Analyse de la mémoire vive d'un téléphone Android :
Échantillon de preuves présentes dans la RAM

TABLE DES MATIÈRES

Analyse de la mémoire vive d'un téléphone Android :

Échantillon de preuves présentes dans la RAM

Sommaire :

L'analyse forensique de la mémoire vive (RAM) des téléphones cellulaires est encore pour le moment une discipline naissante. Pourtant, la mémoire volatile d'un cellulaire contient de l'information pertinente pour un investigateur numérique. L'objectif de notre recherche est de démontrer l'existence de preuves (photo, courriel, etc.) utiles pour un investigateur numérique dans la RAM d'un téléphone cellulaire Android qui a été utilisé en tant que téléphone intelligent (connexion à internet, etc.) ainsi que de donner un aperçu du type de preuves qui peuvent être trouvées.

1. Introduction :

L'utilisation du téléphone intelligent s'est démocratisée à vive allure dans la sphère professionnelle et privée . En décembre 2018, les téléphones portables auraient une part de marché de 47.89% contre 48.44% pour les ordinateurs de bureau [1]. Les téléphones intelligents sont donc devenus une mine d'informations pour les investigateurs numériques. Toutefois, alors que l'analyse de la mémoire non volatile d'un téléphone portable est plutôt bien documentée et éprouvée, l'analyse forensique de la mémoire vive (RAM) des téléphones cellulaires est encore pour le moment une discipline naissante.

Notre analyse portera sur les téléphones utilisant le système d'exploitation Android afin de couvrir la plus grande part d'utilisateurs. En effet, en décembre 2018, les téléphones Android représentaient plus de 75% du marché des téléphones intelligents [2]. L'objectif de notre recherche est de démontrer l'existence de preuves pertinentes (photo, courriel, etc.) pour un investigateur numérique dans la RAM d'un téléphone cellulaire Android qui a été utilisé en tant que téléphone intelligent (connexion à internet, etc.). Cette étude permettra également de faire ressortir les principaux défis liés à l'analyse forensique de la mémoire vive des téléphones intelligents.

2. Revue de la littérature :

Depuis 2007, le *National Institute of Standards and Technology* (NIST) publie des lignes directrices pour l'analyse forensique des téléphones portables [3]. Au fil des années, la communauté scientifique a démontré qu'il est entre autres possible d'extraire la liste de contacts, les SMS, les photos/vidéos, l'historique d'appel et de navigation Internet d'un téléphone cellulaire [4][5][6][7]. Ces informations sont notamment accessibles en analysant la carte SIM du téléphone [5][8][9]. Toutefois, la mémoire d'un cellulaire contient également de l'information pertinente pour un investigateur numérique. Cela justifie que l'analyse forensique des téléphones intelligents puisse être divisée en deux grandes catégories : l'analyse de la SIM et l'analyse de la mémoire [10].

Dans la même veine que le NIST, le *Scientific Working Group on Digital Evidence* (SWGDE) expose des bonnes pratiques pour l'investigation numérique des téléphones portables [11]. En plus des lignes directrices et des meilleures pratiques, certains auteurs ont défini une procédure pour l'analyse forensique des téléphones intelligents [12]. Par

exemple, le *SANS Institute* propose une procédure en neuf étapes [13] tandis que Ali et al. (2017) suggèrent un processus de métamodélisation en huit étapes pour l'investigation numérique des cellulaires [14]. De plus, une classification des différentes méthodes d'extraction de données a été élaborée par le NIST. Cette classification est reprise dans plusieurs articles [7][15][16] et dans des livres spécialisés [17].

Afin d'extraire les données et pour les analyser, les auteurs ont recensé une multitude d'outils forensiques. Ces outils sont principalement utilisés pour l'analyse de données statiques [18][19][20]. Toutefois, afin d'éviter de perdre des éléments de preuve, une analyse en direct (*live forensics*) est également nécessaire [21][22][23][24][25]. Au sujet de l'analyse de la mémoire vive d'un téléphone Android, trois gratuiciels semblent principalement être plébiscités par la communauté scientifique : LiME, Volatility et Autopsy [26][27][28][29][30][31][32]. De leur côté, Yang et al. (2016) suggèrent d'utiliser AMExtractor à la place de LiME [33][34].

LiME et Volatility semblent être devenus des outils incontournables pour l'extraction et l'analyse de la mémoire d'un cellulaire, à tel point que des auteurs dédient certains chapitres de leur livre à leur utilisation [35][36], tandis que des étudiants écrivent des thèses à leur sujet [37]. Toutefois, certains auteurs préfèrent utiliser leurs propres outils, surtout pour l'extraction des données afin de pallier certaines limites reprochées à LiME [24][38][39].

LiME et Volatility restent néanmoins deux outils dont l'utilisation pour l'extraction et l'analyse de la RAM d'un téléphone Android est relativement bien documentée [40][41][42]. Ces outils permettent notamment de pouvoir analyser le comportement d'un logiciel malveillant (*malware*) dans un cellulaire [43][44][45][46][47] ainsi que de retrouver des mots de passe d'applications [48] et des données relatives au stockage infonuagique [49].

3. Protocole expérimental et processus suivi :

La RAM est l'endroit où tout ce qu'un périphérique doit exécuter est chargé, c'est-à-dire le système d'exploitation, les applications utilisées et celles qui s'exécutent en arrière-plan. La RAM est l'endroit où le processeur reçoit directement les informations requises [50]. Avant de pouvoir analyser la RAM d'un cellulaire, il faut déjà l'avoir extraite. Le processus de notre recherche s'est donc déroulé en deux grandes étapes : 1. Extraction et 2. Analyse. Comme nous le verrons, chacune de ces étapes comporte son lot de défis à relever.

3.1. Protocole d'extraction

Afin d'avoir des échantillons de mémoire vive à analyser, nous avons considéré un total de huit alternatives pour l'extraction de la RAM. Nous avons opté pour l'utilisation de trois téléphones Android (Samsung GT-I9505 Galaxy S4, Huawei Y300-051, LG E410B) ainsi que de l'émulateur d'Android Studio. Pour chacune de ces options, nous avons considéré deux outils (LiME et AMExtractor) pour l'extraction de la mémoire vive. Normalement le résultat des extractions à l'aide de ces deux outils est similaire [33].

Tel qu'illustré à l'annexe 1, nous avons réussi à extraire de la RAM trois fois sur huit. En effet, autant LiME que AMExtractor ont certaines limites (voir Tableau 1). LiME ne peut être utilisé que si le téléphone supporte les LKM

(*Loadable Kernel Module*) et si l'investigateur dispose du code source du noyau du système d'exploitation (*kernel*) du téléphone cellulaire examiné et qu'il peut le compiler tandis que AMExtractor nécessite d'effectuer de l'ingénierie inverse (utilisation d'un désassembleur tel qu'IDA Pro [51], voir Annexe 2) et que le téléphone examiné dispose du répertoire */dev/kmem* [34].

Dans les deux cas de figure, que LiME ou AMExtractor soit utilisé, le téléphone doit disposer des droits d'administrateur (*root*) et être en mode développeur (utilisation d'Android Debug Bridge, adb [52], voir Annexe 3). L'ordinateur de l'investigateur numérique doit également contenir Android SDK [53] et Android NDK [54]. Afin de *rooter* les téléphones, nous avons utilisé Kingroot v5.3.7.20180619 [55].

Au sujet de l'option d'utiliser l'émulateur d'Android Studio (v.3.10 installée sous Kali Linux , v.3.18 sous Windows 10), celle-ci a été un échec, car l'émulateur ne dispose pas du fichier */dev/kmem* nécessaire pour utiliser AMExtractor *(*voir Annexes 3 et 4) et, car nous n'avons pas réussi à compiler le noyau utilisé (kernel *Goldfish* [56], voir Annexe 5) en raison d'erreurs constantes de dépendance (Kali Linux ne supporte plus *ncurses*). Dans la même veine que pour l'émulateur d'Android Studio, l'utilisation d'AMExtractor a été impossible pour l'appareil LG E410B car ce dernier ne dispose pas du fichier */dev/kmem.*

Du côté des appareils Samsung GT-I9505 Galaxy S4 et Huawei Y300-051, l'extraction de leur mémoire vive n'a pas pu être réalisée à l'aide de LiME. En ce qui concerne le Huawei Y300-051, nous n'avons pas réussi à trouver le coude source du noyau (*kernel*). Du côté du Samsung GT-I9505 Galaxy S4, bien que nous disposons du code source du noyau [57], il est impossible d'utiliser LiME car cet appareil ne supporte pas les LKM comme LiME.

LiME	**AMExtractor**
1. Rooter le cellulaire	1. Rooter le cellulaire
2. Permettre le débugage USB	2. Permettre le débugage USB
3. Identifier les informations de l'OS : version, kernel, etc.	3. Extraire le kernel du cellulaire
4. Trouver et télécharger les sources du kernel	4. Ingénierie inverse : trouver les informations nécessaires pour définir la configuration
5. Compiler le kernel	5. Compiler
6. Compiler LiME	6. Transférer AMExtractor dans le cellulaire
7. Extraire la mémoire vive	7. Extraire la mémoire vive

Tableau 1 : Principales étapes d'utilisation de LiME et AMExtractor (Inspiré de [34] et [57])

Finalement, nous avons tout de même réussi à extraire la RAM du téléphone LG E410B à l'aide de LiME (voir Annexe 6 pour la procédure suivie). Nous avons également réussi à extraire la mémoire vive des cellulaires Samsung GT-I9505 Galaxy S4 (voir Annexe 2) et Huawei Y300-051 en utilisant AMExtractor.

3.2. Protocole d'analyse

Au sujet de l'analyse des extractions (*dumps*) de mémoire, notre première approche a été de considérer l'utilisation de Volatility étant donné que celle-ci est plutôt bien documentée. Là encore, l'utilisation de Volatility s'est révélée problématique et infructueuse deux fois sur trois en raison des profils nécessaires pour utiliser Volatilty (voir Annexe 1).

Contrairement à la RAM d'une machine Windows où des profils sont prédéfinis dans Volatility, l'analyse de la RAM pour les machines reposant sur un noyau dérivé de Linux, comme Android, nécessite la création d'un profil spécifique dans Volatility [58][59][60] (voir Tableau 2 pour les étapes de création d'un profil Volatility). Malgré nos efforts et nos recherches, nous n'avons réussi qu'à créer un seul profil Volatility fonctionnel, le profil pour l'appareil Samsung GT-I9505 Galaxy S4.

En ce qui concerne le téléphone Huawei Y300-051, le code source du noyau n'étant pas disponible, il est impossible de créer un profil Volatility pour ce modèle de cellulaire. Dans le cas du LG E410B, nous avions tous les éléments nécessaires pour la création d'un profil Volatility et nous avons réussi à créer ledit profil. Toutefois, au moment d'analyser l'image, Volatility n'a pas reconnu cette dernière pour une raison qui nous échappe encore.

Cloner Volatility à partir de GitHub
Éditer le Makefile
Compiler le driver module.ko : commande « make »
Vérifier que le fichier créé, module.dwarf, n'est pas vide
Combiner le fichier module.dwarf et le fichier System.map présent dans le code source du kernel dans un fichier zip
Placer le fichier .zip dans le répertoire volatility/ plugins/ overlays/ linux du paquet Volatility
Examiner la RAM avec Volatility

Tableau 2 : Principales étapes pour créer un profil Volatility (Source : [57], voir Annexe 7)

Suite à ce constat, notre décision a été de créer 5 scénarios spécifiques à la recherche d'une catégorie de preuve (voir Annexe 8 pour le détail des étapes de chaque scénario) et de les mettre en pratique sur l'appareil Samsung GT-I9505 Galaxy S4 :

1. Scénario PDF (*dump_pdf*) : recherche de traces de pdf téléversé / téléchargé
2. Scénario photo (*dump_photo*) : recherche de preuves de l'utilisation d'un appareil photo et traces de la photo
3. Scénario navigation internet (*dump_internet*) : recherche de preuves de l'utilisation d'un navigateur (Google Chrome), historique de navigation et photo téléchargée
4. Scénario courriel (*dump_courriel*) : recherche des adresses courriel, contenu des courriels, historique de courriels, photo jointe à un courriel et de preuve de l'utilisation d'application de courriel (Gmail)
5. Scénario réseau social (*dump_facebook*) : recherche de preuves de l'utilisation des applications Facebook et Messenger et traces de ce qui a été écrit sur le mur Facebook de l'utilisateur et du contenu des messages envoyés par Messenger

Nous avons extrait la mémoire vive (*dump*) du téléphone cellulaire Samsung GT-I9505 Galaxy S4 pour chacun des scénarios ci-dessus. Pour chacune des images, nous avons testé et utilisé plusieurs outils d'analyse (total de 16 outils utilisés voir paragraphe 3.3) afin de vérifier la véracité des 23 hypothèses présentées à l'annexe 9. Ces hypothèses portent sur l'utilisation d'application ainsi que sur les données.

3.3 Outils utilisés pour l'analyse

Bien que Volatility soit l'outil principalement plébiscité par la communauté scientifique pour l'analyse de la mémoire vive d'un téléphone cellulaire Android, afin de ne pas nous limiter dans l'atteinte de notre objectif, nous avons également utilisé de nombreux autres outils.

Parmi les outils disponibles sous Windows, nous nous sommes également servis de Forensics MemDump Extractor [61], de CapLoader [62], de Phone Image Carver [63], de MultiExtractor [64], de PhotoRec [65], de FTK Imager [66], d'Autopsy [30], de HxD [67], et de JPEGExtractor [68] (voir Annexe 1). Certains de ces outils sont des outils d'analyse généraliste tandis que d'autres sont spécialisés.

PhotoRec et FTK Imager sont deux outils fréquemment utilisés par les investigateurs numériques. Toutefois, leur utilisation s'est montrée décevante pour l'analyse de nos images. En effet, il n'est pas possible d'ouvrir les images extraites (format EMiL, voir Annexe 10) avec FTK Imager et PhotoRec ne donne aucun résultat probant pour la découverte des photos.

Malgré le fait que Volatility puisse être utilisé sur Windows, pour des raisons de facilité, nous avons préféré utiliser la version pré installée sur le système d'exploitation Kali Linux [69]. Parmi les outils pré installés sur Kali, nous nous sommes également servis de BinWalk [70] et d'Hexeditor.

Volatility et BinWalk ont été utilisés pour tester les hypothèses concernant l'utilisation d'applications (processus) tandis que les autres outils nous ont été utiles pour les hypothèses liées aux données. Enfin, lorsque tous ces outils se sont révélés incapables de trouver ce que nous cherchons ou pour faciliter la recherche d'informations dans les résultats obtenus, nous avons utilisé des commandes basiques disponibles sur tous les systèmes Linux, notamment Strings et Grep.

4. Résultats de l'analyse :

4.1. Scénario PDF

La signature numérique des fichiers PDF (%PDF [71]) est très bien connue et utilisée par deux nombreux logiciels forensiques tels qu'Autopsy. L'identification et l'extraction des fichiers PDF sont des processus courants lors d'une analyse forensique. Toutefois, bien que ce scénario ait été spécialement conçu pour constater la présence de fichiers PDF dans la mémoire vive d'un téléphone cellulaire, à notre grande surprise, nous n'avons pas réussi à retracer et à extraire les fichiers en question.

Afin d'essayer d'extraire les fichiers PDF, nous avons utilisé les logiciels suivants : Autopsy, MultiExtractor et Phone Image Carver (voir annexes 11, 12 et 13). Bien que ces trois logiciels sont normalement capables de trouver des fichiers PDF (en utilisant une recherche par signature), seulement Phone Image Carver a réussi à détecter la présence de PDF

dans notre image. Nous n'avons toutefois pas pu utiliser ce logiciel pour extraire les fichiers. En effet, cette option n'est disponible que pour la version payante de Phone Image Carver.

De plus, nous avons également essayé, en vain, d'extraire manuellement les fichiers PDF à l'aide de l'éditeur hexadécimal HxD. Bien que la signature numérique des fichiers PDF soit présente dans le *dump* (voir Annexe 14), nous n'avons pas réussi à les extraire. Nous avons pu toutefois constater à plusieurs reprises la présence de 0 au milieu de la chaîne hexadécimale qui aurait dû former le fichier PDF.

En outre, en sachant qu'un éditeur de PDF a été utilisé sur le téléphone cellulaire pour ouvrir ces fichiers, nous avons essayé de retrouver les processus concernés à l'aide de Volatility. Là encore, notre recherche s'est révélée infructueuse. Toutefois, bien que ce n'était pas le but de notre recherche avec cette image, nous avons pu trouver des traces de l'utilisation de Google Chrome, des adresses courriel, des adresses IP ainsi que de nombreuses images au format PNG qui représentent les icônes du téléphone (voir les scénarios suivants pour les méthodes à suivre).

4.2. Scénario photo

L'objectif de ce scénario était de démontrer l'existence de photo, plus précisément de photo au format JPEG, dans la RAM d'un téléphone Android. Dans ce cas, nous n'avons eu aucun problème pour retrouver l'artefact qui nous intéressait à savoir la photo d'un stylo prise avec l'appareil photo du téléphone Samsung GT-I9505 Galaxy S4 .

Afin de trouver et d'extraire la photo au format JPEG, plusieurs outils se sont révélés utiles et efficaces, notamment JPEGExtractor, MultiExtractor, et Phone Image Carver (voir Annexes 15, 16 et 17). Nous soulignerons qu'à l'instar des fichiers PDF, Autopsy ne nous a été d'aucune utilité pour retrouver notre artefact (voir Annexe 18). La version payante du logiciel Forensics MemDump Extractor aurait probablement pu être utilisée pour extraire des fichiers JPEG. Nous n'avons au cours de cette recherche utilisé que la version gratuite de ce logiciel qui se limite à l'extraction des images au format PNG (voir Annexe 19).

Nous avons également extrait manuellement la photo en effectuant une recherche par signature (FF D8 [71]). Enfin, nous avons pu retrouver des processus liés à l'utilisation de l'appareil photo à l'aide de Volatility (voir Annexe 20). Toutefois, pour une raison qui reste encore inconnue, nous n'avons pas réussi à extraire le contenu de ces processus avec Volatility afin de mener une analyse plus poussée. (voir Annexe 21). Enfin, bien que ce n'était pas l'objectif de notre recherche pour ce dump, nous avons là encore pu récupérer des adresses courriel, des adresses IP ainsi qu'une multitude d'images au format PNG.

4.3. Scénario navigation internet

À travers ce scénario, nous voulions prouver qu'il était possible de retrouver l'historique de navigation Internet de l'utilisateur ainsi que de retrouver les adresses http directement tapées dans la barre d'adresse du navigateur. Nous souhaitions également démontrer la présence d'adresses IP dans la RAM d'un cellulaire et qu'il était possible de retrouver une photo téléchargée. Nous n'avons pas rempli nos objectifs en intégralité.

Parmi nos quatre objectifs, nous n'avons réussi qu'à trouver les adresses IP liées aux connexions du cellulaire à l'aide de l'outil Cap Loader (voir Annexe 22). Les adresses IP collectées peuvent ensuite être transformées en nom de domaine. Par exemple, sous Windows, la commande nslookup permet de confirmer que l'utilisateur s'est connecté au Wifi de l'École Polytechnique de Montréal, l'adresse 10.200.27.157 correspondant au réseau Eduroam de cette école (voir Annexe 23).

Du côté de l'historique de consultation internet, nos résultats ont été plus mitigés. En effet, le seul outil qui nous a permis de retrouver quelques traces vraiment probantes de notre navigation internet est Phone Image Carver (voir Annexe 24). Bien que son taux de réussite ne soit pas très élevé, cet outil nous a tout de même permis de retrouver des fragments de pages HTML consultées à partir du téléphone cellulaire analysé.

Nous préciserons tout de même que MultiExtractor ainsi que JPEGExtractor ont réussi à extraire quelques images témoignant de notre navigation internet, comme le logo du site de l'École Polytechnique de Montréal (voir Annexe 25 et 26). JPEGExtractor a également réussi à capter quelques fragments de la photo téléchargée depuis le site Pixabay.

Nous n'avons donc pas réussi à trouver d'historique de consultation en utilisant nos outils d'analyse forensique. Toutefois, afin de confirmer ou infirmer la présence de traces de site internet, nous avons analysé notre image avec les commandes Strings et Grep de Linux (voir Annexe 27).

Nous avons simplement recherché toutes les expressions au format string contenant les mots « http » et « https ». Dans ce cas, nous retrouvons la liste des sites avec lesquels le téléphone a communiqué. Cette commande a été combinée avec la commande Grep afin d'avoir des résultats plus précis. Enfin, nous avons été en mesure de prouver que le navigateur Google Chrome a été utilisé sur l'appareil analysé à l'aide de Volatility (voir Annexe 28).

4.4. Scénario courriel

À travers ce scénario, notre objectif était de démontrer que la mémoire vive d'un cellulaire peut contenir des adresses courriel, le contenu des courriels et les pièces (photo format JPEG) qui y sont jointes. Pour une fois, Autopsy s'est révélé utile pour trouver les adresses courriel (voir Annexe 29). Soulignons également qu'en plus des adresses courriel utilisées pour les échanges de courriel, Autopsy a été en mesure de retrouver l'adresse courriel utilisée par le cellulaire pour se connecter au Wifi. Il est également possible de retrouver des adresses courriel en recherchant par exemple la chaîne « gmail » (voir Annexe 30).

Au sujet des pièces jointes aux courriels, des logiciels qui s'étaient montrés efficaces dans l'analyse des photos au format JPEG (voir Scénario photo) se sont montrés décevants. MultiExtractor et Phone Image Carver n'ont pas réussi à retrouver les photos en pièces jointes des courriels (voir Annexe 31). Rappelons que dans des conditions similaires, ces deux outils n'avaient également pas su retrouver les fichiers PDF joints à des courriels (voir Scénario PDF). Seul JPEGExtractor a été en mesure de détecter et de retrouver les photos recherchées (voir Annexe 32).

De son côté, l'analyse du contenu des courriels a été plus problématique. Aucun des outils utilisés n'a été en mesure de détecter de courriels dans l'image analysée. Toutefois, comme nous avons pu le constater en recherchant des adresses courriel, le contenu de ceux-ci est bien présent dans le *dump* (voir Annexe 30). Nous pouvons donc effectuer une recherche par mots clés, si nous connaissons le contenu des courriels envoyés (voir Annexe 33).

De cette manière, nous avons réussi à retrouver le contenu des courriels envoyés au cours de ce scénario, mais également le contenu de courriels envoyés dans le passé. En effet nous avons retrouvé le contenu des courriels envoyés lors du premier scénario concernant les PDF. À défaut de disposer de mots clés, il faudra effectuer une recherche minutieuse dans l'image pour trouver le contenu des courriels.

Enfin, nous avons pu démontrer l'utilisation de Google Chrome et de l'application Gmail avec l'aide de Volatility (voir Annexe 34). Nous avons toutefois dû croiser les résultats de BinWalk et de Volatility pour arriver à déterminer qu'un processus était bien lié à l'utilisation de l'application Gmail (voir Annexe 35).

4.5. Scénario réseau social

Ce scénario a été conçu afin de prouver que la RAM d'un téléphone cellulaire Android contient des traces des messages écrits par un utilisateur Facebook sur son mur ainsi que du contenu des messages envoyés via l'application Messenger. Encore une fois, les outils traditionnels que nous avons déjà testés au cours des scénarios précédents ne nous ont pas été très utiles.

Toutefois, nous avons réussi à démontrer l'existence de traces de l'utilisation des applications Facebook et Messenger dans la RAM du Samsung GT-I9505 Galaxy S4 en combinant les résultats obtenus par Volatility et BinWalk (voir Annexes 36 et 37). Une recherche dans l'historique des connexions HTTP permet également de confirmer l'utilisation de Messenger (voir Annexe 38).

Au sujet des messages écrits sur le mur et des messages envoyés via Messenger, nous avons suivi une stratégie similaire à celle employée pour retrouver le contenu des courriels (voir Scénario courriel). En ce qui concerne le message publié sur le mur (« SSBsb3ZlIGZvcmVuc2ljcw== »), nous avons réussi à le retrouver uniquement car nous connaissions son contenu (voir Annexe 39). Toutefois, en ce qui concerne les messages envoyés et reçus par Messenger, nous avons remarqué qu'une chaîne de mots permet de les repérer. Cette chaîne est : « "user_key":"FACEBOOK: » (voir Annexe 40)

4.6. Autres résultats

L'utilisation de Phone Image Carver lors de l'analyse des images nous a permis de constater que cet outil était capable de détecter des bases de données SQLite (voir Annexes 14, 18 et 24). Nous n'avons toutefois pas analysé le contenu de ces bases de données lors de notre étude. De la même manière, nous avons pu constater la présence de pièces jointes aux courriels chiffrées en base 64. Nous n'avons pas déchiffré ces pièces jointes.

Enfin, afin de tester notre hypothèse que la RAM d'un téléphone cellulaire Android contient la liste des contacts enregistrés dans le téléphone, nous avons effectué une recherche de chaîne de caractères étant donné que nous connaissions le contenu de la liste de contacts de l'appareil (voir Annexe 41) . Cette recherche nous a permis de confirmer que la liste des contacts de l'appareil est bien présente dans sa RAM. Cette liste est ordonnée avec un identifiant par contact. En suivant la même méthode, nous avons également pu retrouver les numéros de téléphone stockés dans le cellulaire dans la mémoire vive.

5. Conclusion :

L'objectif de notre recherche n'était pas de présenter une ou des méthodes d'analyse formelle de la RAM d'un téléphone cellulaire. Notre but était de démontrer la présence de preuves dans la RAM d'un téléphone portable. Pour cela, nous avons formé une série d'hypothèses. Au final, sur un total de 23 hypothèses testées, nous avons réussi à prouver directement (ou indirectement) la véracité de 19 d'entre elles. Le tableau qui suit présente une synthèse des résultats de notre étude :

Utilisation de l'appareil photo du cellulaire	« Historique » de navigation (pages http/https)	Utilisation de l'application Gmail	« Historique » de courriels (courriels passés)	Utilisation de l'application Messenger	Adresses IP	Base de données SQLite
Photo format JPEG + PNG	Photo téléchargée format JPEG	Adresses courriel	Photo envoyée (JPEG) en pièce jointe d'un courriel	Contenu de ce qui a été écrit sur un mur Facebook	Liste des contacts + numéro de téléphone	Pièces jointes aux courriels (base 64)
Utilisation du navigateur Google Chrome	Contenu des pages (html) consultées	Contenu des courriels	Utilisation de l'application Facebook	Contenu des messages envoyés via Messenger		

Tableau 3 : Synthèse des éléments trouvés (voir Annexe 42 pour plus de détails)

Cette étude aura donc permis de démontrer la présence de preuves pertinentes pour un investigateur numérique dans la RAM d'un téléphone cellulaire Android et aura offert un rapide aperçu des types de preuves. Cette recherche aura également permis, comme l'ont déjà fait d'autres auteurs [58], de constater la difficulté de l'extraction et de l'analyse des *dumps*.

Notre analyse a porté sur l'existence de preuves dans la RAM d'un cellulaire utilisé en tant que téléphone intelligent. Bien que nous avons pu prouver la présence de la liste de contacts (nom et numéro de téléphone) dans la RAM d'un téléphone Android, nous n'avons pas traité la partie des preuves qui découlent de l'utilisation d'un cellulaire en tant que téléphone (appels téléphoniques, SMS, etc.), partie qui pourrait être analysée dans une future recherche.

Médiagraphie :

[1] StatCounter Global Stats, *Desktop vs Mobile vs Tablet Market Share Worldwide*, http://gs.statcounter.com/platform-market-share/desktop-mobile-tablet Consulté le 19 janvier 2019

[2] StatCounter Global Stats, *Mobile Operating System Market Share Worldwide - December 2018*, http://gs.statcounter.com/os-market-share/mobile/worldwide Consulté le 19 janvier 2019

[3] Jansen, W, et R P Ayers. « Guidelines on Cell Phone Forensics ». Gaithersburg, MD: National Institute of Standards and Technology, 2007. https://doi.org/10.6028/NIST.SP.800-101.

[4] Martin, Andrew. « Mobile Device Forensics », SANS Institute, 2009. https://www.sans.org/reading-room/whitepapers/forensics/mobile-device-forensics-32888

[5] Ayers, Rick, Wayne Jansen, Ludovic Moenner, et Aurelien Delaitre. « Cell Phone Forensic Tools : An Overview and Analysis Update ». Gaithersburg, MD: National Institute of Standards and Technology, 2007. https://doi.org/10.6028/NIST.IR.7387.

[6] Christopher Tassone, Ben Martini. « Mobile Device Forensics: A Snapshot ». Australian Institute of Criminology, 3 novembre 2017. https://aic.gov.au/publications/tandi/tandi460.

[7] Mallidi et Palli, « A Comprehensive Analysis of Smartphone Forensics & Data Acquisitions ». International Journal of Advanced Research in Computer Science and Software Engineering 6(2), février 2016, pp. 270-276

[8] Steve Robles. « Mobile Phone Forensics ». Association of Workplace Investigators, AWI Journal, Vol. 4, N. 1, janvier 2013

[9] Willassen, Svein. « Forensic Analysis of Mobile Phone Internal Memory ». In *Advances in Digital Forensics*, édité par Mark Pollitt et Sujeet Shenoi, 194:191-204. Boston: Kluwer Academic Publishers, 2006. https://doi.org/10.1007/0-387-31163-7_16.

[10] Kim et al., « Data Acquisition from Cell Phone using Logical Approach ». World Academy of Science, Engineering and Technology 32 2007, http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.193.1616&rep=rep1&type=pdf

[11] « SWGDE Best Practices for Mobile Phone Forensics ». 11 février 2013, https://www.swgde.org/documents/Current%20Documents/SWGDE%20Best%20Practices%20for%20Mobile%20Phone%20Forensics

[12] Det. Cindy Murphy. « Developing Process for the Examination of Cellular Phone Evidence », Data Extraction & Documentation, s. d., 18. http://ccf.cs.uml.edu/forensicspapers/Cellular%20Phone%20Evidence%20Data%20Extraction%20and%20Documentation.pdf

[13] Det. Cynthia A. Murphy « Developing Process for Mobile Device Forensics », s. d., 9. https://digital-forensics.sans.org/media/mobile-device-forensic-process-v3.pdf

[14] Ali et al., « A Metamodel for Mobile Forensics Investigation Domain ». n. PLoS ONE 12(4): e0176223. 2017, https://doi.org/10.1371/journal.pone.0176223

[15] « Common Mobile Forensics Tools and Techniques ». InfoSec Resources. Consulté le 1 février 2019. https://resources.infosecinstitute.com/category/computerforensics/introduction/mobile-forensics/common-mobile-forensics-tools-and-techniques/.

[16] « Mobile Device Forensics ». I.R.I.S LLC, 1er décembre 2016, http://www.irisinvestigations.com/wordpress/wp-content/uploads/2018/06/MOBILE_DEVICES_12_01_16.pdf

[17] Satish Bommisetty, Rohit Tamma, et Heather Mahalik. « Practical Mobile Forensics ». Packt Publishing, 2014. ISBN 978-1-78328-831-1.

[18] Ogazi-Onyemaechi, B.C., A. Dehghantanha, et K.-K.R. Choo. « Performance of Android Forensics Data Recovery Tools ». In Contemporary Digital Forensic Investigations of Cloud and Mobile Applications, 91-110. Elsevier, 2017. https://doi.org/10.1016/B978-0-12-805303-4.00007-1.

[19] « Getting Started with Android Forensics ». InfoSec Resources, 20 août 2014. https://resources.infosecinstitute.com/getting-started-android-forensics/.

[20] Lessard, Jeff, Gary C Kessler, et Gary Kessler Associates. « Android Forensics: Simplifying Cell Phone Examinations » 4 (2010): 12. https://www.garykessler.net/library/SSDDFJ_V4_1_Lessard_Kessler.pdf

[21] Thing, Ng, et Chang, « Live Memory Forensics of Mobile Phones ». Digital Forensic Research Conference, 2010, https://www.dfrws.org/sites/default/files/session-files/pres-live_memory_forensics_of_mobile_phones.pdf

[22] Vrizlynn L. L. Thing et Zheng-Leong Chua, « Smartphone Volatile Memory Acquisition for Security Analysis and Forensics Investigation ». 2013, https://hal.inria.fr/hal-01463829/document

[23] Joe Sylve. « Android Mind Reading : Memory Acquisition and Analysis with LiME and Volatility », https://www.sans.org/cyber-security-summit/archives/file/summit-archive-1493741700.pdf

[24] Sylve, Joseph T. « Android Memory Capture and Applications for Security and Privacy », . University of New Orleans Theses and Dissertations 1400. 2011, https://scholarworks.uno.edu/td/1400/

[25] Nuno Santos. « Mobile Forensics : Android ». CSF : Forensics Cyber-Security. Automne 2015, https://fenix.tecnico.ulisboa.pt/downloadFile/1970943312266679/csf-12.pdf

[26] Igor Mikhaylov et Oleg Skulkin. « Android Forensic Analysis with Autopsy | Digital Forensics | Computer Forensics | Blog ». https://www.digitalforensics.com/blog/android-forensic-analysis-with-autopsy/

[27] Mark Lohrum, « Using Autopsy to examine an Android image | Free Android Forensics ». 6 novembre 2014, https://freeandroidforensics.blogspot.com/2014/11/using-autopsy-to-examine-android-image.html

[28] LiME, 504ENSICS Labs, 2019. https://github.com/504ensicsLabs/LiME.

[29] Volatility Foundation, https://www.volatilityfoundation.org/

[30] Autopsy Digital Forensic, https://www.autopsy.com/

[31] Broenner, Simon, Hans Höfken, et Marko Schuba. « Streamlining Extraction and Analysis of Android RAM Images »: In Proceedings of the 2nd International Conference on Information Systems Security and Privacy, 255-64. Rome, Italy: SCITEPRESS - Science and and Technology Publications, 2016. https://doi.org/10.5220/0005652802550264.

[32] Mahalik, Heather. « Open Source Mobile Device Forensics », 2014, 21. https://www.nist.gov/sites/default/files/documents/forensics/6-Mahalik_OSMF.pdf

[33] Yang, Haiyu, Jianwei Zhuge, Huiming Liu, et Wei Liu. « A Tool for Volatile Memory Acquisition from Android Devices ». In *Advances in Digital Forensics XII*, édité par Gilbert Peterson et Sujeet Shenoi, 484:365-78. Cham: Springer International Publishing, 2016. https://doi.org/10.1007/978-3-319-46279-0_19.

[34] AMExtractor, https://github.com/ir193/AMExtractor.

[35] Dr. Michael Spreitzenbarth, Dr. Johann Uhrmann « Mastering Python Forensics » Packt Publishing, octobre 2015. ISBN 9781783988044.

[36] Rohit Tamma et Donnie Tindall. « Learning Android Forensics », Packt Publishing, 2015. ISBN 978-1-78217-457-8.

[37] Holger Macht. « Live Memory Forensics on Android with Volatility ». Thèse, Friedrich-Alexander University Erlangen-Nuremberg. Janvier 2013
[38] Sylve, Joe, Andrew Case, Lodovico Marziale, et Golden G. Richard. « Acquisition and Analysis of Volatile Memory from Android Devices ». Digital Investigation 8, nº 3-4 (février 2012): 175-84. https://doi.org/10.1016/j.diin.2011.10.003.
[39] Yang, Seung Jei, Jung Ho Choi, Ki Bom Kim, Rohit Bhatia, Brendan Saltaformaggio, et Dongyan Xu. « Live Acquisition of Main Memory Data from Android Smartphones and Smartwatches ». Digital Investigation 23 (décembre 2017): 50-62. https://doi.org/10.1016/j.diin.2017.09.003.
[40] Ntantogian, Christoforos, Dimitris Apostolopoulos, Giannis Marinakis, et Christos Xenakis. « Evaluating the Privacy of Android Mobile Applications under Forensic Analysis ». Computers & Security 42 (mai 2014): 66-76. https://doi.org/10.1016/j.cose.2014.01.004.
[41] Heriyanto, Andri P. « Procedures And Tools For Acquisition And Analysis Of Volatile Memory On Android Smartphones ». 11th Australian Digital Forensics Conference. Held on the 2nd-4th December 2013 at Edith Cowan University (2013): Western Australia-. https://doi.org/10.4225/75/57b3c9bafb86f.
[42] Broenner, Simon, Hans Höfken, et Marko Schuba. « Streamlining Extraction and Analysis of Android RAM Images »: In Proceedings of the 2nd International Conference on Information Systems Security and Privacy, 255-64. Rome, Italy: SCITEPRESS - Science and and Technology Publications, 2016. https://doi.org/10.5220/0005652802550264.
[43] Angel Alonso-Parrizas. « Forensic Analysis On Android: A Practical Case », GIAC (GMOB) Gold Certification, SANS Institute InfoSec Reading Room, 2015. https://www.sans.org/reading-room/whitepapers/forensics/forensic-analysis-android-practical-case-36317
[44] Bernard Lebel. « Analyse de maliciels sur Android par l'analyse de la mémoire vive » Mémoire, Université Laval, 2018, https://corpus.ulaval.ca/jspui/bitstream/20.500.11794/29851/1/34353.pdf
[45] Dunham, Ken. « Android Malware and Analysis ». Auerbach Publications, 2014. https://doi.org/10.1201/b17598.
[46] Nikos Gkogkos. « Live Memory Forensics on Android devices ». https://fr.slideshare.net/nikoskapios/live-memory-forensics-on-android-devices.
[47] Gazdag, Andras, et Levente Buttyan. « Android Malware Analysis Based On Memory Forensics », s. d., 8. http://www.hit.bme.hu/~buttyan/publications/GazdagB14vocal.pdf
[48] Prof. Christos Xenakis, Dr. Christoforos Ntantogian. « Acquisition and Analysis of Android Memory » Greek Cybercrime Center, http://www.ucd.ie/cci/cync/Acquisition%20and%20Analysis%20of%20Android%20Memory.pdf
[49] Long Chen et Honghua Zhao. « Forensic Analysis of Cloud Storage on Android Volatile Memory ». In May 22-24, 2017 Kuala Lumpur (Malaysia) ICLTET-2017, ACBES-2017. IIE, 2017. https://doi.org/10.15242/IIE.E0517012.
[50] « What Is RAM and ROM Memory? » AndroidPIT. Consulté le 9 mars 2019. https://www.androidpit.com/what-is-ram-rom-memory.
[51] Hex Rays, IDA, https://www.hex-rays.com/products/ida/index.shtml
[52] Android Debug Bridge (adb), https://developer.android.com/studio/command-line/adb
[53] Android SDK, https://developer.android.com/studio
[54] Android NDK, https://developer.android.com/ndk
[55] Kingroot v5.3.7.20180619, https://kingrootofficial.com/download/kingroot-v5-3-7-20180619
[56] Android, Goldfish kernel, https://android.googlesource.com/kernel/goldfish/

[57] Samsung, Code source, https://opensource.samsung.com/reception/receptionSub.do?method=search&searchValue=GT-I9505

[58] Philipp Wächter. « Practical Infeasibility of Android Smartphone Live Forensics – Applicability Constraints of LiME and Volatility » Master's Thesis, Friedrich-Alexander-Universität, 30 avril 2015, https://faui1-files.cs.fau.de/filepool/gruhn/thesis_waechter.pdf

[59] Volatilty Foundation, Linux, https://github.com/volatilityfoundation/volatility/wiki/Linux

[60] Volatility Foundation, Android, https://github.com/volatilityfoundation/volatility/wiki/Android

[61] Techipick, Forensics MemDump Extractor, https://www.techipick.com/forensics-memdump-extractor

[62] Netresec, CapLoader, https://www.netresec.com/?page=CapLoader

[63] Phone Image Carver, http://www.phoneimagecarver.com/

[64] MultiExtractor, https://www.multiextractor.com/index.html

[65] CG Security, PhotoRec, https://www.cgsecurity.org/wiki/PhotoRec

[66] Access Data, FTK Imager, https://accessdata.com/product-download/

[67] HxD, https://mh-nexus.de/en/hxd/

[68] JPEGExtractor, https://sourceforge.net/projects/jpegextractor/

[69] Kali Linux, https://www.kali.org/

[70] ReFirmLabs , BinWalk, https://github.com/ReFirmLabs/binwalk

[71] « File Signatures ». Consulté le 7 mars 2019. https://www.garykessler.net/library/file_sigs.html.

Annexes :

Annexe 1 : Processus suivi : démarches et outils utilisés

Annexe 2 : Étapes pour utiliser AMExtractor appliquées au cas du Samsung GT-I9505 Galaxy S4

Compilation:

ndk-build NDK_PROJECT_PATH=. APP_BUILD_SCRIPT=Android.mk TARGET_ARCH=arm TARGET_ARCH_ABI=armeabi-v7a

Config.h:

SPARSE_MEM <==== Reverse Engineering
STRUCT_PAGE_SIZE 32 <==== Reverse Engineering
USE_SYNC_PTMX <==== Guess : soit USE_SYNC_PTMX soit USE_SEEK_ZERO

Détermination des paramètres:

A- [https://forum.xda-developers.com/android/software-hacking/how-to-reverse-engineer-kernel-t3137384]

1- Extraire le kernel du téléphone cellulaire:

Côté téléphone:
busybox nc -l -p 7777 < /dev/block/bootdevice/by-name/boot
Côté ordinateur:
adb forward tcp:7777 tcp:7777
nc 127.0.0.1 7777 > boot.img

2- Dépaqueter le kernel [https://forum.xda-developers.com/showthread.php?t=2319018]

unpack boot.img

3- Décompresser le kernel [https://forum.xda-developers.com/showthread.php?t=901152]

repack-zImage.sh -u zImage
Cette commande crée d'un répertoire nommé 'zImage_unpacked' qui contient le kernel `kernel.img`

4- Récupérer les symboles du kernel et les convertir en idc

echo 0 > /proc/sys/kernel/kptr_restrict
cat /proc/kallsyms >/sdcard/symbols.txt
open symbols.txt with idccreate.exe [#A]

B- Ingénierie inverse avec IDA pro

1- Ouvrir kernel.img avec IDA pro and choisir ARM petit boutiste comme processeur.

Établir la localisation de la ROM et la localisation de l'input à la première valeur (adresse) de symbols.txt
Lancer le script idc (file -> run script -> select the script to run) <==== Établi les noms des fonctions

2- localiser vm_normal_page

A la fin de vm_normal_page la définition est un appel à pfn_to_page() avec l'adresse de mem_map (FLAT_MEM) ou de mem_section (SPARSE_MEM)

STRUCT_PAGE_SIZE est la taille du registre utilisé pour appeler pfn_to_page

Extraction: Se référer à la documentation de AMExtractor, https://github.com/ir193/AMExtractor

Annexe 3 : Exemple d'utilisation d'*Android Debug Bridge,* adb – *Android Studio (v.3.18) Emulator installé sous Windows 10 (x86)*

adb root
adb devices
adb shell
-> on a un shell dans le cellulaire
C:\Logiciels\Android-SDK\platform-tools>adb devices
List of devices attached
emulator-5554 device
C:\Logiciels\Android-SDK\platform-tools>adb shell
generic_x86:/ #
generic_x86:/ # ls /dev/kmem
ls: /dev/kmem: No such file or directory
generic_x86:/ #
// pas de /dev/kmem -> peut pas utiliser AMExtractor
generic_x86:/ # grep vm_normal_page /proc/kallsyms
0000000000000000 T vm_normal_page
generic_x86:/ # uname
Linux
generic_x86:/ # uname -a
Linux localhost 3.18.91+ #1 SMP PREEMPT Thu Jan 25 02:43:49 UTC 2018 i686
generic_x86:/ # cat /proc/version
Linux version 3.18.91+ (android-build@xpcd3.ams.corp.google.com) (gcc version 4.9 20140827 (prerelease) (GCC))

#1 SMP PREEMPT Thu Jan 25 02:43:49 UTC 2018

generic_--->x86<---:/ #

<u>Annexe 4 :</u> Exemple d'utilisation d'adb pour installer AMExtractor – ***Android Studio (v.3.10) Emulator sur Kali Linux***

root@Host-001:~# cd ndk

root@Host-001:~/ndk# pwd

/root/ndk

root@Host-001:~/ndk# PATH=$PATH:/root/ndk

root@Host-001:~/ndk# cd ..

root@Host-001:~# cd AMExtractor/

root@Host-001:~/AMExtractor# leafpad kernel_struct.h

root@Host-001:~/AMExtractor# ndk-build NDK_PROJECT_PATH=. APP_BUILD_SCRIPT=Android.mk

Android NDK: APP_PLATFORM not set. Defaulting to minimum supported version android-14.

[arm64-v8a] Compile : AMExtractor <= main.c

(…) ⇐ Étapes du processus de compilation + Warnings générés

[x86_64] Executable : AMExtractor

[x86_64] Install : AMExtractor => libs/x86_64/AMExtractor

root@Host-001:~/AMExtractor# PATH=$PATH:/root/Android/Sdk/platform-tools

root@Host-001:~/AMExtractor# adb push libs/x86_64/AMExtractor /data/local/tmp

libs/x86_64/AMExtractor: 1 file pushed. 0.3 MB/s (14280 bytes in 0.041s)

root@Host-001:~/AMExtractor# adb shell

Se déplacer jusqu'au répertoire /data/local/tmp

127|generic_x86:/data/local/tmp # ls

AMExtractor com.example.myapplication-build-id.txt

generic_x86:/data/local/tmp # chmod 777 AMExtractor

generic_x86:/data/local/tmp # **./AMExtractor**

addr iomem_resource not found

addr copy_page not found

addr mem_map not found

addr ptmx_fops not found

open /dev/kmem error

: No such file or directory

1|generic_x86:/data/local/tmp #

<u>Annexe 5 :</u> Exemple d'utilisation d'adb pour compiler le kernel *Goldfih – Android Studio (v.3.18) Emulator sous Windows 10 (x86)*

Microsoft Windows [version 10.0.17134.523]
(c) 2018 Microsoft Corporation. Tous droits réservés.

C:\>cd "adb platform-tools"

C:\adb platform-tools>adb pull /proc/config.gz
/proc/config.gz: 1 file pulled. 0.4 MB/s (23401 bytes in 0.061s)
C:\adb platform-tools>git clone https://android.googlesource.com/platform/prebuilts/gcc/linux-x86/x86/x86_64-linux-android-4.9
Cloning into 'x86_64-linux-android-4.9'...
remote: Sending approximately 388.90 MiB ...
remote: Counting objects: 593, done
remote: Finding sources: 100% (4/4)
remote: Total 2736 (delta 1260), reused 2736 (delta 1260)
Receiving objects: 100% (2736/2736), 388.90 MiB | 1.23 MiB/s, done.
Resolving deltas: 100% (1260/1260), done.
Checking out files: 100% (260/260), done.

C:\adb platform-tools>git clone https://android.googlesource.com/kernel/goldfish
Cloning into 'goldfish'...
remote: Sending approximately 1.10 GiB ...
remote: Counting objects: 25, done
remote: Finding sources: 100% (25/25)
remote: Total 5989850 (delta 5036639), reused 5989848 (delta 5036639)
Receiving objects: 100% (5989850/5989850), 1.09 GiB | 5.04 MiB/s, done.
Resolving deltas: 100% (5036639/5036639), done.

C:\adb platform-tools>cd goldfish

C:\adb platform-tools\goldfish>git checkout android-3.18
error: unable to create file drivers/gpu/drm/nouveau/core/subdev/i2c/aux.c: No such file or directory
Checking out files: 100% (48215/48215), done.
Switched to a new branch 'android-3.18'
D	drivers/gpu/drm/nouveau/core/subdev/i2c/aux.c
Branch 'android-3.18' set up to track remote branch 'android-3.18' from 'origin'.

C:\adb platform-tools\goldfish>set ARCH=x86

C:\adb platform-tools\goldfish>set SUBARCH=x86

C:\adb platform-tools>set CROSS_COMPILE=

Répertoire de C:\adb platform-tools\goldfish\android

2019-02-05 17:24 <DIR> .
2019-02-05 17:24 <DIR> ..
2019-02-05 17:24 <DIR> configs
0 fichier(s) 0 octets
3 Rép(s) 109 941 997 568 octets libres

C:\adb platform-tools\goldfish\android>

Pas de commande « make » sous Windows, impossible d'aller plus loin. Problème similaire sous Kali Linux, nous n'avons pas réussi à faire la compilation du kernel pour des raisons de dépendance liées à la commande « make »

Annexe 6: Procédure d'utilisation de LiME appliquée au cas du LG E410B

1. Téléchargement du noyau, *kernel*, du LGE410B : http://opensource.lge.com/osList/list?m=Mc001&s=Sc002

2. Compilation du noyau en suivant les commandes données dans la documentation de Volatility :
https://github.com/volatilityfoundation/volatility/wiki/Android

2.1. Configuration des variables d'environnement :
$ export ARCH=arm
$ export SUBARCH=arm
$ export CROSS_COMPILE=arm-eabi

2.2. Utilisation d'adb pour récupérer la configuration du cellulaire :
$ cd ~/android-sdk/platform-tools
$ ./adb pull /proc/config.gz

2.3. Décompresser le fichier config.gz et copier le résultat dans le répertoir ~/android-source sous le nom .config

2.4. Vérifier que les instructions suivantes sont présentent dans .config et sont activées :
CONFIG_MODULES=y
CONFIG_MODULES_UNLOAD=y
CONFIG_MODULES_FORCE_UNLOAD=y

2.5. Compilation du noyau :

$ make

.....

CC arch/arm/boot/compressed/misc.o

LD arch/arm/boot/compressed/vmlinux

OBJCOPY arch/arm/boot/zImage

Kernel: **arch/arm/boot/zImage is ready ⇐ Tout c'est bien passé**

Building modules, stage 2.

MODPOST 1 modules

CC drivers/hid/hid-dummy.mod.o

LD [M] drivers/hid/hid-dummy.ko

Voir la documentation de Volatility (https://github.com/volatilityfoundation/volatility/wiki/Android) et de LiME pour la suite des étapes

Annexe 7 : Exemple du contenu du fichier module.dwarf et du fichier System.map

module.dwarf :

.debug_info

<0><0x0+0xb><DW_TAG_compile_unit> DW_AT_producer<GNU C 4.8 -mlittle-endian -marm -mabi=aapcs-linux -mno-thumb-interwork -mcpu=cortex-a15 -mfloat-abi=soft -mfpu=vfp -g -Os -fno-strict-aliasing -fno-common -fno-delete-null-pointer-checks -fno-dwarf2-cfi-asm -fstack-protector -funwind-tables -fomit-frame-pointer -fno-strict-overflow -fconserve-stack> DW_AT_language<DW_LANG_C89> DW_AT_name</tmp/linux/module.c> DW_AT_comp_dir</windows/GT-I9505_EUR_LL_Opensource/Kernel/output> DW_AT_stmt_list<0x00000000>
<1><0x1d><DW_TAG_typedef> DW_AT_name<__s8> DW_AT_decl_file<0x00000001 /windows/GT-I9505_EUR_LL_Opensource/Kernel/include/asm-generic/int-ll64.h> DW_AT_decl_line<0x00000013> DW_AT_type<<0x00000028>>
<1><0x28><DW_TAG_base_type> DW_AT_byte_size<0x00000001> DW_AT_encoding<DW_ATE_signed_char> DW_AT_name<signed char>
<1><0x2f><DW_TAG_typedef> DW_AT_name<__u8> DW_AT_decl_file<0x00000001 /windows/GT-I9505_EUR_LL_Opensource/Kernel/include/asm-generic/int-ll64.h> DW_AT_decl_line<0x00000014> DW_AT_type<<0x0000003a>>
<1><0x3a><DW_TAG_base_type> DW_AT_byte_size<0x00000001> DW_AT_encoding<DW_ATE_unsigned_char> DW_AT_name<unsigned char>
……..
……..

System.map :

00000000 t __vectors_start

00000020 A cpu_v7_suspend_size

00001000 t __stubs_start

00001004 t vector_rst

00001020 t vector_irq

000010c0 t vector_dabt

00001140 t vector_pabt

000011c0 t vector_und

00001240 t vector_addrexcptn

00001244 t vector_fiq

00001244 T vector_fiq_offset

00001248 t .krait_fixup

c0004000 A swapper_pg_dir

c0008000 T _text

c0008000 T stext

c000804c t __create_page_tables

c0008100 t __turn_mmu_on_loc

……

……...

<u>Annexe 8 :</u> Scénarios détaillés pour l'extraction des images (*dumps)*

Protocole - Liste des scénarios et étapes	
✓	**Scénario 1: PDF**
1	Allumer le téléphone cellulaire Samsung GT-I9505 Galaxy S4
2	Se connecter à internet (via wifi) en utilisant l'application Google Chrome
3	Se rendre directement sur Google (via la barre d'adresse: google.com)
4	Chercher dans Google « *RAM Android Analysis* »
5	Télécharger le pdf « *Streamlining Extraction and Analysis of Android RAM Images* »
6	Cliquer sur le bouton envoyer (« *Sent file* ») et sélectionner Gmail
7	Envoyer un courriel à ploymtl.memdump@gmail.com : Objet: Android RAM ; Corps du message: Que penses-tu de ce pdf ? ; Pièce jointe: pdf téléchargé précédemment
8	Se connecter à Gmail en utilisant le navigateur Google Chrome
9	Attendre la réponse de ploymtl.memdump@gmail.com. Réponse: Oui pas mal. Regarde ce pdf ; Pièce jointe: pdf «*Android RAM analysis. Technion student's project with CheckPoint* »
10	Consulter le pdf reçu
11	Imager (*dump*) la RAM du cellulaire avec AMExtractor
12	Éteindre le cellulaire
13	Calculer l'empreinte numérique (MD5) du dump
✓	**Scénario 2: Photographie**
1	Allumer le téléphone cellulaire Samsung GT-I9505 Galaxy S4
2	Prendre une photo d'un stylo
3	Sauvegarder la photo

4	Imager (*dump*) la RAM du cellulaire avec AMExtractor
5	Éteindre le cellulaire
6	Calculer l'empreinte numérique (MD5) du dump
✓	**Scénario 3: Navigation Internet**
1	Allumer le téléphone cellulaire Samsung GT-I9505 Galaxy S4
2	Se connecter à internet (via wifi) en utilisant l'application Google Chrome
3	Se rendre directement sur polymtl.ca (via la barre d'adresse)
4	Se rendre directement sur Google (via la barre d'adresse: google.com)
5	Chercher dans Google « Wiki Forensics »
6	Se rendre sur forensicswiki.org via le lien de résultat de recherche de Google
7	Cliquer sur le lien Topics > File Analysis (en bas à gauche de la page)
8	Cliquer sur le lien hachoir (« *Tools* » en bas de la page)
9	Se rendre directement sur Pixabay (via la barre d'adresse: https://pixabay.com)
10	Faire une recherche pour « cyber crime »
11	Choisir une image et la télécharger
12	Imager (*dump*) la RAM du cellulaire avec AMExtractor
13	Éteindre le cellulaire
14	Calculer l'empreinte numérique (MD5) du dump
✓	**Scénario 4: Courriels**
1	Allumer le téléphone cellulaire Samsung GT-I9505 Galaxy S4
2	Se connecter à internet (via wifi) en utilisant l'application Google Chrome
3	Se rendre directement sur gmail (via la barre d'adresse: google.com/gmail)
4	Se connecter au compte inf8430.memdump@gmail.com
5	Envoyer un courriel à ploymtl.memdump@gmail.com : Objet: photo de stylo ; Corps du message: Salut, regarde mon beau stylo ; Pièce jointe: photo du stylo prise précédemment (tâche 3)
6	Attendre la réponse de ploymtl.memdump@gmail.com: Objet: Re: photo de stylo ; Corps du message: Wow ton stylo irait parfaitement pour prendre des notes lors d'une investigation
7	Lire la réponse de ploymtl.memdump@gmail.com
8	Éteindre Google Chrome
9	Lancer l'application Gmail. Se connecter au compte inf8430.memdump@gmail.com
10	Répondre à ploymtl.memdump@gmail.com: Re:Re: photo de stylo, Corps de message: Les bons outils font les bons ouvriers :)
11	Attendre la réponse de ploymtl.memdump@gmail.com: Objet: Re:Re:Re: photo de stylo ; Corps du message: Élémentaire mon cher Watson
12	Lire la réponse de ploymtl.memdump@gmail.com
13	Imager (dump) la RAM du cellulaire avec AMExtractor
14	Éteindre le cellulaire
15	Calculer l'empreinte numérique (MD5) du dump
✓	**Scénario 5: Réseaux sociaux**
1	Allumer le téléphone cellulaire Samsung GT-I9505 Galaxy S4
2	Se connecter au compte Facebook de Harry Jaimelesdump (= inf8430.memdump@gmail.com) via l'application Facebook
3	Publier: « SSBsb3ZlIGZvcmVuc2ljcw== » sur le mur
4	Envoyer un message à Triphon Tournesol (= ploymtl.memdump@gmail.com). Contenu du message: « Peux tu cracker ce que j'ai écrit sur mon mur? »
5	Fermer l'application Facebook
6	Attendre la réponse de Triphon Tournesol. Réponse: « Bien sur, c'est du base64 !!! »
7	Consulter la réponse de Triphon Tournesol via l'application Messenger
8	Répondre à Triphon Tournesol via Messenger: « Wow t'es trop doué »
9	Fermer l'application Messenger
10	Imager (*dump*) la RAM du cellulaire avec AMExtractor
11	Éteindre le cellulaire
12	Calculer l'empreinte numérique (MD5) du dump

<u>Annexe 9 :</u> Liste des hypothèses testées dans cette étude :

- <u>Hypothèses concernant l'utilisation d'application (processus) :</u>

  - *Hypothèse 1 : La RAM* d'un téléphone cellulaire Android contient des traces de l'utilisation d'un éditeur *PDF*
  - *Hypothèse 2 : La RAM* d'un téléphone cellulaire Android contient des traces de l'utilisation de l'appareil photo du cellulaire
  - *Hypothèse 3 : La RAM* d'un téléphone cellulaire Android contient des traces de l'utilisation d'un navigateur Internet (Google Chrome)
  - *Hypothèse 4 :* La RAM d'un téléphone cellulaire Android contient des traces de l'utilisation d'une application de courriel (Gmail)
  - *Hypothèse 5 :* La RAM d'un téléphone cellulaire Android contient des traces de l'utilisation de l'application de réseau social Facebook
  - *Hypothèse 6 :* La RAM d'un téléphone cellulaire Android contient des traces de l'utilisation d'une application de messagerie instantanée (Facebook Messenger)

- <u>Hypothèses liées aux données :</u>

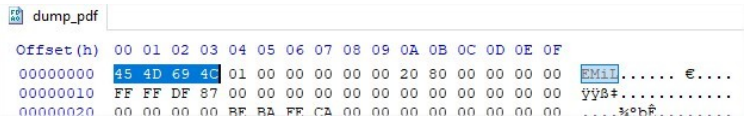
  - *Hypothèse 1 :* La RAM d'un téléphone cellulaire Android contient des traces des fichiers PDF téléversés
  - *Hypothèse 2 :* La RAM d'un téléphone cellulaire Android contient des traces des fichiers PDF téléchargés
  - *Hypothèse 3 :* La RAM d'un téléphone cellulaire Android contient des traces des photos prises (format JPEG) avec le cellulaire
  - *Hypothèse 4 :* La RAM d'un téléphone cellulaire Android contient des traces de l'historique de navigation Internet
  - *Hypothèse 5 :* La RAM d'un téléphone cellulaire Android contient des traces d'adresses IP
  - *Hypothèse 6 :* La RAM d'un téléphone cellulaire Android contient des traces des photos téléchargées
  - *Hypothèse 7 :* La RAM d'un téléphone cellulaire Android contient des traces du contenu de courriels envoyés
  - *Hypothèse 8 :* La RAM d'un téléphone cellulaire Android contient des traces du contenu de courriels reçus
  - *Hypothèse 9 :* La RAM d'un téléphone cellulaire Android contient des traces du contenu de courriels passés (historique de courriel)
  - *Hypothèse 10 :* La RAM d'un téléphone cellulaire Android contient des traces des adresses courriels
  - *Hypothèse 11 :* La RAM d'un téléphone cellulaire Android contient des traces des pièces jointes aux courriels envoyés
  - *Hypothèse 12 :* La RAM d'un téléphone cellulaire Android contient des traces des pièces jointes aux courriels reçus
  - *Hypothèse 13 :* La RAM d'un téléphone cellulaire Android contient des traces de ce qu'écrit un utilisateur Facebook sur son mur

- *Hypothèse 14 :* La RAM d'un téléphone cellulaire Android contient des traces des messages envoyés via l'application Messenger
- *Hypothèse 15 :* La RAM d'un téléphone cellulaire Android contient des traces des messages reçus via l'application Messenger

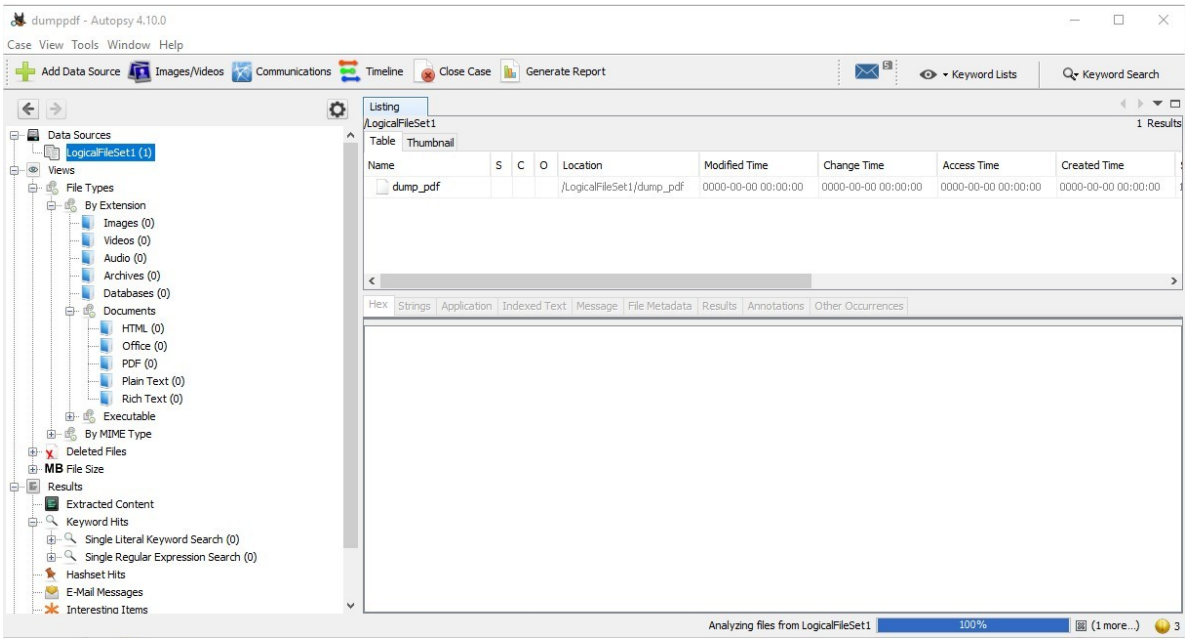
- Autres hypothèses :

  - *Hypothèse 1 :* La RAM d'un téléphone cellulaire Android contient la liste des contacts enregistrés dans le cellulaire
  - *Hypothèse 2 :* La RAM d'un téléphone cellulaire Android contient les numéros de téléphones contenus dans le cellulaire

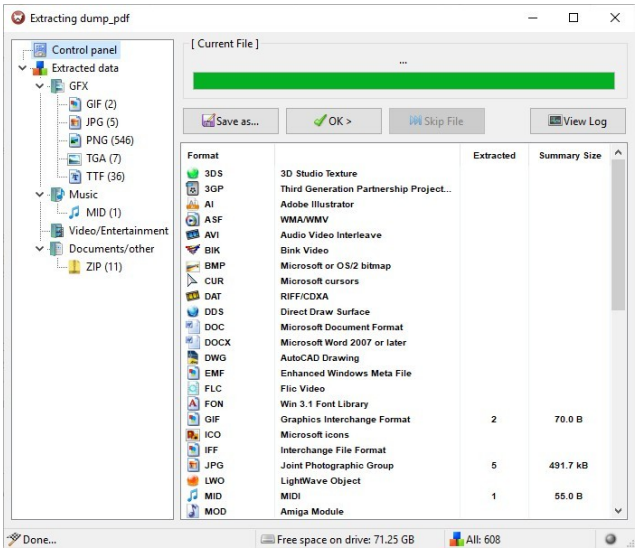
Annexe 10 : Signature (EMiL) d'un *dump* effectué avec AMExtractor



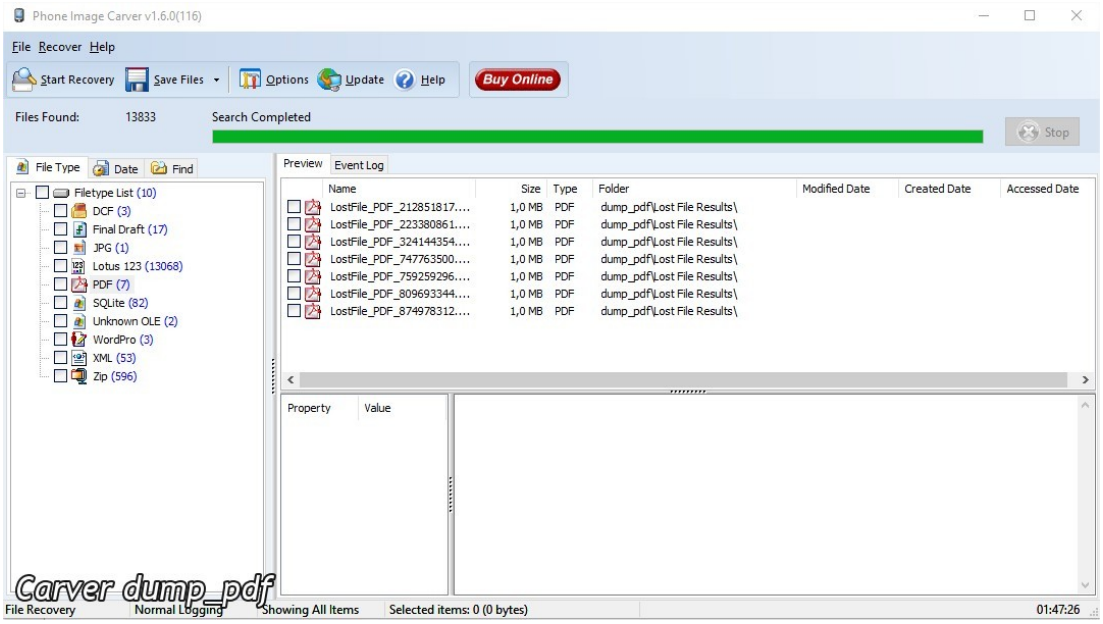
Annexe 11 : Résultats d'Autopsy – *dump_pdf*



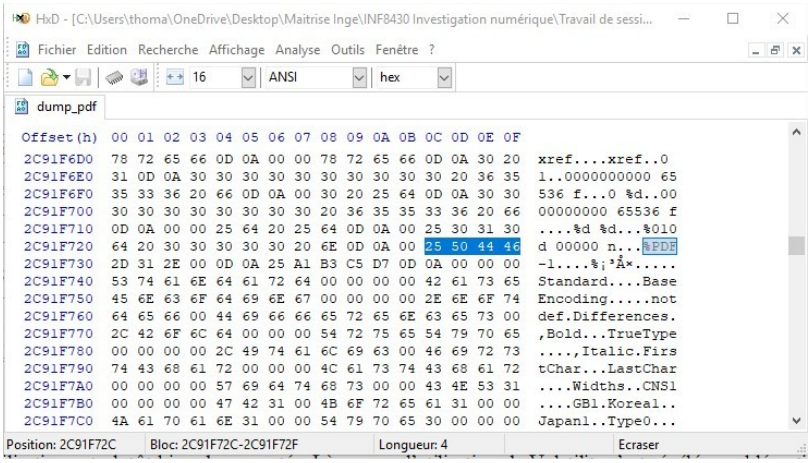
Annexe 12 : **Résultat de MultiExtractor – *dump_pdf***



Annexe 13 : **Résultat de Phone Image Carver – *dump_pdf***



Annexe 14 : **Exemple de signature de fichier pdf dans *dump_pdf***

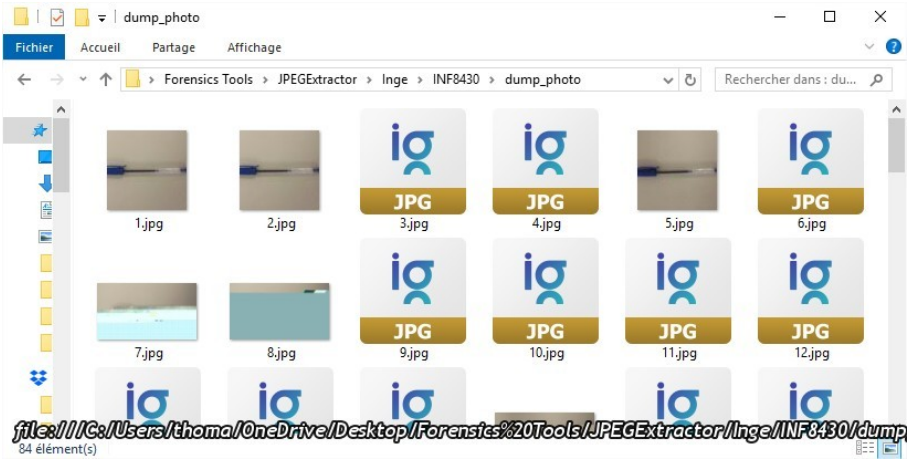


<u>Annexe 15 :</u> Résultat de JPEGExtractor – *dump_photo*

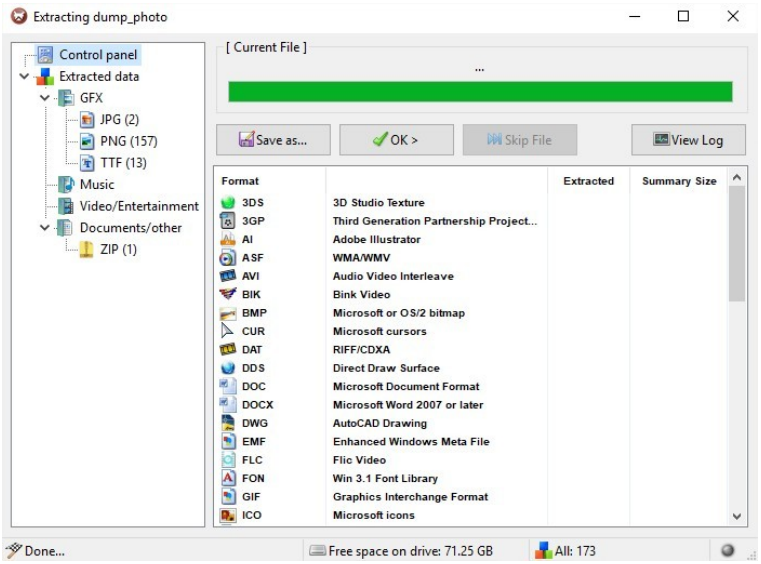
Commande :

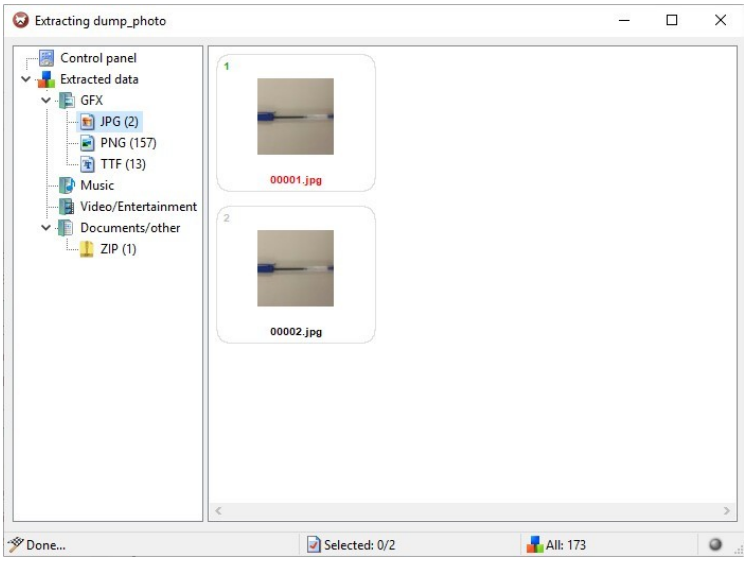
C:\Users\thoma\OneDrive\Desktop\Forensics Tools\JPEGExtractor>java -jar JPEGExtractor_1.0.jar "C:\Users\thoma\OneDrive\Desktop\Maitrise Inge\INF8430 Investigation numérique\Travail de session\dump_photo" "C:\Users\thoma\OneDrive\Desktop\Forensics Tools\JPEGExtractor"

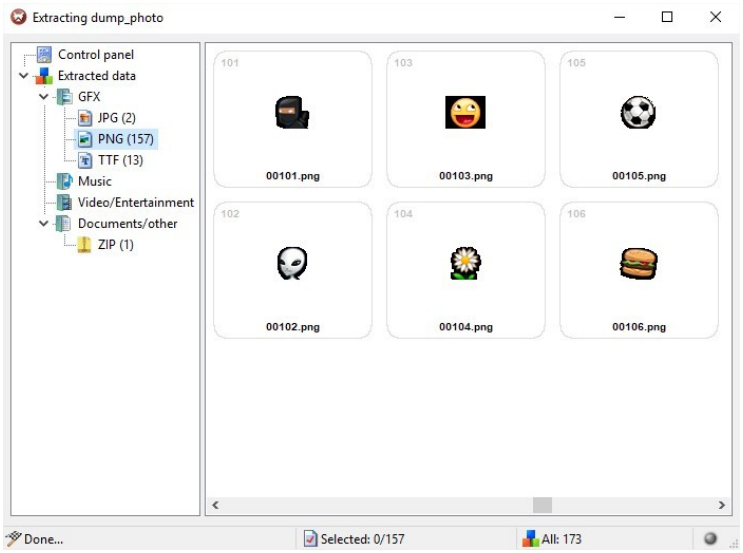
Extracted 84 images



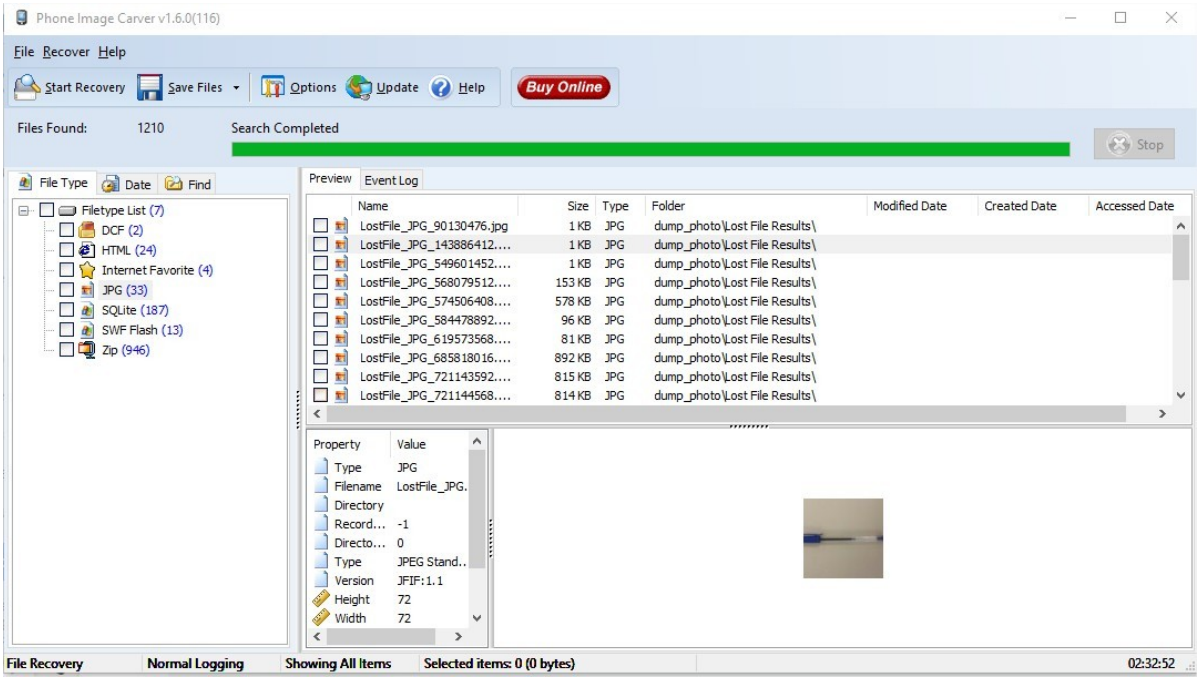
<u>Annexe 16 :</u> Résultat de MultiExtractor – *dump_photo*



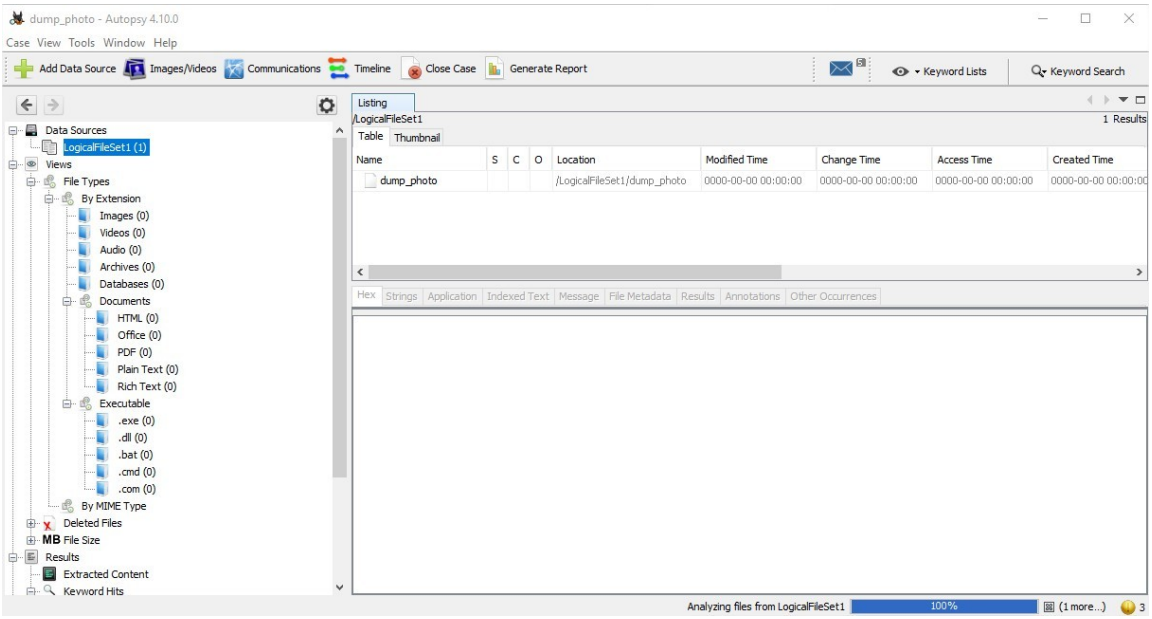




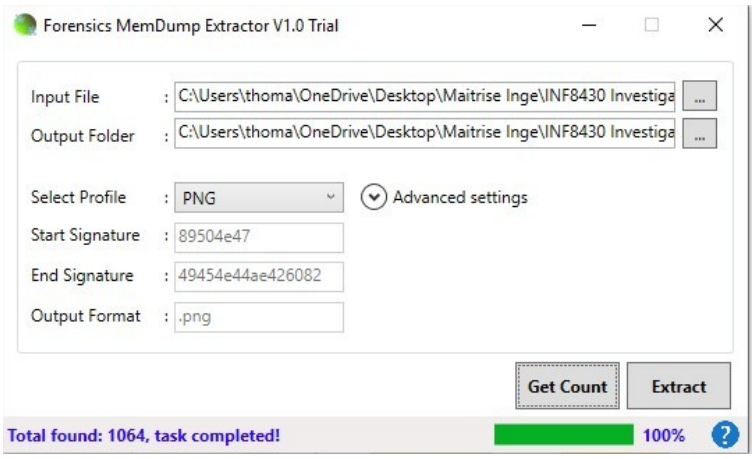
<u>Annexe 17 :</u> Résultat de Phone Image Carver – *dump_photo*



Annexe 18 : **Résultat d'Autopsy –** ***dump_photo***



Annexe 19 : **Résultats de Forensics MemDump Extractor –** ***dump_photo***



Annexe 20 : **Résultats de Volatility –** ***dump_photo***

1) Extraction des résultats de Volatility dans le fichier volatility_dump_courriel.txt – Commande : volatility linux_psxview -f /root/AMExtractor/dump_courriel --profile=LinuxGalaxyS4_jflteARM > Bureau/volatility_dump_courriel.txt

Exemple du contenu de volatility_dump_courriel.txt :

Offset(V) Name PID pslist psscan pid_hash kmem_cache parents leaders

---------- -------------------- ------ ------ ------ -------- ---------- ------- -------

0x80254000 init 1 True True False False False False

0x802543c0 kthreadd 2 True True False False True False

0x80254780 ksoftirqd/0 3 True True False False False False

0x80254b40 kworker/0:0 4 True True False False False False

…….

…….

2) Recherche de résultats probants - Commande: grep -i cam Bureau/volatility_dump_photo.txt

Résultats:

0x8a999e00 CameraHolder 6149 False True False False False False

0x974f0f00 roid.app.camera 5330 False True False False False False

0x9d74cb40 CameraControlle 1226 False True False False False False

0x9d74cf00 CameraControlle 1225 False True False False False False

0x9f753840 mm-qcamera-daem 286 False True False False False False

<u>Annexe 21 :</u> Extraction de l'image d'un processus avec Volatility – *dump_photo*

root@Host-001:~# volatility linux_psxview -f /root/AMExtractor/dump_photo --profile=LinuxGalaxyS4_jflteARM | grep cam

Volatility Foundation Volatility Framework 2.6

INFO : volatility.debug : SLUB is currently unsupported.

0x974f0f00 roid.app.camera 5330 False True False False False False

0x9f753840 mm-qcamera-daem 286 False True False False False False

root@Host-001:~# volatility linux_procdump -f /root/AMExtractor/dump_photo --profile=LinuxGalaxyS4_jflteARM -p 5330 -D /root/Bureau

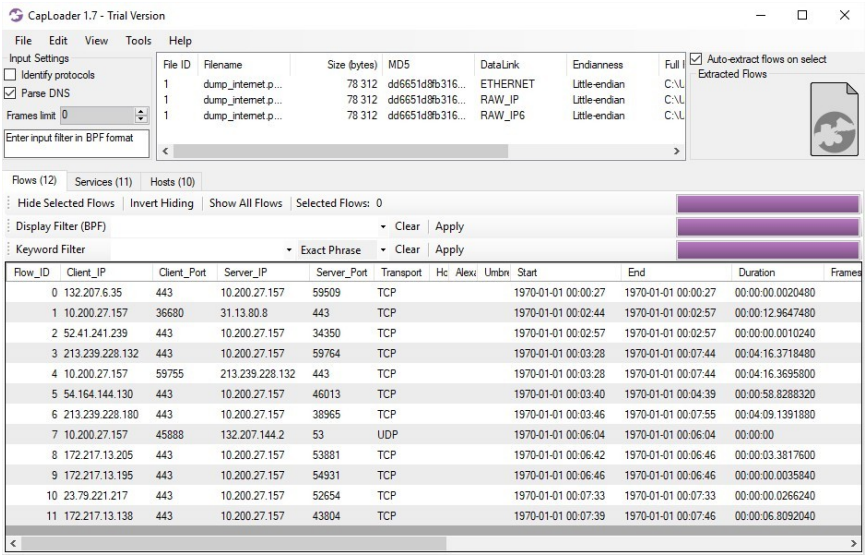
Volatility Foundation Volatility Framework 2.6

Offset Name Pid Address Output File

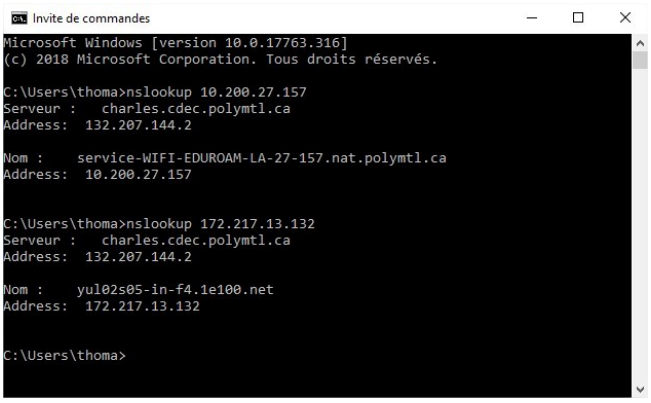
---------- -------------------- --------------- ---------- -----------

root@Host-001:~# **---> Pas d'outup = ne marche pas**

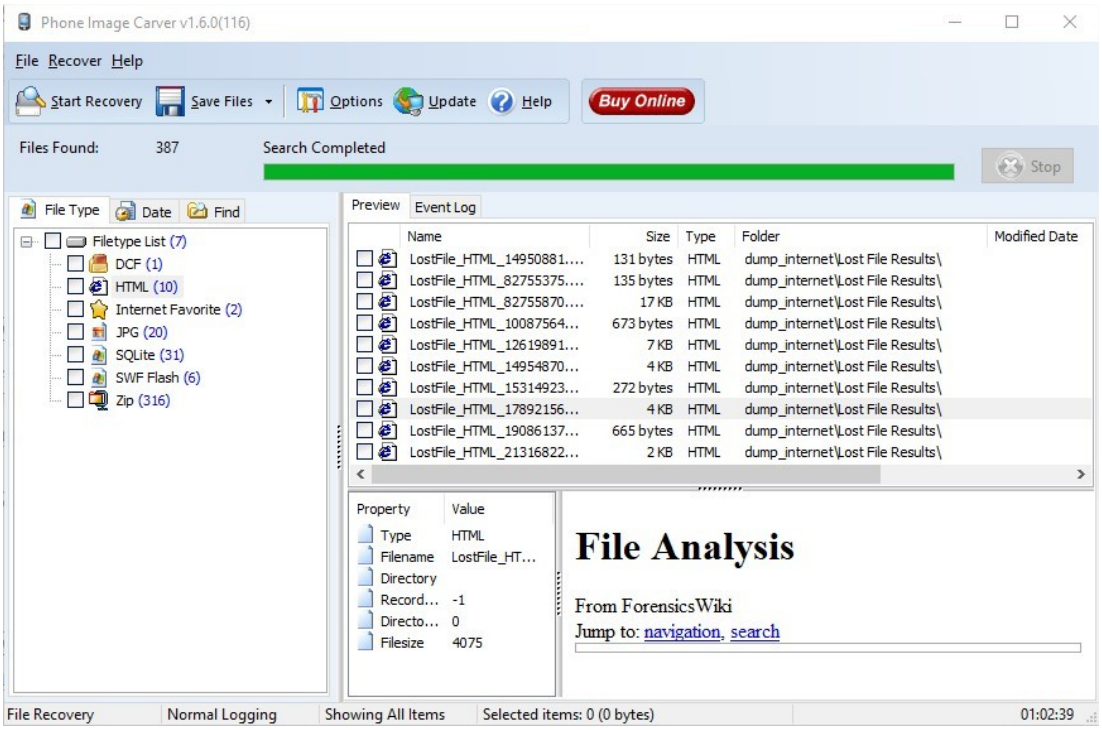
<u>Annexe 22 :</u> Résultat de Cap Loader – *dump_internet*



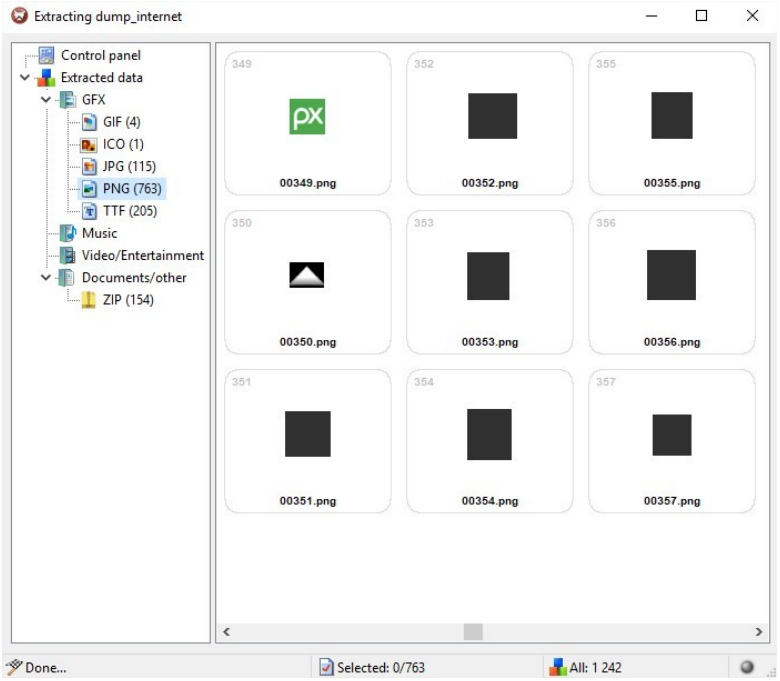
Annexe 23 : **Utilisation de nslookup pour traduire les IP en nom de domaine**

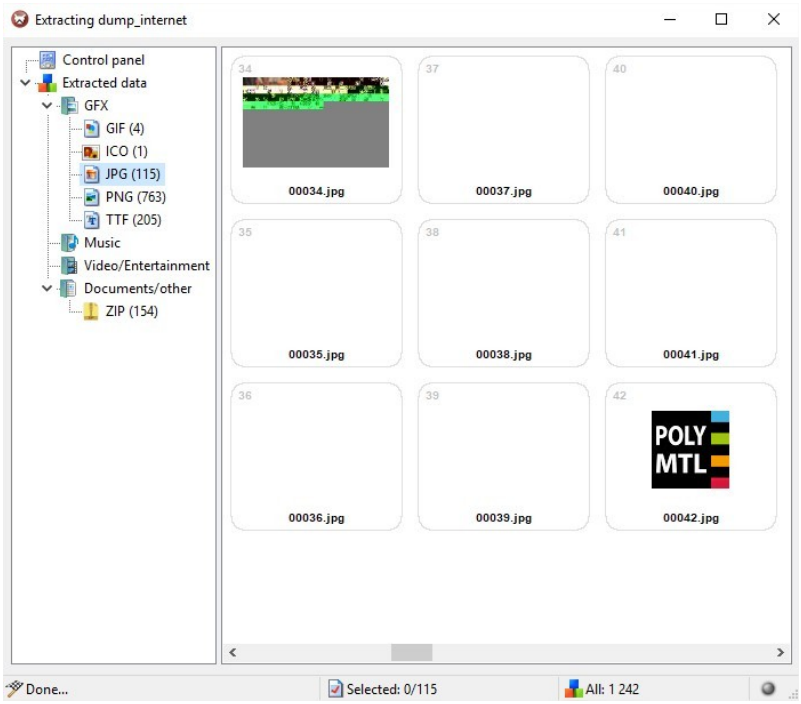


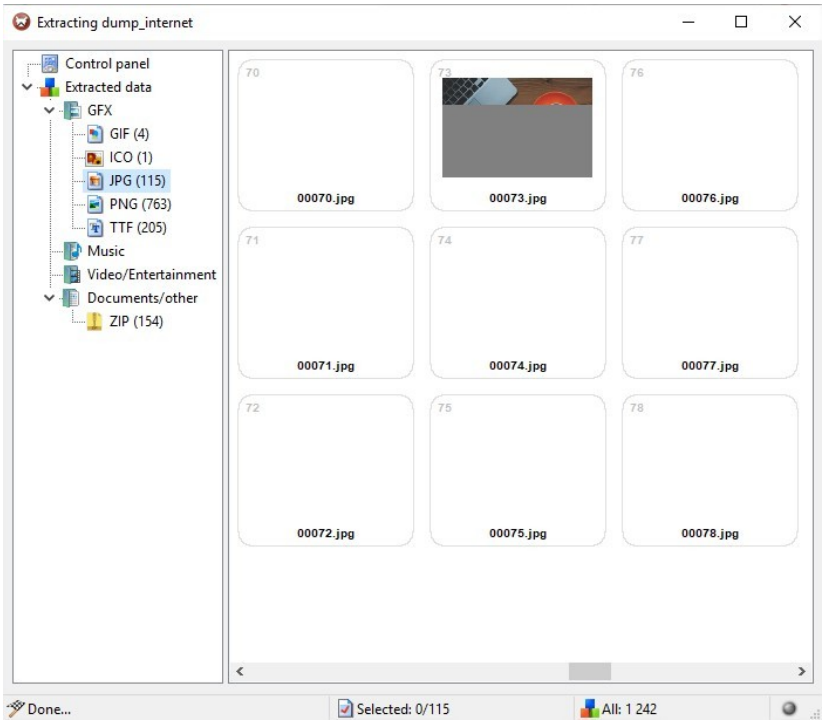
Annexe 24 : **Résultats de Phone Image Carver –** ***dump_internet***



Annexe 25 : **Résultats de MultiExtractor –** ***dump_internet***




Extracting dump_internet
Control panel
Extracted data
GFX
GIF (4)
ICO (1)
JPG (115)
PNG (763)
TTF (205)
Music
Video/Entertainment
Documents/other
ZIP (154)
34
00034.jpg
37
00037.jpg
40
00040.jpg
35
00035.jpg
38
00038.jpg
41
00041.jpg
36
00036.jpg
39
00039.jpg
42
00042.jpg
Done...
Selected: 0/115
All: 1 242

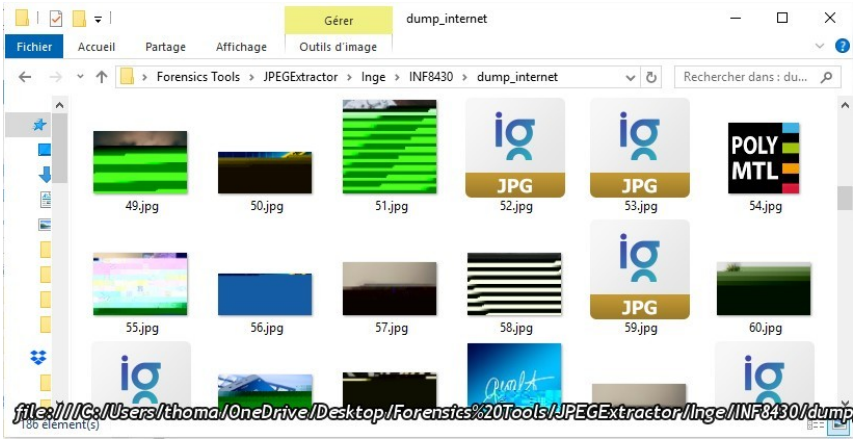

Extracting dump_internet
Control panel
Extracted data
GFX
GIF (4)
ICO (1)
JPG (115)
PNG (763)
TTF (205)
Music
Video/Entertainment
Documents/other
ZIP (154)
70
00070.jpg
73
00073.jpg
76
00076.jpg
71
00071.jpg
74
00074.jpg
77
00077.jpg
72
00072.jpg
75
00075.jpg
78
00078.jpg
Done...
Selected: 0/115
All: 1 242

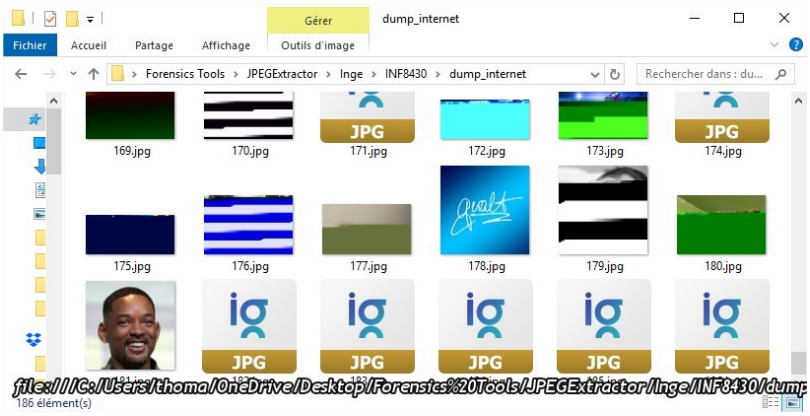
<u>Annexe 26 :</u> Résultats de JPEGExtractor - *dump_internet*

Commande :

C:\Users\thoma\OneDrive\Desktop\Forensics Tools\JPEGExtractor>java -jar JPEGExtractor_1.0.jar "C:\Users\thoma\OneDrive\Desktop\Maitrise Inge\INF8430 Investigation numérique\Travail de session\dump_internet" "C:\Users\thoma\OneDrive\Desktop\Forensics Tools\JPEGExtractor"

Extracted 186 images





<u>Annexe 27 :</u> Historique http, commandes Strings et Grep – *dump_internet*

Commande: strings /root/AMExtractor/dump_internet | grep http | grep google.com | head -n 5

https://www.google.com/js/bg/ytw6ZC4ZfmZw-ulAI3_kb2XLbaHgV_kOR6dFp4_K49Y.js
http://google.com/+Chainfire
!http://plus.google.com/+Chainfire
"icon_url": "https://www.google.com/favicon.ico",
"search_url": "https://www.google.com/search?ie=UTF-8&client=ms-android-samsung&source=android-browser&q={searchTerms}",

Commande: strings /root/AMExtractor/dump_internet | grep http | grep pixabay | head -n 5

https://pixabay.com/android-chrome-192x192.png
https://pixabay.com/android-chrome-192x192.png
https://pixabay.com/en/photos/cyber%20crime/
https://cdn.pixabay.com/photo/2016/12/21/17/39/cyber-security-1923446__480.png
Content-Location: https://pixabay.com/static/img/blank.gif

Commande: strings /root/AMExtractor/dump_internet | grep http | grep polymtl.ca | head -n 5

https://www.polymtl.ca/
http://polymtl.ca/
(...)

strings /root/AMExtractor/dump_internet | grep http | grep forensics | head -n 5

https://forensicswiki.org/images/1/1c/Information_icon.png
https://forensicswiki.org/load.php?debug=false&lang=en&modules=jquery%2Cmediawiki&only=scripts&skin=vector&version=20160831T181752Z
https://forensicswiki.org/load.php?debug=false&lang=en&modules=jquery.checkboxShiftClick%2Ccookie%2CgetAttrs%2ChighlightText%2CmakeCollapsible%2Cmw-jump%2Cplaceholder%2Csuggestions%7Cmediawiki.action.view.postEdit%7Cmediawiki.api%2Ccldr%2Ccookie%2CjqueryMsg%2Clanguage%2CsearchSuggest%2Ctemplate%2Cuser%7Cmediawiki.language.data%2Cinit%7Cmediawiki.libs.pluralruleparser%7Cmediawiki.page.ready%7Cuser.defaults&skin=vector&version=20180709T072851Z&*
https://forensicswiki.org/wiki/Main_Page
https://forensicswiki.org/wiki/Main_Page

<u>Annexe 28 :</u> Résultats de Volatility – *dump_internet*

root@Host-001:~# volatility linux_psxview -f /root/AMExtractor/dump_internet --profile=LinuxGalaxyS4_jflteARM > Bureau/volatility_dump_internet.txt

Volatility Foundation Volatility Framework 2.6

INFO : volatility.debug : SLUB is currently unsupported.

root@Host-001:~# grep -i chr Bureau/volatility_dump_internet.txt

0x89968f00 Chrome_IOThread 5861 False True False False False False

0x89e77480 Chrome_ChildIOT 5948 False True False False False False

0x8a3f3840 Chrome_FileUser 3769 False True False False False False

0x8b200000 Chrome_ChildIOT 3917 False True False False False False

0x8b323c00 ChromiumNet 6988 False True False False False False

0x8bccf840 Chrome_ProcessL 5778 False True False False False False

0x8c2d4b40 Chrome_CacheThr 3772 False True False False False False

0x8c552580 Chrome_HistoryT 5907 False True False False False False

0x8c553c00 Chrome_DevTools 6143 False True False False False False

0x8f15b840 .android.chrome 5544 False True False False False False

0x90eea580 Chrome_ChildIOT 3789 False True False False False False

0x94828000 Chrome_DevTools 3803 False True False False False False

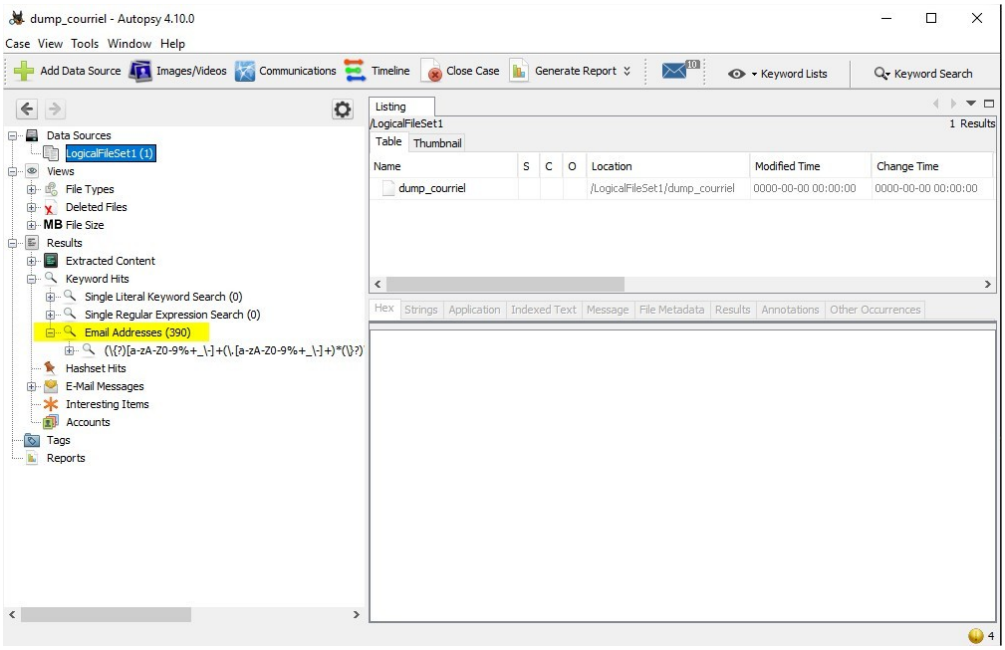
0x94850000 ChromiumNet 2662 False True False False False False

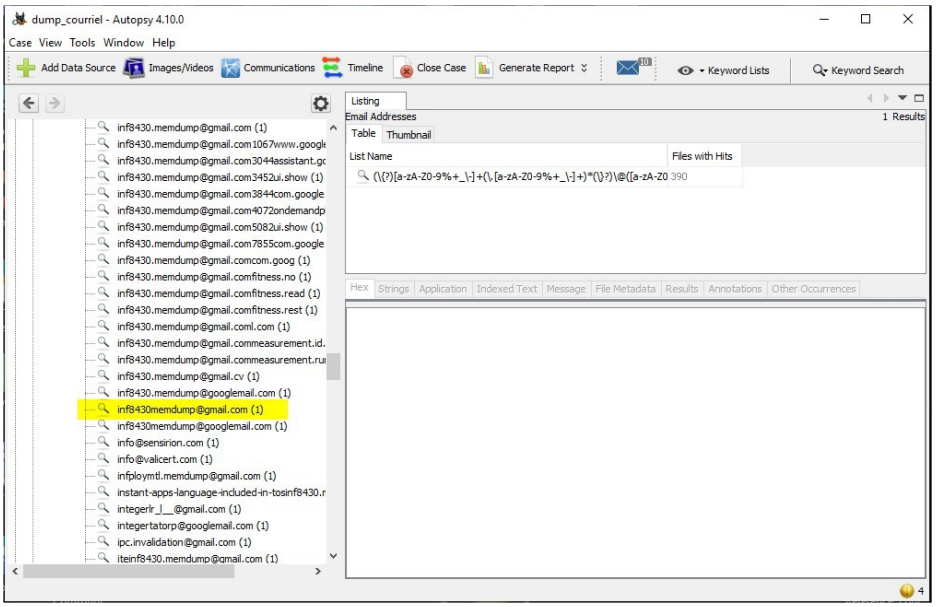
0x94853480 Chrome_ChildIOT 9335 False True False False False False

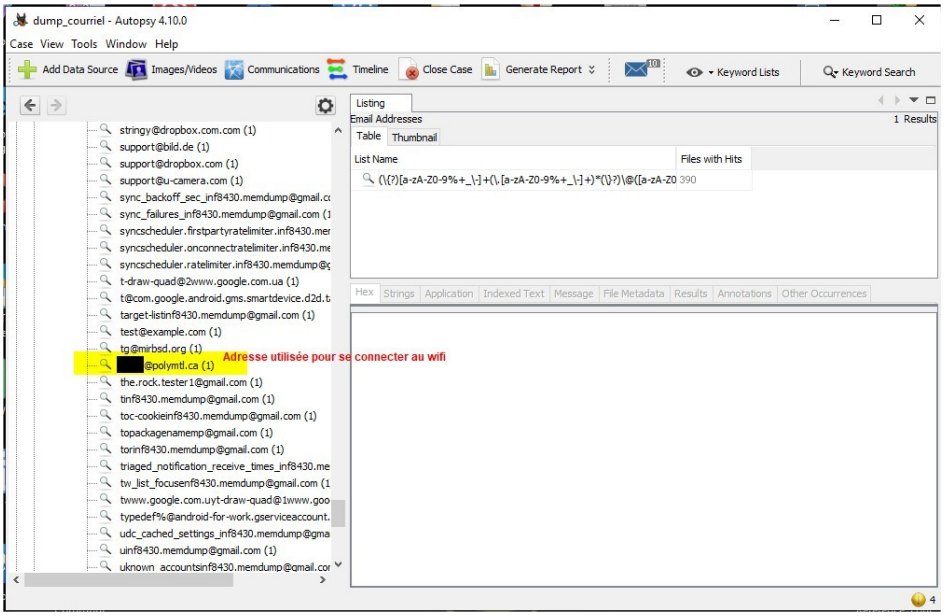
0x99132d00 Chrome_InProcGp 3788 False True False False False False

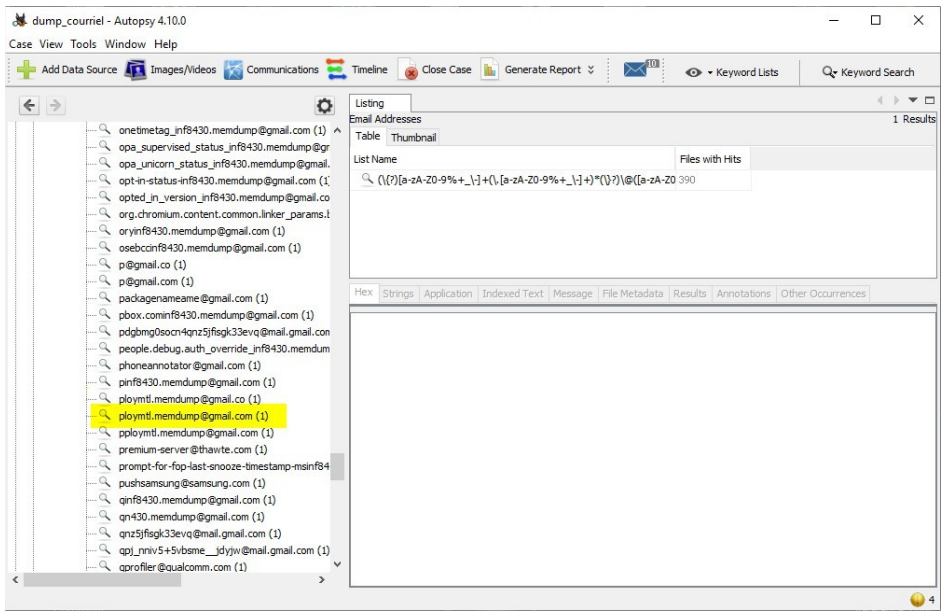
⇒ **Application Google Chrome**

<u>Annexe 29 :</u> Résultats d'Autopsy – *dump_courriel*



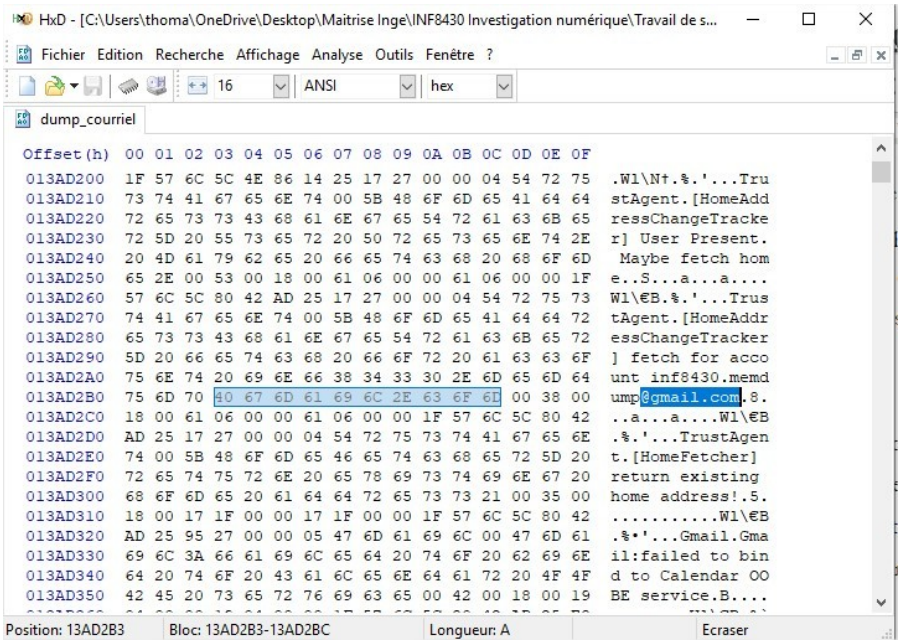

dump_courriel - Autopsy 4.10.0
Case View Tools Window Help
Add Data Source
Images/Videos
Communications
Timeline
Close Case
Generate Report
Keyword Lists
Keyword Search
Listing
Email Addresses
1 Results
Table
Thumbnail
List Name
Files with Hits
Hex
Strings
Application
Indexed Text
Message
File Metadata
Results
Annotations
Other Occurrences
inf8430.memdump@gmail.com (1)
inf8430memdump@gmail.com (1)


dump_courriel - Autopsy 4.10.0
Case View Tools Window Help
Add Data Source
Images/Videos
Communications
Timeline
Close Case
Generate Report
Keyword Lists
Keyword Search
Listing
Email Addresses
1 Results
Table
Thumbnail
List Name
Files with Hits
Hex
Strings
Application
Indexed Text
Message
File Metadata
Results
Annotations
Other Occurrences
stringy@dropbox.com.com (1)
[redacted]@polymtl.ca (1)
Adresse utilisée pour se connecter au wifi


dump_courriel - Autopsy 4.10.0
Case View Tools Window Help
Add Data Source
Images/Videos
Communications
Timeline
Close Case
Generate Report
Keyword Lists
Keyword Search
Listing
Email Addresses
1 Results
Table
Thumbnail
List Name
Files with Hits
Hex
Strings
Application
Indexed Text
Message
File Metadata
Results
Annotations
Other Occurrences
ploymtl.memdump@gmail.com (1)

Annexe 30 : Recherche de « gmail »

Avec HxD :



En utilisant les commandes Strings et Grep

Commande:strings /root/AMExtractor/dump_courriel | grep gmail

Résultats :

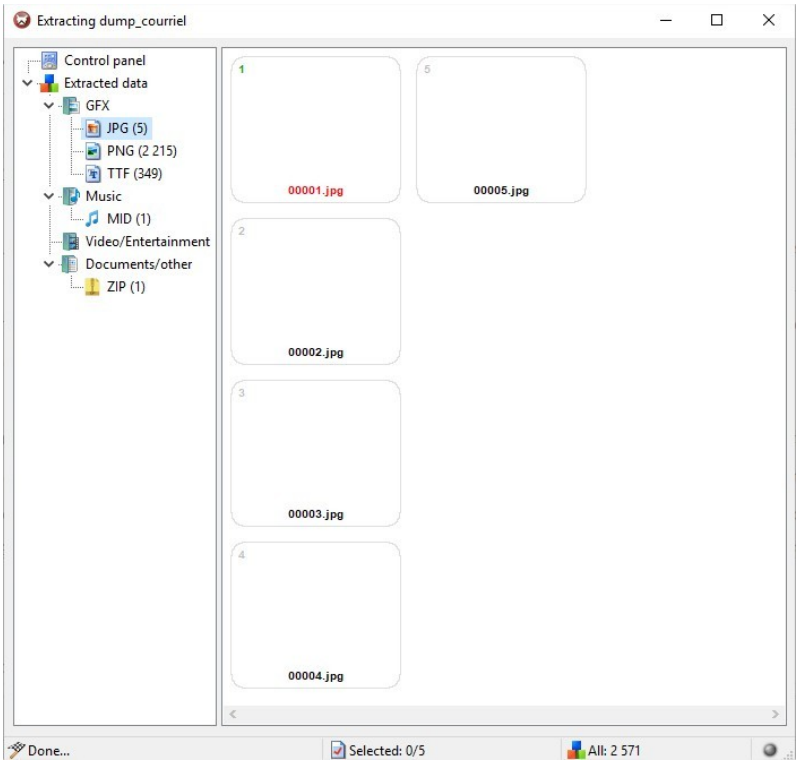
[HomeAddressChangeTracker] fetch for account inf8430.memdump@gmail.com
com.google.android.gms.auth.TokenNotificationManager:inf8430.memdump@gmail.com:com.google
com.google.android.gms.auth.TokenNotificationManager:inf8430.memdump@gmail.com:com.google
content://com.google.android.gm.sapi/inf8430.memdump@gmail.com/message/#thread-a:mmiai-r5629959185229181412/
(...)
<Unknown com.google.apps.bigtop.services.gmail.PersonalLevel.
Photo de styloWow ton stylo irait parfaitement pour prendre des notes lors d'une investigation[null,"ploymtl.memdump@gmail.com","inf inf",null][[null,"inf8430.memdump@gmail.com","John Doe",null]][][][]
inf8430.memdump@gmail.com
ploymtl.memdump@gmail.com

 On Tue, Feb 19, 2019, 20:18 inf inf <ploymtl.memdump@gmail.com wrote:
 <blockquote class="gmail_quote" style="margin:0 0 0 .8ex;border-left:1px #ccc solid;padding-left:1ex;"> <div style="white-space:pre-wrap;font-size:13.0758px;font-family:sans-

serif"> Wow ton stylo irait parfaitement pour prendre des notes lors d'une investigation
D<CAFpM-w=ds_PTB43iyun2uYC3dgBmg0SocN4QNz5JfiSgK33EVQ@mail.gmail.com>
inf8430.memdump@gmail.com
ploymtl.memdump@gmail.com

 On Tue, Feb 19, 2019, 20:18 inf inf <ploymtl.memdump@gmail.com wrote:
 <blockquote class="gmail_quote" style="margin:0 0 0 .8ex;border-left:1px #ccc solid;padding-left:1ex;"> Wow ton stylo irait parfaitement pour prendre des notes lors d'une investigation
D<CAFpM-w=ds_PTB43iyun2uYC3dgBmg0SocN4QNz5JfiSgK33EVQ@mail.gmail.com>
inf8430.memdump@gmail.com
……..
…….

<u>Annexe 31 :</u> Résultats de MultiExtractor – *dump_courriel*

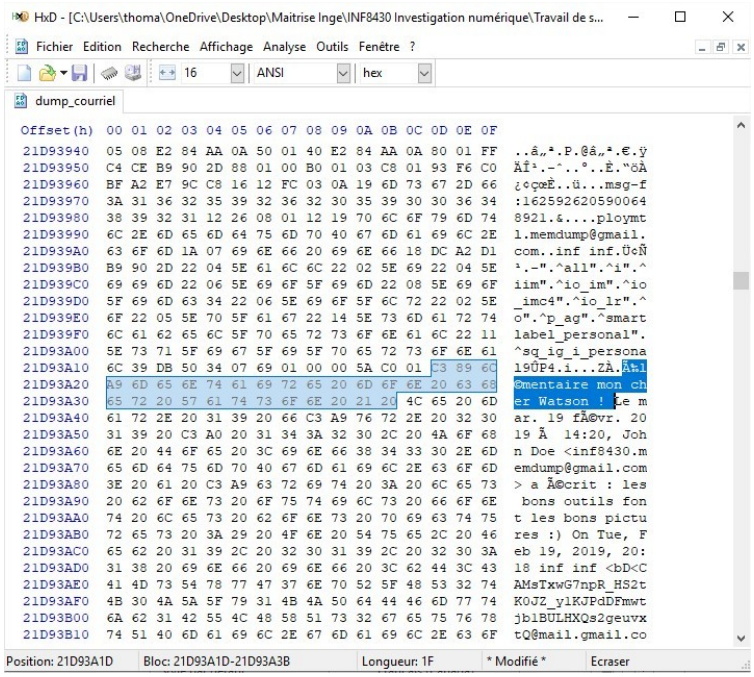


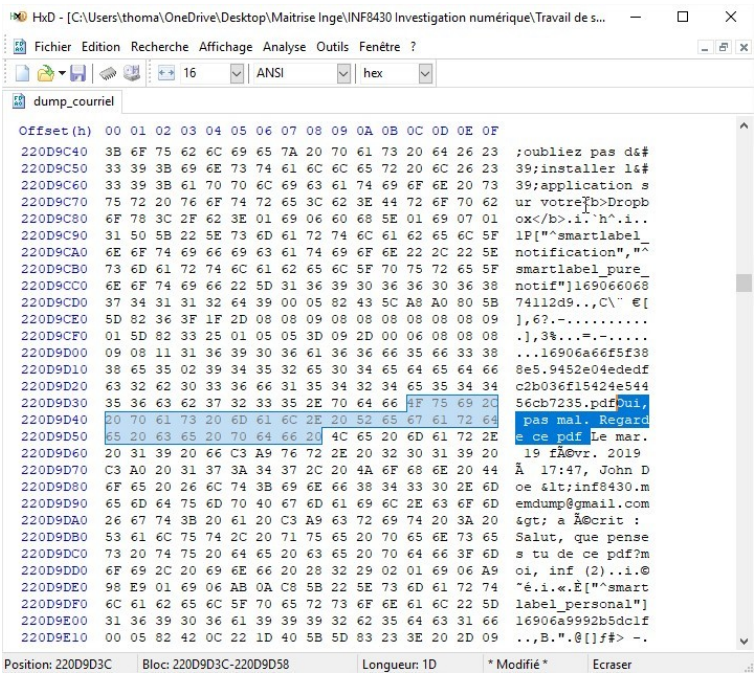
<u>Annexe 32 :</u> Résultats de JPEGExtractor - *dump_courriel*

Commande :
C:\Users\thoma\OneDrive\Desktop\Forensics Tools\JPEGExtractor>java -jar JPEGExtractor_1.0.jar "C:\Users\thoma\OneDrive\Desktop\Maitrise Inge\INF8430 Investigation numérique\Travail de session\dump_courriel" "C:\Users\thoma\OneDrive\Desktop\Forensics Tools\JPEGExtractor"

Extracted 84 images

Annexe 33 : Recherche du contenu des courriels





<u>Annexe 34 :</u> Résultats de Volatility - *dump_courriel*

Commande: volatility linux_psxview -f /root/AMExtractor/dump_courriel --profile=LinuxGalaxyS4_jflteARM > Bureau/volatility_dump_courriel.txt

Résultats intéressants: grep -i chr Bureau/volatility_dump_courriel.txt

0x81787c00 Chrome_ChildIOT 11175 False True False False False False
0x8499f840 Chrome_FileThre 10484 False True False False False False
0x85725a40 Chrome_DBThread 10969 False True False False False False
0x85726580 Chrome_FileUser 10971 False True False False False False
0x85726d00 Chrome_FileThre 10970 False True False False False False
0x85727480 Chrome_ProcessL 10972 False True False False False False
0x862061c0 Chrome_IOThread 3887 False True False False False False
0x86207c00 Chrome_DBThread 10483 False True False False False False
0x8a5acf00 Chrome_CacheThr 10487 False True False False False False
0x8b4ec780 Chrome_DevTools 4249 False True False False False False
0x8f265e00 Chrome_HistoryT 3916 False True False False False False
0x8f268f00 Chrome_IOThread 10488 False True False False False False
0x93984f00 Chrome_FileUser 10485 False True False False False False
0x97d84780 Chrome_ChildIOT 4225 False True False False False False
0x9b4212c0 .android.chrome 3447 False True False False False False
0x9b423c00 Chrome_ChildIOT 10957 False True False False False False
0x9bb66d00 Chrome_CacheThr 10973 False True False False False False
0x9be74f00 ChromiumNet 2600 False True False False False False
0x9c64ad00 Chrome_ProcessL 10486 False True False False False False
0x9cf71680 Chrome_IOThread 10974 False True False False False False
0x9d029e00 Chrome_ProcessL 3848 False True False False False False
0x9d206d00 Chrome_ChildIOT 3982 False True False False False False
-> Google Chrome est ouvert

0x82044780 ogle.android.gm 10934 False True False False False False
-> application Gmail (voir résultat BinWalk)

<u>Annexe 35 :</u> Résultats de BinWalk – *dump_courriel*

Commande: grep inf8430 Bureau/binwalk_dump_courriel.txt

Résultats:

129110080 0x7B21040 Unix path: /data/data/
com.google.android.gm/shared_prefs/com.google.android.apps.gmail.accountDumpState:inf8430.memdump@gmail.com:com.googl
368427168 0x15F5C0A0 Unix path:
/data/data/com.google.android.gm/shared_prefs/com.google.android.apps.gmail.accountDumpState:inf8430.memdump@gmail.com:com.googl
374569366 0x16537996 Unix path:
/www.google.com/m8/feeds/groups/inf8430.memdump@gmail.com/base2_property-android
381221376 0x16B8FA00 Unix path: /data/data/com.android.vending/cache/logs/com.google.inf8430.memdump%40gmail.com.metalog/logs_upload_attempt.log
582119744 0x22B27140 Unix path: /data/data/com.google.android.gm/shared_prefs/Account-inf8430.memdump@gmail.com.xml.bak
595358695 0x237C73E7 Unix path:
/www.google.com/m8/feeds/groups/inf8430.memdump@gmail.com/base2_property-android
622259504 0x2516ED30 Unix path:
/data/data/com.google.android.gms/databases/context_inf8430.memdump_gmail.com.db
803369376 0x2FE271A0 Unix path:
/data/data/com.google.android.gm/shared_prefs/com.google.android.apps.gmail.notifications:inf8430.memdump@gmail.com.xml
1260165992 0x4B1C9B68 Unix path:
/data/data/com.google.android.gms/databases/context_inf8430.memdump_gmail.com.db
1391605096 0x52F23568 Unix path:
/data/data/com.google.android.gms/databases/fitness.db.inf8430.memdump_gmail.com
1442341280 0x55F861A0 Unix path:
/data/data/com.google.android.music/cache/logs:main/com.google.inf8430.memdump%40gmail.com.metalog/logs_upload_attempt.log
1557401824 0x5CD410E0 Unix path: /data/data/com.google.android.gm/shared_prefs/Account-inf8430.memdump@gmail.com.xml
1654844590 0x62A2ECAE Unix path:
/www.google.com/m8/feeds/groups/inf8430.memdump%40gmail.com/base2_property-android/65f1930908666c1b"Rn0yezVSLi17ImA9XBNTEkkIQgE.

Commande: grep JPEG Bureau/binwalk_dump_courriel.txt
Résultat:
186933364 0xB246074 JPEG image data, JFIF standard 1.01
218544742 0xD06BA66 JPEG image data, JFIF standard 1.01
332580075 0x13D2C4EB JPEG image data, JFIF standard 1.01
387836448 0x171DEA20 JPEG image data, JFIF standard 1.01
522212234 0x1F20538A JPEG image data, JFIF standard 1.01
550090226 0x20C9B5F2 JPEG image data, JFIF standard 1.01

597818022 0x23A1FAA6 JPEG image data, JFIF standard 1.01

628203732 0x2571A0D4 JPEG image data, JFIF standard 1.01

658384211 0x273E2553 JPEG image data, JFIF standard 1.01

……

……...

Annexe 36 : Résultats de Volatility – *dump_facebook*

Résultats:

--> Application Facebook (katana)

0x87420000 facebook.katana 15689 False True False False False False

0x8be71680 facebook.katana 16919 False True False False False False

0x99e20000 facebook.katana 5598 False True False False False False

--> Application Messenger (orca)

0x9312b0c0 m.facebook.orca 10237 False True False False False False

0x94b9f840 m.facebook.orca 16090 False True False False False False

Annexe 37 : Résultats de BinWalk - *dump_facebook*

Comande: grep -i messenger Bureau/binwalk_dump_facebook.txt | head -n 5

Résultats:

103537664 0x62BDC00 Unix path: /www.facebook.com/android/messenger/upgrade?app_referrer=app_version_error">www.facebook.com/android/messenger/upgrade

129851473 0x7BD6051 Unix path: /facebook/messenger/neue/MessengerHomeFragmentView;

173351039 0xA55207F Unix path: /facebook/messaging/registration/fragment/MessengerBackedUpAccountRecoveryViewGroup;

182845543 0xAE60067 Unix path: /facebook/messaging/neue/nux/messenger/NeueNuxContactImportFragment;

188621390 0xB3E224E Unix path: /www.facebook.com/android/messenger/upgrade?app_referrer=app_version_error">www.facebook.com/android/messenger/upgrade

Annexe 38 : Recherche dans l'historique des connexions http

Commande: strings /root/AMExtractor/dump_facebook | grep http | grep facebook > grep_http_dump_facebook.txt

Résultats:

NetworkDrawable with id: 2132282534
(https://lookaside.facebook.com/redrawable/GBRPoQKUXQUF1_oBAAAAAADwebkybnsvAAAD/) wasn't hidden before
(...)
:https://b-www.facebook.com/mobile/**orca_android**_crash_logs/
-https://graph.facebook.com/messenger_recovery
;https://www.facebook.com/help/messenger-app/522894144496549
["^/report.*","^/follow/feedsources.*","^/ads/preference.*","^/settings.*","^/help/android-app.*","^/\\d.*/allactivity.*","^/privacy.*","^/about/privacy.*","^/policies.*","^/about/basics/.*","https://m.facebook.com/help/contact/.*","^/terms.*","^/policy.*","^/trust.*","^/communitystandards.*","^/ad_guidelines.*","^/page_guidelines.*","^/payments_terms.*","^/help.*","^/pages/create.*","https://m.facebook.com/groups/create/.*","^/invite/history.*","https://(www|m).facebook.com/safetycheck.*","^(https://m.facebook.com)?/zero/toggle/settings($|\\?.*$|/.*$)","https://(www|m).facebook.com/events/birthdays.*","https://m.facebook.com/.*/about.*","https://m.facebook.com/timeline/app_section/.*","^/allactivity/options\\?id=.*","^/survey.*","^https://m.facebook.com/a/approval_queue/.*","^/legal/thirdpartynotices","^/deactivatewhitelist/.*","^/upsell/loyaltytopup/accept/.*"]
a["https://0.freebasics.com/?ref=android_bookmark", "https://free.facebook.com/zero/optin/legal/"]
",https://www.facebook.com/FacebookAndroidBeta2z
: www.facebook.com/android/messenger/upgrade
: <a href="http://www.facebook.com/android/messenger/upgrade?

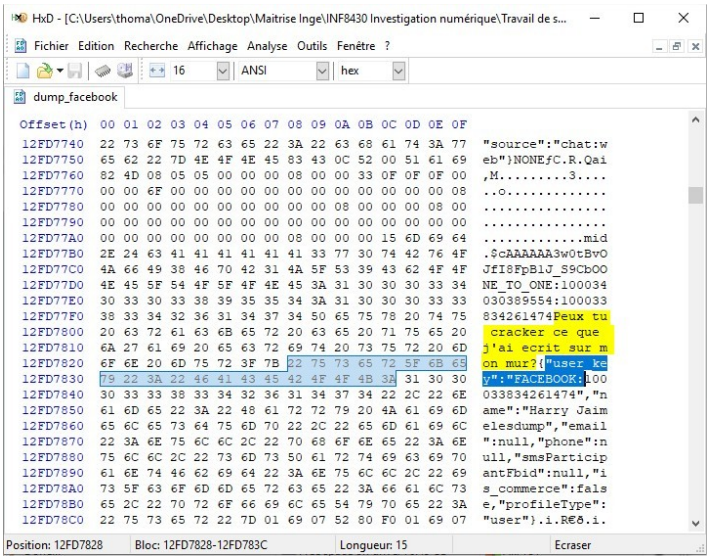
<u>Annexe 39 :</u> Recherche de message écrit sur un mur Facebook

Commande:strings /root/AMExtractor/dump_facebook | grep SSBsb > Bureau/strings_base64_mur_dump_facebook.txt

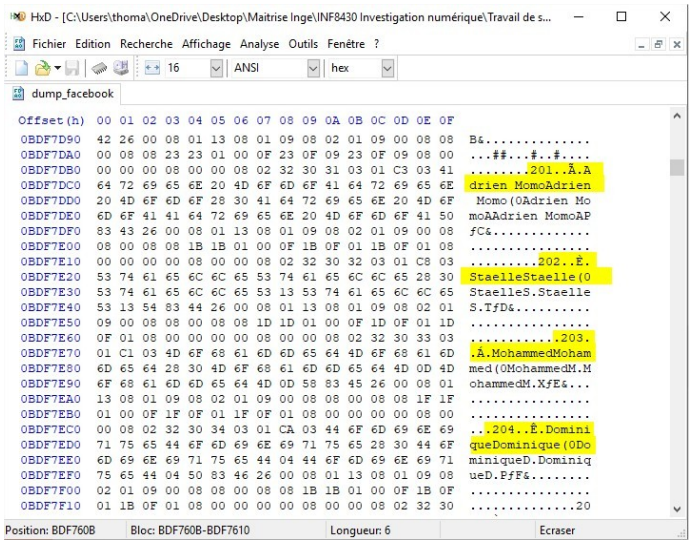
Résultats:

(...)
ALL_PREVIEW\":\"{\\\"width\\\":219,\\\"height\\\":390,\\\"src\\\":\\\"https://scontent.xx.fbcdn.net/v/wl/t1.6435-9/cp0/e15/q65/s780x390/52520530_114644499673358_5879549467874557952_n.jpg?_nc_cat=111&_nc_ad=z-m&_nc_cid=0&_nc_zor=9&_nc_ht=scontent.xx&oh=162eba9dc5fea6c60522aa4f4357887c&oe=5CEF1DBE\\\"}\"}","image_data_source":1,"render_as_sticker":false,"mini_preview":null}]**SSBsb3ZlIGZvcmVuc2ljcw==**
SSBsb3ZlIGZvcmVuc2ljcw==
SSBsb3ZlIGZvcmVuc2ljcw==
tps://scontent.xx.fbcdn.net/v/wl/t1.6435-9/cp0/e15/q65/s1050x525/52520530_114644499673358_5879549467874557952_n.jpg?_nc_cat=111&_nc_ad=z-
…….
…….

Annexe 40 : **Recherche des messages envoyés/reçus par Messenger**



Annexe 41 : **Recherche de la liste de contacts de l'appareil**



<u>Annexe 42 :</u> Tableau récapitulatif des résultats de la recherche :

		Éléments recherchés	Trouvé	Outils utilisés permettant de trouver l'artefact	Remarque
1	*dump_photo*	Utilisation de l'appareil photo du cellulaire	**Trouvé**	Volatility	
2	*dump_photo*	Photo (stylo) format JPEG	**Totalement**	Phone Image Carver; MultiExtractor; JPEGExtractor	
3	*dump_internet*	Utilisation du navigateur Google Chrome	**Totalement**	Volatility; Binwalk (+Grep)	
4	*dump_internet*	« Historique » de navigation (pages http/https)	**Totalement**	Commande Strings et Grep (Linux)	
5	*dump_internet*	Photo téléchargée (« cyber crime ») format JPEG	**partiellement**	JPEGExtractor	Logo de Polytechnique + photo de Will Smith…
6	*dump_internet*	Contenu des pages (html) consultées	**partiellement**	Phone Image Carver	
7	*dump_courriel*	Utilisation de l'application Gmail	**Trouvé**	Volatility; Binwalk (+Grep)	
8	*dump_courriel*	Adresses courriel	**Totalement**	Autopsy; commande Strings (Linux); expressions régulières	Adresse courriel utilisée pour connecter le cellulaire au wifi
9	*dump_courriel*	Contenu des courriels	**Trouvé**	Commande Strings et Grep (Linux)	
10	*dump_courriel*	« Historique » de courriels (courriels passés)	**Trouvé**	Commande Strings (Linux)	
11	*dump_courriel*	Photo envoyée (JPEG) en pièce jointe d'un courriel	**Trouvé**	Binwalk (+Grep); JPEGExtractor	
12	*dump_facebook*	Utilisation de l'application Facebook	**Trouvé**	Volatility; Binwalk (+Grep)	
13	*dump_facebook*	Utilisation de l'application Messenger	**Trouvé**	Volatility; Binwalk (+Grep)	
14	*dump_facebook*	Voir ce qui a été écrit sur un mur Facebook	**Trouvé**	Commande Strings (Linux)	
15	*dump_facebook*	Ce qui a été envoyé via Messenger	**Trouvé**	Commande Strings (Linux)	
16		Autres photos (format PNG)	**Totalement**	Phone Image Carver; MultiExtractor; Forensics MemDump Extractor	Beaucoup d'icônes présentes sur le cellulaire
17		Adresses IP	**Totalement**	CapLoader	
18		Liste des contacts téléphoniques	**Totalement**	Commande Strings (Linux)	
19	*dump_pdf*	pdf téléchargé / téléversé	**Non trouvé**	Tous les outils!!	Présence de la signature de fichiers pdf mais extraction infructueuse
20					
21		**Trouvé mais non exploité**			
22		Bases de données SQLite			
23		Pièces jointes aux courriels (base64)			