République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

جامعة بجاية
Tasdawit n Bgayet
Université de Béjaïa

Université Abderrahmane MIRA - Béjaia
Faculté des Sciences Exactes
Département d'Informatique

Mémoire de fin d'études

Pour l'obtention du diplôme de master en Informatique

Option : Administration et Sécurité des Réseaux

Une solution cloud open source pour Network Forensic

Réalisé par :
M. CHAABNA Youva
M. ASSEF Bilel

Encadré par :
M. AKILAL Abdellah
(U.A/MIRA,Béjaia)

Soutenu le 07 Juillet 2022, Devant le jury composé de :

M. MOKTEFI Mohand : U.A/MIRA,Béjaia - Examinateur
Mme. KESSIRA Dalila : U.A/MIRA,Béjaia - Examinatrice

Promotion : 2021/2022

Dédicaces

“

À NOS CHERS PARENTS, nos frères, nos soeurs et nos familles

À nos amis,

À tous ceux qui ne sont plus-la, qu'Allah les accueil dans son vaste paradis,
À tous ceux qui nous sont chers, à vous tous

Merci.

”

- M.CHAABNA Youva
- M.ASSEF Bilel

Remerciements

Tout d'abord, je remercie **Allah** le tout puissant de nous avoir donné le courage, la santé et la patience nécessaires à mener ce travail à son terme.

En premier lieu, nous tenons à remercier notre encadrant, **M. AKILAL Abdellah** pour sa disponibilité, sa patience et son précieux suivi tout au long de la réalisation de ce travail.

Nous tenons également à remercier les membres du jury à savoir **M. MOKTEFI Mohand** et **Mme. KESSIRA Dalila** pour avoir accepter de faire partie du jury de notre mémoire.

Un grand merci à toute nos famille, et à tous nos amis.

Enfin, nous tenons à remercier à toutes les personnes qui ont participé à ce projet de prés ou de loin afin de réaliser ce travail dans les meilleures conditions possibles.

Table des matières

Table des figures

Liste des acronymes

ADFM	*Abstract Digital Forensic Model*
AKS	*Azure Kloud Service*
AWS	*Amazon Web Services*
BSD	*Berkeley Software Distribution*
DF	*Digital Forensic*
DFRWS	*Digital Forensic Research WorkShop*
DDOS	*Distributed Denial-of-Service*
DNS	*Domain Name System*
eBPF	*Extended Berkeley Packet Filter*
EKS	*Elastic Kubernetes Service*
ESI	*Electronically Stored Information*
FRCP	*Federal Rules of Civil Procedure*
FTK	*Forensic Toolkit*
HTTP	*Hypertext Transfer Protocol*
IaaS	*Infrastructure as a Service*
IBM	*International Business Machines*
IDS	*Intrusion detection System*
IKS	*IBM Cloud Kubernetes Service*

IP *Internet Protocol*

IT *Information Technology*

KVM *Kernel-based Virtual Machine*

K8s *Kubernetes*

LXC *Linux Container*

NIST *National Institute of Standards and Technology*

OS *Operating System*

OSI *Open Systems Interconnection*

PaaS *Platform as a Service*

RKE *Rancher Kubernetes Engine*

SaaS *Software as a Service*

VM *Virtual Machine*

Introduction générale

Aujourd'hui, la majorité des foyers possèdent un accès à internet et il est possible de se connecter avec le bout du monde depuis n'importe où, n'importe quand. Malgré cette avancée technologique, des problèmes surgissent en ce qui concerne la sécurité des données et de la vie privée. Le fait que les données sont stockées sous format numérique et circulent sur internet, donne une opportunité aux criminels d'exploiter certaines vulnérabilités dans le cas d'une brèche de sécurité,ou cybercrime ; Ainsi il est primordial que les enquêteurs réalisent une analyse correcte et fiable de l'incident. Pour cela, plusieurs entités et organismes d'application de la loi ont commencé à développer des outils et processus à adopter dans les enquêtes numériques.

Même les nouvelles technologies sont concernés par ce problème de cybersécurité, alors que le monde de la technologie d'information prend chaque jour des contre-mesures pour maximiser le niveau de confidentialité des données. L'une des ces technologies est le Cloud Computing qui connaît un véritable essor. Le Cloud Computing est est une technologie permettant d'accéder à des services informatiques (logiciels, données, …), à distance et à la demande, via une connexion internet sécurisée. Le Cloud est omniprésent dans notre quotidien, que cela soit sur notre lieu de travail ou dans nos activités personnelles (sauvegarder des fichiers sur Google Drive ou Dropbox, regarder des séries sur Netflix ou HBO …etc).

Tout ce qui devient populaire dans le monde numérique deviendra inévitablement la cible de cyber acteurs malveillants, et les plates-formes de cloud computing ne font pas exception. En 2020, les attaques ciblant le cloud représentaient 20% des cyberattaques de l'année, le troisième environnement le plus ciblé [9]. L'une des attaques les plus marquantes est La violation de données AIS (Automatic Identification System) découverte par le chercheur en cybersécurité Justin Paine lors de la navigation sur BinaryEdge et Shodan. Selon Paine, la base de données divulguée comprenait 8,3 milliards de journaux de flux réseau et de journaux de requêtes DNS des clients AWN de la société de télécommunications basée en Thaïlande [9].

Cela donne une nécessité de mettre en place des méthodes et outils pour répondre aux incidents, ce genre de conduite est réalisée grâce au Network Forensic qui vise à apporter les preuves numériques extraites après le déclenchement d'une alerte de sécurité auprès des tribunaux de justice.

Dans notre projet, nous allons répondre à une problématique bien précise, et qui est de savoir s'il est possible de faire du network forensics dans les environnements de cloud computing. Plus précisément, les environnements Docker et Kubernetes.

Ce mémoire comprend quatre chapitres :

Le premier chapitre introduit la science de “Digital Forensic”, allant des définitions, aux modèles d’investigations, tout en introduisant des concepts tels que “Forensic” et “Digital evidence”.

Le deuxième chapitre présente le “Network Forensic”, allant des définitions a les outils d’investigation .

Le troisième chapitre est consacré au Cloud Computing où nous expliquerons cette nouvelle technologie et ses modèles qui lui sont associés, puis nous définirons la notion de virtualisation .

Le quatrième chapitre sera dédié pour résoudre notre problématique qui s’agit de faire du Network Forensic dans l’environnement cloud .

Enfin, ce mémoire est conclu par un résumé de tout ce qui y a été décrit, et des perspectives futures pour ce projet.

Chapitre 1

Digital Forensics

1.1 Introduction

Dans ce premier chapitre, nous allons introduire la science de l'investigation numérique (Digital Forensics). D'abord, nous commencerons par une introduction, suivie par un historique et définitions de quelques concepts de base associés à cette science. Après, nous citerons quelques modèles de déroulement des enquêtes digitales. Pour finir, nous avancerons quelques outils open source qu'utilise un investigateur numérique.

1.2 Historique et Définitions

Ce point du chapitre nous permettra d'avoir un premier contact avec le digital forensics. Premièrement, un bref historique donnera un résumé de son commencement et croissance au fil du temps, ensuite nous nous focaliserons sur quelques-unes de ses définitions .

Il faut essayer de penser au scénario suivant pour bien comprendre ce que c'est que l'investigation numérique légale :

La police est alertée pour un crime, une fois arrivé sur place, il remarque qu'une partie de la scène de crime comprend des appareils numériques et des médias (CD, DVD, clé USB … etc.). Quelques questions se posent :

· Ces objets ont-ils un lien avec l'affaire en cours ? ou plutôt comment le savoir ?

· Comment la police devrait-elle collecter les preuves numériques, telles que des ordinateurs et des smartphones ? Quelles sont les démarches à suivre s'ils sont encore allumés ?

· Comment transférer la preuve numérique ? Existe-t-il certaines bonnes pratiques à suivre lors du déplacement d'ordinateurs par exemple ?

· Comment analyser les preuves numériques collectées ? Le stockage des appareils personnels varie entre des dizaines de giga-octets et plusieurs téraoctets ; comment cela peut-il être analysé ? [10]

La réponse à ces questions est traitée par Digital Forensics. Mais de quoi s'agit-elle ? Comment a-t-elle vu le jour ? L'origine de la criminalistique numérique est en réalité la conséquence de l'évolution de l'informatique dans les années 70 selon l'auteur [11] . Dans son ouvrage, il précise qu'elle a connu 4 époques de développement avant d'arriver à ce qu'elle est maintenant mais affirme aussi qu'elle est toujours encore en phase de croissance.

Citation 1.2.1: "*A major change took place at the beginning of the 1990s. Investigators and technical support operatives within the UK law enforcement agencies, along with outside specialists, realised that digital forensics (as with other fields) required standard techniques, protocols and procedures. Apart from informal guidelines, these formalisms did not exist but urgently needed to be developed*" [12]

Cette citation confirme la vérité des propos de [11] et annonce l'intérêt urgent de la standardisation des normes et procédures en rapport à l'investigation numérique .

Pour ce qui est des définitions de DF, il en existe plusieurs, parmi elles nous citerons :

Définition 1.2.1 : Forensics
Forensics est un mot venant des mots latins « forensis » et « scientia » signifiant « sur le forum » et « connaissance » respectivement, la science forensique fait fait donc référence au processus d'application de méthodes scientifiques aux procédures pénales et civiles, car dans la Rome antique elles se déroulaient en public sur la place du marché (forum) [13].

Définition 1.2.2 : Digital Forensics

Citation 1.2.2: "*Digital forensics is the scientific acquisition, analysis, and preservation of data contained in electronic media whose information can be used as evidence in a court of law*" [14]

Citation 1.2.3: "*Digital forensics is a branch of forensic science that focuses on identifying, acquiring, processing, analyzing, and reporting on data stored electronically*" [15]

D'après [14] et [15] , le digital forensics est une filiale de la science forensic faisant étude des crimes utilisant des équipements électroniques pour extraire des informations nommées digital evidence.

Définition 1.2.3 : Digital Evidence

Pour ce qui concerne la preuve numérique, nous citerons :

Citation 1.2.4: "*Electronic evidence is a component of almost all criminal activities and digital forensics support is crucial for law enforcement investigations*" [15]

Citation 1.2.5: "*ESI stands for Electronically Stored Information, as defined in the Federal Rules of Civil Procedure and which refers to any type of information that is created, used, and stored in digital form and accessible by digital means. ESI is a formal legal term and includes all data, digital documents or other information contained on any media type, whether tape, hard disk drive, cloud storage or some currently unknown storage technology* " [16]

La définition de l'évidence numérique selon l'interpol est un support crucial pour les parties chargées de l'investigation (citation 1.2.4), FRCP[1] , on la définit comme étant tout type d'information provenant des éléments informatiques stockés pour des raisons légales (citation 1.2.5).

Après avoir introduit les notions de base du digital forensic, nous passerons découvrir des démarches types à parcourir au cours d'une investigation numérique garantissant ainsi

[1] Federal Rules of Civil Procedure (abrégé FRCP) sont les lois fédérales qui gèrent la procédure civile dans les tribunaux de district des États-Unies

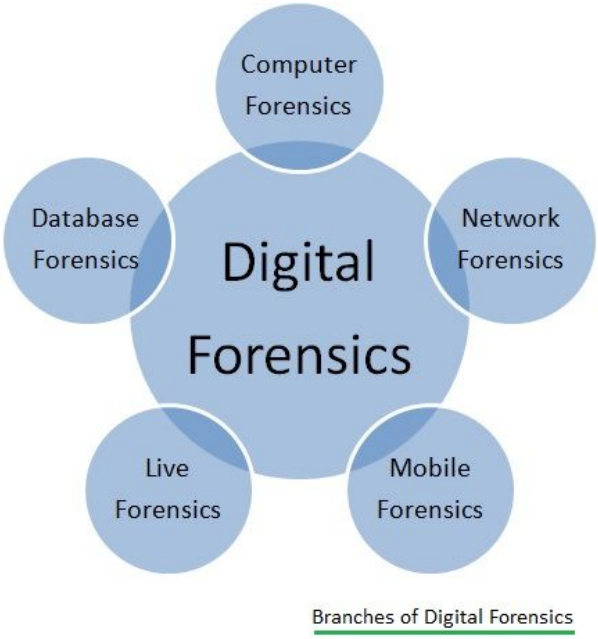

FIG. 1.1 : Les branches du Digital Forensics [1].

l'admissibilité de l'évidence numérique devant un tribunal de justice .

1.3 Modèles d'investigation

Comme on l'a déjà vu, le domaine de la criminalistique numérique est un domaine de recherche relativement nouveau. Malgré ça, elle fait des progrès significatifs pas seulement à propos des programmes de collecte et d'analyse des preuves numériques mais aussi dans la méthodologie. Dans le monde du digital forensics, un modèle de processus est la méthodologie habituelle pour mener une enquête et la guider avec un certain nombre de phases.

Fréquemment, ces modèles sont proposés sur l'expérience des travaux antérieurs, parmi eux sont des modèles généraux expliquant l'ensemble du processus de l'enquête, y en a ceux qui expliquent de manière détaillée une étape spécifique et tandis que d'autres introduisent de nouveaux problèmes avec ou sans exploration de nouvelles méthodes.

Pour notre cas nous verrons en détaille les modèles suivants :

(1) Modèle de Pollitt [2] .

(2) Modèle de DFRWS [2] .

(3) Modèle de ADFM [2].

(4) Modèle de NIST [3] .

1.3.1 Modèle d'investigation de Pollitt

Le tout premier modèle du digital forensic science investigation a été suggéré par MARK Pollitt en 1984 pendant la conférence nationale sur la sécurité de l'information [11]. Il l'introduit en 4 phases principales (figure 1.2).

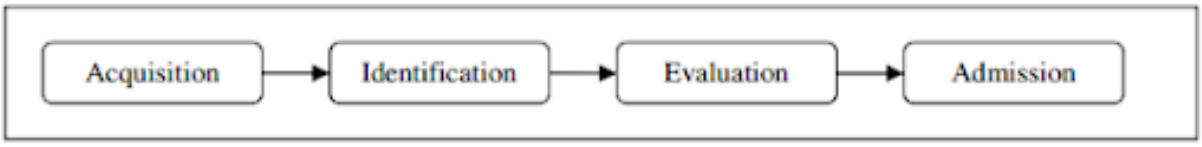

Fig. 1.2 : Modèle de processus de Pollitt [2].

La première phase est l'acquisition, où les preuves sont acquises avec l'accord des autorités , elle est suivie de l'étape d'identification au cours de laquelle toutes les preuves sont transformées du format numérique à un format compréhensible par l'homme. La phase d'évaluation comprend des tâches qui déterminent l'authenticité des preuves recueillies et si elles peuvent effectivement être considérées comme pertinentes pour le cas faisant l'objet d'une enquête. La dernière étape est l'admission où toutes les preuves extraites sont présentées.

1.3.2 Modèle d'investigation de DFRWS

Ce modèle de processus était le modèle de base car il était très cohérent et standardisé. les phases à savoir sont comme l'indique le schéma suivant :

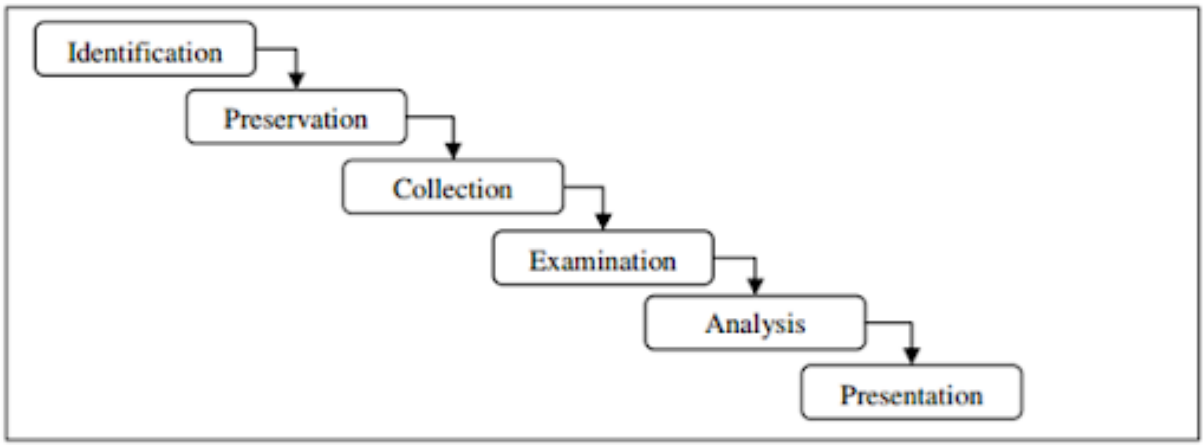

Fig. 1.3 : Modèle proposé par DFRWS [2].

Identification : comprend la détection d'événements ou de crimes, la résolution de signatures, la détection d'anomalies, la surveillance du système ..etc.

Préservation : des technologies d'imagerie sont utilisées et toutes les mesures sont prises pour assurer une "chain of custody" précise et acceptable, la préservation est un principe gardé à travers toutes les phases d'investigation légale numérique.

Collection : les données pertinentes sont collectées sur la base de méthodes de collecte/acquisition avec des logiciels et du matériel approuvés. Dans cette étape, il y a

utilisation également de différentes techniques de récupération et de compression sans perte.

Examination et Analyse : dans cette étape, la traçabilité des preuves, la correspondance des modèles sont garanties, puis les données cachées doivent être découvertes et extraites, à ce stade, l'exploration des données et la chronologie sont effectuées.

Présentation : Les tâches liées à cette étape sont la documentation, la clarification, l'énoncé d'impact de la mission, la recommandation et les contre-mesures sont prises et le témoignage des experts.

1.3.3 Modèle d'investigation de ADFM

En 2002, Mark Reith, Clint Carr et Gregg Gunsch ont formé le modèle abstrait d'investigation numérique, ce modèle inspiré du modèle DFRWS est développé en y ajoutant de légères améliorations comptant ainsi 9 phases.

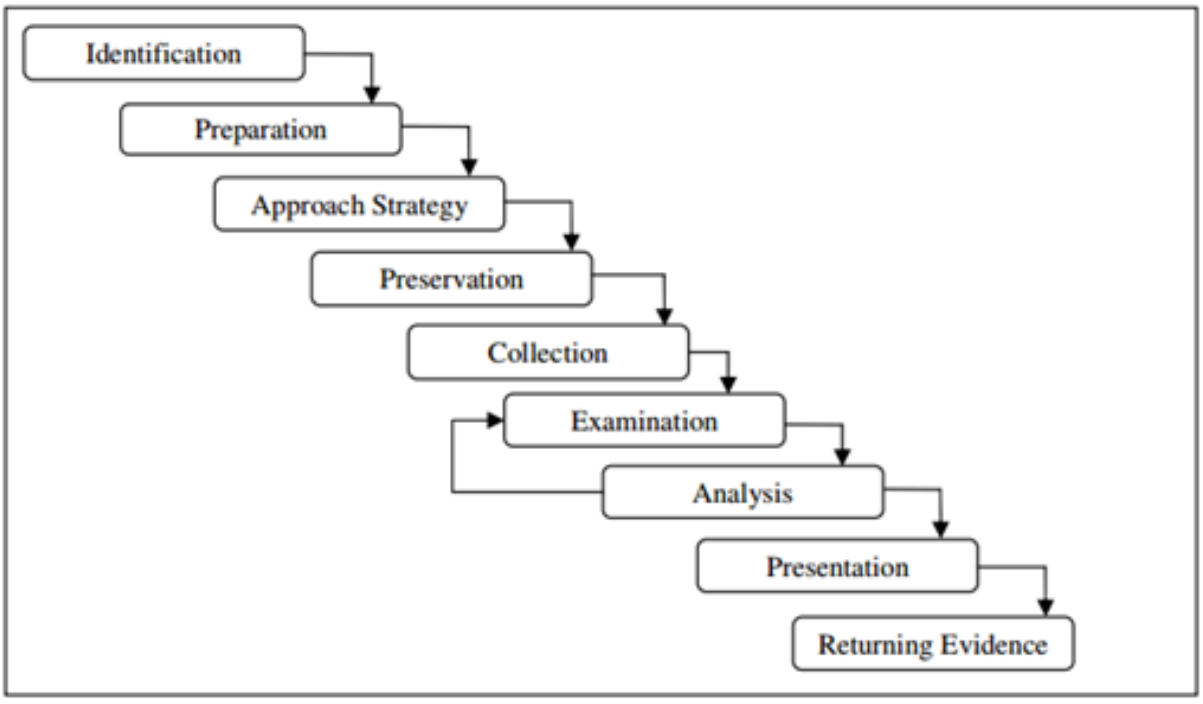

Fig. 1.4 : Les différentes phases de ADFM [2].

Comme selon le modèle de l'organisation DFRWS, l'illustration en dessus divise la feuille de route en ces phases :

Identification : on suppose que le type d'incident est bien reconnu et déterminé, c'est une étape importante puisque toutes les étapes à venir en dépendent (identification des sources d'évidence).

Préparation : il s'agit de la première étape, les outils, les techniques, les mandats de perquisition, l'autorisation de surveillance et le support de gestion sont préparés.

Stratégie d'approche : cette étape est destinée à maximiser la collecte des preuves tout en minimisant l'impact sur la victime en formulant différentes approches et procé-

dures à suivre.

Préservation : toutes les données acquises doivent être isolées et sécurisées pour les conserver dans leur état réel. Toutes les preuves numériques acquises sont dupliquées et la scène physique est enregistrée (photographiée,filmée), sur la base de procédures standardisées, ces tâches sont effectuées dans le cadre de la phase de collecte.

Examen : au cours duquel une analyse systémique approfondie est menée pour rechercher les preuves relatives à l'affaire en cours.

Analyse : durant cette phase, la valeur inculpation disculpatoire des preuves examinées est déterminée.

Présentation : où un résumé du processus est développé et un rapport de l'enquête et des preuves sont établies.

1.3.4 Modèle d'investigation de NIST

Dans la spéciale publication [3] de NIST [2], les auteurs [3] décrivent les phases de base du processus d'un digital forensic : collecte, examen, analyse et rapport.

Au cours de la **collecte**, les données relatives à un événement spécifique sont identifiées, étiquetées, enregistrées et collectées, et leur intégrité est préservée.

Dans la deuxième phase, l' **examen**, les outils forensic et les techniques appropriés aux types de données qui ont été collectées sont exécutés pour identifier et extraire les informations pertinentes des données collectées tout en protégeant leur intégrité. L'examen peut utiliser une combinaison d'outils automatisés et de processus manuels.

La phase suivante, l'**analyse**, consiste à analyser les résultats de l'examen pour en tirer des informations utiles qui répondent aux questions qui ont motivé la réalisation de la collecte et de l'examen.

La phase finale consiste à rendre compte des résultats de l'analyse, ce qui peut inclure la description des actions effectuées, la détermination des autres actions à effectuer et la recommandation d'améliorations des politiques, des directives, des procédures, des outils et d'autres aspects du processus d'investigation numérique.

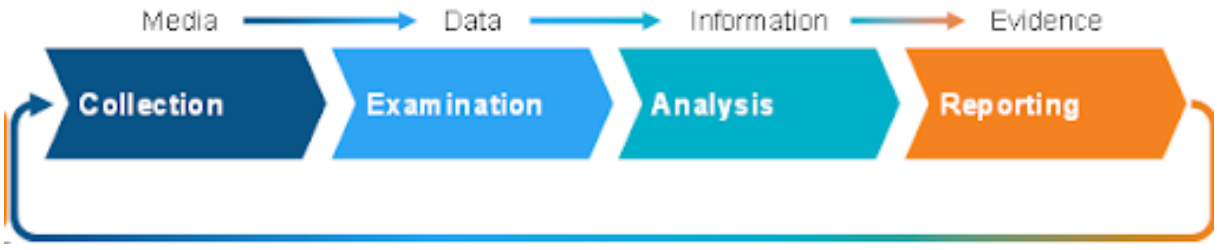

FIG. 1.5 : Les phases basiques selon le NIST [3].

Théoriquement, il y a plusieurs modèles de processus où se trouvent des protocoles qui

[2] Le National Institute of Standards and Technology NIST est un laboratoire de sciences physiques et une agence non réglementaire du Département du commerce des États-Unis. Sa mission est de promouvoir l'innovation américaine et la compétitivité industrielle.

effectuent des analyses d'investigation numérique. Qu'en est-il des moyens ? Le prochain point, nous citerons la manière dont les preuves sont extraites.

1.4 Outils d'investigation

Il existe des dizaines d'outils très performants pour faire de l'analyse forensique. Nous n'allons pas lister ici de manière exhaustive l'ensemble de ces outils. Cependant, nous pouvons de même les regrouper par catégorie :

1.4.1 Analyse de disque : Autopsy/the Sleuth Kit

Autopsy [17] et le Sleuth Kit [18] sont probablement les boîtes à outils les plus connues dans l'enquête numérique. Le Sleuth Kit est un outil de ligne de commande et une bibliothèque à base du langage C qui effectue une analyse forensique des images des disques durs et des smartphones. Autopsy est un système basé sur une interface graphique qui utilise The Sleuth Kit dans les coulisses [18] .

Les outils sont conçus permettant aux utilisateurs d'intégrer facilement des fonctionnalités supplémentaires.

1.4.2 Création d'image : imageur FTK

Autopsy n'a pas de fonctionnalité de création d'image, donc un autre outil doit être utilisé. Alors que les AccessData Forensics Toolkit[19] sont des outils payants, FTK Imager [20] est un produit gratuit. FTK peut être utilisé pour créer des images de disque qui peuvent ensuite être analysées à l'aide d'Autopsy/The Sleuth Kit.

1.4.3 Analyse de la mémoire : Volatility

Le disque dur n'est pas le seul endroit où les données forensic et les artefacts peuvent être stockés sur une machine. Des informations importantes peuvent être stockées dans la RAM, et cette mémoire volatile doit être collectée rapidement et avec soin pour être valide et utile sur le plan d'investigation.

Volatility [21] est l'outil le plus connu et le plus populaire pour l'analyse de la mémoire volatile. Comme The Sleuth Kit, Volatility est gratuit, open-source et prend en charge les plugins tiers.

1.4.4 Analyse du registre Windows : Registry Recon

Le registre Windows agit comme une base de données d'informations de configuration pour le système d'exploitation Windows et les applications qui y sont exécutées. Ces applications peuvent stocker une variété de données différentes dans le registre, et le

registre est l'un des emplacements courants où les logiciels malveillants déploient certaines traces.

Il est possible d'ouvrir et d'afficher le registre Windows via l'application Windows intégrée regedit, et l'analyse du registre est intégrée à certaines plates-formes d'investigation numérique.

Il y a des outils spécialisés tels que Registry Recon [22] sont également disponibles et qui peuvent être utilisés pour reconstruire les registres Windows à partir d'une image forensic et inclut la possibilité de reconstruire les parties supprimées du registre en fonction de l'analyse de l'espace mémoire non alloué.

1.4.5 Mobile forensic : Cellebrite UFED

L'adoption du mobile ne cesse de croître et de nombreuses organisations permettent aux employés d'utiliser ces appareils au travail. Hélas, ces appareils sont une cible croissante de cyberattaques, telles que le phishing, ce qui en fait une source probable d'informations précieuses pour l'investigation.

Avec l'importance croissante de la criminalistique mobile, un outil de criminalistique axé sur le mobile pourrait être une acquisition utile. Cellebrite UFED [23] est largement considéré comme le meilleur outil commercial pour la criminalistique mobile. Il prend en charge un certain nombre de plates-formes différentes (pas seulement les appareils mobiles) et propose des méthodes et des outils exclusifs pour l'analyse des appareils mobiles.

1.5 Conclusion

Dans ce chapitre, nous avons introduit la science d'enquête numérique légale par la définition de ses concepts : forensic , l'illustration des méthodologies à la résolution d'un cas en liaison avec le domaine, et enfin citer des logiciels mis en œuvre pour extraire les preuves numériques légitimes.

Le chapitre suivant offrira davantage une introduction au prochain intitulé “Network Forensic” qui s'agit d'un sous-domaine de la science d'enquête numérique.

Chapitre 2

Network Forensics

2.1 Introduction

Dans ce chapitre, nous allons définir le concept du “network forensic”, ensuite nous allons introduire quelques outils d’investigation utilisés dans cette discipline.

2.2 Définitions

Chaque jour, plus de bits de données circulent sur Internet.

Les prévisions actuelles de Cisco Visual Networking Index (VNI) prévoient que le trafic IP mondial va presque tripler entre 2017 et 2022. Le trafic IP global devrait atteindre 396 EB par mois d’ici 2022, contre 122 exabytes (EB) par mois en 2017, soit un taux de croissance annuel de 26% [24].

Avec l’évolution constante de l’internet et l’augmentation du nombre de machines connectées, les risques liés à la cybercriminalité augmentent eux aussi, ce qui fait qu’un nouveau domaine d’étude sans fin a émergé qui est le “Network Forensic”.

Qu’est ce que donc le “Network Forensic” ?

Pour comprendre ce que le “Network Forensic”, nous allons donner quelques définitions plus formelle :

Définition 2.2.1 : “*Network forensics is a sub-branch of digital forensics relating to the monitoring and analysis of computer network traffic for the purposes of information gathering, legal evidence, or intrusion detection*” [13]

Définition 2.2.2 : “*Network forensics is the science that deals with the capture, recording and analysis of network events and traffic in order to detect and investigate intrusions*” [25]

Définition 2.2.3 : “*The forensic network is defined as the use of scientifically proven techniques to collect, merge, identify, examine, correlate, analyze and document digital evidence from multiple sources of digital processing and active processing in order to discover facts related to the planned intention of unauthorized persons oriented to carry out activities aimed at interrupting, corrupting or compromising system components, as well as providing information to assist in the response or recovery of these activities*” [26]

Définition 2.2.4 : “*Network forensics can be generally defined as a science of discovering and retrieving evidential information in a networked environment about a crime in such a way as to make it admissible in court*” [27]

Le “Network Forensic” est donc une sous discipline du “Digital Forensic” relative à la surveillance, la capture, l’enregistrement et à l’analyse des événements et du trafic du réseau informatique.

L’objectif du “Network Forensic” est donc de découvrir la source des attaques, et collecter des évidences d’intrusion d’une manière qu’elles puissent être utilisées dans une

cour de justice.

2.3 Outils d'investigation

Il existe énormément d'outils d'investigation pour le Network Forensic, que ce soit des outils open source ou propriétaire, ici nous allons introduire certains des outils d'investigation réseau les plus courants et les plus utilisés. L'accent sera mis sur les outils open source :

2.3.1 Libpcap et WinPcap

Libpcap en elle même n'est pas un outil d'investigation mais une bibliothèque UNIX C qui a été écrite dans le cadre d'un programme plus vaste appelé TCPDump [28].

Libpcap fournit une API pour capturer et filtrer les trames de la couche liaison de données à partir d'interfaces réseau arbitraires. Elle a été développée à l'origine au Lawrence Berkeley National Laboratory (LBNL) et initialement rendue public en juin 1994. Il existe également une version WinPcap [29] conçue pour les systèmes Windows.

Le but de libpcap était de fournir une couche d'abstraction afin que les programmeurs puissent concevoir des outils portables de capture et d'analyse de paquets.

Les outils de reniflage et d'analyse de paquets les plus populaires aujourd'hui sont basés sur les bibliothèques libpcap. Ceux-ci incluent tcpdump, Wireshark, Snort, nmap, ngrep et bien d'autres.

Libpcap et WinPcap sont tous deux des logiciels libres publiés sous la « licence BSD », qui a été approuvée par l'Open-Source Initiative.

2.3.2 Tcpdump

Tcpdump [30] est un outil de capture, de filtrage et d'analyse du trafic réseau. Il a été développé au LBNL et rendu public pour la première fois en janvier 1991. Tcpdump s'appuie actuellement sur la bibliothèque libpcap pour ses fonctionnalités.

Tcpdump a été conçu comme un outil UNIX, puis une version WinDump a été développée pour Windows.

L'objectif fondamental de tcpdump est de capturer le trafic réseau, puis d'imprimer ou de stocker le contenu pour analyse. Tcpdump capture le trafic bit par bit lorsqu'il traverse tout support physique adapté au trafic de couche liaison (cuivre, fibre ou même air). Puisque tcpdump est basé sur libpcap, il capture à la couche 2 (la couche liaison de données).

Au-delà de la simple capture de paquets, tcpdump peut décoder les protocoles courants des couches 2 à 4 (ainsi que certains protocoles de couche supérieure), puis afficher leurs informations à l'utilisateur.

2.3.3 Wireshark

Wireshark [31], initialement nommé Ethereal, est un package d'analyse de paquets open source. Il offre à la fois un outil graphique et un analyseur de ligne de commande appelé tshark. Il est conçu pour capturer, filtrer et analyser le trafic.

L'interface graphique facile à utiliser de Wireshark en fait un excellent premier outil pour les analystes novices en criminalistique réseau, tandis que ses capacités avancées de filtrage de paquets, ses fonctionnalités de décodage de protocole et la prise en charge du langage PDML[1] (Packet Details Markup Language) en font également extrêmement utile pour les enquêteurs expérimentés.

2.3.4 TShark

Comme mentionné ci dessus, Tshark [32] est un outil d'analyse de protocole réseau en ligne de commande qui fait partie de la distribution Wireshark.

Comme Wireshark, il est basé sur libpcap et peut lire et enregistrer des fichiers dans les mêmes formats standard que Wireshark. En plus de l'analyse, tshark peut être utilisé pour la capture de paquets.

2.3.5 Dumpcap

La distribution Wireshark est également livrée avec un outil de ligne de commande, " Dumpcap [33] ", qui est spécialement conçu pour capturer des paquets.

Étant donné que dumpcap est un outil spécialisé conçu uniquement pour capturer des paquets, il utilise moins de ressources système, maximisant ainsi les capacités de capture.

En outre, de nombreux systèmes d'exploitation limitent le reniflage du trafic aux programmes exécutés avec des privilèges d'administration.

Wireshark lui-même est un programme complexe et multiforme. Du point de vue de la sécurité, il est plus sûr de limiter l'accès administratif au programme Dumpcap plus petit et plus simple.

2.3.6 Ngrep

Une tâche courante en network forensic et en détection d'intrusion consiste à rechercher un modèle dans le trafic réseau passant par un certain point du réseau. Bien que cela puisse être effectué avec des outils d'analyse complets tels que Wireshark ou IDS comme snort, un outil plus simple et léger est parfois mieux adapté au travail.

Ce créneau est occupé par ngrep [34], qui est un outil simple mais efficace pour rechercher des modèles simples dans le trafic réseau. Il s'agit essentiellement de "grep appliqué

[1] PDML (Panel Definition Markup Language) est un langage de spécification qui définit des balises comme celles définies dans HTML, et permettent à un utilisateur de spécifier l'emplacement exact des composants affichés dans le panel sans connaissance détaillée des langages de programmation informatique complexes.

à la couche réseau".

Il utilise la bibliothèque pcap de tcpdump (libpcap) pour lire les paquets à partir de l'interface réseau et la bibliothèque Perl Compatible Regular Expression (PCRE) pour la correspondance de modèles dans les paquets lus.

2.3.7 Argus

Un flux est une séquence de paquets d'une source à une destination, en examinant les flux au lieu des paquets, on peut rapidement trouver des choses qui se démarquent comme de nouveaux hôtes sur un réseau, des quantités inhabituellement élevées de transferts de données, des flux de données vers des ports inhabituels, etc.

De plus, en stockant des enregistrements de flux au lieu de captures de paquets complets, beaucoup d'espace peut être conservé, car les données de charge utile (couche OSI 5 et supérieures) ne sont pas stockées du tout et les informations de la couche inférieure sont agrégées, permettant ainsi des périodes de surveillance plus longues.

Argus [35] est l'une des premières implémentations d'un système de surveillance des flux réseau, c'était l'acronyme de Audit Record Generation and Utilization System. Développé à l'origine par Carter Bullard, il est maintenant maintenu par Qosient Technologies sous une licence Open Source.

En plus des enregistrements de flux, Argus fournit plusieurs métriques de flux, telles que l'accessibilité, la perte, la gigue, la retransmission ou le retard.

2.3.8 Snort

Snort [36] est un NIDS (Network Intrusion Detection System) open source est, qui fonctionne principalement sur les signatures d'attaque, étant ainsi du type NIDS basé sur les signatures. Il a été initialement développé par Martin Roesch sous une licence Open Source et est maintenant maintenu par Cisco Systems. L'outil lui-même et la base de données des signatures d'attaque sont gérés séparément.

La capture de paquets (appelée acquisition dans la terminologie snort) peut se faire via la libpcap ou via divers frameworks de pare-feu Linux ou BSD comme NFQ (NetFilter Queue) ou IPFW (IP FireWall). Snort peut également lire des fichiers de capture de paquets (nécessitant libpcap).

Snort peut être configuré pour fonctionner en trois modes :

1- Le mode Sniffer, qui lit simplement les paquets du réseau et les affiche dans un flux continu (lisible par l'homme) sur la console.

2- Mode Packet Logger, qui enregistre les paquets sur le disque. Dans les deux modes, snort fonctionnera à peu près comme tcpdump.

3- Le mode NIDS, qui effectue la détection et l'analyse du trafic réseau par rapport à un ensemble de règles défini par l'utilisateur. Il prendra alors une mesure spécifique en fonction de ce qui aura été identifié. Ce qui est exactement détecté et consigné dépend de la base de règles avec laquelle snort est configuré pour s'exécuter.

4- Il existe un quatrième mode, appelé NIPS (Network Intrusion Prevention System) ou snort en ligne, où snort supprime ou réécrit le trafic réseau s'il détecte une activité malveillante afin de prévenir les attaques. Cela nécessite que snort soit positionné de manière à ce que tout le trafic d'attaque passe par le système sur lequel snort s'exécute.

2.4 Conclusion

Dans ce chapitre nous avons défini ce qu'est le "network forensic".

Ensuite, nous avons introduit les outils d'investigation du "network forensic" les plus répandus.

Dans le prochain chapitre, nous parlerons du cloud computing.

Chapitre 3

Cloud Computing

3.1 Introduction

Dans ce chapitre, nous allons définir le domaine du “cloud computing” et présenter ces différents modèles.

Ensuite nous allons parler des différents types de virtualisation, et nous nous intéresserons en particulier au type “OS level virtualisation” et citer ces technologies les plus connues à savoir Docker et LXC.

Enfin, nous allons parler des vulnérabilités de ces technologies. .

3.2 Définitions

Il existe énormément de définitions du “cloud computing”, dans ce chapitre nous allons adopter celle donné par l’institut national des normes et de la technologie “NIST” qui est une des définitions les plus accepté dans l’industrie IT :

Définition 2.1: “ *A model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction*” [37]

Le Cloud computing donne donc aux utilisateurs un accès au services IT comme les réseaux, les serveurs, le stockage, les applications etc. Et la condition préalable à un accès pratique et à la demande aux ressources cloud est la connectivité réseau, d’où le terme “cloud” qui est une métaphore pour l’internet.“Computing” fait référence aux services informatiques fournis par un ordinateur suffisamment puissant capable de fournir une gamme de fonctionnalités, de ressources et de stockage. Ensemble, le cloud computing peut être compris comme la fourniture de services informatiques mesurés à la demande sur Internet.

3.3 Modèles du Cloud Computing

Les modèles du cloud computing peuvent être catégorisés selon leurs déploiements ou selon leurs services offerts.

3.3.1 Selon le modèle de déploiement

3.3.1.1 Public Cloud

Le cloud public est le modèle le plus ancien et le plus connu. Il offre des services sur internet publiquement accessibles comme Google Traduction par exemple.

Le cloud public est maintenu par des fournisseurs de service cloud comme Google Cloud et Amazon AWS, qui gèrent toutes les tâches d’administrations et qui permettent

ainsi aux clients d'accéder et utiliser les ressources cloud gratuitement ou sur la base d'un abonnement payant.

3.3.1.2 Private Cloud

Les clouds privés sont déployés pour un usage interne au sein des entreprises et des organisations. Toutes les données d'un cloud privé sont stockées dans les data center de l'entreprise et l'accès à ces données est autorisé au personnel autorisé seulement, ce qui rend ces données plus sécurisées. Mais cela nécessite aussi que les équipements soient toujours à jour ce qui peut entraîner des coûts élevés.

D'autre part, un contrôle d'accès aux données plus strict signifie également moins de partage de données, même au sein de l' organisme.

3.3.1.3 Hybrid Cloud

Le cloud hybride est un modèle de déploiement flexible, il peut comprendre deux ou plusieurs types de clouds (publics, privés).

Les utilisateurs du cloud peuvent basculer leurs charges de travail entre différents types de clouds comme requis et ainsi profiter des avantages des clouds publics et privés.

Les entreprises peuvent choisir de conserver certaines données sur site pour une sécurité maximale tandis que d'autres données sur les clouds publics pour une meilleure rentabilité en adoptant une base de paiement à l'utilisation, d'où un modèle de cloud hybride.

Mais l'inconvénient est que le cloud hybride nécessite généralement une configuration et une exploitation et une maintenance plus complexes, et il peut également avoir besoin de résoudre des problèmes de compatibilité entre des plates-formes d'infrastructure hétérogènes.

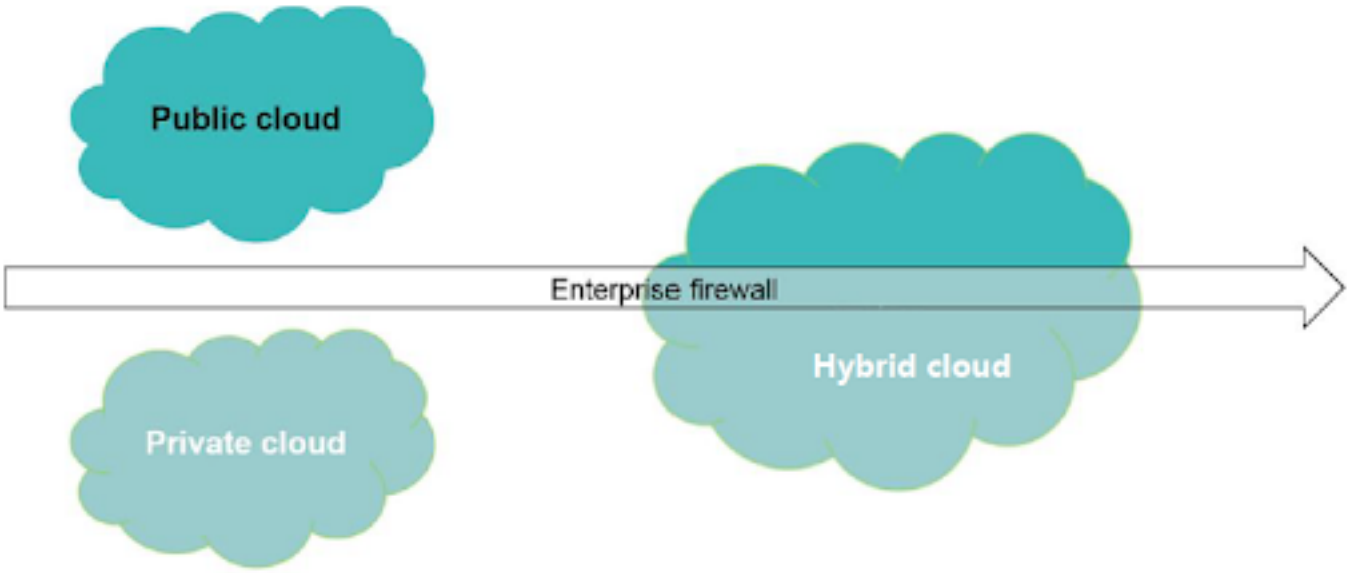

Fig. 3.1 : Les modèles de déploiement du cloud computing [4].

3.3.2 Selon les modèles de services

Il existe trois principaux modèles de services de Cloud Computing : l'infrastructure en tant que service (IaaS), la plate-forme en tant que service (PaaS) et le logiciel en tant que service (SaaS). Chaque modèle de service représente une partie différente du Cloud Computing et comprend sa propre division unique des responsabilités entre le client et le fournisseur de services.

3.3.2.1 Infrastructure as a service (IaaS)

L'infrastructure en tant que service (IaaS) est le modèle de service qui constitue la base du déploiement d'une technologie dans le Cloud. Grâce à un fournisseur IaaS, le client bénéficie d'un accès à la demande via Internet aux ressources informatiques de base, notamment les ordinateurs (matériel virtuel ou dédié), la mise en réseau et le stockage.

L'IaaS permet un accès à une ressource matérielle flexible et de pointe qui peut être adaptée aux besoins de traitement et de stockage, le client peut ainsi fournir les applications, les logiciels sans avoir à en assurer la gestion et la maintenance.

3.3.2.2 Plateform as a service (PaaS)

La plateforme en tant que service (PaaS) est le modèle de service de Cloud dans lequel le client accède à des logiciels combinés par l'intermédiaire d'un fournisseur de services. Le PaaS est le plus souvent utilisé pour le développement d'applications.

Un fournisseur PaaS donne accès à l'infrastructure combinée de Cloud nécessaire au développement d'applications – bases de données, intergiciels, systèmes d'exploitation, serveurs – sans la complexité sous-jacente de sa gestion. Cela permet de devenir plus efficace. Au lieu de passer du temps à installer et à configurer l'infrastructure, le client se concentre uniquement sur le développement, l'exécution et la gestion des applications.

3.3.2.3 Software as a service (SaaS)

Le logiciel en tant que service (SaaS) est le modèle de service de Cloud qui permet d'accéder à un produit logiciel complet, exécuté et géré par le fournisseur de services.

La plupart des solutions SaaS ont tendance à être des applications destinées à l'utilisateur final. Le fournisseur SaaS est responsable de la fourniture, de la maintenance et de la mise à jour du logiciel, y compris de l'infrastructure sous-jacente.

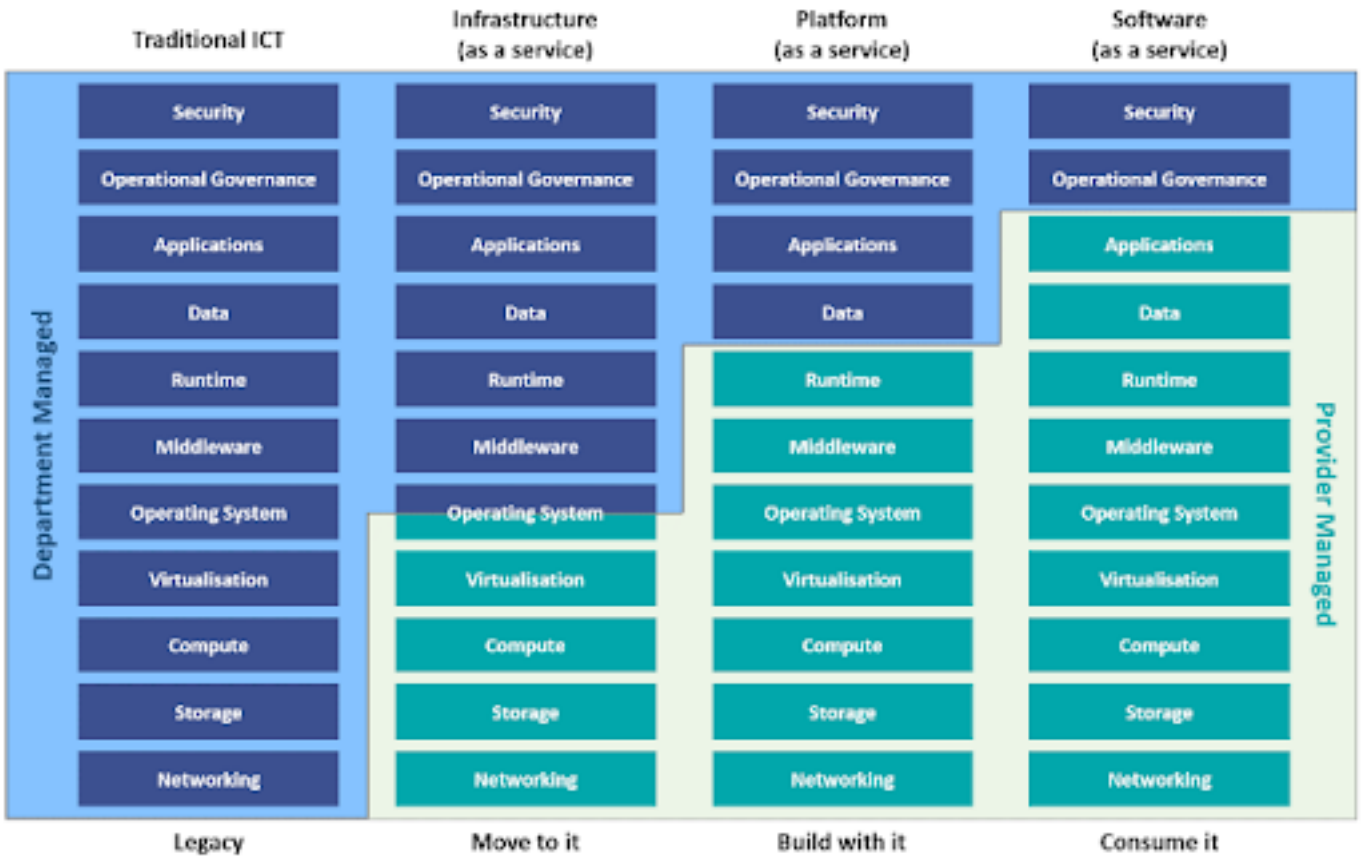

FIG. 3.2 : Les modèles de services du cloud computing [5] .

3.4 Virtualisation

La virtualisation est une technologie qui simule les fonctionnalités matérielles et crée plusieurs machines virtuelles sur une machine physique. Généralement, les applications doivent s'exécuter à l'intérieur d'un système d'exploitation, et un seul système d'exploitation peut s'exécuter sur une machine physique à la fois. La virtualisation permet donc de créer des machines virtuelles résidant sur une machine physique et d'exécuter des systèmes d'exploitation indépendants.

L'essence de la virtualisation est de séparer le logiciel du matériel en convertissant les périphériques "physiques" en dossiers ou fichiers "logiques". Sans virtualisation, l'exécution de plusieurs programmes d'application principaux dans le même système d'exploitation d'un serveur physique peut provoquer des conflits d'exécution et des goulots d'étranglement des performances.

Avec la virtualisation, plusieurs machines virtuelles peuvent s'exécuter sur un seul serveur physique, et chaque machine virtuelle peut exécuter un système d'exploitation indépendant, ce qui améliore l'utilisation des ressources.

3.4.1 Types de virtualisation

3.4.1.1 Virtualisation des postes de travail

La virtualisation des postes de travail est un des types de virtualisation qui est fortement apprécié et usité en entreprise. Cette technique reproduit l'environnement d'un ordinateur, afin d'offrir la possibilité aux professionnels d'accéder à leurs fichiers et applications personnelles depuis n'importe quel poste.

Ce type de virtualisation est rendu possible grâce à l'hébergement du poste de travail virtuel sur un serveur VDI (Virtual Desktop Infrastructure) qui exécute l'ensemble de l'environnement du poste (système d'exploitation et applications). TS2log[1] est un exemple de solution proposant cet accès sécurisé à des desktops virtualisés.

La virtualisation des postes de travail offre beaucoup de flexibilité, notamment en situation de mobilité et facilite le transfert des environnements de travail aux équipes de sous-traitants, elle permet aussi de réduire les coûts liés à la multiplication des postes de travail, particulièrement ceux associés aux licences de logiciels.

3.4.1.2 Virtualisation des applications

La virtualisation d'applications est le processus d'installation d'une application sur un serveur central qui peut virtuellement être exploité sur plusieurs systèmes. Pour les utilisateurs finaux, l'application virtualisée fonctionne exactement comme une application native installée sur une machine physique.

Avec la virtualisation des applications , il est plus facile pour les organisations de mettre à jour, de maintenir et de réparer les applications de manière centralisée. Les administrateurs peuvent contrôler et modifier les autorisations d'accès à l'application sans se connecter au bureau de l'utilisateur.

Un autre avantage de la virtualisation des applications est la portabilité, elle permet aux utilisateurs d'accéder à des applications virtualisées de façon flexible et agile quel que soit le système d'exploitation ou l'appareil utilisé.Ils peuvent ainsi faire tourner plusieurs versions d'une même application sur un seul OS.

Cela permet aussi à l'utilisateur de gagner du temps investi dans les installations d'applications et les opérations de chargement.

3.4.1.3 Virtualisation de serveur

La virtualisation de serveur est un processus de partitionnement des ressources d'un serveur physique en plusieurs serveurs virtuels à l'aide d'une couche logicielle.

Chacun de ces serveurs virtuels créés agit ensuite de manière autonome et isolée, exécutant ses propres systèmes d'exploitation et applications. La virtualisation des serveurs permet aux entreprises d'exécuter plusieurs systèmes d'exploitation indépendants (invités ou virtuels) avec des configurations différentes à l'aide d'un seul serveur (hôte).

[1] Ts2Log est un logiciel de virtualisation pour les entreprises et les professionnels qui permet d'établir une connexion Bureau à distance.

Le processus permet ainsi d'économiser les coûts matériels impliqués dans la conservation d'une multitude de serveurs physiques. Ce type de virtualisation repose sur le rôle de l'hyperviseur, installé sur le serveur physique, qui assure la gestion des différents OS invités et il existe deux types d'hyperviseurs :

- **l'hyperviseur de type 1, ou bare metal** : il opère directement sur le hardware, et devient de ce fait l'outil de contrôle du système d'exploitation. Les OS invités s'exécutent alors par dessus cet hyperviseur, on peut citer comme exemple : vSphere[2] de l'éditeur VMware[3], ou KVM, l'hyperviseur libre pour Linux.
- **l'hyperviseur de type 2, ou host metal** : il fonctionne à l'intérieur d'un autre système d'exploitation, par exemple : VirtualBox, VMware Workstation[4].

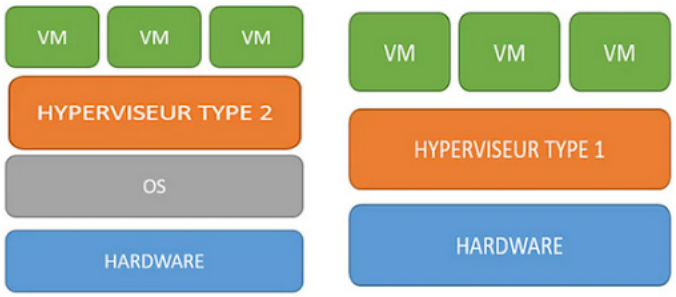

FIG. 3.3 : Différence entre l'hyperviseur de type 1 et type 2 [6] .

3.4.1.4 Virtualisation de réseau

On entend par virtualisation de réseau, ou virtualisation network, le processus reproduisant un réseau physique et ses différents composants (ports, routeurs, etc.).

La virtualisation du réseau permet de gérer et de surveiller l'ensemble du réseau informatique en tant qu'entité administrative unique. Ça permet aux administrateurs de garder une trace de divers éléments de l'infrastructure réseau tels que les routeurs et les commutateurs à partir d'une seule console d'administrateur basée sur un logiciel.

La virtualisation du réseau aide à l'optimisation du réseau pour les taux de transfert de données, la flexibilité, la fiabilité, la sécurité et l'évolutivité. Il améliore la productivité et l'efficacité du réseau global, il devient donc plus facile pour les administrateurs d'allouer et de distribuer les ressources de manière pratique et d'assurer des performances réseau élevées et stables.

3.4.1.5 Virtualisation du stockage

La virtualisation du stockage (appelée également Software Defined Storage, ou SAN virtuel) consiste à regrouper l'ensemble des périphériques de stockage physiques quelles

[2]VMware vSphere est la plate-forme de virtualisation du cloud computing de VMware.
[3]VMware, Inc. est une société de technologie de cloud computing et de virtualisation.
[4]VMware Workstation Pro est un hyperviseur hébergé qui s'exécute sur les systèmes d'exploitation Windows et Linux et qui permet aux utilisateurs de créer et configurer des machines virtuelles.

que soient leurs provenances (différents réseaux, différents centres de données, etc) en un seul périphérique de stockage simulé qui est géré depuis une console centrale.

Pour l'utilisateur, les différentes ressources de stockage qui composent le pool sont invisibles, de sorte que le stockage virtuel apparaît comme un seul lecteur physique. Il peut ainsi ajuster les capacités de stockage en fonction des besoins et de l'évolution structurelle de ses besoins, sans que cela nécessite trop d'investissement et sans avoir à se soucier de la provenance ou de l'emplacement de chaque périphérique de stockage.

3.4.2 Niveaux de virtualisation

Il existe cinq niveaux de virtualisation disponibles [38] qui sont les plus couramment utilisés dans l'industrie. Ceux-ci sont les suivants :

3.4.2.1 Niveau d'architecture du jeu d'instructions Instruction Set Architecture Level (ISA)

Dans ISA, la virtualisation fonctionne via une émulation ISA. Ceci est utile pour exécuter des tas de code hérité qui a été écrit à l'origine pour différentes configurations matérielles.

Ces codes peuvent être exécutés sur la machine virtuelle via un ISA. Un code binaire qui pourrait nécessiter des couches supplémentaires pour s'exécuter peut maintenant s'exécuter sur une machine x86 ou avec quelques ajustements, même sur des machines x64, par exemple, le code binaire MIPS peut s'exécuter sur une machine hôte x86 à l'aide de l'émulation ISA. L'émulation du jeu d'instructions contribue à en faire une machine virtuelle indépendante du matériel.

3.4.2.2 Niveau d'abstraction matérielle (Hardware Abstraction Level) (HAL)

Comme son nom l'indique, ce niveau permet d'effectuer la virtualisation au niveau matériel, il utilise un hyperviseur qui s'exécute directement sur le matériel. D'une part, cette approche génère un environnement matériel virtuel pour une VM.

D'autre part, le processus gère le matériel sous-jacent via la virtualisation ce qui permet de virtualiser les ressources d'un ordinateur, telles que ses processeurs, sa mémoire et ses périphériques d'E/S.

L'idée a été mise en œuvre pour la première fois par IBM dans l' IBM VM / 370 en 1960. Plus récemment, l'hyperviseur Xen a été appliqué pour virtualiser des machines basées sur x86 afin d'exécuter Linux ou d'autres applications de système d'exploitation invité.

3.4.2.3 Niveau bibliothèque (Library Level)

Les appels système du système d'exploitation sont longs. La plupart des applications optent pour les API des bibliothèques au niveau de l'utilisateur. La plupart des API

fournies par les systèmes sont plutôt bien documentées. Par conséquent, la virtualisation au niveau de la bibliothèque est préférée dans de tels scénarios.

La virtualisation avec des interfaces de bibliothèques est rendue possible par les hooks d'API[5] qui contrôlent le lien de communication entre le système et les applications. Certains outils disponibles aujourd'hui, tels que vCUDA'[6] et WINE[7], ont démontré avec succès cette technique.

L'outil logiciel WINE a implémenté cette approche pour prendre en charge les applications Windows sur les hôtes UNIX. Un autre exemple est le vCUDA qui permet aux applications s'exécutant dans des machines virtuelles de tirer parti de l'accélération matérielle GPU.

3.4.2.4 Niveau d'application (Application Level)

La virtualisation au niveau de l'application virtualise une application en tant que machine virtuelle. Sur un système d'exploitation traditionnel, une application s'exécute souvent comme un processus.

Par conséquent, la virtualisation au niveau de l'application est également connue sous le nom de virtualisation au niveau du processus.

C'est généralement utile lors de l'exécution de machines virtuelles avec des langages de haut niveau (HLL[8]).

Microsoft .NET CLR et Java Virtual Machine (JVM) sont deux bons exemples de cette classe de VM.

3.4.2.5 Niveau du système d'exploitation (Operating System Level)

Au niveau du système d'exploitation, le modèle de virtualisation crée une couche abstraite entre les applications et le système d'exploitation.

La virtualisation a ce niveau crée des conteneurs isolés sur un seul serveur physique et les instances du système d'exploitation pour utiliser le matériel et les logiciels dans les centres de données. Chacun de ces conteneurs fonctionne comme un vrai serveur.

La virtualisation au niveau du système d'exploitation, également connue sous le nom de **conteneurisation**, a gagné en popularité au cours des dernières années. La conteneurisation permet à différentes charges de travail de partager les mêmes ressources sous-jacentes

[5] API hooking est une technique par laquelle nous pouvons instrumenter et modifier le comportement et le flux des appels d'API.

[6] vCUDA est un middleware qui permet à une application d'utiliser une unité de traitement graphique (GPU) compatible CUDA installée sur un ordinateur distant comme si elle était installée sur l'ordinateur sur lequel l'application est en cours d'exécution.

[7] Wine est une couche de compatibilité libre et open-source qui permet aux logiciels d'application et aux jeux développés pour Microsoft Windows de fonctionner sur les systèmes d'exploitation Unix-like.

[8] Un langage de haut niveau (HLL) est un langage de programmation tel que C, FORTRAN ou Pascal qui permet à un programmeur d'écrire des programmes qui sont plus ou moins indépendants d'un type particulier d'ordinateur.

d'une manière séparée (isolée).

Comme avec la virtualisation assistée par matériel, les performances sont optimisées car les appels sont effectués directement vers le système d'exploitation sous-jacent sans qu'aucune émulation ne soit nécessaire.

Avec l'émergence de Docker[9] comme moyen simple de créer des charges de travail pouvant être déplacées d'une plate-forme à une autre, tout en minimisant la quantité de ressources utilisées pour fournir la virtualisation, la virtualisation au niveau du système d'exploitation est intégrée à de nombreuses plates-formes cloud et prise en charge par la majorité des DevOps[10] systèmes.

Les autres plateformes qui offrent une virtualisation au niveau du système d'exploitation incluent les conteneurs Linux (LXC) et les partitions de charge de travail IBM pour AIX[11].

Parmi les méthodes de containerisation les plus connues on retrouve LXC et Docker.

3.4.2.5.1 LXC

Linux Container (LXC) désigne à la fois les applications virtualisées basées sur Linux et la plateforme et la technologie de conteneur sous-jacente. Il est nécessaire de garder ceci à l'esprit notamment lorsqu'on parle de plateformes de conteneurs alternatives qui utilisent également les Linux Container comme technologie.

LXC est une méthode de virtualisation au niveau du système d'exploitation et une plateforme de conteneurs open source qui promet une utilisation simple, et une expérience utilisateur intuitive et moderne. Pour cela, la plateforme offre différents outils, langues, modèles et bibliothèques.

En outre, l'environnement de virtualisation peut s'installer et s'utiliser sur toutes les distributions courantes de Linux. LXC permet d'exécuter plusieurs systèmes Linux isolés (conteneurs) sur un hôte de contrôle à l'aide d'un seul noyau Linux en combinant les groupes de contrôle du noyau Linux (cgroups[12]), et la prise en charge des espaces de noms isolés (namespaces) fournis par le noyau Linux afin de fournir un environnement isolé pour les applications.

L'idée de la technologie Linux Container est née en 2001. C'est dans le cadre du projet

[9]Docker est un ensemble de plateformes en tant que produits de service qui utilisent la virtualisation au niveau du système d'exploitation pour fournir des logiciels dans des packages appelés conteneurs.Docker est expliqué plus en détails ultérieurement.

[10] DevOps est un ensemble de pratiques qui combine le développement de logiciels et les opérations informatiques.

[11] AIX est une série de systèmes d'exploitation Unix propriétaires développés et vendus par IBM pour plusieurs de ses plates-formes informatiques.

[12]cgroups est une fonctionnalité du noyau Linux qui limite, comptabilise et isole l'utilisation des ressources d'un ensemble de processus.

Vserver[13], qu'un environnement isolé a été implémenté pour la première fois. Cela a posé les bases de la mise en place de plusieurs espaces de noms contrôlés dans Linux et de ce qui est maintenant appelé le conteneur Linux.

Dans la pratique, LXC assure un développement plus rapide des applications. La technique de conteneurs aide notamment pour le portage, la configuration et l'isolation. Lors de la diffusion de données en temps réel, les conteneurs jouent aussi un rôle majeur en mettant à disposition l'évolutivité nécessaire aux applications.

3.4.2.5.2 Docker

Docker est une technologie open source développée par la société Docker Inc et fournit une virtualisation au niveau du système d'exploitation. Le logiciel publié à l'origine sous le nom " Docker " fut développé sur la base de la technologie Linux Container (LXC) et utilise les fonctionnalités d'isolation des ressources du noyau Linux telles que cgroups, kernel namespaces et OverlayFS[14].

LXC fut ensuite remplacée par le libcontainer[15] de Docker, et de nouveaux composants logiciels ont été ajoutés, alors que Docker a continué à croître et à devenir le standard pour la virtualisation basée sur les conteneurs, à tel point que certains associent Docker aux conteneurs.

Containerd a notamment émergé du développement de Docker en tant que moteur d'exécution de conteneur avec l'implémentation standard runC[16]. Aujourd'hui, ces projets sont gérés par la Cloud Native Computing Foundation (CNCF) et l'Open Container Initiative (OCI).

La fonctionnalité centrale de Docker est la virtualisation d'applications basée sur les conteneurs. Avec Docker, le code de l'application, qui comprend toutes les dépendances, est empaqueté à l'intérieur d'une image. Le logiciel Docker exécute l'application empaquetée dans un conteneur Docker, les images peuvent ainsi être déplacées d'un système à l'autre et être exécutées sur tout système compatible avec Docker.

Une des premières préoccupations des conteneurs Docker est d'isoler l'application en cours d'exécution. À la différence des VM, néanmoins, un système d'exploitation entier n'est pas virtualisé. À la place, Docker alloue certaines ressources issues du système d'exploitation et du matériel à chaque conteneur : on peut créer et diriger en parallèle autant

[13] Linux-VServer est une implémentation de serveur privé virtuel qui a été créée en ajoutant des capacités de virtualisation au niveau du système d'exploitation au noyau Linux.

[14] OverlayFS est une implémentation de système de fichiers de montage union pour Linux. Il combine plusieurs points de montage sous-jacents différents en un seul.

[15] Libcontainer permet aux conteneurs de fonctionner avec les espaces de noms Linux, les groupes de contrôle, les fonctionnalités, les profils de sécurité AppArmor, les interfaces réseau et les règles de pare-feu de manière cohérente et prévisible.

[16] runC est un outil CLI publié par Docker pour générer et exécuter des conteneurs conformément à la spécification OCI.

de conteneurs qu'on veut à partir d'une image Docker. C'est ainsi que les services Cloud extensibles sont mis en place.

Cependant, la création du nombre de conteneurs qu'on souhaite rend leur gestion très compliquée. En effet, gérer un nombre important de conteneurs manuellement est un véritable casse-tête, c'est pourquoi l'orchestration de conteneurs est apparue.

3.4.2.5.3 Orchestration de conteneurs

L'orchestration de conteneurs est simplement l'automatisation de provisionnement, le déploiement, la mise en réseau, la mise à l'échelle, la disponibilité et la gestion du cycle de vie des conteneurs.

L'orchestration de conteneurs est effectuée grâce à des outils tels que Kubernetes, Docker Swarm, Apache Mesos etc.

Aujourd'hui, Kubernetes est la plate-forme d'orchestration de conteneurs la plus populaire.

(1) **Kubernetes**

Kubernetes [39] est un moteur d'orchestration de conteneur open source permettant d'automatiser le déploiement, la mise à l'échelle et la gestion des applications conteneurisées..

Les avantages de Kubernetes par rapport aux autres solutions d'orchestration résultent en grande partie de ses fonctionnalités [40] plus complètes et sophistiquées dans plusieurs domaines, notamment :

- Déploiement de conteneur : Kubernetes déploie un nombre spécifié de conteneurs sur un hôte spécifié et les maintient en cours d'exécution dans l'état souhaité.

- Rollouts : est une modification apportée à un déploiement. Kubernetes permet de lancer, suspendre, reprendre ou annuler des déploiements.

- Découverte des services : Kubernetes peut exposer automatiquement un conteneur à Internet ou à d'autres conteneurs à l'aide d'un nom DNS ou d'une adresse IP.

- Provisionnement du stockage : Les développeurs peuvent configurer Kubernetes pour monter un stockage local ou cloud persistant pour les conteneurs selon leurs besoins.

- Équilibrage de charge : Lorsque le trafic vers un conteneur augmente, Kubernetes peut utiliser l'équilibrage de charge et la mise à l'échelle pour le répartir sur le réseau afin d'assurer la stabilité et les performances.

- Autoréparation : Lorsqu'un conteneur tombe en panne, Kubernetes peut le redémarrer ou le remplacer automatiquement. Il peut également retirer les conteneurs qui ne répondent pas à vos exigences en matière de contrôle de santé.

- Prise en charge et portabilité entre plusieurs fournisseurs de cloud : Kubernetes bénéficie d'un large support parmi tous les principaux fournisseurs de cloud. Ceci est particulièrement important pour les organisations qui déploient des applications dans un cloud hybride ou un environnement multi cloud hybride.

- Écosystème croissant d'outils open source : Kubernetes dispose également d'une gamme d'outils de convivialité et de mise en réseau en constante expansion pour améliorer ses capacités via l'API Kubernetes. Ceux-ci incluent Knative, qui permet aux conteneurs de s'exécuter en tant que charges de travail sans serveur, et Istio, un maillage de services open source.

(2) **Composants de Kubernetes**

Kubernetes suit l'architecture maître-esclave [41], le maître est appelé Master, il gère le cluster Kubernetes et afin d'augmenter la fiabilité d'un cluster, il est possible d'avoir plusieurs masters. Les autres nœuds du cluster ont le rôle de Workers. Ils sont là pour exécuter la charge de travail. Il mettent à jour ou suppriment les conteneurs en fonction de ce qu'ils ont à exécuter et ils sont gérer par les masters.

Un nœud peut être une machine virtuelle ou une machine physique.

(a) **Composants du Master**

(i) Kube-apiserver : un point d'entrée exposant l'API HTTP Rest de k8s (kubernetes) depuis le maître du cluster Kubernetes. Différents outils et bibliothèques peuvent facilement communiquer avec l'API.

(ii) Kube-scheduler : Il est responsable de la répartition et l'utilisation de la charge de travail sur les nœuds du cluster selon les ressources nécessaires et celles disponibles.

(iii) Kube-controller-manager : ce composant est responsable de la plupart des collecteurs qui récupèrent des informations du cluster tout en effectuant des actions de correctives en cas de besoin, en apportant des modifications pour amener l'état actuel du serveur à l'état souhaité.

(iv) Cloud-controller-manager : effectue les mêmes actions que le kube-controller-manager mais pour des fournisseurs de cloud sous-jacents (AWS, Azure, Google Cloud Platform, etc ...).

(v) Etcd : il stocke les informations de configuration pouvant être utilisées par chacun des nœuds du cluster. Ces informations sont conservées sous forme de clé et valeurs à haute disponibilité.

(b) **Composants du Worker**

(i) Kubelet : Un agent qui s'exécute sur chaque nœud du cluster. Il s'assure que les conteneurs fonctionnent dans un pod. kubelet prend un ensemble de PodSpecs fournis par divers mécanismes et s'assure du fonctionnement et de la santé des conteneurs décrits dans ces PodSpecs et ne gère que les conteneurs créés par Kubernetes.

(ii) Kube-proxy : Kube-proxy est un proxy réseau qui s'exécute sur chaque nœud du cluster et implémente une partie du concept Kubernetes de Service et il maintenant les règles du réseau et permet l'exposition des services vers l'extérieur.

(iii) Container Runtime : L'environnement d'exécution de conteneurs est le logiciel responsable de l'exécution des conteneurs tel que : Docker, containerd, cri-o etc.

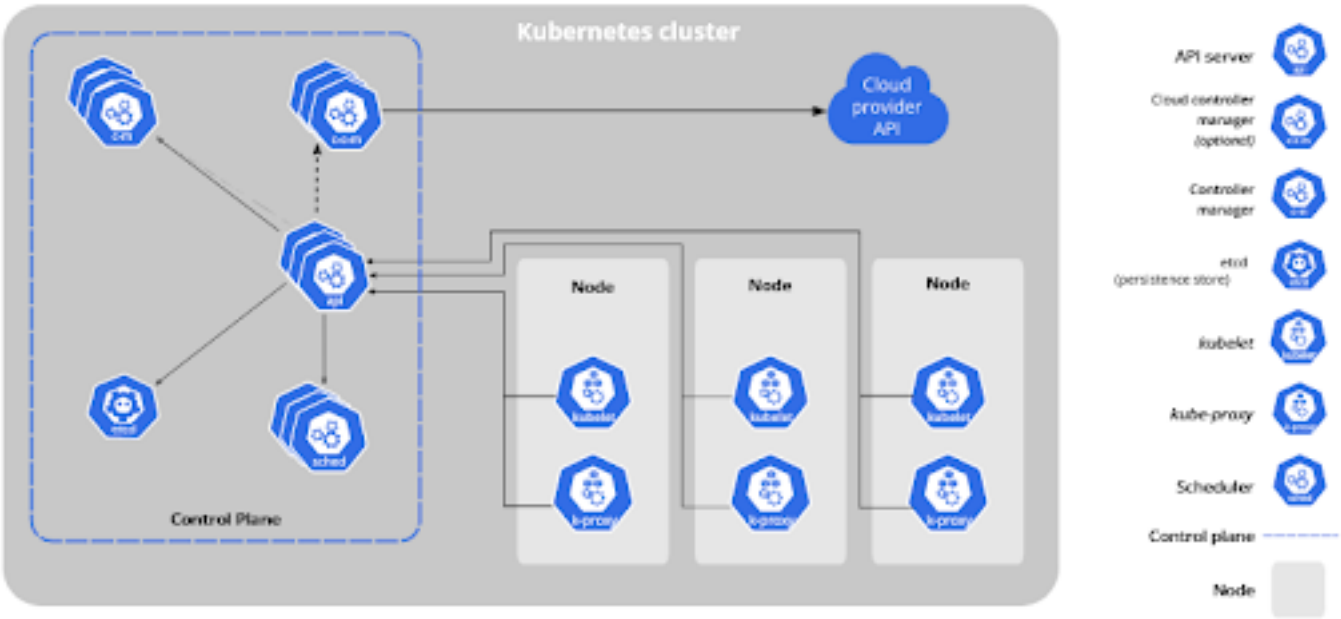

Fig. 3.4 : Les composants de Kubernetes [7].

(3) **Les objets Kubernetes**

Il existe plusieurs objets kubernetes [41], les plus important sont les suivants :

(i) Node : Un node ou noeud est une machine de travail du cluster Kubernetes. Ce sont des unités de travail qui peuvent être physiques, virtuelles mais aussi des instances cloud.

(ii) Pod : Il s'agit de l'unité la plus petite de K8s, un pod encapsule le ou les conteneur(s) formant une application conteneurisée partageant ainsi la même stack réseau et le même stockage. Chaque pod se voit attribuer une adresse IP unique.

(iii) Replicas : c'est le nombre d'instances d'un Pod.

(iv) ReplicaSet : s'assure que les réplicas spécifiés sont actifs.

(v) Deployment : définit l'état désiré et fournit des mises à jour déclaratives des Pods et ReplicaSets.

(vi) Service : Un service peut être défini comme un ensemble logique de pods exposés en tant que service réseau. C'est un niveau d'abstraction au-dessus du pod, qui fournit une adresse IP et un nom DNS unique pour un ensemble de pods.

(vii) Endpoint : Représente l'adresse IP et le port d'un service, il est automatiquement créé lors de la création d'un service avec les pods correspondants.

(viii) Volumes : Fournit un stockage persistant qui existe pendant toute la durée de vie du pod. Ce stockage peut également être utilisé comme espace disque partagé pour les conteneurs à l'intérieur du pod.

(ix) Namespace : Un partitionnement des ressources qu'il gère en différents ensembles qui ne se chevauchent pas appelés namespaces. Ils sont destinés à être utilisés dans des environnements avec de nombreux utilisateurs répartis sur plusieurs équipes ou projets, ou même séparant des environnements tels que le développement, le test et la production.

(4) **Méthodes de communication avec le cluster**

Pour communiquer avec le cluster Kubernetes , on peut utiliser le CLI Kubectl, ou l'interface web nommée Kubernetes Dashboard, on peut également utiliser différents outils et bibliothèques de différents langages comme Java, Go, Python etc pour communiquer avec l'API [41].

(5) **Vulnérabilités et problème d'isolation**

Bien que la technologie de conteneurisation qu'offre LXC et Docker utilisent les cgroups et les namespaces pour garantire un environnement isolé au applications, mais cette isolation n'est tout de même pas parfaite, en effet les conteneurs représente de nombreuses vulnérabilité et parmis les plus connues on retrouve :

(a) **Élévations de privilèges de conteneur**

Ce type d'attaque est basé sur l'exploitation des défauts ou des vulnérabilités qui existent dans le container runtime ou dans le noyau Linux. runC est le runtime de conteneur de bas niveau qui fait le plus gros du travail sous le capot pour Docker, CRIO[17] et containerd. Dans l'un des exploits récents, les conteneurs exploités ont permis aux attaquants d'écraser la bibliothèque runc de l'hôte et d'obtenir un accès root aux hôtes du conteneur [42].

Dans un autre cas, la vulnérabilité du noyau Linux quelque peu ancienne CVE-2017-7308 [43] peut être utilisée pour changer les espaces de noms (namespaces[18]) du processus actuel en processus 1 et les espaces de noms de l'hôte en faisant un appel système du noyau Linux, permettant une évasion complète vers l'hôte.

(b) **Vulnérabilités des images de conteneurs**

L'intégration de programmes malveillants dans les images de conteneurs, en particulier les images disponibles publiquement dans les dépôts en ligne, constituent l'une des menaces les plus courantes pour les conteneurs. Au début de cette année (2019), Docker a annoncé que sa base de données d'utilisateurs Docker Hub avait été compromise par un pirate informatique pendant une brève période [44].

En effet, environ 5% des utilisateurs de Docker Hub ont été touchés par cette violation de données. Les données compromises comprenaient des noms d'utilisateur, des mots de passe et des informations d'identification VCS comme les jetons github, gitlab et bitbucket.

Un incident récent implique le piratage de 17 images Docker accessibles au public, où avait été dissimulé un logiciel de minage de cryptomonnaie [45].

Il est également fréquent de voir des adversaires utiliser des images de base pour détourner un environnement de conteneurs et le transformer en membre d'un réseau zombie en vue d'une attaque DDoS [46] .

(c) **Vulnérabilités d'authentification**

Menaces courantes mais problématiques, qui existent depuis longtemps. Ignorer l'authentification et l'autorisation de base des orchestrateurs tels que Kubernetes et Docker peut offrir aux attaquants un accès complet au déploiement de conteneurs leur permettant ainsi d'accéder à des informations privées, d'injecter du code malveillant ou de faire des ravages en infectant les conteneurs au sein de l'environnement d'exploitation.

[17] CRI-O est une alternative légère à l'utilisation de Docker comme runtime pour kubernetes.

[18] Un espace de noms (namespace) est un ensemble de signes utilisés pour identifier et faire référence à des objets de différents types.

La sécurisation de la console/tableau de bord et des API de l'orchestrateur de conteneurs est la première étape de la sécurisation de l'infrastructure de conteneurs. Les attaques récentes contre Tesla, Shopify et Aviva basées sur des consoles kubernetes non sécurisées sont un bon exemple de ce type d'attaque.

Dans le cas de Tesla, la console Kubernetes a été laissée configurée sans mot de passe et les pirates ont pu prendre le contrôle et trouver les informations d'identification du cloud AWS. Ils ont pu accéder aux compartiments S3 contenant des données sensibles, ainsi qu'exécuter l'extraction de crypto-monnaies dans les pods Kubernetes [47].

Lors d'une attaque similaire contre WeightWatchers, la console de cluster Kubernetes non sécurisée a été découverte en analysant les adresses IP accessibles au public sur le port TCP 10250 de kublet [48].

Un autre exemple de vulnérabilité de sécurité avec une API non sécurisée est l'utilisation de l'API Docker pour créer de nouveaux conteneurs malveillants sur des hôtes de conteneur avec les images communautaires et les logiciels malveillants pour l'extraction de crypto-monnaies [49].

(d) **Vulnérabilités des applications**
Il s'agit de mauvaises pratiques de codage typiques présentant des chances d'exploiter les temps d'exécution de l'application. Ces types de vulnérabilités ne sont pas spécifiques à l'application conteneurisée, mais peuvent être importants en raison des dépendances des applications cloud natives sur les API.

Certains des exemples sont l'injection SQL[19], les scripts inter sites (XSS[20]), la falsification de requête côté serveur (SSRF[21]), les faiblesses dans la génération et l'authentification des clés de chiffrement, ainsi que l'utilisation incorrecte des cookies.

(e) **Vulnérabilités du réseau**
Les conteneurs pouvant communiquer entre eux et avec l'environnement d'orchestration via le réseau, il est alors possible qu'un conteneur compromis puisse propager le logiciel malveillant sur plusieurs hôtes de conteneur.
Les conteneurs exposés à Internet sont aussi soumis aux attaques telles que le déni de services (DDOS[22]), les injections SQL et les attaques XSS ciblant les

[19] L'injection SQL est un vecteur d'attaque courant qui utilise un code SQL malveillant pour la manipulation de la base de données principale afin d'accéder à ses informations stockées.

[20] Les attaques XSS permettent aux attaquants d'injecter des scripts côté client dans les pages Web consultées par d'autres utilisateurs.

[21] Une attaque par falsification de requête côté serveur (SSRF) implique qu'un attaquant abuse des fonctionnalités du serveur pour accéder à des ressources ou les modifier.

[22] DDOS est une attaque par déni de service dans laquelle l'auteur cherche à rendre une machine ou une ressource réseau indisponible.

services réseau qui sont des attaques tout aussi dangereuses pour les environnements conteneurisés que pour les environnements classiques.

(f) **Problème d'isolations des conteneurs**

Le mot "conteneur" est souvent mal compris, car de nombreux développeurs ont tendance à associer le concept d'isolation à un faux sentiment de sécurité, estimant que cette technologie est parfaitement sûre. Il faut comprendre que les conteneurs n'ont aucune fonctionnalité de sécurité par défaut.

Au contraire, leur sécurité dépend entièrement de l'infrastructure de support (OS et plate-forme), de leurs composants logiciels embarqués et de leur configuration d'exécution [50].

L'hypothèse selon laquelle les conteneurs ajoutent une couche de sécurité aux applications qui s'exécutent à l'intérieur est une idée fausse très répandue, en particulier parmi les personnes peu familiarisées avec l'atténuation de la sécurité opérationnelle. Celle-là pourrait entraîner l'exposition des conteneurs à la nature sans configuration appropriée, permettant à un attaquant d'accéder et d'élever les privilèges à l'intérieur de l'hôte.

Une étude récente d'Aqua Security a révélé que "50% des nouvelles instances Docker mal configurées sont attaquées par des botnets dans les 56 minutes suivant leur configuration". Alors que la majorité des attaques se concentrent sur le cryptomining, 40% d'entre elles impliquent également des portes dérobées pour accéder à l'environnement et aux réseaux de la victime.
Les portes dérobées ont été activées en supprimant des logiciels malveillants dédiés ou en créant de nouveaux utilisateurs avec des privilèges root et des clés SSH pour un accès à distance [51].

L'isolation dans les conteneurs peut être très efficace si les conteneurs sont bien configurés, mais ça peut vite causer de terribles dégâts dans le cas contraire.

3.5 Conclusion

Dans ce chapitre nous avons défini le Cloud Computing, et nous avons parlé de ces différents modèles de déploiement et des ces différents modèles de services.

Ensuite nous avons défini la virtualisation et nous avons parlé de ces différents types, en particulier la virtualisation au niveau OS.

Enfin, nous avons parlé des technologies de contarisation les plus connues et utilisées, à savoir docker et LXC puis nous avons évoqué les vulnérabilités liées à ces technologies.

Dans le prochain chapitre on présentera notre solution proposée pour faire du network forensic sur les conteneurs.

Chapitre 4

Network Forensics au niveau du système d'exploitation

4.1 Introduction

Dans ce chapitre, nous allons voir comment faire du Network Forensic dans un conteneur Docker, puis nous allons faire une petite démonstration en mettant en place un simple scénario.

Ensuite, nous allons évoquer le Network Forensic dans le cas d'un orchestrateur de conteneurs Kubernetes en commençant par la mise en place d'un simple cluster, puis en évoquant les techniques qu'on peut utiliser pour y arriver.

Enfin, nous présenterons notre solution pour faire du Network Forensic dans un cluster Kubernetes, plus précisément cibler un pod et récupérer son trafic réseau afin de l'analyser ultérieurement.

4.2 Network Forensic dans Docker

Nous avons vu dans le chapitre précédent ce qu'un conteneur et comment il peut à la fois être sécurisé, ou vulnérable.

Supposant qu'un des conteneurs qui contient une application donnée, présente une activité suspecte, et qu'on veut savoir le trafic réseau que ce conteneur généré, comment peut-on faire cela ?

4.2.1 Capturer le trafic réseau d'un conteneur Docker

Docker offre différentes commandes qui permettent de créer et gérer les conteneurs ainsi que leurs réseau et les images qui les composent etc.

Dans notre cas nous allons nous concentrer sur les commandes qui vont nous permettre de répondre à notre problématique.

La façon la plus simple d'y arriver, c'est de lancer la capture du trafic réseau directement sur le host, mais dans ce cas on aura le trafic de tous les conteneurs qui s'exécute sur la machine, ce qui fait qu'on aura énormément de trafic à analyser par la suite dans le cas ou on a beaucoup de conteneurs qui sont en exécution.

On va donc utiliser l'option "–**net=container :<container-name>**" [52] qui va faire en sorte que notre conteneur qui contient les outils de capture du trafic réseaux, va partager la pile réseau du conteneur cible, et donc va pouvoir voir et capturer le trafic de celui-ci.

Les deux solutions cité, sont structuré dans la figure ci dessous :

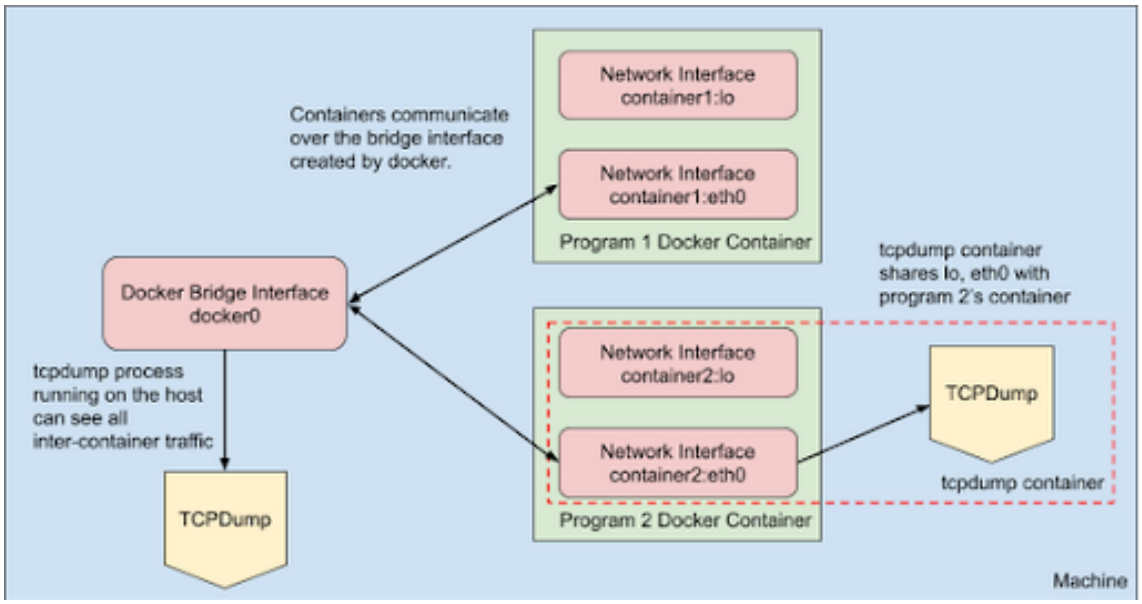

FIG. 4.1 : Configuration du réseau Docker par défaut [8].

Nous allons voir ce que cela donne en pratique dans la prochaine section.

4.2.2 Scénario

Pour mettre en pratique la commande évoqué précédemment, nous allons réaliser le scénario présenté par la figure suivante :

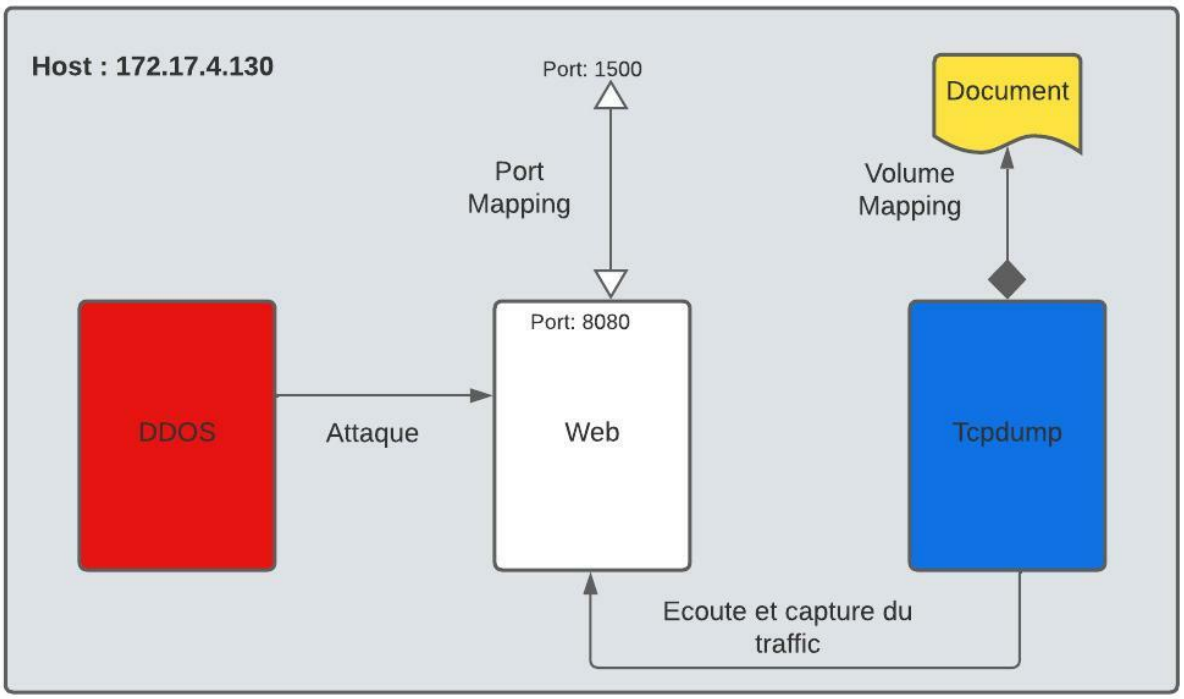

FIG. 4.2 : Enregistrement du trafic du conteneur "Web".

On commence par lancer un conteneur “web” exécutant un simple script python permettant d'afficher une page web.

```
┌──(kali㉿kali)-[~]
└─$ docker run --name web -p 1500:8080 -dt kodekloud/simple-webapp
```

Fig. 4.3 : Lancement de conteneur ”Web”.

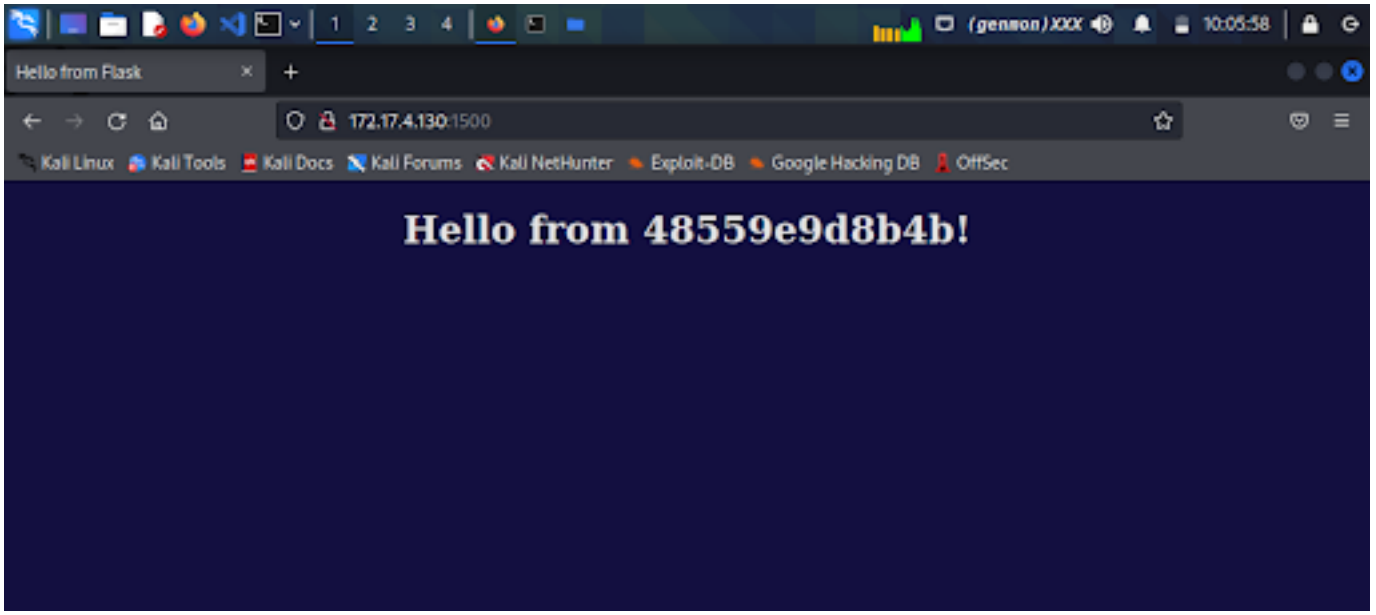

Fig. 4.4 : Affichage de contenu de ”Web” dans le navigateur.

Ensuite, on lance la capture du trafic réseau du conteneur “web” à partir d'un conteneur “tcpdump”, et la sauvegarder dans un fichier pcap qu'on pourra analyser plus tard.

```
┌──(kali㉿kali)-[~]
└─$ docker run --name tcpdump --net=container:web -v /home/kali/recuperation:/data corfr/tcpdump -i eth0 -w /data/capture.pcap
tcpdump: listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
```

Fig. 4.5 : Démarrage de capture.

Enfin, on lance un conteneur “ddos” à base d'une image hébergeant un script [53] qui lance des attaques ddos sur le conteneur “web” créé précédemment.

```
┌──(kali㉿kali)-[~]
└─$ docker run --name ddos flooder 172.17.4.130 1500 1000000
[#] Attack started on 172.17.4.130 (172.17.4.130 ) || Port: 1500 || # Requests: 1000000
 09:05:17 [141] #-#-# Hold Your Tears #-#-#
```

Fig. 4.6 : Début de l'attaque.

On peut voir que les conteneurs sont en exécution :

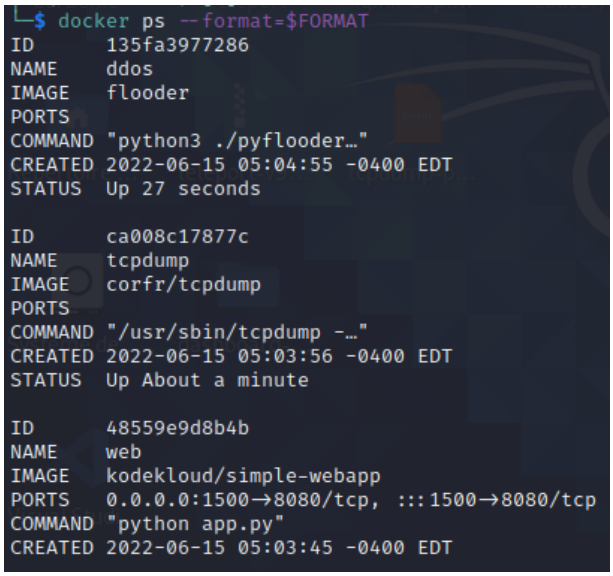

Fig. 4.7 : Liste des conteneurs actifs.

Le fichier "capture.pcap" quant à lui sera stocké dans le volume [54] qu'on a monté au conteneur tcpdump.

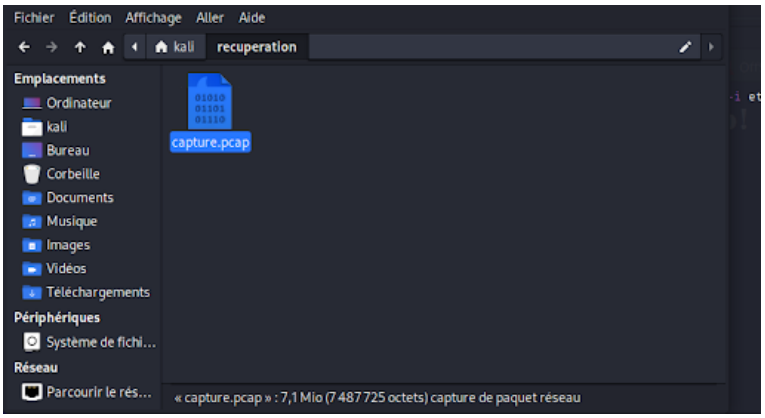

Fig. 4.8 : Génération du fichier Pcap.

On peut maintenant analyser le fichier "capture.pcap" avec Wireshark :

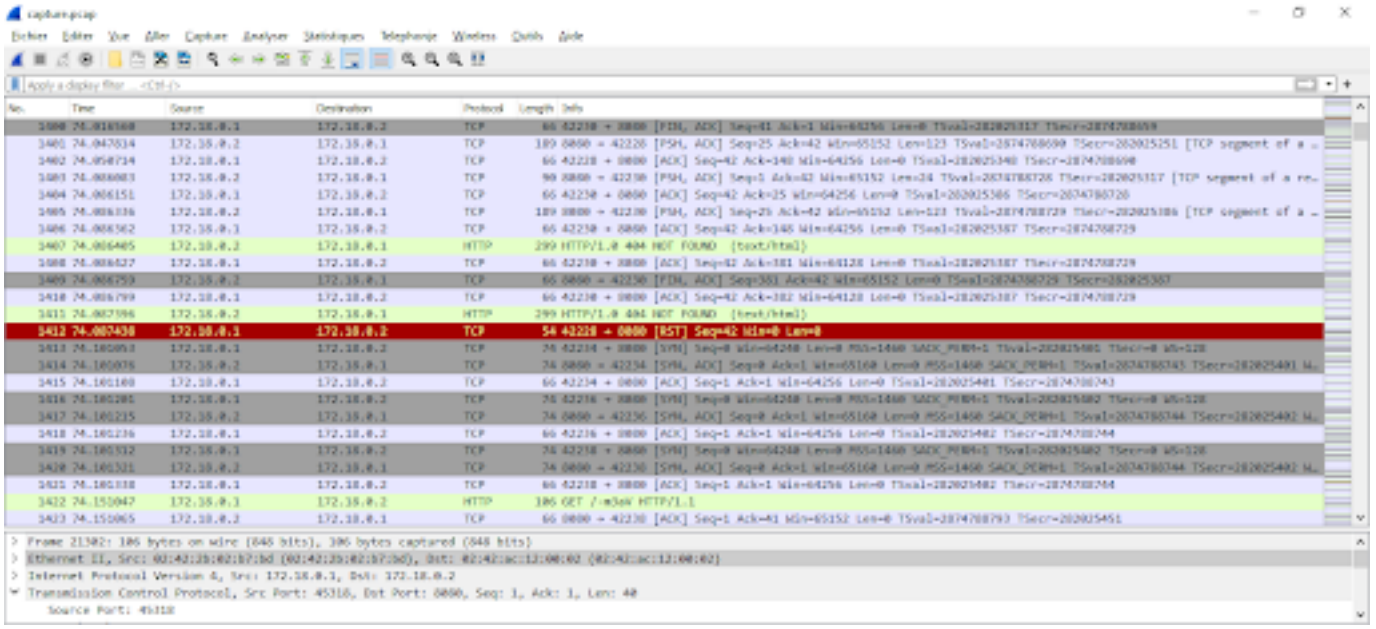

Fig. 4.9 : Liste des paquets affichées par Wireshark.

En filtrant les paquets capturés selon le port destination (8080) et le protocol (HTTP) on peut voir les paquets HTTP résultant d'une HTTP flood attack visant le conteneur "web".

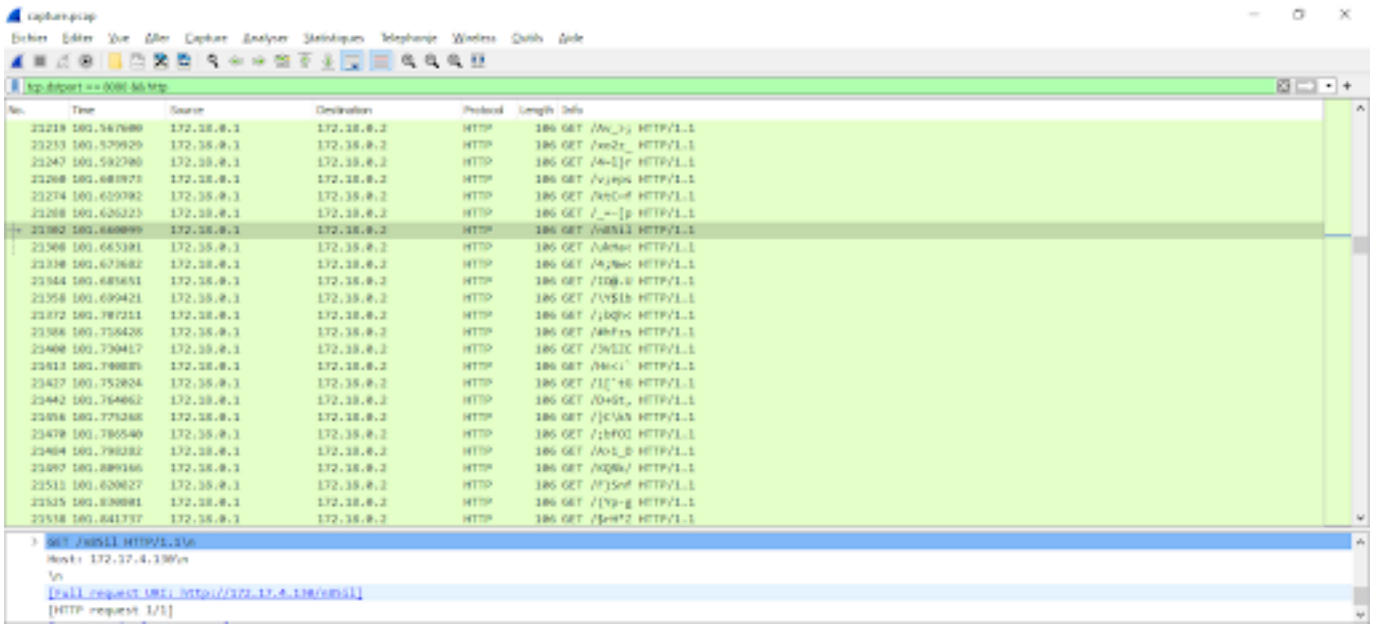

Fig. 4.10 : Filtrage des paquets HTTP.

4.3 Network Forensic dans Kubernetes

Dans cette section nous allons parler du Network Forensic dans Kubernetes, en commençant par les types d'installations d'un cluster Kubernetes.

4.3.1 Installation et Configuration d'un cluster K8s

Il existe plusieurs méthodes pour mettre en place un cluster kubernetes [55], et selon le cluster qu'on veut mettre en place et son utilisation, on peut distinguer plusieurs installation :

(a) **Installation avec un seul nœud (Single node installation)**

Ce type d'installation convient à ceux qui souhaitent tester Kubernetes et pratiquer. Il existe de nombreuses distributions qui permette de mettre en place cette installation, les plus connus étant Minikube [56] et Kind [57].

(b) **Installation manuelle du cluster**

Ce type d'installation est utilisé pour déployer un cluster manuellement en utilisant kubeadm [58]. C'est une méthode privilégiée pour déployer un cluster Kubernetes pour la première fois.

(c) **Installation automatique du cluster**

Ce type d'installation est effectué à l'aide d'outils d'automatisation, de scripts ou de programmes d'installation distribués par le fournisseur. Kubespray [59] et RKE [60] sont un exemple de solution qui permette de mettre en place cette installation.

(d) **Clusters gérés par un fournisseur**

Le cycle de vie des clusters est géré par les fournisseurs. Dans ce type d'installation, un cluster de niveau production peut être déployé avec un minimum d'actions de l'utilisateur. Les fournisseurs sont responsables de la gestion de l'ensemble du cluster ainsi que de l'infrastructure sous-jacente. EKS [61], GKE [62] et AKS [63] sont les fournisseurs les plus connus

(e) **Kubernetes the hard way**

La méthode dure est utilisée pour installer Kubernetes à partir de zéro. C'est utile pour ceux qui veulent apprendre tous les composants de Kubernetes pour comprendre Kubernetes en profondeur [64] [65].

Dans notre cas nous allons utilisé l'installation manuelle, avec kubeadm.

Nous avons 3 nœuds (PCs) physiques (1 master et 2 workers) qui tournent sous xubuntu et qui disposent de la configuration suivante :

- Master node :
  - Ram : 16 Gb.
  - Processeur : intel i5.
  - Stockage : 512 Gb HDD.
  - Système d'exploitation : xubuntu.
- Worker nodes :

- Ram : 4 Gb.
- Processeur : intel Pentium dual-core.
- Stockage : 512 Gb HDD.
- Système d'exploitation : xubuntu.

On commence donc l'installation en installant les dépendances nécessaires pour pouvoir lancer Kubeadm, exécutant ces commandes :

```
$ sudo swapoff -a
$ apt-get update
$ apt-get install -y docker.io containerd apt-transport-https
$ curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | apt-key
 add -
$ cat <<EOF >/etc/apt/sources.list.d/kubernetes.list
deb http://apt.kubernetes.io/ kubernetes-xenial main
EOF
```

Fig. 4.11 : Installation des mise à jour et dépendances .

Ensuite on est prêt pour installer kubeadm, kubelet et kubectl :

```
$ apt-get update
$ apt-get install -y kubelet kubeadm kubectl
```

Fig. 4.12 : Installation des objets kubernetes.

À ce stade, nous avons tous les outils dont on a besoin, on est donc prêt à aller de l'avant et déployer un cluster kubernetes.

Maintenant que l'installation de Kubeadm est terminée, nous allons continuer et créer un nouveau cluster à l'aide de la commande kubeadm init qu'on doit lancer sur le nœud master. Une partie de ce processus consiste à choisir un fournisseur de réseau, et il existe plusieurs choix. Pour notre cluster nous utiliserons Calico [66].

Pour pouvoir utiliser Calico , nous devons ajouter l'option –pod-network-cidr comme arguments de ligne de commande à kubeadm init :

```
$ kubeadm init --pod-network-cidr=192.168.0.0/16
```

Fig. 4.13 : Création du cluster.

Cela va prendre un certain temps pour, a la fin ça va donner quelque chose comme ceci :

```
To start using your cluster, you need to run (as a regular user):

  mkdir -p $HOME/.kube
  sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
  sudo chown $(id -u):$(id -g) $HOME/.kube/config

You should now deploy a pod network to the cluster.
Run "kubectl apply -f [podnetwork].yaml" with one of the options listed
at:
  http://kubernetes.io/docs/admin/addons/

You can now join any number of machines by running the following on each
node as root:

  kubeadm join --token 354502.d6a9a425d5fa8f2e 192.168.0.9:6443 \
--discovery-token-ca-cert-hash
sha256:ad7c5e8a0c909ed36a87452e65fa44b1c2a9729cef7285eb551e2f126a1d6a54
```

Fig. 4.14 : Résultat de l'installation.

On suit alors les instructions indiqué et on exécute ces commandes sur le noeud master :

```
mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

Fig. 4.15 : Installation des commandes de la figure précédente.

Une fois terminé on doit installer un "Network add-on", on a choisi Calico puisqu'il fournit une mise en réseau simple, performante et sécurisée. Calico bénéficie de la confiance des principaux fournisseurs de cloud, EKS, AKS, GKE et IKS ayant tous intégré Calico dans le cadre de leurs offres.

```
kubectl apply -f https://docs.projectcalico.org/manifests/calico.yaml
```

Fig. 4.16 : Installation de Calico.

La dernière étapes est de joindre les noeuds worker au noeud master en exécutant cette commande sur chacune des deux machines :

```
  kubeadm join --token 354502.d6a9a425d5fa8f2e 192.168.0.9:6443 \
--discovery-token-ca-cert-hash
sha256:ad7c5e8a0c909ed36a87452e65fa44b1c2a9729cef7285eb551e2f126a1d6a54
```

Fig. 4.17 : Jointure des workers node au master node.

Une fois cela est fait, on devrait avoir un cluster Kubernetes installé sur nos machines.

On peut s'assurer que tous les noeuds sont opérationnelle avec la commande : kubectl get nodes :

```
kubectl get nodes

NAME        STATUS      ROLES     AGE      VERSION
master      Ready       master    1h       1.23
worker1     Ready       <none>    50m      1.23
worker2     Ready       <none>    48m      1.23
```

Fig. 4.18 : Vérification des états des noeuds.

une installation plus détaillé se trouve sur le site officiel de Kubernetes.

4.3.2 Capture du trafic d'un pod

Pour la capture du trafic vers un Pod, nous avons trouvé qu'il y a plusieurs options et qui sont :

(a) **Monitoring du cluster et filtrage selon l'adresse ip du pod**

Cette solution consiste à récupérer tout le trafic réseau du cluster, puis filtrer selon l'adresse ip du pod que l'on veut voir le trafic généré.

Cela peut s'avérer utile puisque c'est assez simple à réaliser, en effet des outils comme Prometheus permettent de faire ça. Par contre, cette solution peut vite compliquer les choses car dans un vrai cluster de production, on aura énormément de machines, que ce soit physique ou virtuelle qui vont héberger les pods, et il y aura bien évidemment énormément de pods qui s'exécutent.

Le flux réseau sera donc trop volumineux et il y aura surement des pertes, c'est-à-dire des paquets qui ne seront pas capturés.

Un autre point qu'il faut garder à l'esprit, c'est les adresses ip des pods qui sont éphémères, en d'autre terme, un pod qui s'arrête pour n'importe quelle raison se verra attribuer son adresse ip à un autre poids qui sera créé.

Istio et Linkerd sont deux open source qui permettent d'avoir des métriques pour tout le trafic au sein d'un cluster, trafic entrant et sortant.

(b) **Tcpdump en sidecar dans le pod cible**

Normalement les pods sont conçus pour contenir un seul conteneur à la fois, et c'est recommandé par Kubernetes d'ailleurs, mais il arrive qu'un pod puisse contenir un deuxième (ou plusieurs) conteneur, ce deuxième conteneur est appelé "sidecar".

Un conteneur sidecar est donc rien d'autre qu'un deuxième conteneur qui s'exécute cote a cote avec le conteneur principal, les deux conteneurs partage alors les mêmes ressources réseaux et stockage.

On peut donc utiliser ça en notre avantage en lançant un conteneur qui contient des outils de capture (tcpdump ou autre) et le lancer en sidecar avec le conteneur que l'on veut intercepter le trafic.

Cette solution, est très util, mais pour que ça soit efficace, il faut que le sidecar soit ajouté au pod cible avant que celui ci ne soit lancé, autrement, kubernetes doit arrêter le pod en question puis ajouter le sidecar et le lancer de nouveau, ce qui n'est pas une bonne chose, puisque on a pas à toucher à l'intégrité d'un pod ou autre.

(c) **Découverte du noeud auquel est affecté le pod, l'adresse ip et l'interface réseau du pod, puis écoute sur cette interface**

Cette solution vise à cibler exactement le pod que l'on veut capturer le trafic.

Pour y arriver, il faut d'abord avoir quelques informations : l'id du pod cible, son adresse ip, le nœud sur lequel il est exécuté et son interface réseaux.

Une fois que ces informations sont recueillies, on va utiliser le concept du "pod privilégié" et prendre avantage de cela pour y avoir un accès direct au nœud qui lance notre pod ciblé pour ensuite pouvoir récupérer le trafic de ce dernier en écoutant l'interface qu'il utilise.

Cette solution répond exactement à ce qu'on veut faire, et c'est d'ailleurs ce qu'on va utiliser dans notre application, et qu'on verra plus en détails prochainement.

Par contre cette solution n'est pas parfaite non plus, et d'un point de vue sécurité, avoir un accès direct au nœud peut être risqué, c'est d'ailleurs une des raisons pour laquelle les conteneurs privilégiés sont désactivés par défaut par Docker.

Ceci dit, on a la possibilité de réduire les privilèges de notre pod, et le limiter à faire juste ce qu'il doit faire.

(d) **Network tap**

Network tap est un point d'accès de test ou un périphérique matériel placé à un point de réseau spécifique où les données sont accessibles. L'objectif d'une network tap est qu'un tiers surveille le trafic réseau entre deux terminaux.

Le dispositif de network tap envoie une copie du trafic au port de surveillance sans en avertir le réseau, ce dispositif est discret et indétectable et c'est largement utilisé dans les applications de sécurité réseau tel que les systèmes de détection d'intrusion réseau (NIDS).

Cependant, la mise en place d'un network tap dans Kubernetes est un peu plus compliqué étant donné que le réseau dans K8s est géré par un CNI mais aussi quant à l'architecture de K8s et la nature des pods.

Mais il existe tout de même des open source qui permettent de faire une network tap dans K8s tel que kokotap [67].

kokotap fournit des network tap pour les pod Kubernetes en créant une interface VxLAN pour cibler le pod/conteneur, puis effectuer le mirroring des paquets vers l'interface VxLAN grâce à tc-mirred [68]. kokotap peut également créer une interface VxLAN vers un nœud Kubernetes cible pour capturer le trafic.

kokotap peut aussi faire le mirroring des paquets vers un nœud non Kubernetes, mais il faut créer l'interface VxLAN manuellement dans ce cas spécifier l'adresse IP du nœud non Kubernetes.

(e) **eBPF**

eBPF (Extended Berkeley Packet Filter) est une technologie de noyau (à partir de Linux 4.x) qui permet aux programmes de s'exécuter sans avoir à modifier le code source du noyau ou à ajouter des modules supplémentaires. Le kernel intègre un environnement sandbox, une machine virtuelle qui permet l'exécution du bytecode BPF, lequel peut affecter le kernel et utiliser ses ressources, sans le modifier [69].

eBPF a été initialement utilisé pour augmenter l'observabilité et la sécurité lors du filtrage des paquets réseau [70]. Cependant, au fil du temps, c'est devenu un moyen de rendre l'implémentation du code fourni par l'utilisateur plus sûr, plus pratique et plus performant.

En effet, lorsqu'un programme eBPF est employé pour la mise en réseau, les paquets destinés à des destinations spécifiques peuvent contourner le chemin réseau par défaut et se déplacer directement vers la destination prévue. Cela améliore les performances du réseau et est souvent utilisé dans des environnements conteneurisés complexes pour optimiser le comportement des échanges entre microservices ou entre conteneurs.

De même, eBPF peut appliquer une politique de réseau aux paquets, ce qui permet de router les paquets efficacement et d'appliquer des politiques au trafic.

Cilium [71] et Pixie [72] sont des exemples d'open sources qui utilisent cette technologie et l'appliquent dans des clusters Kubernetes.

4.4 Notre application Cloud

Dans cette section, nous allons présenter notre solution qui est une petite application avec une simple interface graphique, et allons faire une démonstration pour montrer son fonctionnement.

4.4.1 Description

L'application offre une interface graphique et permet de cibler un pod précis pour capturer son flux réseau et télécharger le fichier pcap pour une future analyse.

L'application peut aussi s'authentifier sur un cluster distant pour accéder à ses ressources et y faire la capture du trafic réseau d'un pod.

L'idée est de déployer un pod privilégié sur le même nœud sur lequel s'exécute le pod ciblé. Ce pod privilégié contient l'outil tcpdump et a accès aux interfaces réseau du nœud, ce qui lui permet d'écouter directement l'interface réseau du pod ciblé et ainsi récupérer son trafic réseau.

4.4.2 Plan de l'application

Comme montré dans le chapitre précédent, on peut communiquer avec l'API Kubernetes de différentes façons et puisque notre application est déployée dans un pod comme

tout autre application, elle n'aura pas accès à kubectl ou au dashboard de kubernetes.

Pour y remédier , nous avons utilisé un client python pour kubernetes qui permet de faire des requêtes à l'API Kubernetes directement sans passer par kubectl même si l'application est lancée dans un pod.

Notre application décrite dans ce schéma :

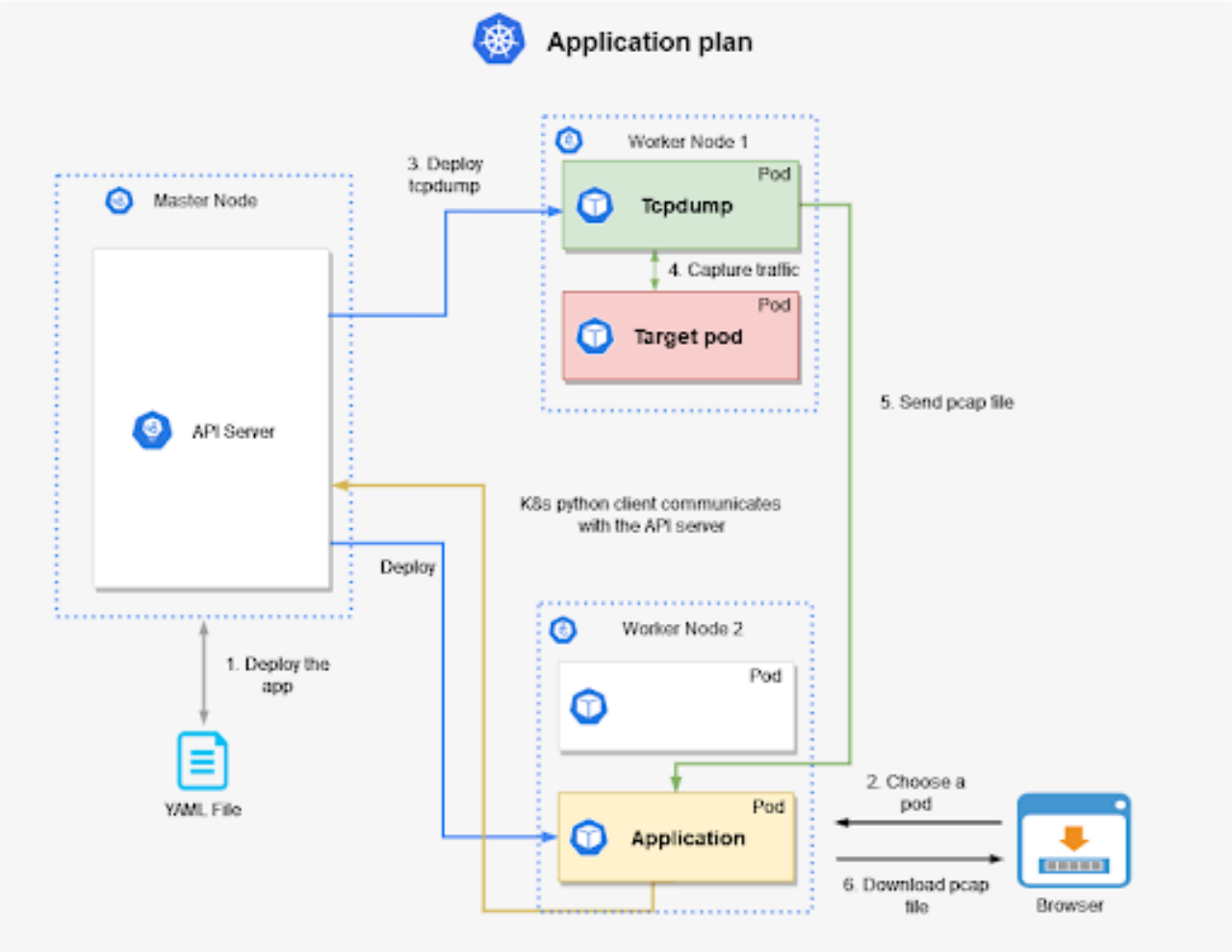

FIG. 4.19 : Plan de l'application.

Comme décrite ci dessus, l'application s'exécute comme suit :

1 . Notre application est déployée manuellement depuis le master node vers un des workers nodes (le choix de l'emplacement de l'application se fait grâce au composant du master "kube-scheduler") à base d'un fichier de configuration avec le langage Yaml.

2 . L'interface graphique de l'application est accédé à partir d'un navigateur web où nous séléctionerons un pod dans l'objectif d'acquérir son trafic réseau.

3 . L'application crée un pod privilégié ayant accès aux interfaces réseau sur le nœud qui exécute le pod ciblé précédemment et qui contient l'outil de capture Tcpdump écoutant l'interface du pod.

4 . Le pod Tcpdump (privilégié) envoie le fichier pcap du pod capturé à l'application pour que l'utilisateur puisse le télécharger (une fois la capture est terminée le pod priviligié sera supprimé).

5 . Les fichiers pcap sont sous la forme "**nom du pod - jour-mois-année - heure-minute-seconde.pcap** ".

4.4.3 Outils et API utilisé

(a) **Python**

On a utilisé Python pour le développement du backend de cette application, en effet Python est un langage de programmation open source multi-plateformes et orienté objet très populaire, qui grâce à ses nombreuses bibliothèques spécialisées, peut être utilisé pour de nombreuses situations d'une manière simple et rapide. Dans notre cas on a utilisé la bibliothèque officielle pour Kubernetes.

Fig. 4.20 : Logo de Python.

(b) **Kubernetes Python Client** C'est la bibliothèque officielle pour Kubernetes qui permet d'interagir avec l'API Kubernetes et y faire toutes les tâches que kubectl peut accomplir, comme la création, la modification, la suppression etc des objets kubernetes.

(c) **Flask**

Flask est un micro framework open-source, simple et très léger de développement web en Python. Il nous permet de lier le backend et le front end de notre application d'une manière simple et efficace.

Fig. 4.21 : Logo du framework Flask.

(d) **Wireshark**

Wireshark est le plus célèbre analyseur des réseaux informatiques, doté d'une interface graphique et de décodage des paquets, il fournit des fonctionnalités de tri et de

filtrage des paquets capturés par pcap qui est un des éléments qui le compose. Ce dernier supporte des centaines de protocoles (HTTP, DNS, ARP, TCP, UDP ...etc) assurant ainsi une analyse efficace pour divers besoins, surtout dans notre cas.

FIG. 4.22 : Logo de Wireshark.

4.4.4 Aperçu de l'application

En accédant à l'application via un navigateur, on a une première interface graphique dans laquelle on peut choisir le cluster dans lequel on va faire la capture du trafic réseau d'un pod, on aura donc à choisir entre un cluster local et un cluster distant.

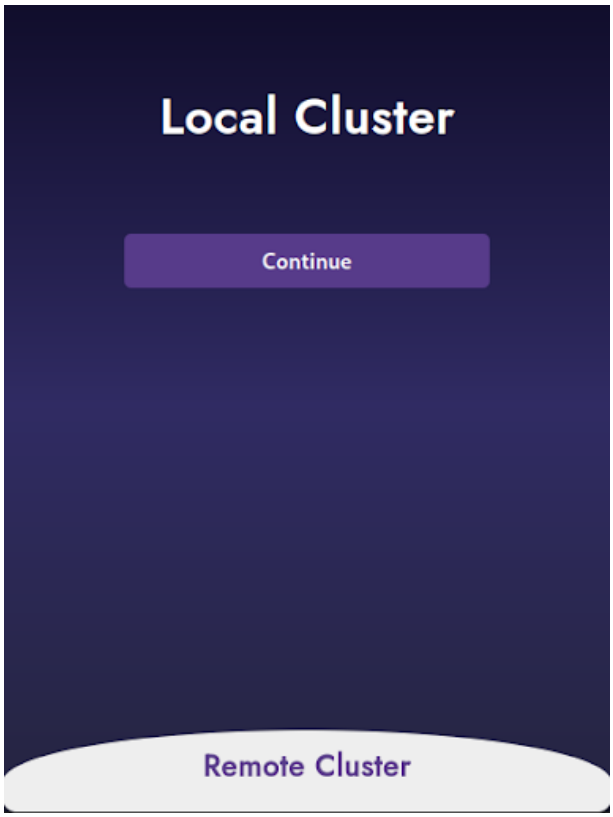

FIG. 4.23 : Page d'authentification "Local cluster".

Si on choisit d'accéder à un cluster distant, il faut alors fournir l'adresse ip du serveur api, le port et le token d'authentification.

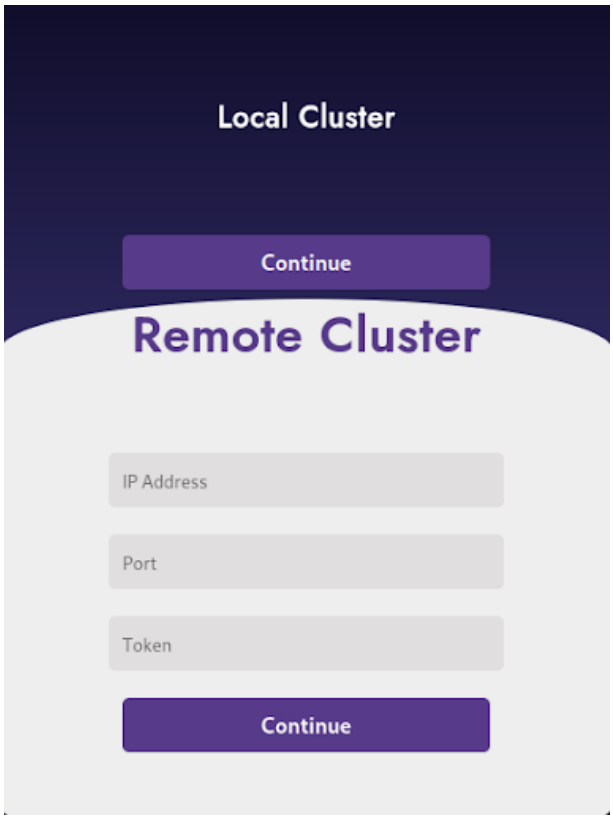

FIG. 4.24 : Page d'authentification "Remote cluster".

Une fois le cluster choisi on sera redirigé vers la deuxième interface dans laquelle s'affiche tous les pods en exécution présent dans le cluster.

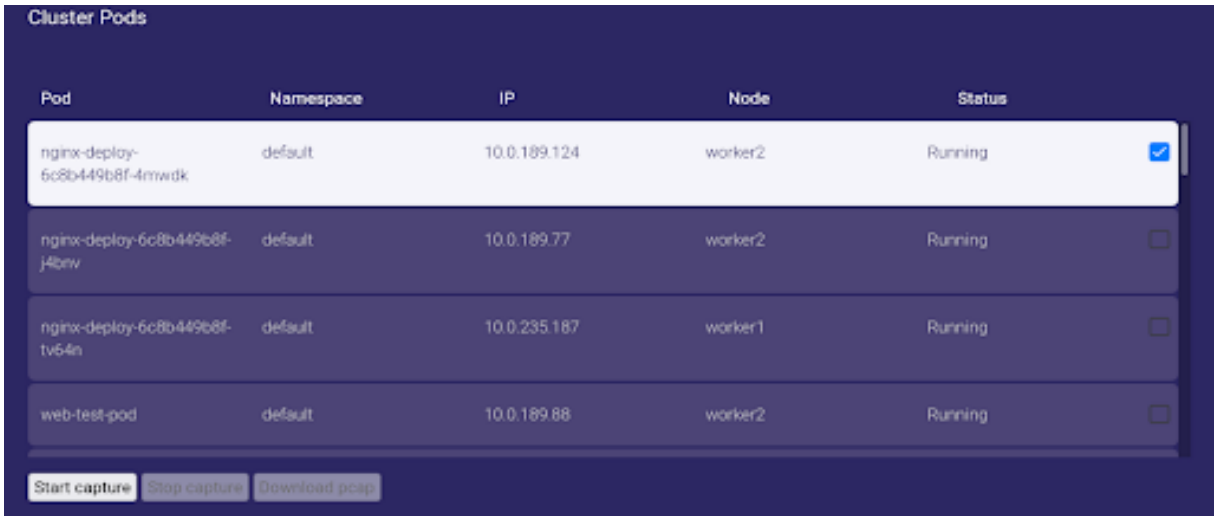

FIG. 4.25 : Page d'accueil affichant la liste des pods actifs.

On peut alors choisir le pod dont on veut capturer le trafic réseau, et lancer la capture en appuyant sur le bouton "start capture". On peut arrêter la capture à tout moment en appuyant sur le bouton "stop capture" et télécharger le fichier pcap en appuyant sur "download capture".

4.5 Test de validation

Pour tester le bon fonctionnement de l'application, nous avons déployé un pod "victim-pod" et nous avons tenté de récupérer son trafic réseau en utilisant notre application. On choisit donc notre pod "victim-pod" et on lance la capture de son flux réseau.

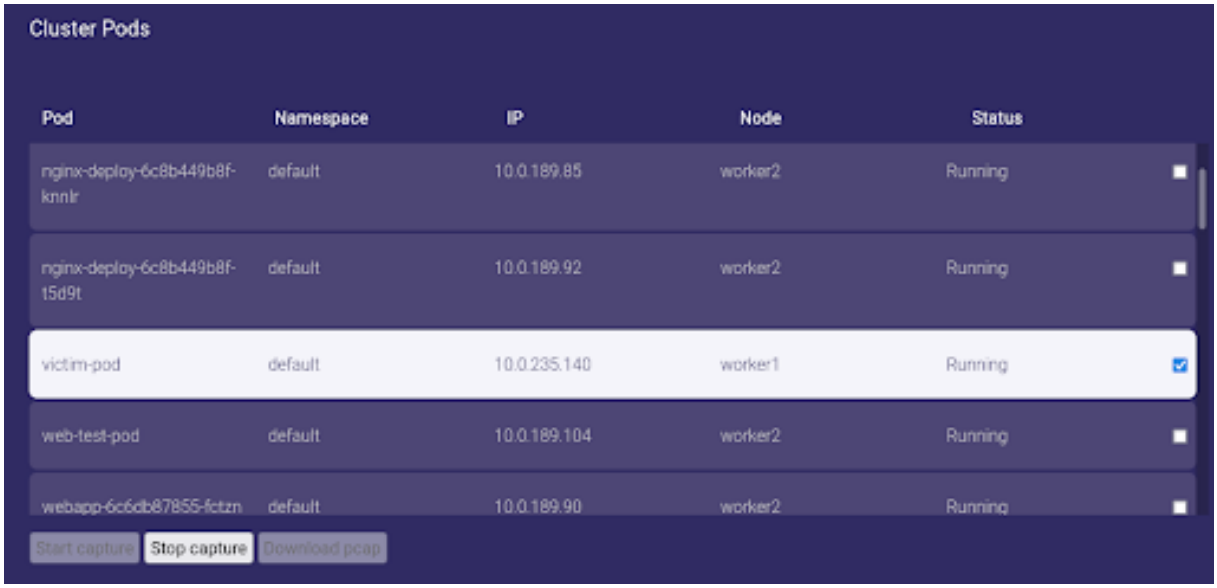

Fig. 4.26 : Séléction et début de capture de pod.

Après une certaine durée, on arrête la capture du flux réseau en appuyant sur "stop capture" puis on télécharge le fichier pcap en appuyant sur "download pcap".

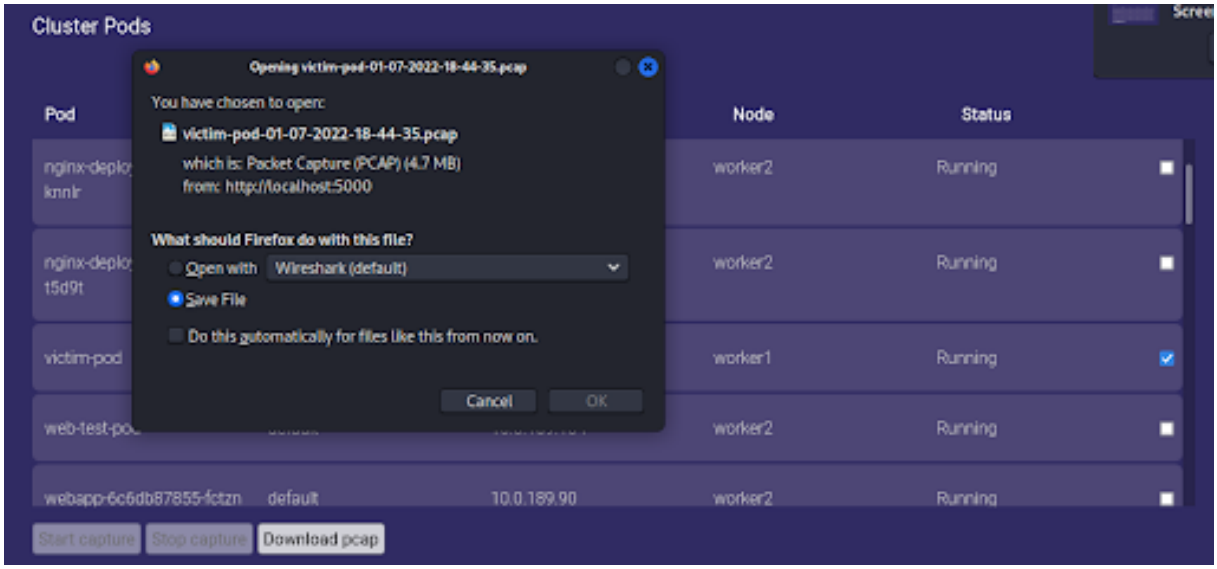

Fig. 4.27 : Arrêt et téléchargement du fichier pcap.

On a le fichier pcap qui se télécharge et qu'on peut analyser par la suite avec Wireshark.

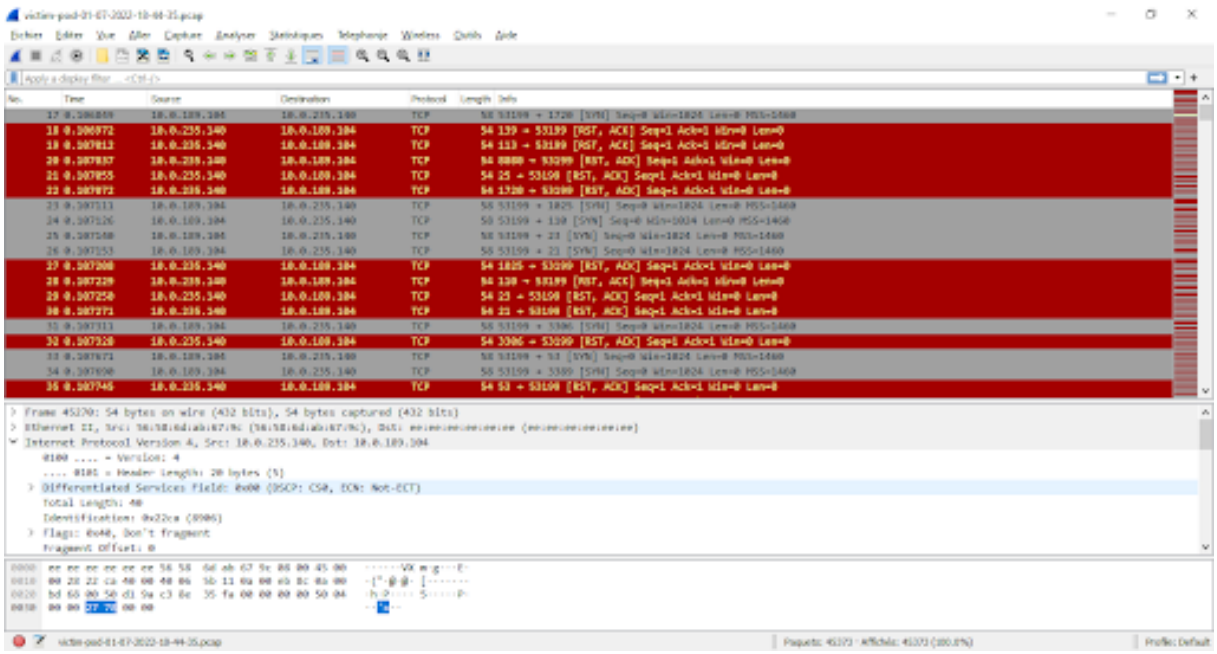

FIG. 4.28 : Liste des paquets.

En analysant le fichier pcap, on peut voir beaucoup de demandes de connexion TCP à destination de notre "victim-pod" qui ne sont pas acquitté avec un ACK (figure 4.29) en effet c'est une connexion TCP half-open, c'est à dire que la connexion TCP n'est pas complète (une connexion TCP ordinaire passe par un processus bien connu sous le nom du ***Three way hanshake*** ou prise de contact à trois voies qui se finalise par une phase d'acquittement avec un paquet ACK).

Cependant, la communication entre la source et le pod "victim-pod" est terminée à chaque fois par la source qui envoie un paquet RST (figure 4.28). Ce comportement peut être normal s'il y a eu une erreur qui fait que la connexion doit être fermée immédiatement, mais dans ce cas le pod "victim-pod" reçoit beaucoup de demandes connexion dans une courte durée, ces connextions cible différent ports et elles sont toutes fermé avec l'envoie d'un paquet RST.

Tout cela laisse dire que le pod "victim-pod" subit un scan de port furtif "stealth scan", probablement lancé à l'aide d'un outil comme nmap.

```
[TCP Segment Len: 0]
Sequence Number: 0    (relative sequence number)
Sequence Number (raw): 2794360483
[Next Sequence Number: 1    (relative sequence number)]
Acknowledgment Number: 0
Acknowledgment number (raw): 0
0110 .... = Header Length: 24 bytes (6)
Flags: 0x002 (SYN)
Window: 1024
[Calculated window size: 1024]
Checksum: 0xcd38 [unverified]
[Checksum Status: Unverified]
Urgent Pointer: 0
Options: (4 bytes), Maximum segment size
 > TCP Option - Maximum segment size: 1460 bytes
```

FIG. 4.29 : Analyse de paquets.

4.6 Conclusion

Dans ce chapitre, nous avons parlé du Network forensic dans Docker et avons montré comment on peut cibler le trafic réseau d'un conteneur avec un scénario comme pratique.

Ensuite, nous avons parlé des différentes méthodes d'installation d'un cluster kubernetes et la manière laquelle avons mis en place notre cluster.

Enfin, nous avons évoqué les différentes solutions pour faire du network forensic dans Kubernetes et avons présenté une solution à notre problématique par une application récupérant le trafic d'un pod de notre cluster avec le concept du pod privilégié.

Conclusion et perspectives

Conclusion générale

Nous vivons dans une ère où les technologies évoluent chaque jour. Que ce soit les technologies cloud Computing ou autres, elles évoluent si vite et nous sommes au point où tout est devenu connecté (ordinateurs, smartphones …).

Hélas, toute évolution entraîne des risques. Parmi ces risques, ceux liés à la sécurité ou à la vie privée. En effet, autres que les incidents de sécurité, on y retrouve des cybercrimes qui sont menés par des utilisateurs malveillants dont l'objectif est la prise du pouvoir, gagner de l'argent ou encore dans le but de se venger.

Afin de pouvoir résoudre ces cybercrimes, le domaine du digital Forensic (investigation légale numérique) est apparu.

Network Forensic est un sous domaine du Digital Forensics, qui permet aux investigateurs de cybercrimes de trouver des preuves et des évidences associées à un incident survenu dans un réseau de communication, et ce par l'interception, l'inspection et l'analyse des paquets.

Dans notre projet, nous avons réussi à répondre à notre problématique, et qui de savoir s'il est possible de faire du network forensics dans les environnement de Cloud Computing. Plus précisément, nous nous sommes concentrés sur les environnements Docker et Kubernetes étant donné qu'ils sont les plus utilisés dans les environnements Cloud.

Nous avons exploré différentes solutions open source pour pouvoir atteindre notre objectif, et nous avons choisi d'utiliser les pods privilégiés pour notre propre solution que nous avons proposée.

Ce projet nous a permis de nous projeter dans le domaine du Network Forensic et de l'appliquer au cloud Computing et ça nous a aussi permis de nous défier en choisissant de travailler sur une problématique aussi récente. Ce projet nous a également permis d'acquérir de nouvelles compétences dans les domaines du Forensic et du Cloud Computing.

Cependant, il n ' y a pas eu encore de solution parfaite qui répond aux besoins de faire du Network Forensic dans un environnement Kubernetes, étant donné que la partie réseau de celui-ci est plus compliqué, et gérer par des tiers partie qui proposent chacun, une diffèrent approche pour gérer cette partie. Il reste donc beaucoup de progrès à faire pour y arriver.

Pour conclure, le domaine du Network Forensic est plein de potentiel et avec l'arrivée des nouvelles technologies comme le Cloud Computing et le nombre de cybercrimes qui augmente, le Network Forensic aura encore plus de défis et de problématiques à résoudre.

Bibliographie

[1] Digital forensics investigation | what is digital forensics. https://www.rfwireless-world.com/Articles/what-is-digital-forensics.html.

[2] Soufiane Tahiri. Digital forensics models. https://resources.infosecinstitute.com/topic/digital-forensics-models/, january 2016.

[3] Karen Kent, Suzanne Chevalier, and Tim Grance. Guide to integrating forensic techniques into incident. *Tech. Rep. 800-86*.

[4] Hcia cloud computing learning guide version 4.0. https://e.huawei.com/en/talent/#/cert/product-details?certifiedProductId=139&authenticationLevel=CTYPE_CARE_HCIA&technicalField=PSC&version=4.0, 2019.

[5] Cloud strategy. https://www.awe.gov.au/sites/default/files/documents/cloud-strategy.pdf, february 2019.

[6] Jennifer Montérémal. Quel type de virtualisation pour optimiser vos ressources informatiques ? https://www.appvizer.fr/magazine/services-informatiques/virtualisation/type-virtualisation, july 2020.

[7] Kubernetes components. https://kubernetes.io/docs/concepts/overview/components/.

[8] Kevin Ku. Advanced fun and profit with packet capture. https://www.akitasoftware.com/blog-posts/advanced-fun-and-profit-with-packet-capture, may 2021.

[9] Cloud cyber attacks : The latest cloud computing security issues. https://www.triskelelabs.com/blog/cloud-cyber-attacks-the-latest-cloud-computing-security-issues.

[10] Tryhackme and Strategos. Intro to digital forensics. http://https://tryhackme.com/room/introdigitalforensics.

[11] Mark Pollitt. A history of digital forensics. In *IFIP International Conference on Digital Forensics*, pages 3–15. Springer, 2010.

[12] Open Learn. A brief history of digital forensics. https://www.open.edu/openlearn/science-maths-technology/digital-forensics/content-section-4.2.

[13] In *Introduction to Network Forensics,version 1.1 Final*, page 10. ENISA, 2019.

[14] Dale Liu. *Cisco router and switch forensics : Investigating and analyzing malicious network activity*. Syngress, 2009.

[15] Interpol. Digital forensics. https://www.interpol.int/How-we-work/Innovation/Digital-forensics.

[16] BIA. Electronically stored information (esi). https://www.biaprotect.com/glossary/esi/.

[17] https://www.autopsy.com/.

[18] Open source digital forensics. https://www.sleuthkit.org/.

[19] Howard Poston. 7 best computer forensics tools [updated 2021]. https://resources.infosecinstitute.com/topic/7-best-computer-forensics-tools/, january 2021.

[20] Ftk® forensic toolkit. https://www.exterro.com/forensic-toolkit.

[21] Volatility foundation. https://www.volatilityfoundation.org/.

[22] https://arsenalrecon.com/products/registry-recon.

[23] https://cellebrite.com/fr/cellebrite-ufed-fr/.

[24] V Cisco. Cisco visual networking index : Forecast and trends, 2017–2022. *White paper*, 1(1), 2018.

[25] Changwei Liu, Anoop Singhal, and Duminda Wijesekera. A logic-based network forensic model for evidence analysis. In *IFIP International Conference on Digital Forensics*, pages 129–145. Springer, 2015.

[26] Ahmad Almulhem. Network forensics : Notions and challenges. In *2009 IEEE International Symposium on Signal Processing and Information Technology (ISSPIT)*, pages 463–466. IEEE, 2009.

[27] Yong Guan. In *Computer and Information Security Handbook (Third Edition)*, 2013.

[28] https://www.tcpdump.org/.

[29] https://www.winpcap.org/.

[30] Man page of tcpdump. https://www.tcpdump.org/manpages/tcpdump.1.html.

[31] Chapter 1. introduction. https://www.wireshark.org/docs/wsug_html_chunked/ChapterIntroduction.html#ChIntroWhatIs.

[32] tshark(1) manual page. https://www.wireshark.org/docs/man-pages/tshark.html.

[33] D.4. dumpcap : Capturing with “dumpcap” for viewing with wireshark. https://www.wireshark.org/docs/wsug_html_chunked/AppToolsdumpcap.html.

[34] ngrep - network grep. http://ngrep.sourceforge.net/usage.html.

[35] Documentation. https://openargus.org/documentation.

[36] Snort®users manual 2.9.16. http://manual-snort-org.s3-website-us-east-1.amazonaws.com/.

[37] Peter Mell, Tim Grance, et al. The nist definition of cloud computing. 2011.

[38] Levels of virtualization implementation. https://www.brainkart.com/article/Levels-of-Virtualization-Implementation_11329/.

[39] Documentation de kubernetes. https://kubernetes.io/fr/docs/home/.

[40] Container orchestration. https://www.ibm.com/cloud/learn/container-orchestration.

[41] ajdaini hatim. Les différents objets et composants de kubernetes. https://devopssec.fr/article/differents-objets-composants-kubernetes, july 2019.

[42] National vulnerability database. https://nvd.nist.gov/vuln/detail/CVE-2019-5736.

[43] https://www.cvedetails.com/cve/CVE-2017-7308/.

[44] Wang Wei. Docker hub suffers a data breach, asks users to reset password. https://thehackernews.com/2019/04/docker-hub-data-breach.html, april 2019.

[45] Catalin Cimpanu. 17 backdoored docker images removed from docker hub. https://www.bleepingcomputer.com/news/security/17-backdoored-docker-images-removed-from-docker-hub/, june 2018.

[46] Chris Doman. Botnet deploys cloud and container attack techniques. https://www.cadosecurity.com/botnet-deploys-cloud-and-container-attack-techniques/, january 2021.

[47] RedLock CSI Team. Lessons from the cryptojacking attack at tesla. https://redlock.io/blog/cryptojacking-tesla, february 2018.

[48] Catalin Cimpanu. Weight watchers it infrastructure exposed via no-password kubernetes server. https://www.bleepingcomputer.com/news/security/weight-watchers-it-infrastructure-exposed-via-no-password-kubernetes-server/, june 2018.

[49] Doug Olenick. Docker api vulnerability allows hackers to mine monero. https://www.cadosecurity.com/botnet-deploys-cloud-and-container-attack-techniques/, march 2019.

[50] Docker security. https://docs.docker.com/engine/security/.

[51] VB Staff. Aqua security : 50% of new docker instances attacked within 56 minutes. https://venturebeat.com/2021/06/28/aqua-security-50-of-new-docker-instances-attacked-within-56-minutes/, june 2021.

[52] Network settings. https://docs.docker.com/engine/reference/run/#network-settings.

[53] D4Vinci. Pyflooder. https://github.com/D4Vinci/PyFlooder, june 2021.

[54] Use volumes. https://docs.docker.com/storage/volumes/.

[55] Saeid Bostandoust. Kubernetes installation methods the complete guide. https://itnext.io/kubernetes-installation-methods-the-complete-guide-1036c860a2b3, july 2021.

[56] minikube start. https://minikube.sigs.k8s.io/docs/start/, june 2022.

[57] https://kind.sigs.k8s.io/.

[58] https://kubernetes.io/docs/reference/setup-tools/kubeadm/, september 2021.

[59] https://github.com/kubernetes-sigs/kubespray.

[60] https://github.com/rancher/rke.

[61] https://aws.amazon.com/fr/eks/.

[62] https://cloud.google.com/kubernetes-engine.

[63] https://azure.microsoft.com/en-us/services/kubernetes-service/.

[64] Kubernetes the hard way. https://github.com/kelseyhightower/kubernetes-the-hard-way.

[65] Kubernetes the hard way on virtualbox. https://github.com/mmumshad/kubernetes-the-hard-way.

[66] What is calico ? https://projectcalico.docs.tigera.io/about/about-calico.

[67] kokotap : Tapping pod traffic to vxlan interface. https://github.com/redhat-nfvpe/kokotap.

[68] tc-mirred(8) — linux manual page. https://man7.org/linux/man-pages/man8/tc-mirred.8.html.

[69] Stephen J. Bigelow. Réseau, sécurité, observabilité : l'essentiel sur ebpf. https://www.lemagit.fr/conseil/Reseau-securite-observabilite-lessentiel-sur-eBPF, august 2021.

[70] Lavanya Chockalingam. What is ebpf and why does it matter for observability ? https://newrelic.com/blog/best-practices/what-is-ebpf, april 2021.

[71] ebpf-based networking, observability, security. https://cilium.io/.

[72] Instantly troubleshoot your applications on kubernetes. https://pixielabs.ai/.

Résumé

Dans ce projet nous nous sommes initié au domaine du Network Forensic et avons expérimenté les outils utilisés dans ce domaine. En effet, s. L'occurrence d'un cybercrime n'est pas une question de 'si' mais de 'quant'.

Network forensic est un domaine qui aspire à aider les investigateurs de cybercrimes à trouver des preuves associées à un incident dans les réseaux de communication, et ce par la collecte, l'inspection et l'analyse des paquets. Cependant, avec l'évolution des technologies de virtualisation et la montée en puissance du cloud computing, de nouveaux risques et défis de sécurité informatique sont apparus.

L'objectif de notre travail est de voir dans un premier temps à quel point les outils d'interception de paquets actuels peuvent être utilisés pour intercepter des paquets dans des systèmes basés sur la virtualisation au niveau OS (OS-Level). Puis, dans un second temps, essayer d'adapter, configurer et déployer une solution cloud computing basée sur les outils open sources existants associés au Network Forensic pour permettre une interception légale des paquets dans les environnements Docker et Kubernetes.

Mots clés : Network Forensic, Cloud Computing, Network interception, OS-level Virtualization, Containers, Open source, Docker, Kubernetes.

Abstract

In this project we were introduced to the field of Network Forensic and experimented with the tools used in this field. Indeed. The occurrence of a cybercrime is not a question of 'if' but of 'when'.

Network forensic is a field that aspires to help cybercrime investigators find evidence associated with an incident in communication networks through the collection, inspection and analysis of packets. However, with the evolution of virtualization technologies and the rise of cloud computing, new IT security risks and challenges have emerged.

The objective of our work is to first see how current packet interception tools can be used to intercept packets in systems based on virtualization at OS level (OS-Level). Then, in a second step, try to adapt, configure and deploy a cloud computing solution based on existing open source tools associated with Network Forensic to allow legal interception of packets in Docker and Kubernetes environments.

Keywords : Network Forensic, Cloud Computing, Network interception, OS-level Virtualization, Containers, Open source, Docker, Kubernetes.

ملخص

في هذا المشروع ، تعرفنا على مجال الطب الشرعي الشبكي وجربنا الأدوات المستخدمة في هذا المجال. في الواقع ، ليس وقوع الجريمة الإلكترونية مسألة "إذا" بل "متى".

الطب الشرعي للشبكة هو مجال يطمح إلى مساعدة محققي الجرائم الإلكترونية في العثور على أدلة مرتبطة بحادث في شبكات الاتصالات من خلال جمع وفحص وتحليل الحزم. ومع ذلك ، مع تطور تقنيات المحاكاة الافتراضية وظهور الحوسبة السحابية ، ظهرت مخاطر وتحديات جديدة لأمن تكنولوجيا المعلومات.

اإلهدف من عملنا هو معرفة كيفية استخدام أدوات اعتراض الحزم الحالية لاعتراض الحزم في الأنظمة القائمة على المحاكاة الافتراضية على مستوى نظام التشغيل (مستوى نظام التشغيل). بعد ذلك ، في الخطوة الثانية ، حاول تكييف وتهيئة ونشر حل الحوسبة السحابية استنادًا إلى الأدوات الحالية مفتوحة المصدر المرتبطة بشبكة الطب الشرعي للسماح بالاعتراض القانوني للحزم في بيئات Docker و .Kubernetes

---

كلمات مفتاحية : الطب الشرعي للشبكة ، الحوسبة السحابية ، اعتراض الشبكة ، المحاكاة الافتراضية على مستوى نظام التشغيل ، الحاويات ، المصدر المفتوح ، Docker ، .Kubernetes

---