATCTG--TCTTGGGTGAC-TGCGATACAAGTTA--CCTT
```

3.3.1.1 Algorithme de Needelman et Wunsch

En 1970, Needelman et Wunch ont proposé un algorithme basé sur la programmation dynamique pour calculer l'alignement global optimal entre deux séquences protéiques ou nucléotidiques. Dans cet algorithme, il s'agit de trouver les similarités entre deux protéines en comparant leurs séquences d'acides aminés. Le nombre de correspondances (*match*) est défini comme le nombre maximal d'acides aminés d'une protéine qui peuvent être alignés avec ceux de l'autre protéine.

Fonctionnement de l'algorithme :

Soient $A=a_1a_2a_3...a_m$ et $B=b_1b_2b_3...b_n$ deux séquences protéiques de longueurs m et n respectivement. Les étapes à suivre sont les suivantes.

Remplir une matrice M de dimension mxn. Contrairement au calcul de la distance d'édition qui représente un coût minimal, ici on cherche à calculer un score maximal (ce qui veut dire que les coûts sont inversés). Chaque case $M[i,j]$ représente le score maximal entre les deux sous-séquences $a_1a_2a_3...a_i$ et $b_1b_2b_3...b_j$. Le score maximal entre les séquences protéiques A et B est donné par le contenu de la case $M[m,n]$. Pour remplir cette matrice, on utilise la relation de récurrence suivante :

$$M(i,j) = \max\begin{cases} M(i-1,j-1) + S(x_i, x_j) \\ M(i-1,j) + D \\ M(i,j-1) + D \end{cases} \quad \text{..... (1)}$$

avec :

- Les valeurs $S(x_i, x_j)$ sont les scores associés au remplacement de l'acide aminé x_i par l'acide aminé x_j. Ces scores sont des paramètres expérimentaux rangés dan une « matrice de substitution ». Cette matrice donne les probabilités pour qu'un acide aminé ou un acide nucléotidique soit muté en un autre pendant une période de temps de son évolution (Mazouzi, 2005). Parmi les matrices de substitution les plus utilisées, on trouve PAM et BLOSUM (voir appendice D).

- D : la pénalité d'ouverture d'un *gap*.

- La pénalité d'un *gap* de longueur n est calculée par la fonction suivante :

$$f(n) = D + e(n-1) \quad \text{..... (2)}$$

où : n = la longueur du *gap* et e = la pénalité d'extension d'un *gap*.

Exemple : Avec une pénalité d'ouverture d'un gap égale à -15 et une pénalité d'extension d'un gap égale à -3, la pénalité d'un gap de longueur 2 sera égale à:

$f(2) = -15 + (-3)(2-1) = -18.$

Exemple d'une matrice d'alignement (Mazouzi, 2005).

Prenons deux séquences S = *DEFLK* et T = *DCEF* et déterminons la matrice d'alignement en utilisant la matrice de substitution BLOSUM62 et avec D = -15 et e = -3.

On commence par initialiser la première ligne et la première colonne en utilisant la relation récurrente (2) de pénalités d'ouverture et d'extension de gap. Ensuite, on remplit le reste de la matrice en utilisant la relation de récurrence (1) relative à l'algorithme en question. Le tableau 3.3 montre à quoi ressemble la matrice après remplissage.

Tableau 3.3 Exemple de matrice d'alignement (alignement global)

	j=0	*D*	*E*	*F*	*L*	*K*
i=0	0	-15	-18	-21	-24	-27
D	-15	6	-9	-15	-25	-25
C	-18	-9	2	-11	-16	-28
E	-21	-16	-4	-1	-14	-15
F	-24	-18	-19	2	-1	-16

Dans cet exemple, les flèches indiquent, pour chaque case, d'où vient la valeur maximale (dans certains cas, on peut l'obtenir de deux ou trois cases). Les cases en gris, représentent l'alignement global optimal. Pour identifier cet alignement, on fait un retour en arrière (*trace back*) en commençant à partir de la dernière case *M[m,n]* et en remontant jusqu'à la case *M[0,0]* de la manière suivante :

- Si la flèche vient de la case (*i-1*, *j-1*), les deux symboles a_i et b_j seront placés en correspondance (l'un au dessous de l'autre).

- Si la flèche vient de la case (*i-1*, *j*), le symbole b_i est aligné avec un *gap* (-).

- Si la flèche vient de la case (*i*, *j-1*), le symbole a_j est aligné avec un *gap* (-).

Le résultat de l'alignement sera pour cet exemple :

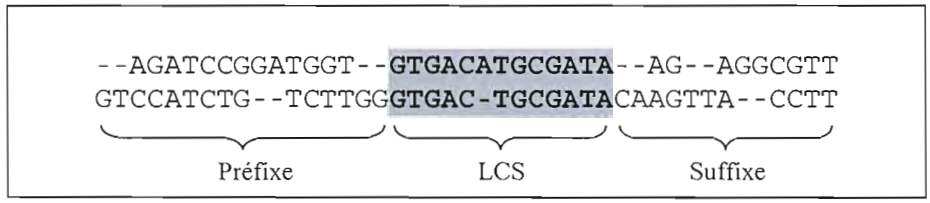
```
DEFLK
DCEF-
```

3.3.2 Alignement local

Une version modifiée de l'algorithme de Needelman et Wunch a été proposée par Smith et Waterman (Smith et Waterman, 1981) pour calculer l'alignement local.

Étant données deux séquences *A* et *B*, l'algorithme vise à trouver deux sous-séquences C_A et C_B de *A* et *B* respectivement, dont le score d'alignement est le maximum de toutes les paires de sous-séquences possibles. Seulement les deux sous-séquences contribueront au calcul du coût global de l'alignement. Cela implique que la sous-séquence qui précède la sous-séquence C_A (préfixe dans *A*) et la sous-séquence qui précède la sous-séquence C_B (préfixe dans *B*) doivent être ignorées. De même, la sous-séquence qui succède à la sous-séquence C_A (suffixe dans *A*) et la sous-séquence qui succède à la sous-séquence C_B (suffixe dans *B*) doivent être ignorées comme illustré dans la figure 3.2.

Figure 3.2 Exemple d'alignement local



Malheureusement, cette similarité locale ne peut être détectée par les algorithmes qui recherchent un alignement global. L'algorithme de Smith et Waterman est utilisé pour comparer les séquences dont les longueurs sont significativement différentes.

3.3.2.1 Algorithme de Smith et Waterman

Le but de cet algorithme est de trouver la plus longue sous-séquence commune (LCS) entre deux séquences moléculaires.

Principe de fonctionnement de l'algorithme

Cet algorithme est une modification de l'algorithme de Needelman et Wunsch. Le but de la modification est de supprimer les scores négatifs dans la matrice, ce qui permet à n'importe quelle case de la matrice d'être un point de départ pour le calcul des scores finals. Si la valeur d'une case devient inférieure à zéro, elle est réinitialisée à zéro et elle peut être utilisée comme un nouveau point de départ (Jaspard, 2006; Touzin, 2003).

La matrice est remplie comme suit :

- La première ligne et la première colonne sont initialisées à zéro.
- Les autres cases sont calculées à partir de la relation de récurrence suivante :

$$M(i,j) = \max \begin{cases} M(i-1,j-1) + S(x_i, x_j) \\ M(i-1,j) + D \\ M(i,j-1) + D \\ 0 \end{cases}$$

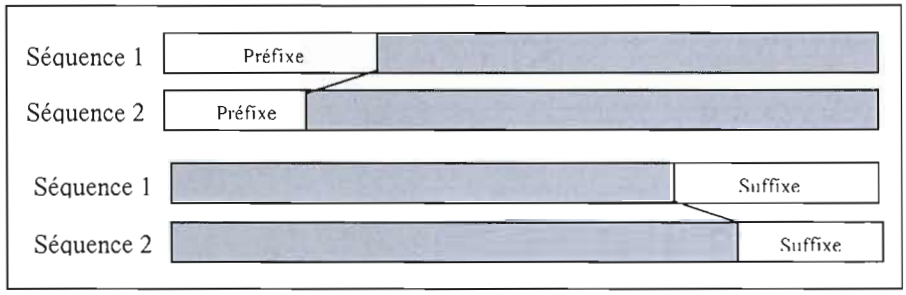
avec : $S(x_i,x_j)$ extrait de la matrice de substitution et D la pénalité d'ouverture d'un *gap* (comme mentionné pour l'algorithme précédent.)

Pour trouver un alignement local de score maximal, on fait un retour en arrière en commençant cette fois-ci par la plus grande valeur de la matrice et en remontant jusqu'à trouver une valeur nulle. On ne parcourt donc pas toute la matrice. Ce chemin qu'on cherche à établir représente la plus longue sous-séquence commune (LCS).

3.3.3 Alignement semi-global

Un autre type d'alignement a été proposé par Coull, Branch, Szymanski et Breimer (Coull, Brach, Szymanski et Breimer, 2003) pour résoudre un problème spécifique de la détection d'intrusion (*masquerade detection*) : l'alignement semi-global. Avec ce type d'alignement, on ignore soit un préfixe, soit un suffixe, mais pas les deux en même temps. Il s'agit d'une solution mitoyenne entre les alignements local et global. La figure 3.3 montre à quoi ressemble schématiquement un alignement semi-global.

Figure 3.3 Exemple d'alignement semi-global



3.3.3.1 Algorithme de Coull, Branch, Szymanski et Breimer

Les auteurs ont conçu un nouvel algorithme d'alignement qui est en fait une version modifiée de l'algorithme de Wunch et Needleman. Leur objectif est de comparer la séquence de commandes entrées par un utilisateur avec sa signature. La signature d'un utilisateur est une séquence de commandes qui définit son comportement habituel.

Fonctionnement de l'algorithme

L'algorithme est présenté à la figure 3.4, Il s'agit de remplir une matrice M dont les dimensions correspondent aux tailles des deux séquences de commandes à comparer. Pour remplir une case donnée *M[i,j]*, on évalue d'abord trois variables : *top*, *left* et *diagonal* comme suit :

- La 1ère ligne et la 1ère colonne de la matrice sont initialisées à zéro (ligne 5 de l'algorithme).

- Si la case à remplir appartient à la dernière ligne ou à la dernière colonne (lignes 7, 8, 9 de l'algorithme) :
    - *top = M[i, j-1]*
    - *left = M[i-1, j]*
- Pour les autres cases (lignes 11, 12, 13, 14 de l'algorithme) :
    - *top = max { M[i, j-1], M[i, j-1] + coût_Suppression }*
    - *left = max { M[i-1, j], M[i-1, j] + coût_Insertion }*
- *diagonal = M[i, j] + coût_Match.* (ligne 14 de l'algorithme)

Enfin: *M[i, j] = max {top, left, diagonal}.* (ligne 16 de l'algorithme)

On remarque que le score ne change pas lors du remplissage des cases de la dernière ligne et la dernière colonne (les coûts ne sont pas appliqués), ce qui permet d'ignorer les suffixes. Aussi, lors du remplissage des autres cases, si l'ajout du coût à la case adjacente (haut ou gauche) diminue le score, on n'applique pas ce coût, ce qui revient à toujours prendre le maximum entre la valeur de la case adjacente et la valeur de cette case plus le coût. Cela permet d'ignorer les préfixes. Et bien sûr, si le score devient négatif, il est réinitialisé à zéro parce que la première ligne et la première colonne sont déjà initialisées à zéro.

Détermination des coûts

Les variables *coût_Insertion* (*gInterBlk* dans l'algorithme) et *coût_Suppression* (*gUserSig* dans l'algorithme) sont les coûts associés à l'insertion et suppression d'un *gap* respectivement. Au départ, les auteurs ont attribué des coûts tel qu'indiqué dans le tableau 3.4.

Tableau 3.4 Coût associés aux opérations (alignement semi-global)

Opération	Coût
Insertion d'un *gap* dans la séquence analysée	-2
Insertion d'un *gap* dans la signature	-3
Remplacement d'une commande par un autre	0
Match (les commandes sont les mêmes)	1

Il n y a aucune justification théorique ou autre de cette définition de coûts : ils ont simplement été déterminés empiriquement pour répondre aux besoins de l'application. Ces coûts peuvent être ajustés afin d'atteindre les résultats voulus qui consistent ici à minimiser les faux négatifs (des intrusions non-détectées) et les faux positifs (des blocs légitimes détectés comme intrusions).

Figure 3.4 L'algorithme d'alignement semi-global

```
Input: string UserSig of length m, string IntrBlck of length n
1. Initialize a matrix, D, of type integer
2. for i=0 to m
3.     for j=0 to n
4.         if(j=0 or i=0)
5.             D[i][j]=0;
6.         else
7.             if(j=n or i=m)
8.                 top=D[i][j-1];
9.                 left=D[i-1][j];
10.            else
11.                top=D[i][j-1] - gUserSig;
12.                left=D[i-1][j] - gIntrBlck;
13.                if(top<0) top=D[i][j-1];
14.                if(left<0) left=D[i-1][j];
15.            diagonal=D[i-1][j-1] + matchScore(UserSig_{j-1},IntrBlck_{j-1});
16.            D[i][j]=maximum(top,left,diagonal);
17. return D[m][n];
```

3.3.4 Complexité temporelle et spatiale

Tous les algorithmes que nous avons vus sont basés sur la programmation dynamique et utilisent par conséquent une matrice dont les dimensions correspondent directement aux

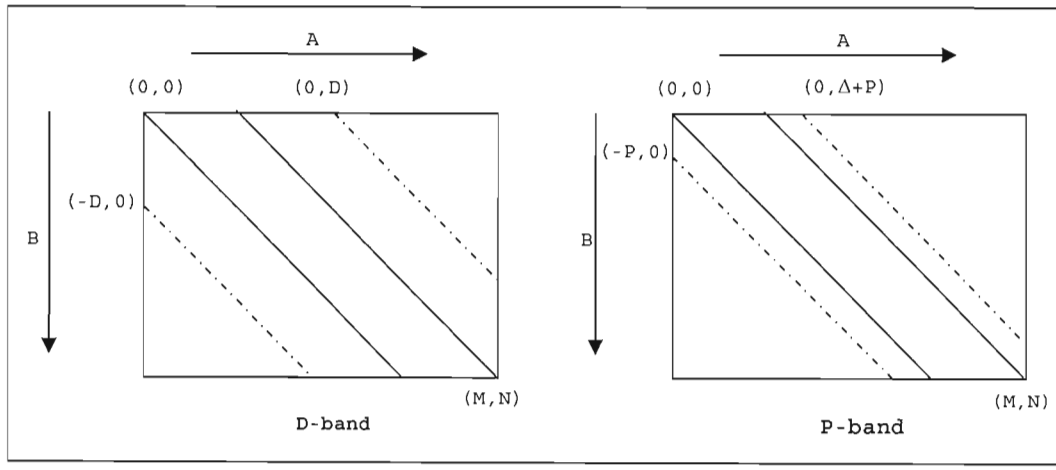
tailles des séquences à comparer. On doit dans un premier temps remplir cette matrice pour calculer le coût d'édition des deux séquences et faire par la suite un retour en arrière pour calculer les différents alignements.

On remarque que la complexité de ces algorithmes réside dans le calcul des éléments de la matrice, d'où il ressort que les complexités temporelle et spatiale sont de l'ordre de *O(MxN)* pour une matrice de dimensions *M* par *N*.

Afin de réduire la complexité spatiale, plusieurs solutions linéaires ont été proposées (Miller et Myer, 1988). Toutefois, si on ne s'intéresse qu'au calcul du coût d'édition, on pourra réduire cette complexité en utilisant seulement deux vecteurs de dimensions identiques ; dans les différents algorithmes d'alignements que nous avons vus, pour remplir une case *(i, j)* de la matrice, on a besoin seulement des trois cases adjacentes haut, gauche et diagonal. Donc à n'importe quel moment du calcul, on utilise seulement les données de la ligne en cours et de la ligne précédente.

Par contre, on peut réduire la complexité temporelle si on arrive à déterminer les cases qui entrent effectivement dans le calcul du coût d'édition et des différents alignements, et à ne calculer, autant que possible, que celles-ci. Deux algorithmes intéressants ont été proposés pour ce faire. Leur complexité temporelle dépend d'une valeur de sortie de l'algorithme. L'algorithme de E. W. Myer (Myers, 1986) présente une complexité en *O(ND)* où *D* est la distance d'édition entre les deux séquences, alors que l'algorithme de Sun Wu, Udi Manber et Gene Myers (Wu, Manber, Myers et Miller, 1989) présente une complexité en *O(NP)* où *P* est le nombre de suppressions dans le script d'édition minimal (SES). La figure 3.5 montre à quoi ressemble schématiquement le calcul de la matrice pour ces deux algorithmes. On remarque que les valeurs calculées sont confinées entre deux lignes en pointillés. Une autre remarque importante est que plus les deux séquences se ressemblent moins il y a de calculs à faire, car si la distance d'édition est moindre, on aura moins de calculs et ce qui correspond bien au fait que les deux séquences se ressemblent. Le même principe s'applique au nombre de suppressions *P* (moins il y a de suppressions, moins il y a de calculs à effectuer). Étant donné que nous n'avons pas fait appel à ces algorithmes pour notre application, nous n'allons pas les détailler davantage.

Figure 3.5 Les algorithmes D-band et P-band (Wu, Manber, Myers et Miller, 1989)



CHAPITRE IV

UN NOUVEL OUTIL POUR LA DÉTECTION DES COPIES PARTIELLES « FCOMPARE »

4.1 Introduction

Comme nous avons vu dans l'introduction, si un investigateur, en analysant l'ordinateur d'un suspect, découvre un fichier texte douteux attaché à un courriel provenant d'une deuxième personne, il va sûrement chercher à localiser la version originale de ce courriel. Après avoir saisi l'ordinateur de cette deuxième personne, comment va-il procéder pour trouver cette version originale, sachant qu'elle a peut-être été modifiée (par ajout d'autres informations, correction d'orthographe, etc.) ?

Une approche naïve consisterait à comparer la version du fichier que l'investigateur a entre les mains avec les autres fichiers textes de la machine saisie, soit ligne par ligne ou encore caractère par caractère. La comparaison ligne par ligne ne donne pas nécessairement les résultats que l'on recherche, car il se peut que des fichiers présentant de fortes similarités ne soient pas détectés, par exemple s'il y a eu un simple décalage dans les lignes. La comparaison ligne par ligne donne de meilleurs résultats si on cherche des copies identiques, mais dans ce cas, il vaut encore mieux utiliser la technique de comparaison des hachés (MD5, SHA-1, etc.) qui est moins coûteuse en temps de calcul. Par analogie, la comparaison des fichiers caractère par caractère ne nous mène à rien car un seul caractère inséré quelque part dans le document créera un décalage et par conséquent la comparaison échouera. Donc, l'investigateur a plutôt besoin d'un outil se situant entre les outils utilisant le hachage et les outils utilisant la comparaison ligne par ligne, un outil qui peut repérer les fichiers textes qui ont été légèrement modifiés, ce que nous appelons des copies partielles.

4.2 Modélisation du problème

En considérant deux fichiers à comparer comme deux séquences de caractères, on peut utiliser les outils de la bioinformatique qui ont été vus au chapitre précédent pour retrouver des copies partielles d'un fichier texte. Les calculs de l'alignement optimal entre deux séquences et de la distance d'édition peuvent nous aider grandement dans la comparaison des fichiers. La distance d'édition entre les deux fichiers nous donne une mesure de similarité entre ces deux fichiers alors que l'alignement permet de visualiser cette similarité.

4.2.1 Exigences de l'outil « fCompare »

Nous avons développé un outil logiciel (nommé « *fCompare* ») basé sur les techniques d'alignement de séquences pour localiser les copies partielles d'un fichier texte dans un système de fichiers (Bégin et Louafi, 2006). Cet outil prend en entrée un fichier texte (original) et parcourt le système de fichiers afin d'évaluer les fichiers textes existants en associant à chacun une valeur (score) qui mesure le degré de similarité avec le fichier original. Cet outil calcule aussi une valeur (seuil) qui sert à distinguer les fichiers potentiellement intéressants pour l'investigateur des autres. D'autre part, nous avons donné la possibilité à l'investigateur de visualiser un alignement optimal (c'est-à-dire un des alignements qui ont conduit au calcul du score maximal), ce qui peut aider, le cas échéant, à identifier les modifications apportées à la copie localisée. L'outil peut aussi générer un rapport détaillé imprimable des résultats de la recherche. Bien évidemment, cet outil doit être adapté à la tâche selon certains paramètres qui vont guider sa recherche, et que nous allons maintenant voir en détails.

4.2.2 Choix du type d'alignement

Nous avons donné la possibilité à l'investigateur de choisir le type d'alignement qui sera recherché. Bien sûr, chaque type est applicable dans un contexte particulier. Nous avons utilisé les algorithmes d'alignement qui ont été décrits dans le chapitre précédent, en apportant quelques modifications au niveau de la définition des coûts. Les différents types d'alignement offerts par l'outil sont :

- Alignement global

- Alignement local
- Alignement semi-global
- Automatique

Alignement global : L'utilisateur devrait utiliser cet alignement quand le fichier original et la copie recherchée sont de tailles comparables : en d'autres termes quand les deux fichiers présentent des similarités sur toute la longueur des chaînes représentant les deux fichiers. L'investigateur peut utiliser cet alignement s'il pense que le fichier recherché n'est pas susceptible d'avoir été modifié fréquemment (pas beaucoup d'ajouts, possibilité de corrections, quelques suppressions, etc.). On peut penser, par exemple, à un fichier contenant la liste des personnes contacts, à des fichiers contenant des transactions financières dont la fréquence des mises à jour est faible, etc.

Alignement local : L'investigateur devrait utiliser l'alignement local quand le fichier original et la copie recherchée sont de tailles significativement différentes, en d'autres termes, quand les deux fichiers sont susceptibles de présenter des similarités seulement sur une partie des chaînes représentant les deux fichiers. Cet alignement est intéressant pour les fichiers dont la fréquence de modifications est importante, les modifications pouvant se trouver n'importe où dans le fichier. On peut penser, par exemple, au cas des fichiers de journalisation (fichiers *log*) ou à celui d'un fichier contenant des transactions financières dont la fréquence des mises à jour est importante.

Alignement semi-global : Comme le principe de cet alignement est d'ignorer soit le préfixe ou le suffixe, il est conseillé de l'utiliser quand on pense que la copie recherchée peut avoir été modifiée par ajout ou suppression de morceaux de textes soit au début ou à la fin. Un fichier de journalisation en est un exemple typique. Même avec un fichier des transactions, si l'investigateur est sûr qu'il n'y aura que des ajouts soit au début, soit à la fin du fichier, cet alignement pourra être utile. Peu importe la taille des ajouts en préfixe ou en suffixe, ils seront ignorés dans le calcul.

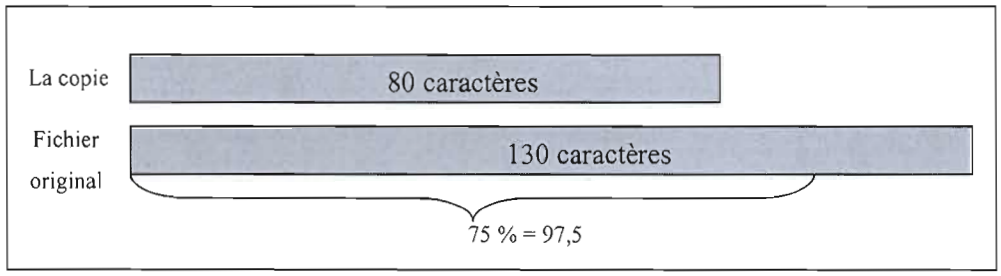
Automatique : Cette option a été ajoutée pour donner plus de flexibilité à l'outil, surtout quand l'investigateur n'est pas en mesure de prédire la taille des copies recherchées et la localisation des modifications possibles. Le principe est simple : l'outil demande à l'investigateur d'entrer un pourcentage qui le représente le « *rapport de tailles* » entre les deux fichiers analysés (réglé par défaut à 75 %) pour quantifier la différence de taille entre le fichier original et la copie recherchée afin de pouvoir décider quel alignement utiliser. Si la taille du plus petit fichier des deux est inférieure au pourcentage spécifié de la taille de l'autre fichier, on considère que les deux fichiers sont de tailles significativement différentes et par conséquent l'outil utilise un alignement local. Dans le cas contraire, les deux fichiers seront considérés de tailles comparables et par conséquent l'outil recherchera un alignement global, comme le montre la figure 4.1. Avec cette option, le type d'alignement utilisé dépendra donc de la taille de la copie.

Exemple : Supposons que le fichier original est de taille 80 caractères et que la copie qu'on veut aligner est de taille 130 caractères. Supposons aussi que l'investigateur a choisi un rapport de tailles de 75 %.

Donc, 75% de la taille du plus gros fichier des deux (ici, c'est la copie) est égale à : 130 x 0,75 = 97,5

On a : 80 (taille du plus petit fichier) < 97,5 donc les deux fichiers sont de tailles significativement différentes et par conséquent l'outil utilisera l'alignement local.

Figure 4.1 Alignement automatique



4.2.3 Attribution des coûts aux opérations

Nous avons vu au chapitre précédent que l'alignement global a été défini en fonction de coûts calculés en utilisant une matrice de substitution et que cette matrice représente dans le contexte bioinformatique des probabilités de mutations. Ces coûts sont en fait des valeurs heuristiques basées sur des théories biologiques. Mais, dans le contexte du comportement humain qui apporte des modifications à des fichiers, on ne dispose pas de données permettant de caractériser les coûts. C'est d'ailleurs pourquoi les auteurs de l'alignement semi-global (Coull, Brach, Szymanski et Breimer, 2003), défini dans un contexte non bioinformatique, ont utilisé des jeux de coûts jugés raisonnables. Il faut noter que, dans ce cas, il n' y a pas de justification théorique derrière les valeurs de coûts choisies. Ils résultent plutôt d'une validation empirique consistant à attribuer des coûts, effectuer la recherche et, à chaque fois, ajuster les coûts pour atteindre l'objectif de l'application.

Dans notre cas, nous avons choisi pour nos expérimentations des coûts qui nous paraissent raisonnables, mais l'application permet à l'investigateur de modifier les coûts à sa guise. Il est recommandé que l'investigateur effectue quelques évaluations sur des cas réels pour pouvoir donner un sens pratique aux coûts.

Les coûts que nous avons attribués par défaut sont donnés au tableau 4.1.

Tableau 4.1 Attribution des coûts aux opérations (fCompare)

Opération	Coût
Insertion (insertion d'un *gap* dans la copie)	-2
Suppression (insertion d'un *gap* dans le fichier original)	-3
Remplacement	0
Match	1

Dans cette combinaison de coûts, nous avons choisi de pénaliser l'insertion d'un *gap* dans le fichier original (-3). La pénalité est moins sévère quand un *gap* est inséré dans la

copie (-2). En effet, nous estimons qu'il faut favoriser l'insertion de *gap* dans la copie plutôt que dans le fichier original, pour localiser les fichiers qui sont le plus proche possible du fichier original (et ainsi minimiser les modifications dans le fichier original). À chaque fois qu'il y a correspondance de caractères (*match*), on ajoute 1 au score. Enfin, dans le but de pénaliser la création de décalage dans le fichier original ou dans la copie, on favorise le remplacement d'un caractère par un autre plutôt que l'insertion d'un *gap* dans les deux fichiers.

4.2.4 Définition d'un seuil d'acceptation

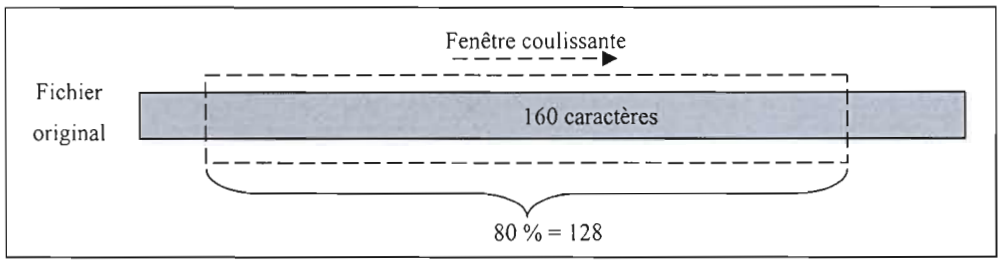
Pour aider l'investigateur à décider quels sont les fichiers qui peuvent être considérés comme des copies partielles et quels sont les fichiers qui sont loin de l'être, nous avons introduit un seuil de distance. Rappelons que notre outil donne en sortie une liste de fichiers et qu'à chaque fichier est associé un score calculé après alignement. Plus le score est grand, plus le fichier est proche du fichier original. Si le nombre de fichiers présentés est important, l'investigateur aura besoin d'un indice pour lui dire à partir de quel score il devrait commencer à fouiller dans les copies identifiées. Comment peut-on établir ce seuil ?

L'investigateur doit indiquer un « *pourcentage de tolérance* » qui caractérise indirectement le score de la copie qu'il peut accepter comme copie partielle, en fonction de la taille du fichier original. Le principe est le suivant : on sélectionne des portions du fichier original au moyen d'une fenêtre coulissante dont la taille est établie au moyen du pourcentage spécifié. Chacune des sous-séquences ainsi obtenues est alignée avec le fichier original, ce qui nous donne un score pour chaque fenêtre, comme le montre le schéma de la figure 4.2. La moyenne de ces scores donne le seuil. En d'autres termes, il s'agit de faire une évaluation d'auto-similitude du fichier original en utilisant des portions de celui-ci pour l'évaluation du seuil. La taille de ces portions, définie par le pourcentage spécifié par l'investigateur, permet donc indirectement d'établir un score (seuil) qui s'appliquera ensuite à la sélection des copies partielles : les copies retenues sont celles dont le score est supérieur ou égal à ce seuil.

Exemple : Si la taille du fichier original est de 160 caractères et le pourcentage de tolérance est de 80 %, alors la taille des sous-séquences utilisées pour établir le seuil sera :

160 x 0,80 = 128. Les sous-séquences seront alors : [1...128], [2...129], [3...130], ..., [33...160].

Figure 4.2 Définition d'un seuil d'acceptation



4.2.5 Définition d'un alignement optimal

Le meilleur score mesurant le degré de similarité entre le fichier original et les copies partielles peut en général être obtenu par plusieurs alignements optimaux distincts.

Exemple : Prenons les deux chaînes de caractères suivantes :

```
A = PARAPSYCHOLOGIQUE
B = APICULTURE
```

En utilisant un alignement global avec les coûts suivants :

Insertion = 1, Suppression = 1, Remplacement = 1, Match = 0

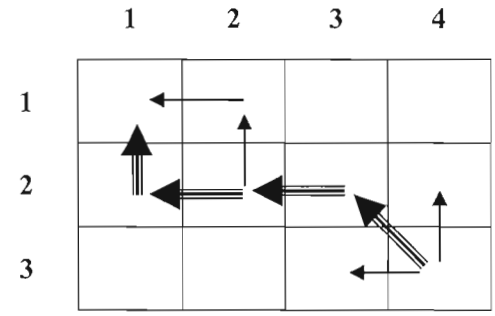
on obtient 450 alignements optimaux.

Étant donné que le nombre de ces alignements optimaux sera souvent considérable, on a choisi d'afficher un seul alignement optimal. Le principe permettant d'identifier cet alignement est le suivant : en faisant le retour sur trace, on favorise la transition diagonale sur les deux autres. Dans le cas, où il n'y a que la transition horizontale et la transition verticale, on favorise la transition horizontale, pour afficher l'alignement qui ne modifie pas le fichier original (on préfère insérer un *gap* dans le fichier analysé que de l'insérer dans le fichier

original). L'objectif ici est de mettre en évidence les changements apportés à la copie plutôt qu'au fichier original.

Par exemple, dans le schéma de la figure 4.3, la valeur de la case (3,4) peut être obtenue à partir des trois cases adjacentes. On favorisera pour l'affichage la diagonale. De même, la valeur de la case (2,2) peut être obtenue à partir de la case (1,2) qui est sur la même ligne verticale ou de la case (2,1) qui est sur la même ligne horizontale. On préfèrera dans ce cas la case (2,1) qui correspond à introduire un *gap* dans la copie.

Figure 4.3 Choix de l'alignement optimal à afficher



4.3 Conception et fonctionnement de « fCompare »

Comme première version, nous avons préféré développer une application avec une interface graphique. Une des raisons qui nous ont poussé à aller dans ce sens est que nous avons voulu montrer au moins un alignement optimal, c'est-à-dire, comment se comparent les deux fichiers après alignement. Il ne faut pas oublier que les algorithmes d'alignement peuvent nous donner plus d'un alignement optimal. Après avoir fait ce choix, nous avons vu qu'il serait très intéressant de donner la possibilité à l'investigateur de générer un rapport détaillé avec des statistiques relatives aux fichiers identifiés.

4.3.1 Choix du langage de programmation

Dans notre cas, on peut dire que tous les langages de programmation disponibles sur le marché auraient pu faire l'affaire. Vu que les systèmes Windows sont utilisés par la plupart des ordinateurs qui existent à travers le monde, et vu que l'environnement de développement

offert par Microsoft, « *Visual Studio.NET* », est plus que convivial et permet le développement rapide d'applications (RAD), nous avons choisi de développer notre outil avec le plus récent langage intégré dans cet environnement, à savoir, C#. Cela implique que pour pouvoir utiliser *fCompare*, l'investigateur doit installer la dernière version du « *Microsoft.NET Framework* » qui peut être téléchargé gratuitement[17]. Notre choix ne s'est pas basé sur des contraintes techniques mais plutôt sur deux points : convivialité et rapidité de développement.

4.3.2 Plateforme et système de fichiers

Nous avons développé et testé notre outil sur une plateforme Windows XP avec un système de fichiers NTFS. Comme notre analyse s'effectue au niveau fichier, nous pensons qu'il n'y aura aucun problème pour faire la même analyse sur d'autres systèmes de fichiers (FAT, FAT32, etc.) ou même sur d'autres plateformes (UFS d'UNIX, EXT2FS de Linux, etc.) parce que le principe de l'outil consiste, tout simplement, à convertir tout le fichier en une chaîne de caractères, y compris les caractères de contrôles (par exemple la fin de ligne: <CR>< LF> pour Windows, <LF> pour Unix, etc.). Donc, les caractères de contrôle qui sont la principale différence entre les fichiers des différents systèmes de fichiers (du point de vue du contenu) sont préservés et, par conséquent, participent au calcul des alignements.

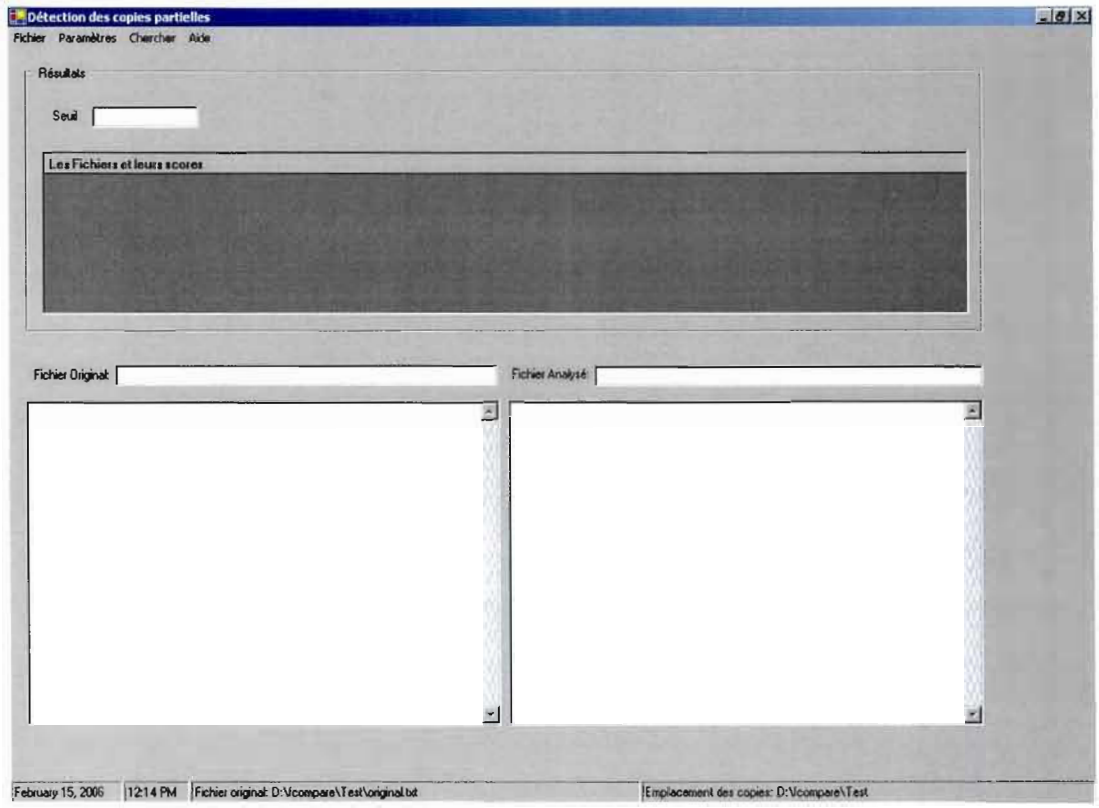
4.3.3 Fonctionnement de « fCompare »

La figure 4.4 montre l'interface principale de l'outil que nous avons développé. Elle est composée d'un menu déroulant principal. Nous allons voir en détail comment utiliser ce menu afin localiser les copies partielles d'un fichier texte.

---

[17] http://www.microsoft.com/downloads/details.aspx?familyid=0856EACB-4362-4B0D-8EDD-AAB15C5E04F5&displaylang=fr

Figure 4.4 Interface principale de « fCompare »



L'investigateur doit d'abord indiquer le fichier original à comparer, en sélectionnant le menu <Fichier><Fichier original>. Une fenêtre s'ouvre alors pour lui permettre de saisir le nom du fichier. Il est aussi possible de chercher sur le disque en utilisant le bouton <Parcourir…> (voir figure 4.5).

Figure 4.5 Sélection du fichier original

L'investigateur doit ensuite sélectionner la partie du disque où il veut effectuer la recherche des copies partielles : pour cela, il doit sélectionner le menu <Fichier><Emplacement des copies>. Une fenêtre s'ouvre afin qu'il puisse saisir le chemin complet du répertoire où la recherche va se faire, ou tout simplement sélectionner un répertoire en utilisant le bouton <Parcourir…> (voir figure 4.6).

Figure 4.6 Sélection de l'emplacement des copies partielles

4.3.3.1 Réglage des différents paramètres

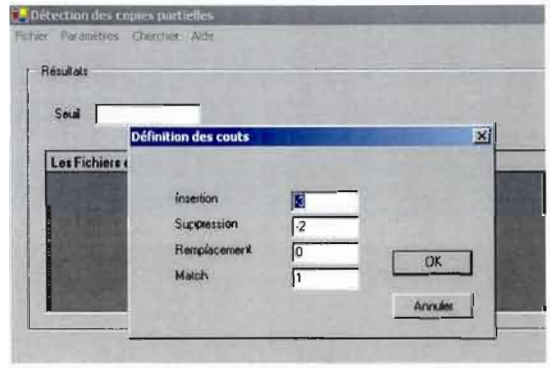
Après avoir sélectionné le fichier original et l'endroit où chercher les copies partielles, on doit par la suite régler les paramètres de la recherche. Ces paramètres sont accessibles via le menu <Paramètres> tel que décrit dans les paragraphes suivants.

Le menu <Paramètres><Type d'alignement> sert à choisir le type d'alignement qui sera utilisé pour la recherche : alignement global, local, semi-global ou automatique. Comme mentionné plus haut, si on choisit d'utiliser l'alignement automatique, l'investigateur doit régler le « *rapport de tailles* » qui caractérise la différence de taille entre le fichier original et les fichiers identifiés sur le disque (voir figure 4.7).

Figure 4.7 Sélection du type d'alignement

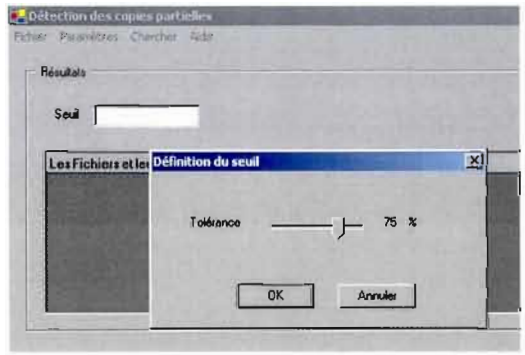
Le menu <Paramètres><Coûts> sert à régler les coûts pour les différentes opérations que l'algorithme d'alignement considère (voir figure 4.8)

Figure 4.8 Sélection des coûts des opérations



Le menu <Paramètres><Seuil> sert à régler le pourcentage de « *tolérance* » qui sera utilisé pour calculer le seuil d'acceptation (voir figure 4.9).

Figure 4.9 Réglage du pourcentage de tolérance qui entre dans le calcul du seuil



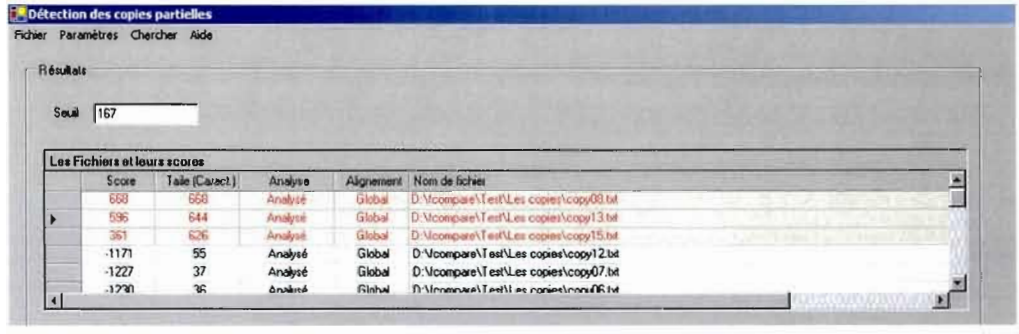
4.3.3.2 Résultats de la recherche des copies partielles

À cette étape, tous les paramètres de l'application sont réglés. L'investigateur, pour lancer la recherche, doit utiliser le menu <Recherche><Lancer la recherche>, une fenêtre s'ouvre alors affichant les fichiers textes identifiés sur le disque. Ensuite, dans la même fenêtre, une barre de progression montre la progression du calcul d'alignement. Une fois le calcul terminé, les résultats de la recherche sont affichés dans une grille placée dans

l'interface principale de l'application. La grille présente tous les fichiers identifiés sur le disque, et à chaque fichier est associé un score qui mesure le degré de ressemblance du fichier avec le fichier original. Ces fichiers sont affichés par ordre décroissant de score : les premiers fichiers, dont le score est le plus élevé, ressemblent le plus au fichier original. Les fichiers identifiés comme copies partielles sont affichés avec une couleur différente pour les distinguer des autres fichiers. Cette distinction est basée sur la valeur du seuil, qui est affichée elle aussi. D'autres informations sont affichées pour aider l'investigateur dans son analyse (voir la figure 4.10) :

- Taille du fichier (nombre de caractères).
- Type d'alignement utilisé. Dans le cas où l'investigateur a choisi un alignement automatique, il est intéressant de voir quel alignement a effectivement été utilisé (global ou local).
- Analyse : indique si le fichier a été analysé ou non : il se peut que le fichier n'ait pu être analysé parce qu'il est bloqué par une autre application durant l'analyse, ou parce que sa taille est très grande (voir section 4.3.4 plus loin).

Figure 4.10 Résultats de la recherche

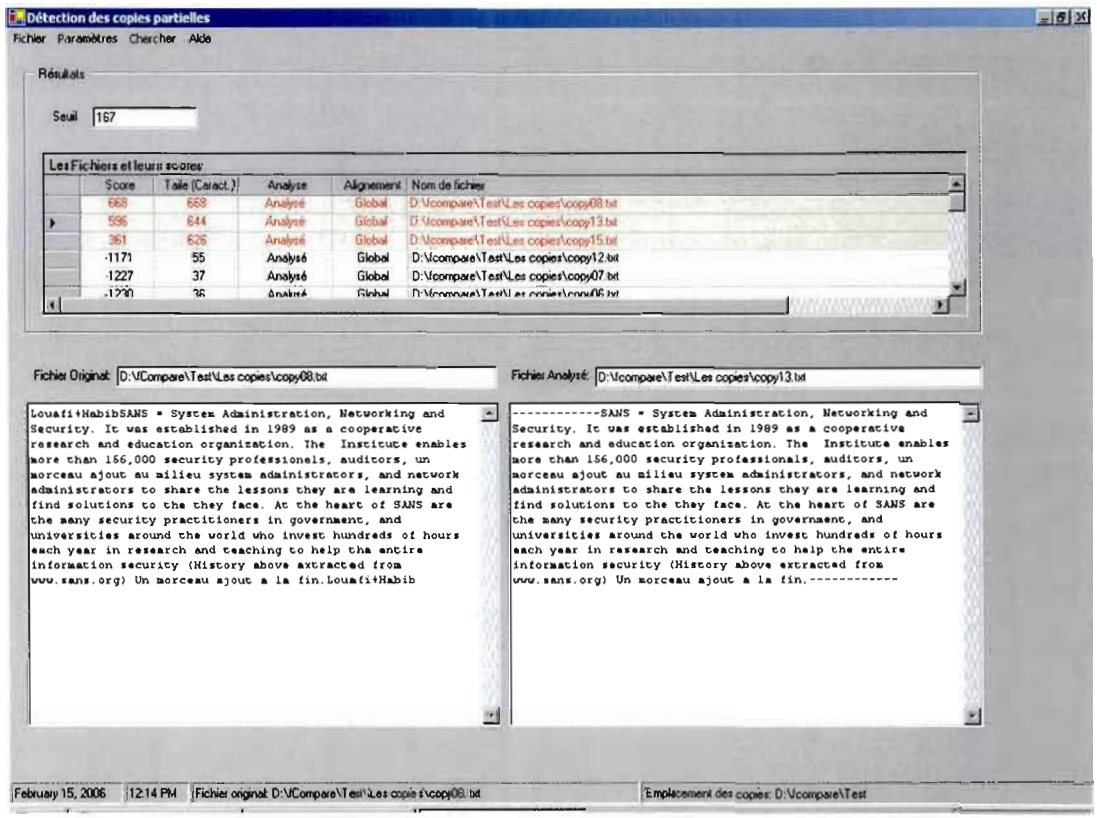


4.3.3.3 Affichage d'un alignement optimal

En sélectionnant un fichier dans la grille et utilisant le menu <Recherche><Un alignement optimal>, on peut afficher cet alignement optimal. Au bas de la grille, il y a deux

fenêtres qui présentent le fichier original et la copie partielle retenue pour affichage après alignement (voir figure 4.11).

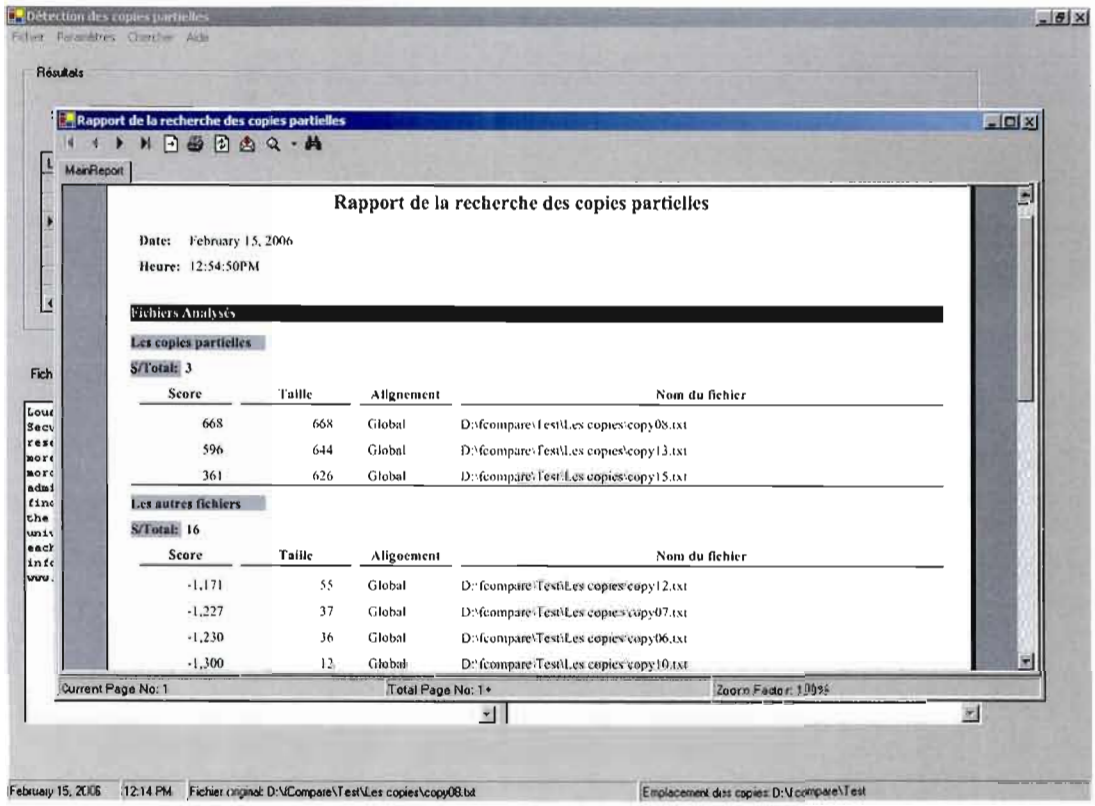
Figure 4.11 Affichage d'un alignement optimal



4.3.3.4 Rapport de la recherche des copies partielles

En choisissant le menu <Recherche><Générer un rapport>, l'investigateur peut générer un rapport qui sera affiché et qu'il est possible d'imprimer. Le rapport contient, en plus des informations de la grille, la date et l'heure du rapport ainsi que quelques statistiques : nombre total de fichiers, nombre de fichiers analysés, nombre de fichiers non analysés et le nombre de copies partielles (voir la figure 4.12).

Figure 4.12 Rapport de la recherche des copies partielles



4.3.3.5 Guide de l'utilisateur

Le menu <Aide><A propos…> affiche une fenêtre qui explique brièvement le but de l'outil, en plus d'identifier l'auteur et le numéro de version.

Le menu <Aide><Aide> sert à afficher une page HTML, qui sert de guide pour l'utilisateur de l'outil.

4.3.4 Limites de « fCompare »

La seule limitation que nous avons identifiée est la taille du fichier texte à traiter (fichier original ou fichier recherché). Comme notre approche utilise une matrice dont les dimensions sont les tailles des deux fichiers qu'on veut aligner, on est souvent limité par la taille mémoire de la machine.

Si on ne désire pas afficher un alignement optimal, on peut réduire la quantité de mémoire nécessaire en utilisant seulement deux vecteurs V1 et V2 de dimensions égales à la taille du plus petit fichier des deux. Dans les différents types d'alignements que nous avons utilisés, pour remplir une case (i, j) de la matrice, on a besoin seulement des trois cases adjacentes (i-1, j-1), (i-1, j) et (i, j-1). Donc à n'importe quel moment du calcul, on utilise seulement les données de la ligne en cours et de la ligne précédente.

On utilise ces deux vecteurs de la façon suivante. On remplit le vecteur V1 comme on fait pour remplir la première ligne de la matrice d'alignement. Le vecteur V2 sera rempli comme on remplit la deuxième ligne de la matrice mais en utilisant les valeurs du vecteur V1. À ce stade, le vecteur V1 contient exactement les mêmes valeurs que devrait contenir la première ligne de la matrice et le vecteur V2 contient les mêmes valeurs que devrait contenir la deuxième ligne de la matrice. On se sert à nouveau du vecteur V1 pour calculer les valeurs qui correspondent à la troisième ligne de la matrice parce qu'à ce niveau du calcul, on n'aura besoin que des valeurs du vecteur V2. On refait le même calcul jusqu'à calculer les valeurs du vecteur V1 ou V2 qui correspondent aux valeurs de la dernière ligne de la matrice. La dernière case du dernier vecteur calculé contiendra le score maximal. De cette façon, il est possible de traiter encore des fichiers plus volumineux. Nous n'avons pas implémenté cette technique parce qu'on a préféré donner à l'investigateur la possibilité de visualiser un alignement optimal afin de l'aider à identifier les changements apportés à la copie localisée.

CHAPITRE V

RÉSULTATS ET DISCUSSION

5.1 Introduction

Afin de valider notre approche, nous avons préparé un ensemble de scénarios d'utilisation de l'outil. L'expérimentation consiste à chercher les copies partielles d'un fichier texte donné. Dans ce but, nous avons préparé un fichier contenant un extrait de texte, nommé *original.txt*, et nous avons apporté différentes modifications à ce fichier pour générer un ensemble de fichiers qui peuvent être des copies partielles. Ces fichiers sont nommés *copy1.txt, copy2.txt,..., copy17.txt*. Pour que l'expérimentation soit la plus réaliste possible, nous avons inséré d'autres fichiers qui n'ont aucune relation avec le fichier original, ces fichiers sont nommés *other1.txt, other2.txt, other3.txt*. Le fichier *other3.tx* est un fichier vide.

5.2 Environnement d'expérimentation

Nous avons réalisé les différents tests sur un ordinateur personnel compatible IBM dont voici les caractéristiques :

- Microprocesseur : INTEL Pentium 4 à 3,20 GHz
- Mémoire vive : 1 GO
- Disque dur divisé en deux partitions :
    - la partition C est de taille : 29 GO
    - la partition D est de taille 58 GO.

- Système d'exploitation : Windows XP, édition professionnelle, version 2002 avec *service pack 2* installé

- Système de fichiers : NTFS.

5.3 Description et déroulement des tests

Les contenus des fichiers *original.txt, copy1.txt, copy2.txt,..., copy17.txt* et les fichiers *other1.txt, other2.txt, other3.txt* sont décrits à l'appendice E. Les modifications apportées au fichier original afin de créer les fichiers *copyX.txt* sont identifiables de la façon suivante : les ajouts sont identifiés en rouge et les suppressions en bleu.

Afin de faciliter la recherche, nous avons sauvegardé tous ces fichiers dans le même répertoire dont le chemin d'accès est : « *D:\fCompare\Test\ Copies*» alors que le fichier *original.txt* a été sauvegardé dans le répertoire parent «*D:\fCompare\Test* ». Nous allons dans un premier temps présenter et exécuter les différents scénarios et discuter les résultats par la suite.

Scénario 1 : Paramètres par défaut

Dans ce premier scénario, nous utilisons les paramètres par défaut. Nous voulons vérifier dans un premier temps si les paramètres que nous avons fixés par défaut permettent à l'outil de détecter les copies partielles et de voir quelle est la nature de ces copies détectées (nombre de copies, types de modifications, emplacement des modifications). Est-ce que ce choix de combinaison de coûts est justifié ? Rappelons les paramètres par défaut :

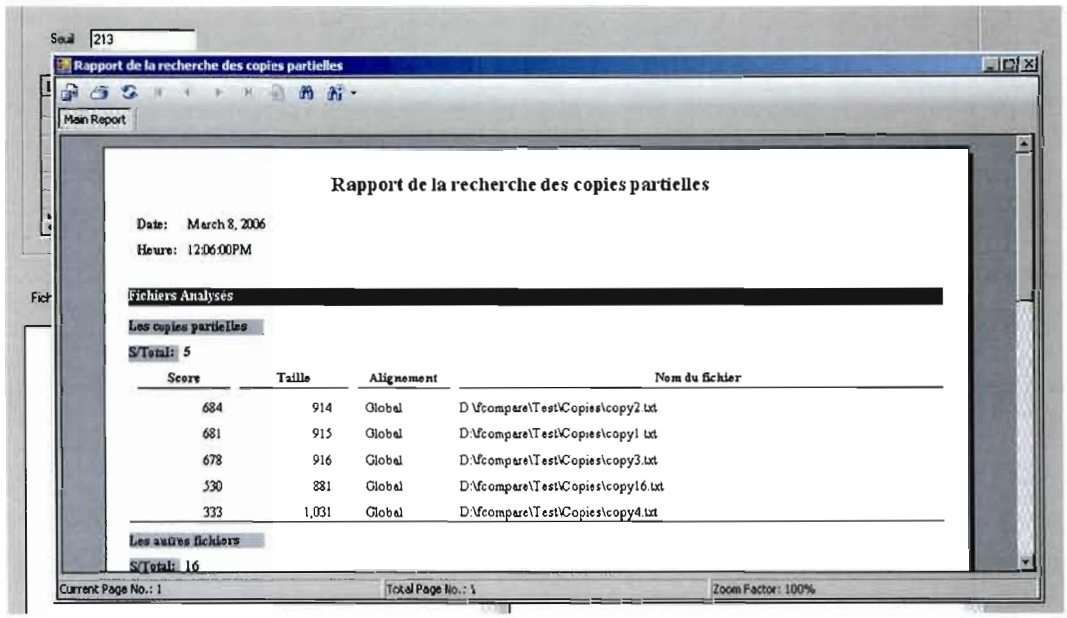
- Type d'alignement : global.

- Coûts des opérations : insertion = -2, suppression = -3, remplacement = 0, match = 1.

- Pourcentage de tolérance : 75 %.

Résultats : La figure 5.1 montre le rapport de recherche de copies partielles généré pour le scénario 1.

Figure 5.1 Résultats du scénario 1 (paramètres par défaut)

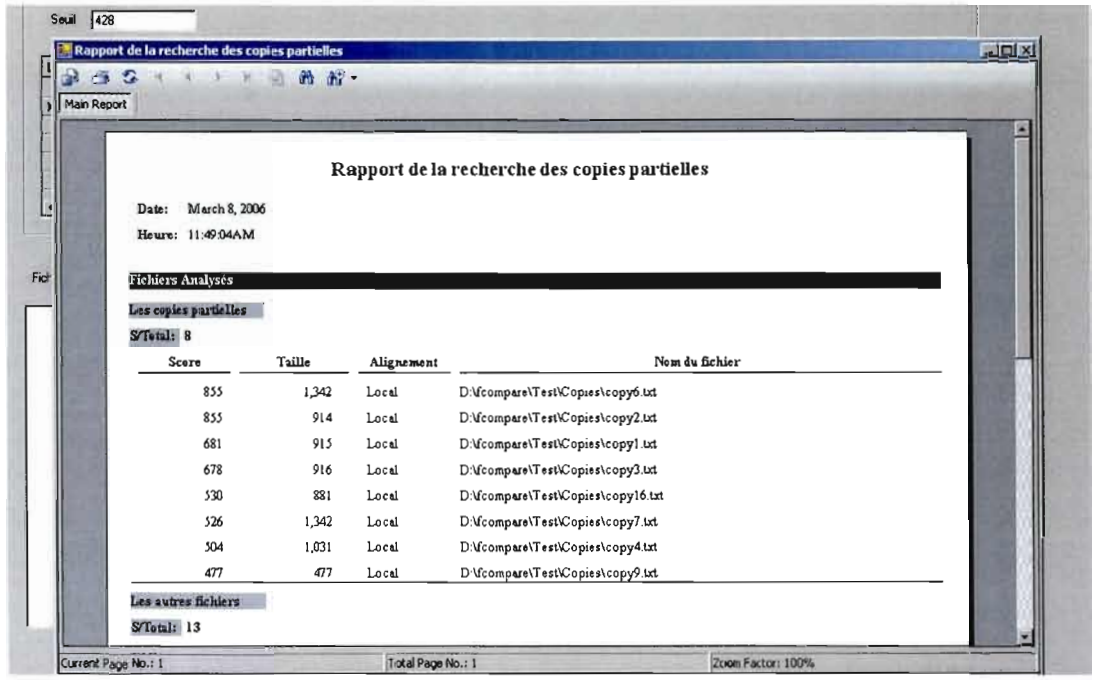


Scénario 2 : Alignement local

Dans ce scénario, nous choisissons un alignement local; les autres paramètres sont inchangés. Par ce scénario, on veut voir les nouveaux fichiers détectés par l'alignement local que l'alignement global n'a pas détecté dans le scénario précédent et montrer qu'il y a un type de similarité locale que l'alignement global ne peut pas détecter, d'où l'importance de ce type d'alignement pour la recherche de copies partielles.

Résultats : La figure 5.2 montre le rapport de recherche de copies partielles généré pour le scénario 2.

Figure 5.2 Résultats du scénario 2 (alignement local)

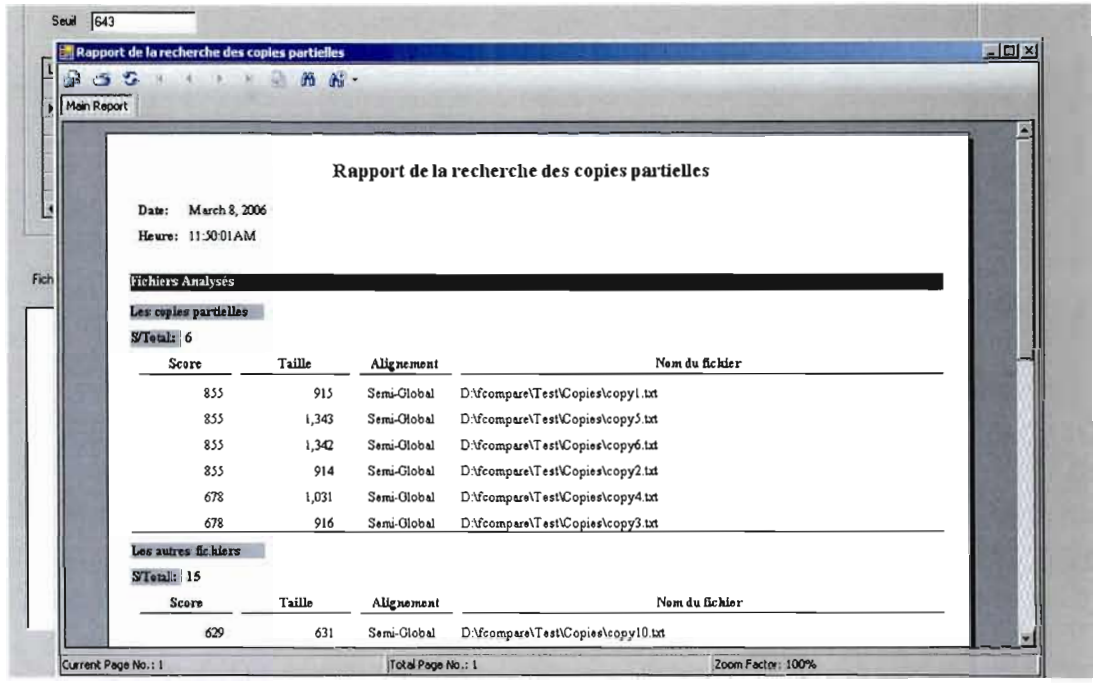


Scénario 3 : Alignement semi-global

Dans ce scénario, nous choisissons un alignement semi-global; les autres paramètres sont inchangés. Le but de ce scénario est de voir ce que ce type d'alignement apporte de nouveau par rapport aux alignements global et local utilisés dans les scénarios 1 et 2 respectivement. Est-ce qu'il y a une similarité particulière que seul l'alignement semi-global peut détecter ? Il est important aussi de montrer que chaque alignement détecte un type particulier de similarité et qu'il faut tous les considérer. On veut aussi s'assurer, par ces trois scénarios (1, 2 et 3), qu'aucun des fichiers « *otherX.txt* » ne soit détecté.

Résultats : La figure 5.3 montre le rapport de recherche de copies partielles généré pour le scénario 3.

Figure 5.3 Résultats du scénario 3 (alignement semi-global)

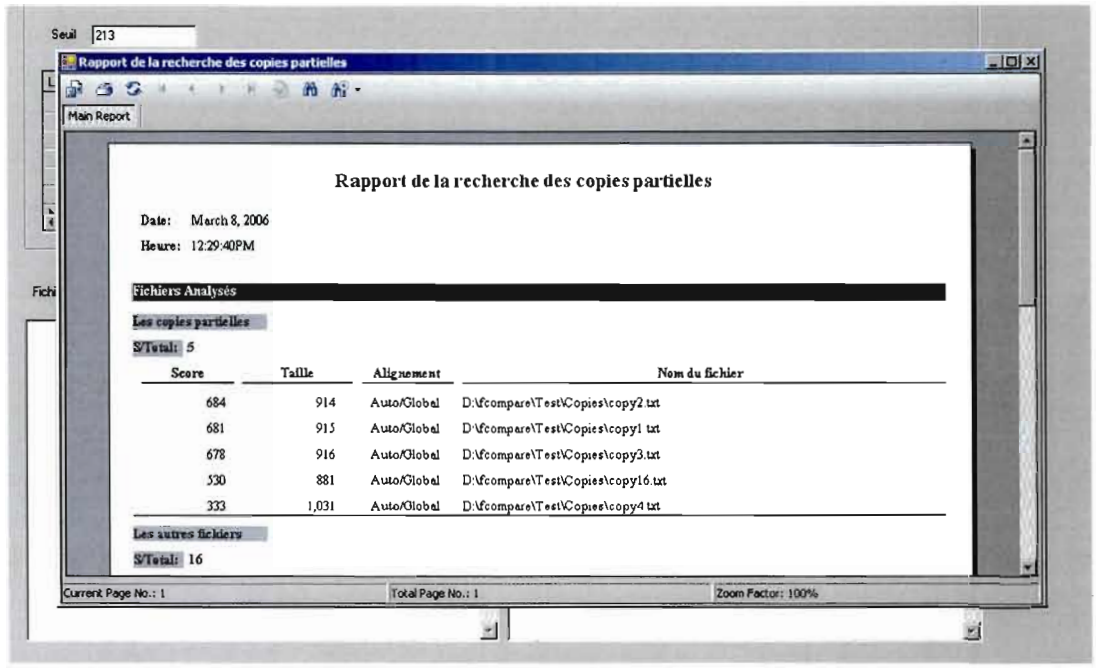


Scénario 4 : Alignement automatique

Dans ce scénario, c'est l'outil qui décide le type de l'alignement à utiliser (alignement automatique). Nous avons fixé le rapport de tailles à 75%. L'objectif de ce scénario est de montrer l'utilité de l'alignement automatique et la flexibilité qu'il donne à l'outil. Nous voulons montrer que si l'investigateur ne peut pas savoir quel type d'alignement utiliser, il pourra confier la tâche de décider du type d'alignement à l'outil. Il n'a qu'à spécifier le rapport de taille entre le fichier original et la copie.

Résultats : La figure 5.4 montre le rapport de recherche de copies partielles généré pour le scénario 4.

Figure 5.4 Résultats du scénario 4 (alignement automatique)



Scénario 5 : Effet d'un changement des coûts des opérations

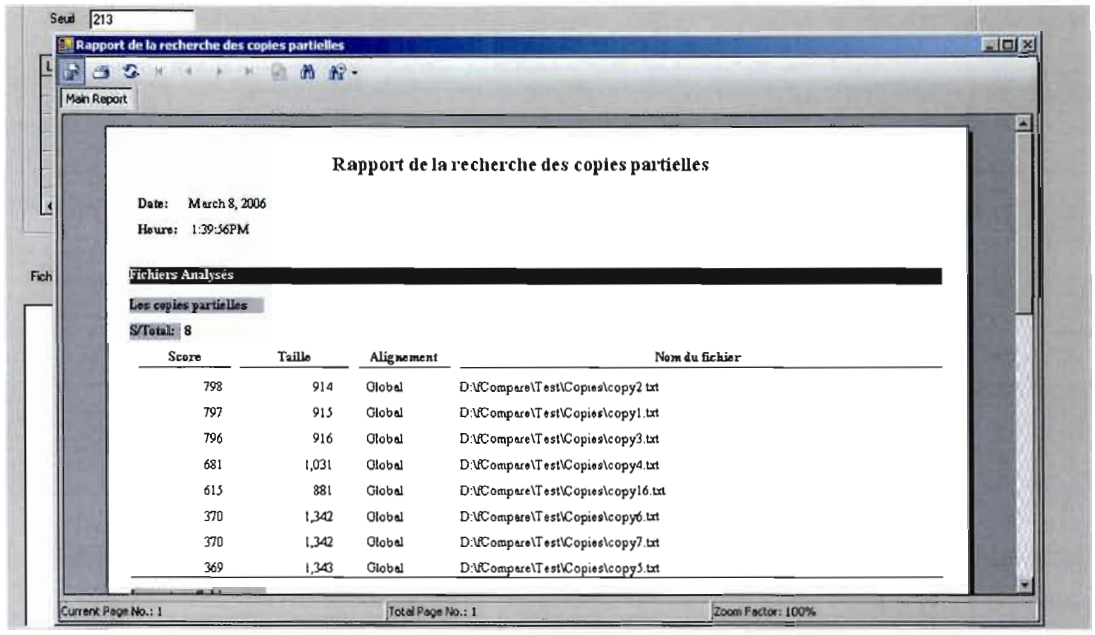
Dans ce scénario, nous utilisons un nouveau jeu de coûts pour les opérations, à savoir :

Insertion = -1, Suppression = -2, Remplacement = -10, Match = 1.

Nous avons pénalisé davantage le remplacement d'un caractère par un autre et diminué les coûts d'insertion et de suppression. Dans cette combinaison de coûts, le remplacement de caractère coûte plus cher que l'insertion de *gap* dans les deux fichiers. Le but de ce scénario est de montrer que les coûts ont une grande influence sur la définition même de copie partielle.

Résultats : La figure 5.5 montre le rapport de recherche de copies partielles généré pour le scénario 5.

Figure 5.5 Résultats du scénario 5 (effet d'un changement des coûts des opérations)

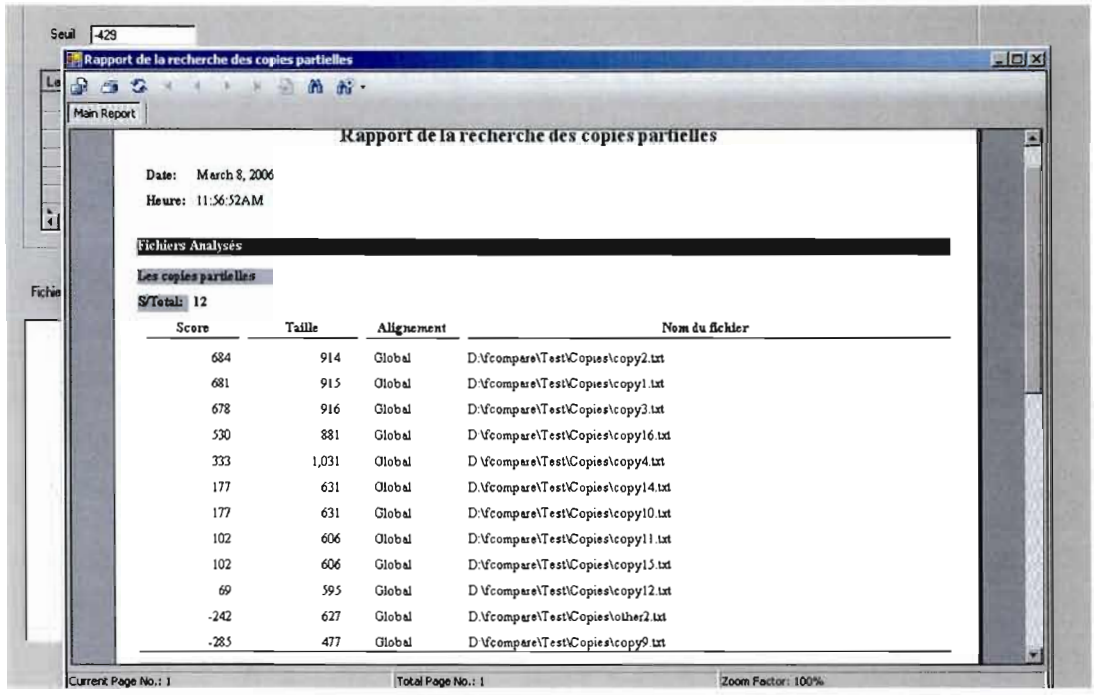


Scénario 6 : Influence du pourcentage de tolérance

Dans ce scénario, nous vérifions l'influence du pourcentage de tolérance. Nous fixons ce pourcentage de tolérance à 50 % et nous gardons les autres paramètres par défaut inchangés. Avec ce scénario, nous voulons vérifier si l'outil va détecter plus de fichiers que ceux détectés dans le scénario 1 et si, parmi les nouveaux fichiers détectés, il y a des fichiers qui ne devraient pas être considérés comme des copies. Il est important de voir à la fois l'influence et l'importance du pourcentage de tolérance dans la définition même de copie partielle.

Résultats : La figure 5.6 montre le rapport de recherche de copies partielles généré pour le scénario 6.

Figure 5.6 Résultats du scénario 6 (influence du pourcentage de tolérance)

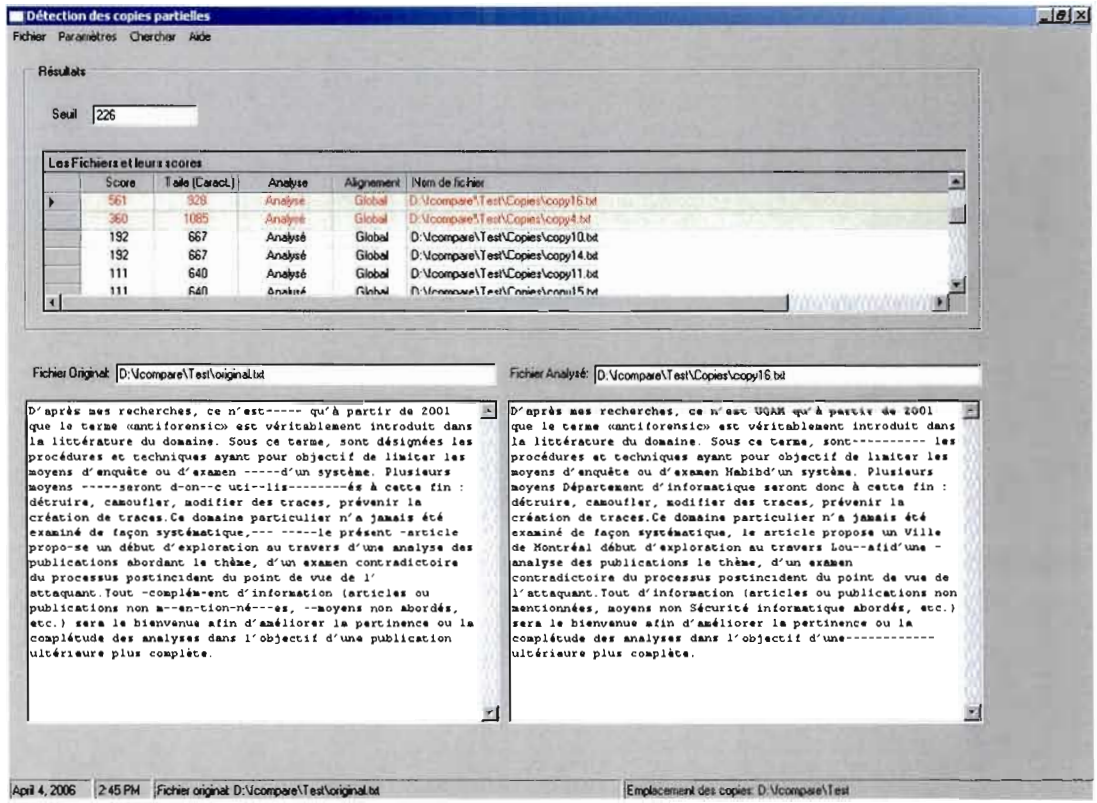


Scénario 7 : Un alignement optimal

Le but de ce scénario est de montrer à quoi ressemblent les deux fichiers après alignement. Cette visualisation peut aider à voir les similarités ou les différences entre les deux fichiers analysés. On garde les paramètres par défaut.

Résultats : La figure 5.7 montre le rapport de recherche de copies partielles généré pour le scénario 7.

Figure 5.7 Résultats du scénario 7 (un alignement optimal)

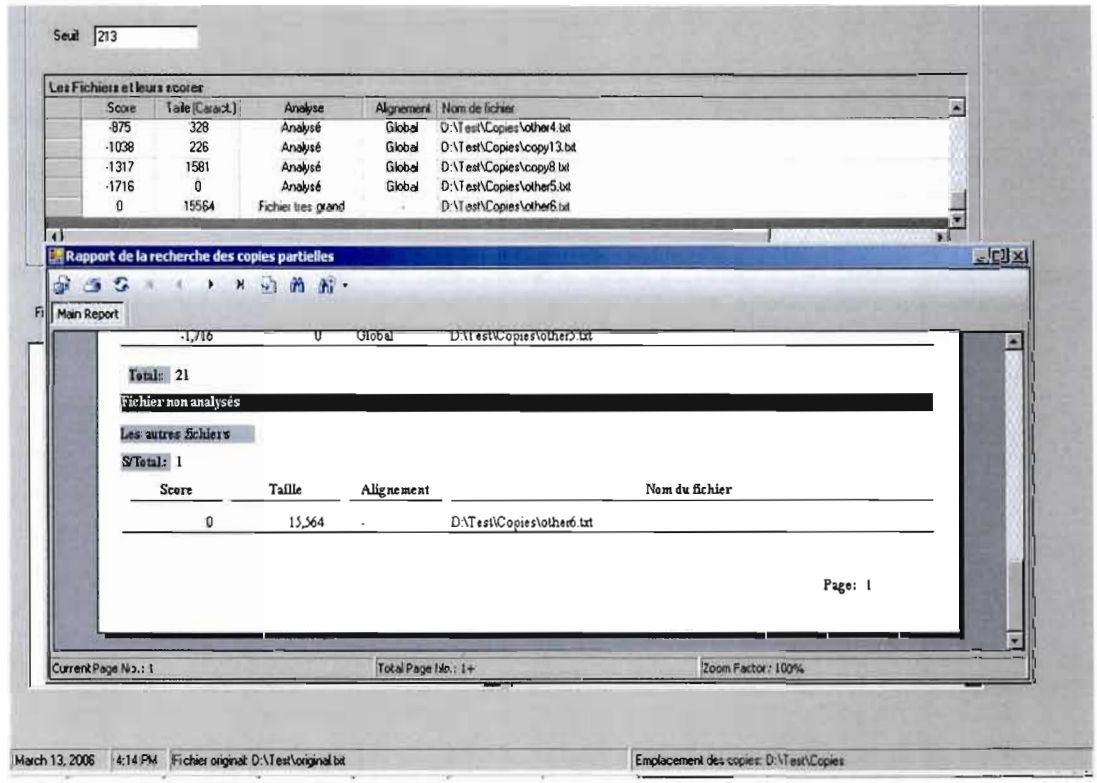


Scénario 8 : Limites de « fCompare »

Dans ce scénario, nous voulons montrer les limites de notre outil. On sait qu'il utilise une matrice dont la taille correspond au produit des tailles des deux fichiers à comparer. Nous avons augmenté graduellement la taille de cette matrice pour voir la taille maximale de la matrice que l'outil peut charger en mémoire. D'autre part, nous avons fixé cette taille et nous avons ajouté à l'ensemble des fichiers soumis au test un autre fichier texte volumineux que nous avons nommé other6.txt. Nous voulons par cela montrer qu'une fois l'outil chargé en mémoire, s'il rencontre un fichier dont la taille dépasse la taille de la matrice, il sera classé parmi les fichiers non analysés.

Résultats : La taille limite de la matrice que l'outil peut charger en mémoire est de 13965 entiers car les éléments de la matrice sont déclarés comme des entiers « *int* ». Avec la configuration matérielle et logicielle sur laquelle nous avons réalisé ce test, on peut comparer des fichiers texte dont la taille ne dépasse pas 13965 caractères. La figure 5.8 montre le rapport de recherche de copies partielles générées pour le scénario 8.

Figure 5.8 Résultats du scénario 8 (limites de « fCompare »)



5.4 Discussion

Dans le scénario 1, avec un alignement global, l'outil a détecté cinq copies partielles, à savoir dans l'ordre : *copy2.txt, copy1.txt, copy3.txt, copy16.txt et copy4.txt*. Comme nous l'avons prévu, aucun des fichiers *otherX.txt* n'a été détecté. L'outil a détecté seulement cinq fichiers en se basant sur le seuil qu'il a calculé et qui est basé sur le pourcentage de tolérance que nous avons établi par défaut à 75%. Il se peut qu'il y ait d'autres copies partielles que cet alignement n'a pas détectées, c'est ce que nous allons avoir avec les deux scénarios suivants.

Dans le scénario 2, avec un alignement local, l'outil a détecté trois autres fichiers en plus des fichiers détectés dans le scénario 1, soit *copy6.txt, copy7.txt, copy9.txt*. En regardant de près le contenu de ces fichiers (appendice D), on comprend bien qu'il s'agit d'une similarité locale que l'alignement global (scénario 1) n'a pas détectée. Encore une fois, aucun des fichiers « *otherX.txt* » n'a été détecté. Comme prévu, nous avons pu détecter d'autres copies qui présentent des similarités locales non détectées avec l'alignement global dans le premier scénario.

Le scénario 3 utilise un alignement semi-global. On remarque que l'outil a détecté trois fichiers que les deux alignements précédents (global et local) ont détectés (*copy2.txt, copy4.txt, copy3.txt*). Il a aussi détecté deux autres fichiers : un déjà détecté par l'alignement global mais non détecté par l'alignement local (*copy1.txt*) et un autre détecté par l'alignement local mais non par l'alignement global (*copy6.txt*). Ce scénario permet de constater que cet alignement est une solution mitoyenne entre les alignements global et local. Un sixième fichier (*copy5.txt*) a été détecté uniquement dans ce scénario. Dans ce fichier, une partie importante jouant le rôle d'un suffixe a été supprimée par rapport au fichier original. Cette similarité, un peu particulière, n'a été détectée ni par l'alignement global, ni par l'alignement local. Cet exemple montre bien l'importance de l'alignement semi-global par rapport aux autres.

On remarque avec ces scénarios que le choix du type d'alignement est décisif pour la recherche de copies partielles. Si l'investigateur est en mesure d'avoir une idée de la nature de la copie qu'il est en train de chercher (taille, type de modifications possibles : ajouts, suppressions ou les deux, emplacements possibles des modifications), il pourra alors choisir le type d'alignement approprié (voir le chapitre 4, section 4.2.2).

Dans le scénario 4, nous avons sélectionné un alignement automatique. Les fichiers détectés sont les mêmes que ceux qui ont été détectés par l'alignement global (scénario 1). En effet, étant donné la taille des fichiers soumis au test et le rapport de tailles fixé à 75 %, l'alignement global a été sélectionné par l'outil pour toutes les analyses sauf pour le fichier *other5.txt* qui a été traité par alignement local en raison de sa taille (fichier vide). Bien que ce scénario n'a pas détecté de nouveaux fichiers par rapport au scénario 1, nous avons montré

par ce scénario la flexibilité que l'outil propose : en une seule analyse l'outil a utilisé les deux types d'alignement global et local en se basant sur le rapport de tailles fixé à 75%.

Dans le scénario 5, les coûts des opérations ont été modifiés : nous avons pénalisé le remplacement d'un caractère par un autre (-10 au lieu de 0) et nous avons diminué les pénalités relatives aux opérations d'insertion (-1 au lieu de -2) et de suppression (-2 au lieu de -3). Avec cette combinaison de coûts, on remarque que d'autres fichiers ont été détectés qui viennent s'ajouter aux fichiers détectés par l'alignement global (scénario 1). Ce sont les fichiers (*copy6.txt, copy7.txt* et *copy5.txt*). Ce scénario montre que le choix des coûts a une grande influence sur la définition même de copie partielle (voir figure 5.5). L'outil calcule, pour chaque fichier analysé, un score qui mesure son degré de ressemblance avec le fichier original. Le calcul de ce score est basé sur la définition des coûts des opérations, d'où l'importance de bien faire attention à donner des coûts raisonnables. Si, par exemple, on inverse complètement les coûts des opérations comme suit : insertion = 2, suppression = 1, remplacement = 0, match = -1, la liste de fichiers détectés sera tout autre et ne comportera pas forcément des copies partielles comme on l'entend normalement. Il faut que le jeu de coûts ait un sens qui ne soit pas en contradiction avec le principe même du fonctionnement de l'outil.

Dans le scénario 6, nous avons diminué le pourcentage de tolérance utilisé pour le calcul du seuil (50 % au lieu de 75 %). Cela veut dire que l'investigateur accepte des copies dont les rapports de tailles avec le fichier original sont supérieurs ou égales à 50 %, ce qui explique les résultats produits par l'outil (voir figure 5.6) : 12 fichiers ont été détectés dont sept fichiers qui n'ont pas été détectés dans le scénario 1 (*copy14.txt, copy10.txt, copy11.txt, copy15.txt, copy12.txt, copy2.txt, other2.txt, copy9.txt*), même le fichier *other2.txt*, qui est très différent du fichier original, a été détecté. Ce scénario montre clairement que le pourcentage de tolérance joue un rôle important dans le calcul du seuil et, par conséquent, dans la définition de copie partielle.

Dans le scénario 7, nous avons choisi de visualiser le fichier *original.txt* et le fichier *copy16.txt* après alignement car dans ce dernier fichier il y a des ajouts et des suppressions un peu partout dans le document. On voit sur la figure 5.7 que les *gaps* insérés dans le fichier

original correspondent aux chaînes de caractères qui ont été insérées dans la copie et que les *gaps* insérés dans la copie correspondent aux chaînes de caractères qu'on a supprimés de la copie. De cette façon, l'investigateur peut clairement voir les modifications apportées à la copie.

Le scénario 8 montre les limites de *fCompare*. On voit dans la figure 5.8 que le fichier *other6.txt* n'a pas été analysé parce que sa taille est trop grande comme mentionné dans le chapitre 4, section 4.3.4. Le rapport de recherche l'a classé parmi les fichiers non analysés et la fenêtre principale de l'outil montre que ce fichier n'a pas été analysé parce que sa taille est trop grande. Comme nous l'avons indiqué précédemment, nous avons aussi mesuré avec exactitude la taille du plus grand fichier texte que l'outil peut traiter qui est de 13965 caractères.

Enfin, pour que l'outil donne des résultats acceptables, il est conseillé que l'investigateur réalise des simulations sur des cas réels pour donner un sens pratique aux différents paramètres.

5.4.1 Comparaison avec d'autres outils de comparaison de fichiers

Vu l'absence d'outils forensiques servant à localiser les copies partielles d'un fichier texte, nous avons fait une recherche sur Internet afin d'identifier les outils de comparaison de fichiers susceptibles d'être comparés avec notre outil. Voici ces outils de comparaison de fichiers avec leurs caractéristiques.

diff[18] : outil Unix/Linux qui compare deux fichiers et affiche les différences entre eux. Pour les fichiers identiques, il n'affiche rien et quand il s'agit de deux fichiers binaires, il indique seulement s'ils sont différents ou non. La comparaison utilisée est ligne par ligne. Cet outil a beaucoup évolué au fil du temps, la version actuelle est basée sur le calcul de la LCS et de la distance d'édition (Miller et Myers, 1985; Myers, 1986). Mais, la comparaison reste tout de même ligne par ligne, il ne considère pas le fichier au complet. Si par exemple toutes les lignes ont été affectées par de simples corrections d'orthographe, il dira que toutes les

[18] http://unixhelp.ed.ac.uk/CGI/man-cgi?diff

lignes sont différentes. Il affiche en sortie les lignes qui sont identiques et les lignes qui sont différentes.

diff3[19] : variante de *diff* qui sert à comparer trois fichiers en même temps, mais il utilise le même principe de comparaison que *diff*.

fComp[20] : compare deux fichiers ligne par ligne. Il peut comparer deux fichiers binaire octet par octet. Cet outil utilise le même algorithme utilisé par *diff* (Miller et Myers, 1985; Myers, 1986).

Comp[21] : outil Windows qui compare deux fichiers et affiche les caractères d'un fichier qui ne correspondent pas aux caractères de l'autre fichier. Il fait la comparaison caractère par caractère. Un caractère inséré quelque part dans le document créera un décalage qui faussera la comparaison, même si les deux fichiers présentent de fortes similarités.

FC[22] : outil Windows qui compare deux fichiers et affiche les lignes d'un fichier qui ne correspondent pas aux lignes de l'autre fichier. Cet outil fait aussi la comparaison ligne par ligne.

cmp[23] : outil Linux qui compare deux fichiers caractère par caractère. Il indique le premier octet et le numéro de la première ligne où les deux fichiers diffèrent. Si les deux fichiers sont identiques, il ne produit aucune sortie.

Nous avons identifié d'autres outils de comparaison de fichiers, à savoir *comm*[24], *sdiff*[25], etc., mais dans tous les cas le principe de comparaison reste le même, soit ligne par ligne ou encore caractère par caractère.

---

[19] http://unixhelp.ed.ac.uk/CGI/man-cgi?diff
[20] http://www.ibiblio.org/pub/Linux/devel/vc/fhist-1.10.README
[21] http://www.microsoft.com/technet/prodtechnol/windowsserver2003/fr/library/ServerHelp/c8ee75f8-96d5-4cff-9b32-2b486c99f048.mspx?mfr=true
[22] http://www.microsoft.com/technet/prodtechnol/windowsserver2003/fr/library/ServerHelp/c8ee75f8-96d5-4cff-9b32-2b486c99f048.mspx?mfr=true
[23] http://www.ss64.com/bash/cmp.html
[24] http://www.ss64.com/bash/comm.html

On remarque que tous les outils que nous avons identifiés utilisent une comparaison ligne par ligne ou caractère par caractère. On peut aussi utiliser les fonctions de hachage (MD5, SHA-1, etc.) pour détecter les copies identiques. Mais, quand il s'agit de trouver les copies qui ont été légèrement modifiées, aucun de ces outils ne peut être utilisé.

De plus, l'outil que nous avons développé ne fait pas que comparer deux fichiers textes et afficher les lignes ou les caractères qui différent ou qui se ressemblent. Étant donné un fichier texte, il parcourt le disque et identifie tous les fichiers qui peuvent être des copies légèrement modifiées. La comparaison que fait *fCompare* est : un fichier à plusieurs. Les outils que nous identifiés ne donnent pas de résultats semblables même si on les utilisent plusieurs fois afin de simuler une recherche de copies partielles car ces outils ne prennent pas les fichiers comparés en entier, mais plutôt ligne par ligne ou caractères par caractères.

---

[25] http://www.ss64.com/bash/sdiff.html

CONCLUSION ET PERSPECTIVES

Les recherches en sécurité de l'information sont focalisées sur la protection des informations contre les attaques avant qu'elles ne se produisent. La prévention d'intrusion (*IPS : intrusion prevention system*) en est un exemple typique. En ce qui concerne l'après attaque, peu de travaux ont été développés. Un des facteurs qui vont à l'encontre du développement de ce genre de recherches est que, pour ne pas nuire à leur réputation, la plupart des entreprises préfèrent absorber les pertes liées aux attaques plutôt que de les dévoiler au public. La forensique informatique, connue sous le terme anglais « *computer forensic* », est le domaine de recherche qui s'intéresse aux enquêtes après incidents. Elle représente l'ensemble des principes scientifiques et des méthodes techniques appliquées à l'investigation criminelle pour prouver l'existence d'un crime et aider la justice à déterminer l'identité de l'auteur et son mode opératoire. La forensique informatique est un domaine à la fois légal, scientifique et technique.

Au cours de ce mémoire, nous avons présenté en quoi consiste la forensique informatique, allant de l'aspect juridique à l'aspect purement technique, en passant par les différentes techniques de recherche de preuves. Nous avons notamment présenté plusieurs outils logiciels pouvant être utilisés par les investigateurs dans leurs enquêtes. Un grand nombre de ces outils sont à code source libre, ce qui a inspiré de nombreux développeurs à créer d'autres outils forensiques. Il existe également des outils commerciaux qui sont généralement fiables et bénéficient d'un bon support.

En analysant les différents outils disponibles, nous avons constaté l'absence et la nécessité d'un outil logiciel pour détecter les copies partielles d'un fichier texte, c'est-à-dire un outil permettant de localiser tous les fichiers qui ressemblent à un fichier texte donné. Nous avons adapté les algorithmes d'alignements de séquences utilisés en bioinformatique afin de développer cet outil que nous avons nommé « *fCompare* ». Cet outil vient combler un

vide entre les outils classiques de hachage et les outils de comparaison de fichiers ligne par ligne ou caractère par caractère. Pour que cet outil donne de bons résultats, il est important que l'investigateur sache en régler les paramètres.

Les principales contributions que nous avons apportées sont :

- Introduction de la notion de « *copie partielle* » pour représenter les fichiers légèrement modifiés.

- Adaptation, pour la recherche de copies partielles, de deux algorithmes d'alignement de séquences génomiques utilisés en bioinformatique (global et local) ainsi que d'un troisième algorithme d'alignement dit « semi-global » qui a été développé pour résoudre un problème spécifique de la détection d'intrusion.

- Définition d'un autre type d'alignement « *automatique* » qui utilise la notion de « *rapport de tailles* » entre le fichier original et le fichier analysé pour décider quel alignement sera effectivement utilisé (global ou local).

- Définition d'un « *seuil d'acceptation* » pour distinguer les copies partielles des autres fichiers. Ce seuil utilise un « *pourcentage de tolérance* » qui caractérise la taille de la copie recherchée par rapport au fichier original.

Travaux futurs

La bioinformatique est une discipline intéressante qui a introduit des algorithmes pouvant être utilisés dans d'autres domaines, à savoir ici la forensique informatique. En adaptant ces algorithmes avec un paramétrage approprié, il a été possible d'obtenir de bons outils de comparaison. Les algorithmes d'alignement peuvent produire en plus un certain nombre des statistiques que nous n'avons malheureusement pas eu l'occasion d'explorer afin de voir comment on pourrait les utiliser pour améliorer notre outil. Parmi ces statistiques, on peut citer : la longueur maximale, la longueur minimale ou moyenne d'un *gap*, la densité des *gaps* (les sites les plus affectés par les modifications), le nombre total des gaps, le nombre total des correspondances (*match*), le nombre total des remplacements (*mismatch*).

On sait que les versions originales d'algorithmes d'alignement utilisent des matrices de substitution qui sont liées aux probabilités de mutation, ce qui donne un aspect dynamique aux coûts associés aux différentes opérations. Malheureusement, nous n'avons pas de données empiriques qui caractérisent le comportement humain ou la langue utilisée pour produire une définition semblable des coûts des différentes opérations pour notre outil. On peut penser à l'analyse fréquentielle utilisée en cryptographie pour casser une clé par exemple. Cette analyse fréquentielle utilise des tableaux contenant les probabilités pour que deux caractères se suivent dans une langue donnée. Il serait intéressant d'explorer cette piste pour essayer de donner une structure ou une fonction que notre outil pourrait utiliser afin d'avoir des coûts non statiques qui varient en fonction des caractères rencontrés lors de l'analyse.

Les deux algorithmes cités au chapitre 3 (section 3.3.4) peuvent être utilisés pour améliorer la complexité temporelle des algorithmes d'alignement, ce qui pourrait accélérer le temps d'analyse de fichiers, surtout quand il s'agit de copies peu modifiées. Il serait aussi intéressant d'implémenter la solution proposée par (Miller et Myers, 1988) afin de réduire la complexité spatiale. De plus, l'expérimentation de notre outil sur d'autres plates-formes et sur d'autres systèmes de fichiers serait nécessaire pour étudier et, si nécessaire, améliorer sa portabilité.

Enfin, il serait intéressant de développer une version « ligne de commande » de notre outil afin de l'intégrer à des ensembles d'outils forensiques comme, par exemple, TSK ou AFB.

APPENDICE A

LES OUTILS NÉCESSAIRES POUR LE PLAN DE RÉPONSE INITIAL

Les outils nécessaires pour le plan de réponse initial sont classés en deux groupes : réutilisables et consommables.

Les composantes réutilisables comptent:

- Des stations de travail pour la duplication et l'analyse forensique. Ces stations doivent supporter les interfaces courantes (par ex., IDE et SCSI) pour la duplication des disques et offrent une plate-forme sur laquelle seront exécutés les logiciels forensiques.

- Des logiciels forensiques pour faire l'analyse, la collecte et la documentation des preuves, et dans certains cas, la duplication de disques.

- Un renifleur réseau, pour capturer le trafic du réseau.

- Du câblage réseau utilisé avec le matériel de duplication forensique et le renifleur réseau.

- Des concentrateurs (*hubs*) utilisés avec le renifleur et le matériel de duplication forensique. Ils sont aussi un moyen simple pour interconnecter les stations de travail.

- Un graveur CD ou DVD, ou n'importe quel autre équipement pour stocker et transporter les copies et les preuves.

- Des disques IDE et SCSI de grandes capacités pour stocker les copies forensiques.
- Des connecteurs SCSI et IDE de différents types et des câbles pour pouvoir connecter plusieurs disques aux équipements de duplication.
- Un jeu de tournevis pour pouvoir enlever les disques durs.
- Un disque de démarrage contenant les outils forensique de tous les différents matériels et systèmes d'exploitations qui peuvent se retrouver dans l'environnement de travail.
- Un disque comprenant des outils avec des liens exécutables, un système d'exploitation de base et des applications forensiques pour tous les matériels et systèmes d'exploitations de l'environnement pour pouvoir faire des analyses sur les systèmes en exécution.

Les composantes consommables comptent:

- Des DVD-R et/ou des CD-R vierges.
- Des étiquettes de preuves.
- Des marqueurs permanents.
- Des sacs pour les preuves.
- Un bloc-notes.

APPENDICE B

LES ÉTAPES À SUIVRE POUR L'ACQUISITION DES PREUVES

Afin que les preuves collectées soient d'une qualité acceptables, l'investigateur doit suivre les étapes suivantes (National Institute of Justice, 2004).

1. Sécuriser les preuves numériques conformément aux directives du département responsable de l'enquête. En absence de telles directives, on peut suivre le guide relatif au plan de réponse initial (*A guide for first responders*) publié par l'institut national de la justice des États-Unis (*National Institute of Justice*, 2001).

2. Documenter la configuration matérielle et logicielle du système utilisé par l'examinateur.

3. S'assurer du bon fonctionnement du système de l'examinateur (matériel et logiciel).

4. Désassembler l'ordinateur à examiner pour pouvoir accéder aux médias de stockage. Il faut faire attention à l'électricité électrostatique et aux champs magnétiques.

5. Identifier les médias de stockage pour faire l'acquisition. Ces médias peuvent être internes et/ou externes.

6. Documenter les médias internes de stockage et la configuration matérielle :

    a. État des lecteurs (fabrication, modèle, dimensions, taille, état des cavaliers, endroit, interface).

b. Les composantes internes (carte son, carte graphique, carte réseau, adresses MAC[26], carte PCMCIA, etc.)

7. Déconnecter les médias de stockage pour éviter la destruction ou l'altération des données.

8. Rechercher la configuration du système suspect en utilisant des démarrages contrôlés (*controlled boots*) :

    a. Faire un démarrage contrôlé pour capturer les informations de configuration matérielle (par ex., CMOS[27]/BIOS[28]) et les tests de fonctionnement.

        i. Modifier la séquence de démarrage pour démarrer dans un environnement d'investigation à partir d'un support amovible (par exemple, à l'aide d'un lecteur de disquette ou d'un lecteur CD).

        ii. Noter la date et l'heure.

        iii. Désactiver le mot de passe.

    b. Faire un second démarrage contrôlé pour tester les fonctionnalités de l'ordinateur et tester l'environnement de démarrage forensique.

        i. S'assurer que le lecteur de disquette et le lecteur CD sont bien connectés à l'alimentation électrique et aux câbles de données et s'assurer aussi que les médias de stockage de données sont encore déconnectés de l'alimentation électriques et des câbles de données.

---

[26] Adresses MAC (*Media Access Control*) : un numéro unique intégré (gravé) par le constructeur dans l'interface de la carte réseau.
[27] CMOS : Un type de *chipset* qui sert à stocker les informations de configuration du BIOS.
[28] BIOS (*Basic Input Output System*) : Ensemble de routines stockées dans le CMOS qui permettent à l'ordinateur de démarrer le système d'exploitation et faire les tests de fonctionnement des autres périphériques.

ii. Placer le disque de démarrage forensique dans le lecteur de disquette ou dans le lecteur CD, démarrer l'ordinateur et s'assurer que l'ordinateur démarre à partir du disque de démarrage forensique.

c. Reconnecter les médias de stockage et faire un troisième démarrage contrôlé pour capturer la configuration des lecteurs à partir du CMOS/BIOS

i. S'assurer que le disque de démarrage forensique est inséré dans le lecteur de disquette ou dans le lecteur CD afin d'éviter que l'ordinateur ne démarre accidentellement à partir des médias de stockage.

9. Éteindre l'ordinateur.

10. Quand c'est possible, déconnecter les supports de stockage et les reconnecter au système d'examen[29]. Il faut les configurer pour que le système d'examen puisse les reconnaître et ensuite procéder à l'acquisition des données.

11. Dans certains cas, il est recommandé de ne pas déconnecter ces supports de stockage. Voici quelques exemples :

a. Systèmes à disques redondants (RAID : *redundant array of independent disks*). Déconnecter ce genre de disques et faire l'acquisition séparément peut ne pas donner de meilleurs résultats car les données peuvent être stockées ou dupliquées sur différents disques et paraissent pour le système d'exploitation comme si elles sont stockées sur un seul disque. Ces disques sont utilisés pour augmenter la tolérance aux fautes.

b. Ordinateurs portables : pour ce genre de systèmes, il est parfois difficile d'accéder au disque. De plus, il peut être inutilisable une fois déconnecté du système.

---

[29] Le système que l'investigateur utilise pour collecter et analyser les preuves.

c. Équipements désuets (*legacy equipment*) : les anciens disques ou lecteurs peuvent ne pas être lus par les nouveaux systèmes.

d. Disponibilité des équipements : l'examinateur peut ne pas avoir la possibilité d'accéder aux disques.

e. Stockage réseau : il peut être nécessaire d'utiliser des équipement réseaux pour acquérir les données.

12. Quand l'examinateur veut utiliser le système suspect pour acquérir les données, il doit connecter à ce système les supports de stockage sur lesquels il veut stocker les preuves (disque dur, lecteur de bandes, graveur CD-RW, etc.).

13. Il faut s'assurer que les supports utilisés pour recevoir les preuves soient stérilisés (*forensically clean*[30]).

14. Autant que possible, il faut protéger les supports de stockage de l'ordinateur suspect contre l'écriture pour préserver l'authenticité des preuves.

15. Afin de garantir l'authenticité des données, l'examinateur doit, avant de commencer l'acquisition des preuves, calculer une valeur de protection d'intégrité (CRC, hachage) relative aux données en question.

16. Si la protection utilisée contre l'écriture est matérielle, on doit :

    a. Installer un lecteur protégé contre l'écriture.

    b. Démarrer le système avec le système d'exploitation contrôlé de l'examinateur.

17. Si la protection utilisée contre l'écriture est logicielle, on doit :

---

[30] Support numérique nettoyé de toute information inutile et de toute donnée résiduelle. Scanné contre les virus et vérifié avant d'être utilisé.

a. Démarrer le système avec le système d'exploitation contrôlé de l'examinateur.

b. Activer la protection contre l'écriture.

18. Vérifier la géométrie de tous les supports de stockage pour s'assurer que tous les espaces sont pris en compte, y compris les espaces protégés; en d'autres termes, il faut s'assurer que l'espace physique corresponde au total des espaces de toutes les partitions présentes.

19. Commencer l'acquisition des preuves en utilisant l'équipement et le logiciel approprié.

20. Vérifier si l'opération d'acquisition s'est bien déroulée en comparant la valeur calculée avant l'acquisition (CRC, hachage) avec celle calculée après l'acquisition ou faire tout simplement une comparaison secteur par secteur.

APPENDICE C

RÉDACTION DU RAPPORT D'INVESTIGATION

Tout au long de son enquête, l'investigateur prend des notes qui vont l'aider lors de la rédaction de son rapport.

Prise de notes

Parmi les notes que l'investigateur doit tenir, on trouve :

- Prendre notes de tous les entretiens.
- Garder une copie de l'autorisation d'enquête avec les notes du cas.
- Garder la requête initiale d'assistance avec les notes du cas.
- Garder une copie relative à la chaîne de continuité des preuves.
- Prendre des notes assez détaillées pour permettre une duplication complète des actions.
- Inclure dans les notes les dates, les horaires, les descriptions et les résultats des actions faites durant l'investigation.
- Documenter toutes les problèmes rencontrés durant l'investigation, ainsi que toutes les actions entreprises contre ces irrégularités.

- Inclure dans le rapport d'autres informations telles que la topologie du réseau s'il existe, la liste des utilisateurs autorisés, la coopération des utilisateurs, les mots de passes, etc.

- Documenter les modifications apportées au système ou au réseau par l'investigateur ou par les autorités légales.

- Documenter le système d'exploitation ainsi que tous les logiciels et les mises à jour (*patches*) installés.

- Documenter les informations obtenues sur le site mais concernant les stockages distants, les accès des utilisateurs distants et les sauvegardes à l'écart (*offsite*).

- Si des informations relatives à des preuves qui sont au-delà du contexte de l'enquête ont été trouvées, elles doivent être documenté et mentionnées, car elles peuvent faire l'objet d'une autorisation additionnelle d'enquête.

Rédaction du rapport

Le rapport de l'examinateur doit inclure les éléments suivants :

- Identité de l'agence faisant le rapport.

- Identificateur du cas ou le numéro de la présentation.

- Cas d'investigation.

- Identité du rapporteur.

- Date de réception.

- Date du rapport.

- Descriptions des items soumis à l'investigation (numéro de série, fabrication, modèle).

- Identité et signature de l'examinateur.
- Description des étapes suivies durant l'examen des preuves : recherches de chaînes de caractères, recherches d'images, récupérations des fichiers supprimés.
- Les résultas trouvés et la conclusion.

Aussi, le rapport doit contenir les rubriques suivantes :

- Sommaire des résultats trouvés : cette section est consacrée à une brève description des résultas de l'examen mené sur les items soumis à l'analyse. Tous les éléments cités ici doivent être détaillés dans la section suivante.
- Les détails des résultats trouvés : cette section décrit d'une façon très détaillée tous les résultats de l'examen :
    - Les fichiers concernant l'enquête.
    - Les autres fichiers, y compris les fichiers supprimés, qui appuient l'enquête.
    - Les chaînes de caractères, les mots clés et les textes recherchés.
    - Les preuves relatives à l'Internet telles que l'analyse du trafic web, les fichiers *log* du clavardage, les fichiers du cache, les courriels.
    - Analyses d'images.
    - Indications sur la possession et la propriété.
    - Analyses des données.
    - Techniques utilisées pour dissimuler les données telles que le chiffrage, la stéganographie, attributs de fichiers cachées, attributs des partitions cachées et les anomalies dans les noms de fichiers.

- Matériel à l'appui : listes du matériel pour appuyer la validité de l'enquête tel que : impression de quelques preuves spécifiques, copies numériques des preuves, documentation de la chaîne de continuité des preuves.

- Glossaire : pour aider le lecteur du rapport à comprendre les termes techniques.

APPENDICE D

LA MATRICE DE SUBSTITUTION "BLOSUM62"

Le contenu de la matrice de substitution « Blosum62 » utilisée en bioinformatique.

	A	C	D	E	F	G	H	I	K	L	M	N	P	Q	R	S	T	V	W	Y
A	4	0	-2	-1	-2	0	-2	-1	-1	-1	-1	-2	-1	-1	-1	1	0	0	-3	-2
C		9	-3	-4	-2	-3	-3	-1	-3	-1	-1	-3	-3	-3	-3	-1	-1	-1	-2	-2
D			6	2	3	-1	-1	-3	-1	-4	-3	1	-1	0	-2	0	-1	-3	-4	-3
E				5	-3	-3	0	-3	1	-3	-2	0	-1	2	0	0	-1	-2	-3	-2
F					6	-3	-1	0	-3	0	0	-3	-4	-3	-3	-2	-2	-1	1	3
G						6	-2	-4	-2	-4	-3	0	-2	-2	-2	0	-2	-3	-2	-3
H							8	-3	-1	-3	-2	1	-2	0	0	-1	-2	-3	-2	2
I								4	-3	2	1	-3	-3	-3	-3	-2	-1	3	-3	-1
K									5	-2	-1	0	-1	1	2	0	-1	-2	-3	-2
L										4	2	-3	-3	-2	-2	-2	-1	1	-2	-1
M											5	-2	-2	0	-1	-1	-1	1	-1	-1
N												6	-2	0	0	1	0	-3	-4	-2
P													7	-1	-2	-1	-1	-2	-4	-3
Q														5	1	0	-1	-2	-2	-1
R															5	-1	-1	-3	-3	-2
S																4	1	-2	-3	-2
T																	5	0	-2	-2
V																		4	-3	-1
W																			11	2
Y																				7

APPENDICE E

CONTENU DES FICHIERS DE VALIDATION

Dans cet appendice nous présentons le contenu des fichiers qui ont été utilisés lors de la validation de notre outil. Au début nous avons préparé un fichier texte que nous avons nommé *original.txt* et nous avons généré un ensemble de fichiers textes en apportant différentes modifications au fichier original. Ces fichiers sont nommés *copy1.txt, copy2.txt, ..., copy17.txt*. Les modifications sont identifiables comme suit : les ajouts sont en rouge et les suppressions en bleu.

Nous avons ajouté d'autres fichiers à cet ensemble. Ces fichiers qui n'ont aucune relation avec le fichier original sont nommés *other1.txt, other2.txt, ..., other5.txt*.

Original.txt
D'après mes recherches, ce n'est qu'à partir de 2001 que le terme «antiforensic» est véritablement introduit dans la littérature du domaine. Sous ce terme, sont désignées les procédures et techniques ayant pour objectif de limiter les moyens d'enquête ou d'examen d'un système. Plusieurs moyens seront donc utilisés à cette fin : détruire, camoufler, modifier des traces, prévenir la création de traces. Ce domaine particulier n'a jamais été examiné de façon systématique, le présent article propose un début d'exploration au travers d'une analyse des publications abordant le thème, d'un examen

contradictoire du processus postincident du point de vue de l'attaquant.
Tout complément d'information (articles ou publications non mentionnées, moyens non abordés, etc.) sera le bienvenue afin d'améliorer la pertinence ou la complétude des analyses dans l'objectif d'une publication ultérieure plus complète.

Copy1.txt : une phrase insérée à la fin du document

D'après mes recherches, ce n'est qu'à partir de 2001 que le terme «antiforensic» est véritablement introduit dans la littérature du domaine. Sous ce terme, sont désignées les procédures et techniques ayant pour objectif de limiter les moyens d'enquête ou d'examen d'un système. Plusieurs moyens seront donc utilisés à cette fin : détruire, camoufler, modifier des traces, prévenir la création de traces.
Ce domaine particulier n'a jamais été examiné de façon systématique, le présent article propose un début d'exploration au travers d'une analyse des publications abordant le thème, d'un examen contradictoire du processus postincident du point de vue de l'attaquant.
Tout complément d'information (articles ou publications non mentionnées, moyens non abordés, etc.) sera le bienvenue afin d'améliorer la pertinence ou la complétude des analyses dans l'objectif d'une publication ultérieure plus complète. Une phrase insérée a la fin du document pour faire le test.

Copy2.txt : une phrase insérée au début du document

Une phrase insérée au début du document pour faire le test. D'après mes recherches, ce n'est qu'à partir de 2001 que le terme «antiforensic» est véritablement introduit dans la littérature du domaine. Sous ce terme, sont désignées les procédures et techniques ayant pour objectif de limiter les moyens d'enquête ou d'examen d'un

système. Plusieurs moyens seront donc utilisés à cette fin : détruire, camoufler, modifier des traces, prévenir la création de traces.
Ce domaine particulier n'a jamais été examiné de façon systématique, le présent article propose un début d'exploration au travers d'une analyse des publications abordant le thème, d'un examen contradictoire du processus postincident du point de vue de l'attaquant.
Tout complément d'information (articles ou publications non mentionnées, moyens non abordés, etc.) sera le bienvenue afin d'améliorer la pertinence ou la complétude des analyses dans l'objectif d'une publication ultérieure plus complète.

Copy3.txt : une phrase insérée au milieu du document

D'après mes recherches, ce n'est qu'à partir de 2001 que le terme «antiforensic» est véritablement introduit dans la littérature du domaine. Sous ce terme, sont désignées les procédures et techniques ayant pour objectif de limiter les moyens d'enquête ou d'examen d'un système. Plusieurs moyens seront donc utilisés à cette fin : détruire, camoufler, modifier des traces, prévenir la création de traces.
Une phrase insérée au milieu du document pour faire le test. Ce domaine particulier n'a jamais été examiné de façon systématique, le présent article propose un début d'exploration au travers d'une analyse des publications abordant le thème, d'un examen contradictoire du processus postincident du point de vue de l'attaquant.
Tout complément d'information (articles ou publications non mentionnées, moyens non abordés, etc.) sera le bienvenue afin d'améliorer la pertinence ou la complétude des analyses dans l'objectif d'une publication ultérieure plus complète.

Copy4.txt : trois phrases insérées au début, au milieu et à la fin du document

Une phrase insérée au début du document pour faire le test. D'après mes recherches, ce n'est qu'à partir de 2001 que le terme «antiforensic» est véritablement introduit dans la littérature du domaine. Sous ce terme, sont désignées les procédures et techniques ayant pour objectif de limiter les moyens d'enquête ou d'examen d'un système. Plusieurs moyens seront donc utilisés à cette fin : détruire, camoufler, modifier des traces, prévenir la création de traces.

Une phrase insérée au milieu du document pour faire le test. Ce domaine particulier n'a jamais été examiné de façon systématique, le présent article propose un début d'exploration au travers d'une analyse des publications abordant le thème, d'un examen contradictoire du processus postincident du point de vue de l'attaquant.

Tout complément d'information (articles ou publications non mentionnées, moyens non abordés, etc.) sera le bienvenue afin d'améliorer la pertinence ou la complétude des analyses dans l'objectif d'une publication ultérieure plus complète. Une phrase insérée a la fin du document pour faire le test.

Copy5.txt : un morceau de texte important inséré à la fin du document

D'après mes recherches, ce n'est qu'à partir de 2001 que le terme «antiforensic» est véritablement introduit dans la littérature du domaine. Sous ce terme, sont désignées les procédures et techniques ayant pour objectif de limiter les moyens d'enquête ou d'examen d'un système. Plusieurs moyens seront donc utilisés à cette fin : détruire, camoufler, modifier des traces, prévenir la création de traces.

Ce domaine particulier n'a jamais été examiné de façon systématique, le présent article propose un début d'exploration au travers d'une analyse des publications abordant le thème, d'un examen

contradictoire du processus postincident du point de vue de l'attaquant.
Tout complément d'information (articles ou publications non mentionnées, moyens non abordés, etc.) sera le bienvenue afin d'améliorer la pertinence ou la complétude des analyses dans l'objectif d'une publication ultérieure plus complète. La première publication à ma connaissance qui traite du sujet est publiée en Juillet 1999 [1] par un professeur de l'université de Georgetown (D.Denning) et un ancien du Bureau Fédéral d'Investigation (FBI - W.Baugh). Elle met en exergue l'utilisation du chiffrement, de l'anonymat et d'autres techniques pour masquer des activités criminelles : les mots de passe, la compression, la stéganographie, le stockage à distance des informations, l'inactivation ou la suppression des enregistrements d'audit.

Copy6.txt : un morceau de texte important inséré au début du document

La première publication à ma connaissance qui traite du sujet est publiée en Juillet 1999 [1] par un professeur de l'université de Georgetown (D.Denning) et un ancien du Bureau Fédéral d'Investigation (FBI - W.Baugh). Elle met en exergue l'utilisation du chiffrement, de l'anonymat et d'autres techniques pour masquer des activités criminelles : les mots de passe, la compression, la stéganographie, le stockage à distance des informations, l'inactivation ou la suppression des enregistrements d'audit.
D'après mes recherches, ce n'est qu'à partir de 2001 que le terme «antiforensic» est véritablement introduit dans la littérature du domaine. Sous ce terme, sont désignées les procédures et techniques ayant pour objectif de limiter les moyens d'enquête ou d'examen d'un système. Plusieurs moyens seront donc utilisés à cette fin : détruire, camoufler, modifier des traces, prévenir la création de traces.
Ce domaine particulier n'a jamais été examiné de façon systématique, le présent article propose un début d'exploration au travers d'une analyse des publications abordant le thème, d'un examen

contradictoire du processus postincident du point de vue de l'attaquant.
Tout complément d'information (articles ou publications non mentionnées, moyens non abordés, etc.) sera le bienvenue afin d'améliorer la pertinence ou la complétude des analyses dans l'objectif d'une publication ultérieure plus complète.

Copy7.txt : un morceau de texte important inséré au milieu du document

D'après mes recherches, ce n'est qu'à partir de 2001 que le terme «antiforensic» est véritablement introduit dans la littérature du domaine. Sous ce terme, sont désignées les procédures et techniques ayant pour objectif de limiter les moyens d'enquête ou d'examen d'un système. Plusieurs moyens seront donc utilisés à cette fin : détruire, camoufler, modifier des traces, prévenir la création de traces.
La première publication à ma connaissance qui traite du sujet est publiée en Juillet 1999 [1] par un professeur de l'université de Georgetown (D.Denning) et un ancien du Bureau Fédéral d'Investigation (FBI - W.Baugh). Elle met en exergue l'utilisation du chiffrement, de l'anonymat et d'autres techniques pour masquer des activités criminelles : les mots de passe, la compression, la stéganographie, le stockage à distance des informations, l'inactivation ou la suppression des enregistrements d'audit. Ce domaine particulier n'a jamais été examiné de façon systématique, le présent article propose un début d'exploration au travers d'une analyse des publications abordant le thème, d'un examen contradictoire du processus postincident du point de vue de l'attaquant.
Tout complément d'information (articles ou publications non mentionnées, moyens non abordés, etc.) sera le bienvenue afin d'améliorer la pertinence ou la complétude des analyses dans l'objectif d'une publication ultérieure plus complète.

Copy8.txt : des morceaux de texte insérés un peu partout dans le document

D'après mes recherches, ce n'est qu'à partir de 2001 que le terme «antiforensic» est véritablement introduit dans la littérature du domaine. Sous ce terme, sont désignées les procédures et techniques ayant pour objectif de limiter les moyens d'enquête ou d'examen d'un système. Dans son étude du processus d'attaque, [23] l'auteur du site trapkit.de, prend en compte la phase de furtivité basée sur les techniques qui permettent à l'attaquant d'être intraçable pendant le déroulement de l'exploit et dans la phase post incident.

Plusieurs moyens seront donc utilisés à cette fin : détruire, camoufler, modifier des traces, prévenir la création de traces.

La première publication à ma connaissance qui traite du sujet est publiée en Juillet 1999 [1] par un professeur de l'université de Georgetown (D.Denning) et un ancien du Bureau Fédéral d'Investigation (FBI - W.Baugh). Ce domaine particulier n'a jamais été examiné de façon systématique, le présent article propose un début d'exploration au travers d'une analyse des publications abordant le thème, d'un examen contradictoire du processus postincident du point de vue de l'attaquant. Elle met en exergue l'utilisation du chiffrement, de l'anonymat et d'autres techniques pour masquer des activités criminelles : les mots de passe, la compression, la stéganographie, le stockage à distance des informations, l'inactivation ou la suppression des enregistrements d'audit.

Tout complément d'information (articles ou publications non mentionnées, moyens non abordés, etc.) sera le bienvenue afin d'améliorer la pertinence ou la complétude des analyses dans l'objectif d'une publication ultérieure plus complète.

Copy9.txt : un paragraphe supprimé au début document

D'après mes recherches, ce n'est qu'à partir de 2001 que le terme «antiforensic» est véritablement introduit dans la littérature du

domaine. Sous ce terme, sont désignées les procédures et techniques ayant pour objectif de limiter les moyens d'enquête ou d'examen d'un système. Plusieurs moyens seront donc utilisés à cette fin : détruire, camoufler, modifier des traces, prévenir la création de traces.

Ce domaine particulier n'a jamais été examiné de façon systématique, le présent article propose un début d'exploration au travers d'une analyse des publications abordant le thème, d'un examen contradictoire du processus postincident du point de vue de l'attaquant.

Tout complément d'information (articles ou publications non mentionnées, moyens non abordés, etc.) sera le bienvenue afin d'améliorer la pertinence ou la complétude des analyses dans l'objectif d'une publication ultérieure plus complète.

Copy10.txt : un morceau de texte important supprimé à la fin du document

D'après mes recherches, ce n'est qu'à partir de 2001 que le terme «antiforensic» est véritablement introduit dans la littérature du domaine. Sous ce terme, sont désignées les procédures et techniques ayant pour objectif de limiter les moyens d'enquête ou d'examen d'un système. Plusieurs moyens seront donc utilisés à cette fin : détruire, camoufler, modifier des traces, prévenir la création de traces.

Ce domaine particulier n'a jamais été examiné de façon systématique, le présent article propose un début d'exploration au travers d'une analyse des publications abordant le thème, d'un examen contradictoire du processus postincident du point de vue de l'attaquant.

Tout complément d'information (articles ou publications non mentionnées, moyens non abordés, etc.) sera le bienvenue afin d'améliorer la pertinence ou la complétude des analyses dans l'objectif d'une publication ultérieure plus complète.

Copy11.txt : un paragraphe supprimé au milieu du document

D'après mes recherches, ce n'est qu'à partir de 2001 que le terme «antiforensic» est véritablement introduit dans la littérature du domaine. Sous ce terme, sont désignées les procédures et techniques ayant pour objectif de limiter les moyens d'enquête ou d'examen d'un système. Plusieurs moyens seront donc utilisés à cette fin : détruire, camoufler, modifier des traces, prévenir la création de traces.

Ce domaine particulier n'a jamais été examiné de façon systématique, le présent article propose un début d'exploration au travers d'une analyse des publications abordant le thème, d'un examen contradictoire du processus postincident du point de vue de l'attaquant.

Tout complément d'information (articles ou publications non mentionnées, moyens non abordés, etc.) sera le bienvenue afin d'améliorer la pertinence ou la complétude des analyses dans l'objectif d'une publication ultérieure plus complète.

Copy12.txt : quelques phrases supprimées à différents endroits dans le document

D'après mes recherches, ce n'est qu'à partir de 2001 que le terme «antiforensic» est véritablement introduit dans la littérature du domaine. Sous ce terme, sont désignées les procédures et techniques ayant pour objectif de limiter les moyens d'enquête ou d'examen d'un système. Plusieurs moyens seront donc utilisés à cette fin : détruire, camoufler, modifier des traces, prévenir la création de traces.

Ce domaine particulier n'a jamais été examiné de façon systématique, le présent article propose un début d'exploration au travers d'une analyse des publications abordant le thème, d'un examen contradictoire du processus postincident du point de vue de l'attaquant.

Tout complément d'information (articles ou publications non mentionnées, moyens non abordés, etc.) sera le bienvenue afin

d'améliorer la pertinence ou la complétude des analyses dans l'objectif d'une publication ultérieure plus complète.

Copy13.txt : une bonne partie supprimée en début du document

D'après mes recherches, ce n'est qu'à partir de 2001 que le terme «antiforensic» est véritablement introduit dans la littérature du domaine. Sous ce terme, sont désignées les procédures et techniques ayant pour objectif de limiter les moyens d'enquête ou d'examen d'un système. Plusieurs moyens seront donc utilisés à cette fin : détruire, camoufler, modifier des traces, prévenir la création de traces.

Ce domaine particulier n'a jamais été examiné de façon systématique, le présent article propose un début d'exploration au travers d'une analyse des publications abordant le thème, d'un examen contradictoire du processus postincident du point de vue de l'attaquant.

Tout complément d'information (articles ou publications non mentionnées, moyens non abordés, etc.) sera le bienvenue afin d'améliorer la pertinence ou la complétude des analyses dans l'objectif d'une publication ultérieure plus complète.

Copy14.txt : une bonne partie supprimée en fin du document

D'après mes recherches, ce n'est qu'à partir de 2001 que le terme «antiforensic» est véritablement introduit dans la littérature du domaine. Sous ce terme, sont désignées les procédures et techniques ayant pour objectif de limiter les moyens d'enquête ou d'examen d'un système. Plusieurs moyens seront donc utilisés à cette fin : détruire, camoufler, modifier des traces, prévenir la création de traces.

Ce domaine particulier n'a jamais été examiné de façon systématique, le présent article propose un début d'exploration au travers d'une analyse des publications abordant le thème, d'un examen

contradictoire du processus postincident du point de vue de l'attaquant.
Tout complément d'information (articles ou publications non mentionnées, moyens non abordés, etc.) sera le bienvenue afin d'améliorer la pertinence ou la complétude des analyses dans l'objectif d'une publication ultérieure plus complète.

Copy15.txt : une bonne partie supprimée au milieu du document
D'après mes recherches, ce n'est qu'à partir de 2001 que le terme «antiforensic» est véritablement introduit dans la littérature du domaine. Sous ce terme, sont désignées les procédures et techniques ayant pour objectif de limiter les moyens d'enquête ou d'examen d'un système. Plusieurs moyens seront donc utilisés à cette fin : détruire, camoufler, modifier des traces, prévenir la création de traces. Ce domaine particulier n'a jamais été examiné de façon systématique, le présent article propose un début d'exploration au travers d'une analyse des publications abordant le thème, d'un examen contradictoire du processus postincident du point de vue de l'attaquant. Tout complément d'information (articles ou publications non mentionnées, moyens non abordés, etc.) sera le bienvenue afin d'améliorer la pertinence ou la complétude des analyses dans l'objectif d'une publication ultérieure plus complète.

Copy16.txt : des insertions et des suppression un peu partout dans le document
D'après mes recherches, ce n'est UQAM qu'à partir de 2001 que le terme «antiforensic» est véritablement introduit dans la littérature du domaine. Sous ce terme, sont désignées les procédures et techniques ayant pour objectif de limiter les moyens d'enquête ou d'examen Habibd'un système. Plusieurs moyens Département

d'informatique seront donc utilisés à cette fin : détruire, camoufler, modifier des traces, prévenir la création de traces.
Ce domaine particulier n'a jamais été examiné de façon systématique, le présent article propose un Ville de Montréal début d'exploration au travers Louafid'une analyse des publications abordant le thème, d'un examen contradictoire du processus postincident du point de vue de l'attaquant.
Tout complément d'information (articles ou publications non mentionnées, moyens non Sécurité informatique abordés, etc.) sera le bienvenue afin d'améliorer la pertinence ou la complétude des analyses dans l'objectif d'une publication ultérieure plus complète.

Other1.txt :

Les origines de la philosophie grecque, et donc de la pensée occidentale tout entière, sont mystérieuses. Selon la tradition érudite, la philosophie naît avec Thalès et Anaximandre: au dix-huitième siècle, on a recherché ses origines les plus lointaines dans des contacts légendaires avec les cultures orientales, avec la pensée égyptienne et la pensée indienne. Rien n'a pu être attesté dans cette direction, et l'on s'est contenté d'établir des analogies et des parallèles. En réalité, l'époque qui correspond aux origines de la philosophie grecque est bien plus proche de nous. Platon appelle «philosophie», amour de la sagesse, sa propre recherche, sa propre activité éducative, liée à une expression écrite, à la forme littéraire du dialogue. Et Platon se tourne avec vénération vers le passé, vers ce monde où les «sages» avaient véritablement vécu. D'autre part, la philosophie postérieure, notre philosophie, n'est autre qu'une continuation, qu'un développement de la forme littéraire introduite par Platon; et pourtant cette dernière surgit comme un phénomène de décadence, puisque «l'amour de la sagesse» est en deçà de la «sagesse». En effet l'amour de la sagesse ne signifiait pas pour Platon une aspiration à quelque chose qui

n'avait jamais été atteint, mais plutôt une tendance à récupérer ce qui déjà avait été réalisé et vécu.

Other2.txt:

Identifiable object-oriented modeling languages began to appear between mid-1970 and the late 1980s as various methodologists experimented with different approaches to object-oriented analysis and design. The number of identified modeling languages increased from less than 10 to more than 50 during the period between 1989-1994. Many users of OO methods had trouble finding complete satisfaction in any one modeling language, fueling the "method wars." By the mid-1990s, new iterations of these methods began to appear and these methods began to incorporate each other's techniques, and a few clearly prominent methods emerged.

Other3.txt :

Quand on considère un système (pris au sens large du terme), la logique permet d'exprimer pour un état donné du système, les propriétés de cet état. Par exemple, on peut donner une formule de la logique usuelle des entiers qui exprime qu'une variable contient un nombre premier. Cependant cette propriété peut être vérifiée pour un état, mais peut être fausse pour un autre état. Une propriété n'est, en général, pas toujours vérifiée. D'autre part, il peut être intéressant d'exprimer des relations sur les propriétés de différents états.

Other4.txt:

In cryptography, RSA is an algorithm for public-key encryption. It was the first algorithm known to be suitable for signing as well as encryption, and one of the first great advances in public key

```
cryptography. RSA is still widely used in electronic commerce
protocols, and is believed to be secure given sufficiently long
keys.
```

Other5.txt : fichier vide.

BIBLIOGRAPHIE

Anderson, Michael R. S.d. *Computer evidence processing*. En ligne <http://www.forensics-intl.com/art10.html>. Consulté le 05 février 2005.

_____ S.d. *Electronic fingerprints*. En ligne. <http://www.forensics-intl.com/art2.html>. Consulté le 5 février 2005.

Bégin, Guy et Habib Louafi. 2006. *FCompare: A forensic tool for comparing partial copies of text files*. En preparation. The 6th Annual Digital Forensic Research Workshop (DFRWS 2006).

Carrier, Brian. 2001. *TCTUTILs*. En ligne. <http://www.digital-evidence.org/tools/index.html>. Consulté le 10 mai 2005.

_____ 2002 *Mac-Robber*. En ligne. <http://www.sleuthkit.org/mac-robber/download.php>. Consulté le 10 mai 2005.

_____ 2003. *Autopsy Forensic Browser (AFB)*. En ligne. <http://www.sleuthkit.org/autopsy>. Consulté le 10 mai 2005.

_____ 2003. *The Sleuth Kit (TSK)*. En ligne. <http://www.sleuthkit.org/sleuthkit>. Consulté le 10 mai 2005.

Casey, Eoghan. 2001. *Handbook of Computer Crime Investigation: Forensic Tools and Technology*. New York: Academic Press, 448 p.

_____ 2004. *Digital Evidence and Computer Crime: Forensic Science, Computers and the Internet*. 2e Ed. New York: Academic Press, 690 p.

Computer Crime and Intellectual Property Section. 2002. Voir U.S. Department of Justice, Criminal division, Computer Crime and Intellectual Property Section. 2002. *Searching and Seizing Computers and Obtaining Electronic Evidence in Criminal Investigations*. En ligne <http://www.cybercrime.gov/s&smanual2002.htm>. Consulté le 5 février 2005

Coul, S., J. Branch, B. Szymanski, et E. Breimer. 2003. *Intrusion Detection: A bioinformatics Approach*. 19th Annual Computer Security Applications Conference (ACSAC '03), p. 24-34.

Eckstein, Knut. 2002. *TCT and TCTUTILs for HP-UX.* En ligne. <http://www.xs4all.nl/~eknut/tct-hp.html>. Consulté le 10 mai 2005.

Farmer, Dan et Wietse Venema. 1999. *The Coroner's ToolKit (TCT).* En ligne. <http://www.porcupine.org/forensics/tct.html >. Consulté le 10 mai 2005.

Giegerich, Robert et Wheeler David. 1996. *Pairwise Sequence Alignment.* En ligne. <http://www.techfak.uni-bielefeld.de/bcd/Curric/PrwAli/prwali.html>. Consulté le 15 février 2006.

Goad, W. et M. Kanehisa. 1982. *Pattern recognition in nucleic acid sequences: a general method for finding local homologies and symmetries.* Nucleic Acids Research, no 10, p. 247-263.

HashKeeper. S.d. *Users of the HashKeeper Forensic utility.* En ligne. <http://groups.Yahoo.com/group/hashkeeper/ >. Consulté le 18 mars 2006.

InfoMètre. 2004. *Enquête sur la sécurité de l'information.* En ligne. <http://www.infometre.cefrio.qc.ca/loupe/enquetes/securite.asp>. Consulté le 5 février 2005.

Jaspard, Emanuel. 2006. *Algorithmes et programmes de comparaison de séquences.* En ligne. <http://ead.univ-angers.fr/~jaspard/Page2/BIOINFORMATIQUE/7ModuleBioInfoJMGE/9AlgoProgramme/1ProgAlgo.htm. >. Consulté le 15 février 2006.

Mandia, Kevin et Chris Prosise. 2001. *Incident Response: Investigating Computer Crime.* California: Osborne/McGraw-Hill p.16-17

Marcella, A. et R. Greenfield. 2002. *Cyber Forensics: A Field Manual for Collecting, Examining, and Preserving Evidence of Computer Crimes.* New York: Auerbach, 443 p.

Matthew, Brad. 2001. *Collecting Electronic Evidence after a System Compromise.* En ligne. < http://www.auscert.org.au/render.html?it=2247 >. Consulté le 5 février 2005.

Mazouzi, Abdallah. 2005. « Développement de modules bioinformatiques pour la reconstruction d'arbres phylogéniques et de réticulogrammes ». Mémoire de maîtrise, Montréal, Université du Québec à Montréal, 129 p.

Myers, E. W. 1986. *An O(ND) Difference Algorithm and its Variations.* Algorithmica, no 1, p. 251-266.

Miller, W. et E. W. Myers. 1985. *A File Comparison Program.* Software-Practice & Experience, no 15, p. 1025-1040.

Miller, W. et E. W. Myers. 1988. *Optimal Alignments In Lineare Space.* Compu Appl Biosci., no 4(1) 15, p. 1025-1040.

National Institute of Justice. 2001. Voir U.S. Department of justice, Office of Justice Programs, National Institute of Justice. 2001. *Electronic Crime Scene Investigation: A Guide for First Responders.* En ligne. <http://www.ncjrs.gov/pdffiles1/nij/187736.pdf.>. Consulté le 18 juillet 2005.

National Institute of Justice. 2004. Voir U.S. Department of Justice, Office of Justice Programs, National Institute of Justice. 2004. *Forensic Examination of Digital Evidence: A Guide for Law Enforcement.* En ligne. <http://www.ncjrs.gov/pdffiles1/nij/199408.pdf >. Consulté le 10 mai 2005.

National Institute of Standards and Technology (NIST). S.d. *National Software Reference Library (NSRL).* En ligne. <http://www.nsrl.nist.gov/>. Consulté le 18 juillet 2005

Needleman, S. et C. Wunch. 1970. *A general Method Applicable to the Search for Similarities in the Amino Acid Sequences Of Two Proteins.* Journal of Molecular Biology, no 48, p. 443-453.

Radcliff, Deborah. 1989. *Crime in the 21st century.* En ligne. <http://www.infoworld.com/cgi-bin/displayStory.pl?/features/981214crime.htm>. Consulté le 5 février 2005.

Schauer, Hervé. 2001. *Compte-rendu de la conférence du FIRST.* En ligne. <http://www.hsc.fr/ressources/veille/first2001.html.fr#forensique.>. Consulté le 15 février 2006.

Schultz, E. et R. Shumway. 2001. *Incident Response: A Strategic Guide to Handling System and Network Security Breaches.* Boston: New Riders Publishing, 384 p.

Shinder, D. et E. Tittel. 2002. *Scene of the CyberCrime: Computer Forensics Handbook.* Massachusetts: Syngress Publishing, 718 p.

Smith, T. F. et M.S. Waterman. 1981. *Identification of common molecular subsequences.* Journal of molecular Biology, no 147, p. 195-197.

Standards On-line Select. 2003. *Information security management: Specification for information security management systems.* En ligne. <http://www.standards.com.au/select/Script/Details.asp?docn=AS956629867290>. Consulté le 15 février 2006.

_____ 2005. *Information technology : Security techniques - Code of practice for information security management.* En ligne.

<http://www.standards.com.au/Select/Script/Details.asp?docn=ISOA00022_2305.>. Consulté le 15 février 2006.

Touzin, Hélene. 2003. *Alignement local, BLAST.* En ligne. <http://www.lifl.fr/~touzet/BI/Cours/local.pdf>. Consulté le 15 février 2006.

Venema, Wietse Zweitze. 1999. *Computer Forensic Analysis Class.* En ligne. <http://www.porcupine.org/forensics/handouts.html>. Consulté le 5 février 2005.

Wu, S., U. Manber, G. Myers et W. Miller. 1989. *An O(NP) sequence comparison algorithm.* Information Processing Letters, no 35, p. 317-323