ASSEMBLAGE SEMI-AUTOMATIQUE DE LOGS POUR ENTRAINEMENT D'OPÉRATEURS DE SIEM

SEMI-AUTOMATED LOG SEQUENCE GENERATION FOR SIEM OPERATOR TRAINING

Une thèse soumise à la Division des études supérieures

du Collège militaire royal du Canada

par

Sébastien Ménard

Capitaine

en vue de l'obtention du diplôme

Maîtrise ès sciences appliquées

Août 2016

REMERCIEMENTS

J’aimerais remercier tous mes collègues du Laboratoire de sécurité informatique du département de génie électrique et de génie informatique pour m’avoir aidé et encouragé pendant mes deux années d’études au Collège militaire royal du Canada. J’aimerais particulièrement remercier mon superviseur, Sylvain Leblanc pour m’avoir aidé à réaliser cette thèse.

RÉSUMÉ

Les logiciels de sécurité de l'information et de gestion d'événements de sécurité (SIEM) permettent l'agrégation des informations générées par tous les senseurs de sécurité d'un réseau dans le but d'obtenir une visibilité optimale sur les alertes de sécurité. Les logiciels SIEM sont devenus les outils principaux de gestion d'information pour organiser les logs et les alertes de sécurité pour le personnel des centres d'exploitation de réseau (CER). L'entraînement du personnel qui surveille et défend de vastes réseaux informatiques est un défi récurrent qui est coûteux en temps et en argent. Cette recherche contribue au développement de méthodes d'entraînement qui ne reposent pas sur l'utilisation d'équipes de test de pénétration de réseaux informatiques. Le domaine de recherche de cette thèse est le développement de nouvelles techniques pour entraîner des défenseurs de réseaux, particulièrement les opérateurs de logiciel SIEM. L'utilisation de connecteur de reprise pour entraîner des opérateurs de logiciel SIEM est une alternative viable. Cependant, l'organisation de séquences des logs incorporant des scénarios d'entraînement intéressants et utilisables par des connecteurs de reprises n'est pas triviale. Cette recherche développe des techniques et propose une architecture pour assembler des séquences de logs utilisables pour l'entrainement d'opérateur SIEM basée sur des moyens semi-automatiques.

ABSTRACT

Security information and Event management (SIEM) software enable the aggregation of information generated by all security sensors within a defended network providing optimal visibility on security alerts. SIEMS have become the main information management tool used by system defenders to organize logs and security alerts for an organization's Network Operation Center (NOC). The training of system defenders is a recurring challenge, which is costly in terms of both money and time. This research contributes to the development of training methods that does not depend on network penetration teams. This research is based on the development of new techniques to train network defenders, particularly SIEM operators. We intend to develop a new approach that does not rely on the presence of a penetration testing team. The use of SIEM replay connectors is a viable alternative to train SIEMS operators. However, organizing logs characteristic of malicious scenarios in a way which can be useable by SIEM replay connectors is not trivial. This research has developed techniques and proposes an architecture to assemble logs useable by replay connectors to train SIEM operators in a semi-automatic fashion.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTES DES FIGURES

LISTE DES SYMBOLES, ABRÉVIATIONS ET ACRONYMES

ADP	Adobe Portable Format
CEF	Common Event Format
CER	Centres d'exploitation de réseau
CSTE	Collaborative Security Test Environment
CSV	Comma-separated values
DP	Device Profiler
DNS	Domain Name Server
ECTS	Environnement collaboratif de test de sécurité
ESM	Enterprise Security Manager
FAC	Forces armées canadiennes
HTTP	Hypertext Transfer Protocol
IEG	Integrated Exchange Gateway
IP	Internet Protocol
JSON	JavaScript Object Notation
MAST	Malicious Activity Simulation Tool
MDN	Ministère de la Défense nationale
SDI	Système de détection d'intrusion
SEV	Security Engineering Validation
SIEM	Sécurité de l'information et de gestion d'événements de sécurité
SMTP	Simple Mail Transfer Protocol
SP	Service Pack
SQL	Structured Query Language
URL	Uniform Resource Locator
VNE	Vulnerability and Exposure
XML	Extensible Markup Language

Chapitre 1 : Introduction

1.1 Introduction

Les logiciels de sécurité de l'information et de gestion d'événements de sécurité (SIEM[1]) permettent l'agrégation des informations générées par tous les senseurs de sécurité d'un réseau dans le but d'obtenir une visibilité optimale sur les alertes de sécurité. Les logiciels SIEM sont devenus les outils principaux de gestion d'information pour organiser les logs[2] pour le personnel des centres d'exploitation de réseau (CER) [1]. L'entraînement du personnel qui surveille et défend de vastes réseaux informatiques est un défi récurrent qui est coûteux en temps et en argent. De plus, les organisations ont l'obligation de réentraîner les défenseurs de leurs réseaux régulièrement pour préserver leurs expertises à un haut niveau [2]. Le domaine de recherche de cette thèse est le développement de nouvelles techniques pour entraîner des défenseurs de réseaux, particulièrement les opérateurs de logiciel SIEM. Cette recherche contribue au développement de méthodes d'entraînement qui ne reposent pas sur l'utilisation d'équipes de test de pénétration de réseaux informatiques.

La prestation d'entraînement cybernétique est un problème complexe pour les organisations militaires dû au nombre important d'opérateurs à entraîner et à la complexité des réseaux à gérer. Traditionnellement, les organisations militaires utilisent des équipes de test de pénétration de réseaux informatiques surnommées en anglais *« Red Team »*, ces équipes sont en charge de conduire des attaques contrôlées vers un réseau protégé dans le but de tester son dispositif de sécurité. L'utilisation de ces équipes est une approche réaliste, efficace et exhaustive pour fournir des scénarios d'entraînement pour le personnel qui a la tâche de défendre un réseau [3]. Cependant, ces équipes sont rares et dispendieuses, c'est principalement la raison pour laquelle des démarches de recherche ont été entreprises pour développer des méthodes d'entraînement alternatives plus économiques. Wong a développé un cadre d'entraînement renouvelable pour l'éducation des opérateurs de logiciel SIEM [4]. Sa solution est basée sur l'utilisation de connecteurs de reprise appelés en anglais *replay connectors*, une fonctionnalité qui est généralement disponible dans une implémentation de logiciel SIEM. Ces connecteurs permettent de rejouer des séquences de logs collectées d'un réseau informatique sur la console

---

[1] L'acronyme SIEM vient du terme d'anglais *Security Information and Event Management* (SIEM). L'acronyme SIEM est reconnu et couramment utilisé dans la littérature et l'industrie de la sécurité informatique. Afin de simplification, le terme SIEM sera utilisé dans ce mémoire.

[2] Le terme log dans ce mémoire désigne une entrée dans un journal d'activité informatique; certains pourraient parler de « log entries » ou « d'entrées de journal », mais nous utilisons le terme log pour se conformer à la définition communément acceptée de tels logs représentent parfois les alertes qui sont générées par des senseurs et collectées par un SIEM.

du SIEM, ce qui est fait habituellement pour tester ses différentes configurations. Wong a démontré que ces connecteurs de reprise peuvent être utilisés avec succès pour entraîner des opérateurs de logiciel SIEM. Wong a développé un cadre d'entraînement pour les opérateurs de logiciel SIEM qui repose sur l'utilisation des connecteurs de reprise. Il a cependant identifié certains points faibles avec son approche, particulièrement au niveau de l'utilisation d'un processus d'ingénierie inverse ayant pour but de reconstruire un réseau à partir de séquence de logs liée à des scénarios d'entraînement. L'objectif étant de reconstruire un réseau en laboratoire dans le but de configurer un SIEM pour surveiller ce réseau et pour détecter les incidents quand les logs liés aux scénarios d'entraînement sont rejoués. Il a conclu que le processus d'ingénierie inverse utilisé est non intuitif pour l'entraînement des opérateurs. Cette thèse adresse cette limitation et est une continuité du travail accompli par Wong pour développer de nouvelles approches d'entraînement pour les opérateurs de logiciel SIEM.

La partie gauche de la Figure 1-1 illustre les concepts reliés à l'entraînement des opérateurs de logiciel SIEM quand une équipe de type *« Red Team »* est utilisée, ce qui est la méthode la plus performante pour entraîner des opérateurs. Un réseau typique surveillé à l'aide d'un logiciel SIEM comprend plusieurs différents types de senseurs, qui surveillent certaines activités qui ont lieu sur le réseau et qui génèrent des logs en concordance avec celles-ci. Un exemple typique de senseur est un système de détection d'intrusion (SDI), qui génère des alertes relatives à des détections de menaces ou d'activités malicieuses potentielles. Le logiciel SIEM utilise des logiciels appelés connecteurs pour collecter les logs provenant de ces senseurs dans le but de les agréger et de les normaliser dans une base de données centralisée. Ainsi, un logiciel SIEM permet d'agréger l'information provenant des multiples senseurs existants sur un réseau dans le but d'obtenir une visibilité optimale sur les événements. La plupart des logs générés par les senseurs proviennent d'activités courantes qui ont lieu sur le réseau et ne sont pas nécessairement reliés à des activités malicieuses ou des menaces potentielles. Ces logs reliés à des activités courantes demandent quand même à être analysés par les opérateurs de logiciel SIEM pour y découvrir d'éventuelles activités suspicieuses. Ils constituent le bruit de fond essentiel à l'entraînement des opérateurs de logiciel SIEM.

Une équipe de type *« Red Team »* peut être utilisée pour générer des événements malicieux contrôlés sur un réseau opérationnel dans le but de tester l'habilité des opérateurs de logiciel SIEM à détecter ces attaques. Les activités créées par les actions de l'équipe « *Red Team* » sont détectées par les senseurs et des logs sont générés pour celles-ci. Ces logs qui sont générés par les activités de l'équipe de testeurs sont essentiels pour entraîner les opérateurs de logiciel SIEM. Ces logs qui sont collectés et gérés par le logiciel SIEM, peuvent être par la suite utilisés par l'opérateur SIEM pour détecter les activités malicieuses de l'équipe *« Red Team »*. Cette méthode d'entraînement est la plus performante dû à l'utilisation d'un réseau opérationnel, qui est habituellement surveillé par les opérateurs qui

reçoivent l'entraînement, ce qui garantit un haut niveau de réalisme. L'utilisation d'un environnement opérationnel facilite la conception de l'entraînement, car l'émulation ou la réplication des activités courantes d'un réseau qui sont essentielles pour l'entraînement n'ont pas besoin d'être produite. Les activités courantes, qui sont produites la plupart du temps par l'action des utilisateurs du réseau, fournissent l'arrière-plan qui est nécessaire pour l'entraînement. Sans cet arrière-plan, qui sert de bruit de fond, il n'est pas possible de fournir un entraînement efficace aux opérateurs de logiciel SIEM.

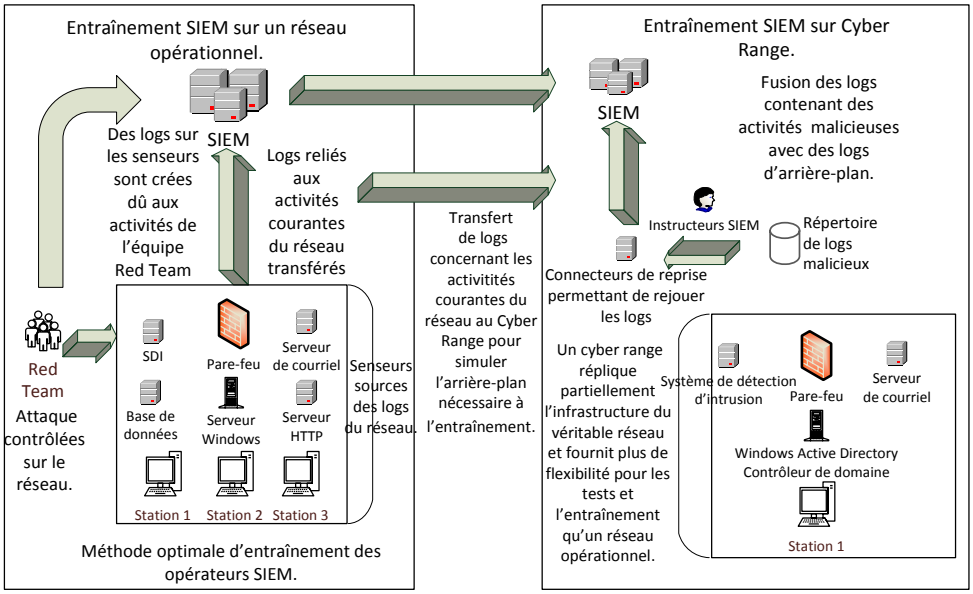

Figure 1-1. Méthodes d'entraînement pour opérateurs de logiciel SIEM

Le côté droit de la Figure 1-1 illustre un scénario d'entraînement pour des opérateurs de logiciel SIEM qui est un compromis par rapport au scénario d'entraînement optimal. Dans ce cas-ci, un réseau de simulation appelé en anglais *Cyber Range* est utilisé pour exécuter des scénarios d'entraînement. Un réseau *Cyber Range* peut être considéré comme une réplique partielle d'un véritable réseau. Le réseau *Cyber Range* inclut une excellente approximation de l'instance du logiciel SIEM déployée sur le véritable réseau qui est émulé. Un réseau *Cyber Range* réplique également partiellement les senseurs qui sont installés sur le véritable réseau afin d'émuler suffisamment de logs d'arrière-plan. Il est possible de récolter des logs reliés aux activités courantes du véritable réseau et de les ajouter aux autres logs déjà émulés par les senseurs du *Cyber Range.* Cette méthode est particulièrement utile lorsque le nombre de logs d'arrière-plan émulés n'est pas suffisant pour produire un environnement convenable pour l'entraînement; Quand nous obtenons un arrière-plan suffisamment riche, une équipe de type *« Red Team»* peut par la suite être utilisée pour entraîner les opérateurs sur le réseau *Cyber Range*. L'équipe peut attaquer le réseau test pour tester les dispositifs défensifs sans aucune restriction car un réseau *Cyber Range* est un environnement test qui peut être remis à son état initial au besoin ou à la

fin de la séance d'entraînement. Cependant, comme précédemment mentionné, les équipes de type *« Red Team »* sont rares et onéreuses. Une solution pour émuler leurs attaques, est d'assembler ou de "fusionner" les logs qui sont produits par les activités de l'équipe *« Red Team »* avec les logs qui servent d'arrière-plan dans le but de les rejouer à l'aide de la fonctionnalité des connecteurs de reprise. En premier lieu, nous devons collecter les logs qui sont liés aux activités malicieuses générées des équipes *« Red Team ».* Ces séquences de logs, qui sont liées à des activités malicieuses, peuvent potentiellement être récoltées d'un véritable réseau, ou générées minutieusement pour un entraînement spécifique pour par la suite être stockées dans un répertoire et ou au besoin une base de données. Nous devons par la suite les incorporer c'est-à-dire les "fusionner" dans une séquence de logs qui sont reliées aux activités courantes d'un réseau pour une période limitée pour que l'entraînement soit efficace. Le produit final sera une séquence de logs incorporant les activités malicieuses qui sont d'intérêt pour l'entrainement d'opérateurs avec les logs liés à des activités courantes du réseau. Cette séquence de logs peut par la suite être rejouée par des connecteurs de reprise pour entraîner les opérateurs à configurer le logiciel SIEM pour détecter les activités malicieuses. La "fusion" consiste à assembler une nouvelle séquence de logs et la rendre cohésive pour qu'elle soit utilisable pour l'entraînement. La séquence de logs assemblée va inclure un scénario d'entraînement camouflé dans un arrière-plan adéquat, elle peut par la suite être rejouée durant différentes sessions d'entraînement sur le réseau *Cyber Range*.

L'entraînement des défenseurs de réseau est un défi opérationnel récurrent pour les organisations militaires. L'utilisation d'équipe de type *« Red Team »* fait partie de la solution optimale d'entraînement, cependant elles sont des ressources rares et onéreuses. Des démarches de recherche sont présentement entreprises pour développer des méthodes pour entraîner des défenseurs de réseaux qui ne reposent pas uniquement sur l'utilisation d'équipe de type *« Red Team »*.

1.2 Difficultés reliées à l'assemblage de logs

L'utilisation de connecteurs de reprise SIEM pour rejouer les logs est une option viable pour entraîner les administrateurs de logiciel SIEM à configurer les divers composantes de celui-ci et à détecter des activités malicieuses sur un réseau [4]. Cependant, l'organisation et l'assemblage des séquences de logs comportant des scénarios intéressants dans le but de les rejouer à l'aide de connecteurs de reprise pour l'entraînement n'est pas triviale.

Les séquences de logs comportant des activités malicieuses se doivent d'être récoltées ou générées. Ces séquences doivent être par la suite adaptées au réseau d'entraînement si elles ne sont pas récoltées directement de celui-ci. Pour rendre les entrées logs utilisables, nous devons les adapter aux réseaux utilisés, un exemple simple d'adaptation est de changer les champs des adresses IP des logs pour une adresse IP qui existe sur notre réseau d'entraînement. Le log

suivant provenant du SDI *Suricata* suite à la détection d'un balayage réseau illustre le problème.

Tableau 1-1. Adaptation d'un log


```
{"timestamp":"2016-02-17T11:48:19.727182-
0500","flow_id":32075200,"in_iface":"eno16777736","event_typ
e":"alert","src_ip":"192.168.1.2","src_port":80,"dest_ip":"1
92.168.1.3","dest_port":49167,"proto":"TCP","alert":{"action
":"allowed","gid":1,"signature_id":2210021,"rev":2,"signatur
e":"SURICATA STREAM ESTABLISHED retransmission packet before
last ack","category":"","severity":3}}
```


Les champs du log surligné en rouge se doivent d'être personnalisés pour rendre le log réutilisable sur un réseau d'entraînement pour un connecteur de reprise. Dans ce cas, nous devons adapter le nom de l'interface (`in_iface`), les ports et les adresses IP de la source et de la destination du balayage à notre réseau d'entraînement. Le champ '*timestamp*' est le temps où l'information reporté par le log a lieu, il devra aussi être personnalisé de manière particulière. Ce processus de personnalisation des logs se doit d'être semi-automatisé pour faciliter la création de séquences de logs utilisables pour entraîner des opérateurs de logiciel SIEM. La semi-automatisation est nécessaire pour s'assurer que la solution soit utilisable de façon générale et que le processus d'assemblage de logs soit efficace. Le processus ne peut pas être totalement automatisé car des entrées d'informations provenant de l'utilisateur sont parfois nécessaires. L'utilisateur pourrait par exemple vouloir personnaliser un élément du scénario d'entraînement inséré, en choisissant l'usager du réseau qui est touché par un événement malicieux. Sans ce processus de personnalisation, les séquences de logs rejouées n'auront pas de sens lorsqu'elles seront rejouées sur un réseau d'entraînement.

Un autre problème à solutionner pour obtenir des séquences de logs utilisables pour entraîner des opérateurs est d'être en mesure d'incorporer les séquences de logs reliées à des scénarios d'entraînement à des séquences de logs d'arrière-plan comportant des activités courantes d'un réseau. Ces séquences d'arrière-plan sont nécessaires à l'entraînement car elles fournissent le bruit d'arrière-fond toujours présent sur un réseau actif. Les séquences de logs comportant le scénario d'entraînement doivent être insérées dans l'arrière-plan, en préservant la cohésion du produit fini. L'analogie que nous pourrions utilisée est que nous avons besoin d'insérer un nouveau chapitre dans un livre déjà écrit, tout en préservant la cohésion de l'histoire. Nous avons surnommé ce processus la "fusion". Le développement de techniques pour fusionner les logs servant d'arrière-plan avec les logs reliés à un scénario d'entraînement par des moyens semi-automatiques facilite l'assemblage des séquences de logs utilisables pour l'entraînement. Nous pouvons ainsi assembler un scénario intéressant et être en mesure de le rejouer dans un arrière-plan que nous pouvons altérer à volonté et ainsi être en mesure de fournir un entraînement non redondant.

1.3 Objectif de la recherche

Notre objectif de recherche est l'investigation des méthodes et des techniques nécessaires pour assembler des séquences de logs. Ces logs contiendront des scénarios d'entraînement et des séquences d'activités courantes servant d'arrière-plan et ils seront utiles pour l'entraînement des opérateurs de logiciel SIEM.

Notre démarche de recherche propose une architecture pour faciliter la génération de séquence de logs utilisables pour l'entraînement des opérateurs de logiciel SIEM. La priorité de notre recherche a été axée sur les aspects suivants de l'architecture proposée :

1. Répertoire contenant les séquences de logs qui sont reliées à des scénarios d'entraînement pour les opérateurs de SIEM. Notre objective est de démontré qu'il est possible d'utiliser un répertoire pour stocker les séquences de logs intéressantes pour l'entraînement en support de notre démarche de recherche. La génération de ces séquences est un domaine de recherche en soit et ne fait pas partie de cette thèse.
2. Développement de méthodes et de techniques pour adapter une séquence de logs reliées à des scénarios intéressants pour l'entraînement. L'objectif étant de rendre ces séquences de logs entreposées dans un répertoire rejouable sur un réseau d'entraînement à l'aide de moyens semi-automatiques.
3. Développement de méthodes semi-automatiques pour insérer et fusionner une séquence de logs reliée à un scénario d'entraînement, qui a été personnalisée pour le réseau d'entraînement utilisé, avec une séquence de logs reliée à des activités courantes d'un réseau qui sert d'arrière-plan.

La démarche de recherche a inclus le développement d'un prototype implémentant les trois éléments précédents. Il est important de noter que notre démarche de recherche ne touche pas au domaine de la création de séquence de logs comportant des événements intéressants pour l'entraînement à l'aide de moyens synthétiques ou semi-synthétiques.

Une validation de notre démarche de recherche a été également accomplie à l'aide de notre prototype que nous avons utilisé pour assembler une séquence de logs reliée à un scénario d'entraînement non-trivial et réaliste à différentes séquences de logs servant d'arrière-plan adéquates.

1.4 Contributions

La contribution de cette recherche est de développer des techniques pour : :

1. La personnalisation de séquences de logs reliées à des scénarios d'entraînement qui sont générées ou qui sont récoltées d'un réseau. La

personnalisation est faîte de manière semi-automatique, dans le but d'adapter la séquence de logs à un réseau de destination vers lequel elle pourra être rejouée par des connecteurs de reprise dans un but d'entraînement.

2. L'insertion et la fusion de séquences de logs reliées à des scénarios d'entraînement dans une séquence de logs servant d'arrière-plan collecté d'un réseau actif. L'objectif de la fusion et de l'insertion étant de produire une nouvelle séquence de logs contenant un scénario d'entraînement qui est incorporée le plus efficacement possible avec une séquence de logs servant d'arrière-plan. La séquence fusionnée doit avoir une cohésion suffisante pour être en mesure d'être utilisable pour entraîner des opérateurs de logiciel SIEM.

Ces contributions vont directement faciliter à la création de séquences de logs dans un but d'entraînement des opérateurs de logiciel SIEM à l'aide de connecteurs de reprise. Le développement de nouvelles techniques d'entraînement d'opérateurs de logiciel SIEM permettra à des organisations comme les Forces armées canadiennes(FAC) d'entraîner plus facilement son personnel qui a la tâche de défendre ses réseaux de grande taille.

1.5 Plan du mémoire

Ce mémoire de maîtrise est constitué de cinq chapitres. Le chapitre 2 est une revue de la littérature qui présente des concepts reliés aux réseaux d'entraînement *Cyber Range* et aux logiciels SIEM. Le chapitre 2 revoit également les méthodes d'entraînement d'opérateurs de logiciel SIEM existantes et couvre certains éléments applicables à notre recherche provenant de domaines connexes. Le chapitre 3 propose une architecture servant à assembler des séquences des logs utilisables pour entraîner des opérateurs de logiciels SIEM, pour par la suite décrire les processus de personnalisation, d'insertion et de fusion. Le chapitre 4 décrit l'implémentation de notre prototype et la validation de notre démarche de recherche. Le chapitre 5 est une brève conclusion qui discute des résultats de la recherche et du travail futur à accomplir.

Chapitre 2 : Revue de la littérature

2.1 Introduction

Lors du chapitre 1, nous avons discuté de la nécessité d'entraîner les opérateurs de CER à opérer un logiciel SIEM pour être en mesure de protéger efficacement de vastes réseaux informatiques. Ce chapitre est une introduction aux différents concepts jugés nécessaires à la compréhension de la présente démarche de recherche. En premier lieu, nous allons présenter les concepts reliés au réseau *cyber-range*, réseau de simulation informatique couramment utilisé pour l'entraînement cybernétique. En deuxième lieu, nous allons discuter brièvement de l'architecture et des fonctionnalités principales des logiciels SIEM et de leurs rôles primordiaux dans la défense d'un réseau de grande envergure. Nous allons également brièvement discuter des normes de format de logs utilisés par les logiciels SIEM, qui sont un facteur à considérer pour notre démarche de recherche. Nous allons par la suite couvrir les exigences d'entraînement pour les opérateurs de logiciel SIEM et présenter des méthodes d'entraînement existantes. Pour conclure, nous allons discuter de quelques éléments provenant de domaines de recherche connexes qui sont applicables à notre projet.

2.2 Concepts reliés aux réseaux d'entraînement *cyber-range*

Cette section discute brièvement des concepts reliés au réseau de simulation cybernétique de type *cyber-range*. La dépendance des organisations militaires contemporaines aux réseaux informatiques et à leur opérationnalisation pose plusieurs défis. Un de ces défis est de fournir la prestation d'entraînement cybernétique réaliste de manière récurrente. Un autre défi important est de surmonter les capacités limitées à tester de manière précise et rapide des systèmes d'information distribués. Ces deux problèmes se doivent d'être absolument surmontés pour être en mesure de maintenir un réseau sécuritaire.

Pour être en mesure de solutionner ces problèmes complexes, les organisations militaires ont développé des réseaux de simulation appelés en anglais cyber-range. Ces réseaux simulés fournissent un environnement réaliste pour exécuter les tests de sécurité et sont une plate-forme de choix pour l'entraînement du personnel à la sécurité informatique. Un réseau de type cyber-range inclut la simulation de plusieurs types de serveurs et de services réseaux, de systèmes de défense et de stations de travail. Ils sont souvent déployés dans un environnement virtuel pour minimiser les coûts de déploiement et d'exploitation. L'objectif principal de ces réseaux simulés est de répliquer le plus fidèlement possible un environnement opérationnel. Pour être en mesure d'atteindre les buts recherchés, les activités des utilisateurs du véritable réseau se doivent d'être simulées le plus fidèlement possible pour assurer un environnement d'entraînement efficace. Plusieurs méthodes et techniques sont requises pour assembler un cyber-range, on utilise souvent un agencement de machines virtuelles combiné avec des dispositifs physiques comme des serveurs, augmenté avec des simulations de trafic de réseau et d'activités d'utilisateurs. Dans le but

d'obtenir un environnement réaliste, la simulation d'événements malicieux ou d'événements qui sont intéressants pour l'entraînement des défenseurs de réseau est requise.

La Figure 2-1 est une vue d'ensemble d'un réseau test des FC appelé en anglais *Collaborative Security Test Environment (CSTE)* ou en français *l'Environnement collaboratif de test de sécurité* (ECTS). L'architecture du réseau ECTS inclut différentes composantes qui reproduisent l'environnement courant du réseau du Ministère de la Défense nationale (MDN). La section des clients reproduit les stations des utilisateurs internes du réseau. Ces clients sont des stations virtuelles hautement configurables qui fournissent un haut degré de flexibilité à la simulation. L'ECTS est muni d'une section de serveurs qui émule des services réseaux typiques comme des serveurs de courriel, serveurs web, serveurs *Windows Active Directory* et des serveurs DNS. La section *Integrated Exchange Gateway* (IEG) reproduit l'architecture des périmètres de défense présents dans le réseau du MDN. Cette section inclut des pare-feux à états les plus récents, des outils de prévention d'intrusion et des outils de prévention de fuites de données. L'autre section de l'architecture est une simulation de l'environnement externe du réseau de la défense qui inclut une émulation de l'Internet.

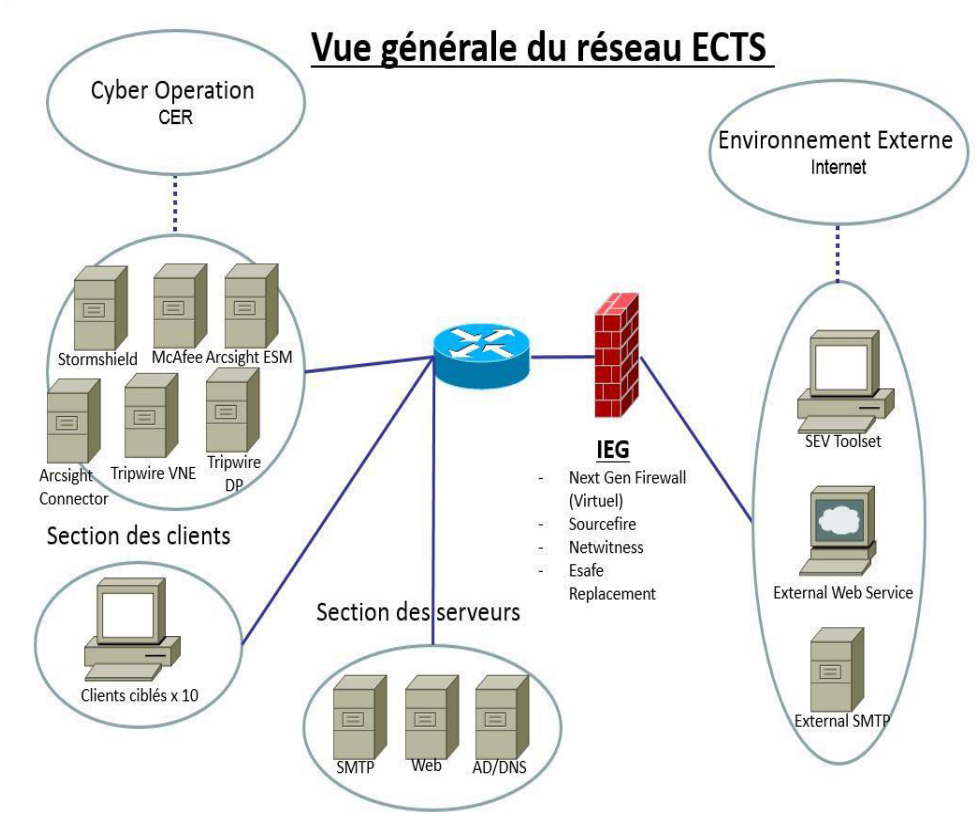

Figure 2-1. Architecture de l'environnement ECTS [5].

Cette composante inclut spécifiquement la simulation de menaces provenant de l'extérieur du réseau. Cette section comprend des stations de travail utilisées pour la validation de la sécurité du réseau interne. La section comprend aussi la simulation de services web et plusieurs stations de travail hautement configurables. Finalement, le réseau ECTS réplique les fonctionnalités du CER du MDN, l'environnement de travail du personnel qui a la tâche de surveiller et de protéger le réseau interne. Le CER inclut des stations de travail munies de logiciel de surveillance et de dépistage hautement performants. Le logiciel *ArcSight Enterprise Security Management* (ESM) est le SIEM utilisé pour la gestion de l'information produite par l'ensemble des senseurs qui sont déployés pour surveiller le réseau.

2.3 Concepts reliés aux logiciels SIEM

Cette section présente brièvement les rôles et les fonctionnalités reliés aux logiciels SIEM. Les logiciels SIEM sont devenus l'outil principal de surveillance pour défendre des réseaux de taille importante et sont maintenant au cœur du processus de la défense dans les CER [1]. La raison principale est que les logiciels SIEM permettent la gestion efficace de l'énorme quantité de logs produits par les senseurs de sécurités installés sur un réseau. Les logiciels SIEM sont les outils principaux de gestion d'information utilisés par les défenseurs pour organiser et gérer les alertes de sécurité. Les logiciels SIEM centralisent la gestion des logs et augmentent le niveau de sécurité et de protection sur un réseau [5]. Ils permettent d'agréger l'information générée par tous les senseurs existants sur un réseau pour créer une représentation de l'information optimale pour surveiller le réseau.

Une étude des diverses architectures des logiciels SIEM proposées par divers vendeurs montre que sept importantes fonctionnalités sont implémentées pour aider à la gestion des logs, elles sont définies de la manière suivante [6]:

1. La normalisation constitue l'habileté à assembler les logs générés par tous les senseurs d'un réseau et de les présenter dans un format commun. Cette capacité facilite la corrélation entre les événements reportés par les différents senseurs et facilite le processus d'analyse et la recherche pour les surveillants du réseau.
2. Le filtrage des événements est la suppression de certaines entrées de logs provenant de processus d'analyse réalisé à l'étape de la collection ou du pré- stockage des logs dans la base de données. Les logs sont simplement rejetés quand ils sont jugés de ne pas contenir d'information valable.
3. La réduction est définie par l'omission des logs non-nécessaires et par le transfert de certains champs des logs intéressants dans le

stockage à long terme. La réduction se doit d'être faite à l'étape du stockage des logs dans la base de données centrale.

4. L'agrégation est définie par la consolidation des logs qui sont similaires dans une entrée unique à qui le nombre d'occurrence de l'événement va être ajouté.
5. La rotation est définie par la gestion des archives des logs. Quand un répertoire de logs n'est plus pertinent pour la situation présente, il est fermé et stocké dans un répertoire d'archive et un nouveau répertoire est ouvert pour stocker les logs les plus récents.
6. La synchronisation du temps est définie par le changement du temps dans les logs capturés à un format de temps commun pour s'assurer que la normalisation soit faîte.
7. La vérification de l'intégrité est l'action d'entreposer et de gérer les archives des logs pour s'assurer de préserver leur intégrité.

La Figure 2-2 illustre l'architecture du logiciel SIEM *ArcSight ESM*, qui est relativement semblable à l'architecture de la plupart des logiciels SIEM existants. La couche 1 contient les éléments du réseau qui génèrent les logs, les senseurs et les programmes appelés connecteurs qui collectent les logs de ceux-ci. Un exemple trivial de senseurs est un pare-feu qui protège un réseau, le SIEM va utiliser un connecteur pour collecter les logs produits par celui-ci. La couche 2 contient les systèmes centraux qui peuvent inclure plusieurs serveurs. Ces serveurs collectent les logs originaux des senseurs en se servant des connecteurs présents dans la couche 1. Après que les logs aient été collectés, ils sont consolidés et stockés dans la base de données centralisée. La base de données centralisée emmagasine généralement les logs dans un format normalisé pour optimiser et faciliter la tâche de la couche 3 qui est la couche de contrôle et de surveillance. La couche 3 est composée des interfaces que les opérateurs du CER utilisent pour surveiller et réexaminer les événements. Ces interfaces sont aussi utilisées pour configurer les composantes de la couche 2 et même de la couche 1 quand les fonctionnalités du système le permettent.

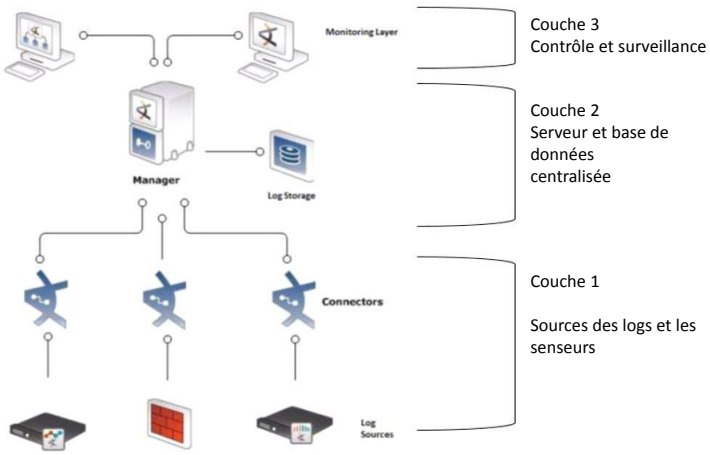

Figure 2-2. Architecture typique d'un SIEM, Arcsight ESM dans le cas présent [5].

2.4 Normes pour les logs

La gestion des logs est une des fonctions principales des logiciels SIEM et notre démarche de recherche est intimement reliée à la manipulation et l'altération de logs. Il est donc important de comprendre ce que constitue un log. Malheureusement, comme reporté par Kühnel, plusieurs définitions différentes à propos de ce que constitue un log existent. Cependant, Kühnel définit un log : « comme étant de l'information que le développeur d'une application pense être utile et intéressante pour décrire l'état courant d'une application, accompagnée d'un horodatage[3], et reportée quand l'état se produit » [7]. Cette définition simple est convenable, il existe de multitudes normes pour les logs selon les divers senseurs, cependant ils ont tous un élément en commun, de l'information reliée à un temps d'entré. Autrement dit, la plus simple norme de log est constituée de deux éléments, un horodatage et des données ne possédant aucune structure. Cependant, l'information qui est contenue dans les logs peut être structurée de différentes façons et sous différents formats. Gerhards a catégorisé les structures de différentes normes de logs existants selon ces catégories [8]:

1. Semi-structuré, similaire à la norme de logs du système d'exploitation Linux, c'est-à-dire qu'il n'y pas de distinction entre les champs et séparateurs de champ et certaines données qui sont du bruit fond.
2. Faiblement structuré, similaire à la norme CSV, de l'information externe est requise pour comprendre le contenu des champs, cependant les données des champs sont inclus.

[3]Horodatage est la traduction de mot anglais *timestamp*.

3. Fortement structuré, similaire à la norme RFC5424, c'est-à-dire le protocole Syslog. Les données structurées incluent des noms pour les champs et les valeurs des champs. Un autre exemple de ces types de logs est la norme JavaScript Object Notation (JSON) et Extensible Markup Language (XML).
4. Fortement structuré basé sur la validation du profile, c'est-à-dire avec les données structurées incluant des noms pour les champs et les valeurs des champs, augmentés avec un système pour accepter différents noms de champs ou différents types de donnée. Par exemple pour un champ contenant l'adresse IP, on pourrait l'avoir nommé 'ip', ou 'addresse_IP' ou 'addressIP'. Dans le cas des données, on pourrait être en mesure d'accepter une valeur pour une date dans un format différent, par exemple '12 OCT 2014', ou '12-10-2014'.

Une norme de log ayant une structure fortement structurée facilite la manipulation de l'information contenue dans les champs, ce qui est souhaitable pour être en mesure d'altérer facilement un log de manière automatique ou semi-automatique. Pour notre démarche de recherche, l'utilisation de normes pour le format de log ayant une structure structurée ou fortement structurée est fortement souhaitable.

Sur un réseau informatique, chacun des senseurs génère habituellement des logs selon leur propre norme. C'est un problème technique que les concepteurs de logiciel SIEM ont solutionné en adaptant une norme pour normaliser et stocker les différents logs dans une base de données centralisée. Par exemple, la compagnie *Hewlett Packard* a développé pour son implémentation SIEM *ArcSight,* la norme ouverte appelée en anglais *Common Event Format(CEF)* [9]. Cette norme est parfaitement extensible permet de convertir des logs produits par plus de 310 senseurs dans un format commun . Adapter une norme commune pour les logs est une obligation pour être en mesure de manipuler efficacement les différents champs des logs et être en mesure de les rejouer à l'aide de connecteurs de reprise.

2.5 Techniques d'entraînements des opérateurs de logiciel SIEM

La recherche sur le développement de nouvelles techniques d'entraînement pour les défenseurs de réseaux est motivée par le besoins de préserver les ressources rares et dispendieuses que sont des membres d'équipe de type *« Red Team »* et également d'être en mesure de fournir de l'entraînement à des opérateurs à grande échelle. L'implémentation d'un logiciel SIEM est un processus complexe car sa configuration se doit d'être personnalisée au réseau qui est protégé. Les opérateurs doivent être entraînés de manière adéquate dans le but de bien comprendre les nombreuses fonctionnalités du SIEM pour être en mesure d'en tirer tous les avantages[4]. Les opérateurs de logiciel SIEM doivent apprendre à configurer les différentes composantes pour être en mesure de détecter les

incidents de sécurité efficacement. Ils doivent aussi apprendre à reconnaître les motifs reliés à un incident de sécurité parmi les nombreux logs agrégés par un logiciel SIEM.

Traditionnellement, dans les organisations militaires, les équipes de type *« Red Team »* sont utilisées pour tester les dispositifs de défense d'un réseau et la réponse du personnel du CER aux différents incidents de sécurité. L'utilisation de ces équipes est la plus performante d'entraînement pour les défenseurs de réseau [3]. Le personnel des équipes de type *« Red Team »* est en mesure de répliquer des attaques sophistiquées de manière contrôlées. Ce qui est très important lors de tests réalisés en environnement où l'opérabilité du réseau doit être maintenue.

L'utilisation de cadre d'outils d'aide à l'attaque semi-automatique, comme par exemple la suite *Metasploit*, est efficace pour tester les dispositifs de défenses et la configuration d'un logiciel SIEM. Ces outils sont particulièrement efficaces dans un environnement test comme un réseau *Cyber Range* où un strict contrôle sur les attaques est moins nécessaire qu'en utilisant un environnement réel [2]. Aussi, l'utilisation d'un réseau *Cyber Range* permet de rejouer les scénarios à volonté sans impact négatif sur l'environnement. C'est un avantage important comparativement à l'utilisation d'un environnement opérationnel pour l'entraînement où les activités du réseau ne peuvent parfois pas être perturbées. Cependant, pour pouvoir utiliser un *Cyber Range* avec succès pour entraîner des opérateurs de logiciel SIEM, les activités courantes des utilisateurs d'un réseau se doivent d'être répliquées. Sans cet essentiel bruit de fond, il est impossible d'entraîner les défenseurs efficacement. Émuler les activités courantes des utilisateurs pour récréer un arrière-plan réaliste pour l'entraînement n'est pas une tâche simple. C'est spécialement le cas pour l'entraînement d'opérateurs de logiciel SIEM, où l'émulation de séquences de logs d'une manière cohérente et suffisamment détaillée est requise.

Basé sur ces faits, l'école de la *Naval Postgraduate School* de la *U.S. Navy* a développé un projet pour démontrer qu'il était possible de fournir de l'entraînement pour sécurité informatique en se servant de l'environnement opérationnel de l'audience ciblée. Ils ont développé un outil appelé en anglais le *Malicious Activity Simulation Tool (MAST)*. Cet outil simule des activités malicieuses sur certains éléments d'un réseau opérationnel, par exemple une station de travail. L'objectif est de simuler des scénarios d'activités malicieuses qui peuvent être utilisés pour entrainer les défenseurs de réseau dans leur environnement de travail habituel de manière aussi efficace que des méthodes d'entraînement traditionnelles[10]. L'avantage de cette approche est que l'entraînement est réalisé dans un cadre extrêmement réaliste et que les activités courantes du réseau, qui sont essentielles à l'entraînement, n'ont pas besoin d'être émulées. La Figure 2-3 illustre l'architecture client-serveur des outils *MAST*.

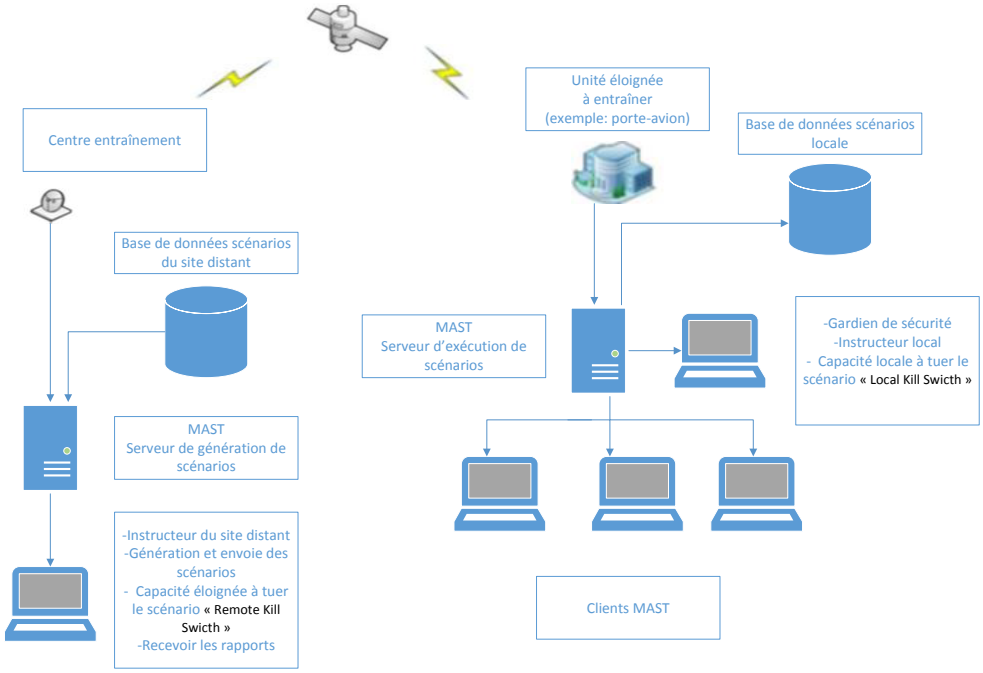

Figure 2-3. Architecture des outils MAST [10].

Sur le réseau opérationnel, qui peut être situé dans un lieu éloigné comme un porte-avion en opération par exemple, chacune des stations de travail sont munies d'un client *MAST* dormant. Quand un instructeur local décide d'exécuter un scénario d'entraînement, par exemple la propagation d'un ver malicieux. Les clients *MAST* ciblés vont recevoir la charge utile délivrée par le serveur des scénarios. Les clients *MAST* vont par la suite exécuter la charge utile, le scénario va émuler le comportement d'un ver malicieux sur les stations de travail ciblées et conséquemment sur l'ensemble du réseau. Les effets auront un effet visible pour tous les utilisateurs, par exemple le ver pourrait fermer une fenêtre de l'interface utilisateur sur une des stations de travail ciblée. Alternativement, le ver pourrait être invisible aux utilisateurs, par exemple, le ver pourrait exécuter un balayage des ports et tenter de s'auto-propager sur le réseau. Il est à noter que la simulation des effets malicieux peut être arrêtée à tout moment et que le réseau peut être restauré à son état normal à tout instant en cas de besoin. Les effets malicieux créés sont cependant suffisamment réalistes pour être en mesure d'être détectés par les senseurs du réseau. Conséquemment un scénario d'entraînement réaliste est créé pour entraîner le personnel du CER et les opérateurs de logiciel SIEM. Évidemment, les limitations des outils *MAST* sont reliées aux exigences de sécurités du réseau opérationnel utilisé pour l'entraînement et aussi à l'obligation d'être en mesure d'arrêter les effets malicieux et de pouvoir restaurer l'état normal à tout moment. Les outils *MAST* se sont montrés comme une solution viable pour réduire la dépendance aux équipes de type « *Red Team* » et efficaces pour l'entraînement des défenseurs de réseau[2]. Il est important de noter que les outils *MAST* ont été conçus pour

entraîner les défenseurs de réseau et non spécifiquement pour entrainer les opérateurs de logiciel SIEM. Néanmoins, quand les outils MAST créent les artifices reliés à un incident malicieux, les configurations du SIEM en place et la réponse des opérateurs aux incidents malicieux sont testées.

Une autre approche pour entraîner les opérateurs de logiciel SIEM qui a été explorée est l'utilisation de connecteurs de logiciel SIEM qui permettent de rejouer des séquences de logs collectées sur une période limitée de temps. Ces connecteurs de reprise permettent de tester la configuration du SIEM, particulièrement les signatures et les règles de corrélations qui sont essentielles pour détecter les incidents de sécurité [5]. Wong a développé un cadre d'entraînement renouvelable pour l'éducation des opérateurs de logiciel SIEM qui est efficace[4]. Lors de son expérimentation, Wong a utilisé les connecteurs de reprise pour rejouer un scénario de propagation de ver malicieux sur un réseau en laboratoire. L'utilisation de connecteurs de reprise offre plusieurs avantages, spécialement quand un réseau *Cyber Range* est utilisé. Les séquences de log qui sont rejouées peuvent être récoltées d'un réseau opérationnel de manière relativement simple, cependant leurs configurations nécessaires à leur utilisation sur un réseau *Cyber Range* sont parfois complexes. Ces entrées, lorsqu'elles sont rejouées sur un réseau *Cyber Range* sont particulièrement utiles pour émuler les activités courantes d'un réseau, l'arrière-plan essentiel pour entraîner efficacement des opérateurs de logiciel SIEM.

2.6 Connecteurs de reprise utilisés pour rejouer des séquences de logs

Cette recherche propose de développer des séquences de logs dans le but de les rejouer avec des connecteurs de reprise, il est donc important de revoir brièvement leur fonctionnement. La capacité de rejouer des séquences de logs est une fonction importante pour un logiciel SIEM. Les connecteurs de reprise permettant de rejouer des séquences de logs sont largement utilisés pour tester la configuration d'un SIEM, développer des règles de corrélations d'événements et rejouer des événements archivés intéressants. La Figure 2-4 illustre le fonctionnement des connecteurs de reprise qui rejouent des séquences de logs qui sont stockées dans un fichier. Le connecteur prend simplement la place d'un senseur réel et rejoue une séquence de logs reliée à ce senseur. Seulement le temps d'entré de chacun des logs sont changé quand la séquence est rejouée par le connecteur. Le gestionnaire d'événement du SIEM va gérer les événements rejoués comme si elles provenaient d'un véritable senseur du réseau surveillé. Le format du fichier acceptable par un connecteur peut varier grandement selon l'implémentation du SIEM, les connecteurs de reprise de ArcSight acceptent entre autres les formats CSV, SYSLOG et CEF.

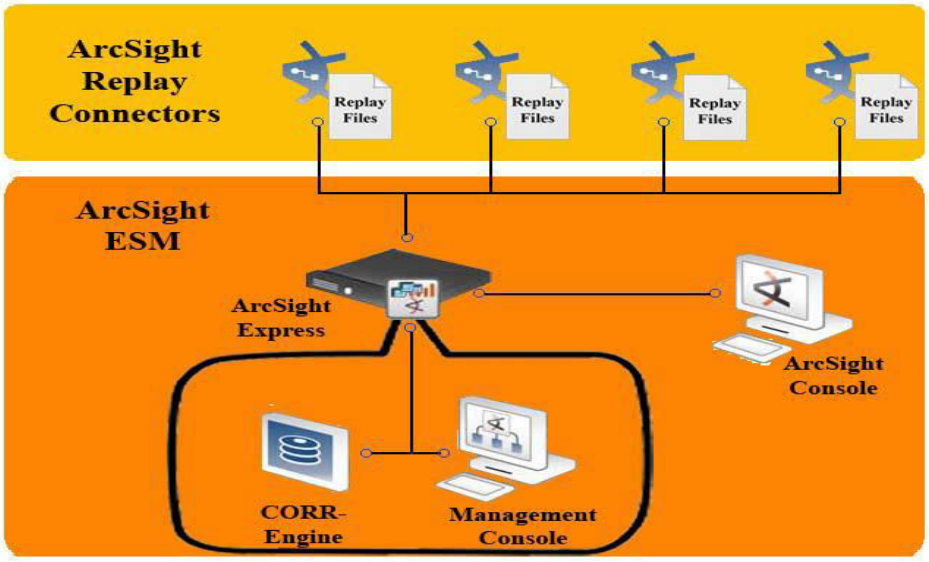

Figure 2-4. Architecture des connecteurs de reprise du logiciel ArcSight pour rejouer des séquences de logs [4]

Le logiciel SIEM *Splunk* implémente aussi une capacité de rejouer des logs à partir d'un fichier [11]. Le générateur d'événement de *Splunk* implémente des fonctionnalités supplémentaires comparativement à l'implémentation de SIEM *ArcSight*. De plus, l'implémentation de *Splunk* permet de personnaliser le processus par lequel les logs sont rejoués. Par exemple, la valeur pour un des champs des logs peut être allouée totalement au hasard lorsque que la séquence de logs est rejouée. Il est également possible d'écrire une fonction spécifique qui va calculer la valeur du champ selon certains paramètres, les possibilités de personnalisation des logs sont pratiquement illimitées. Également, le générateur de logs a été écrit avec le langage de programmation *Python* et son code est ouvert, ce qui rend cet outil très flexible.

La plupart des implémentations des logiciels SIEM permettent l'utilisation des connecteurs de reprise pour rejouer des séquences de logs en concurrence avec de véritables senseurs. La capacité des logiciels SIEM à rejouer des logs peut être utilisée sur un réseau *Cyber Range* ou sur un véritable réseau pour divers types de scénarios d'entraînement ou de tests. Les connecteurs de reprise peuvent rejouer des logs qui proviennent de différents réseaux à condition que le format des séquences de logs utilisées soit accepté par le SIEM. Un point important à souligner est la capacité pour la plupart des implémentations de logiciel SIEM à être configurées pour seulement surveiller un nombre restreint de senseurs à la fois à l'aide d'un filtre sur un canal [12]. Un canal pour un logiciel SIEM est l'équivalent d'une chaîne pour un téléviseur, les opérateurs ont la possibilité de syntoniser un canal pour le surveiller. Le canal peut être personnalisé à l'aide de multiples filtres dans le but d'ajuster la recherche. Cette fonctionnalité est très utile pour l'entraînement des opérateurs, qui pourrait seulement à avoir à syntoniser un canal configuré pour l'entraînement sur un réseau opérationnel.

Pour saisir les défis reliés avec l'utilisation de connecteur de reprise pour entraîner des opérateurs de logiciel SIEM, il faut comprendre comment les expérimentations de Wong ont été accomplies. Les objectifs d'entraînement de Wong étaient la compréhension du comportement des menaces cybernétiques et l'installation et la configuration d'un logiciel SIEM. Un scénario d'entraînement de base a été adopté comme méthodologie et un laboratoire d'entraînement a été utilisé pour entraîner les opérateurs. Essentiellement, les étudiants agissant comme opérateurs ont reçu une séquence de logs comportant un scénario qui était relié à une attaque de ver malicieux. En premier lieu, les étudiants se devaient de comprendre le comportement de la menace cybernétique en étudiant les séquences de logs reçues. En deuxième lieu, les étudiants devaient configurer l'environnement du réseau en laboratoire en utilisant un processus d'ingénierie inverse sur les séquences de logs fournies. Le but du processus était de reconstruire le réseau en laboratoire pour être en mesure de rejouer la séquence de logs. La dernière étape du processus était de configurer le SIEM pour être mesure de détecter et de reporter les attaques trouvées dans les séquences de logs reçues.

Wong a découvert que le processus d'ingénierie inverse utilisé pour reconstruire le réseau à partir de la séquence de logs fournie n'est pas intuitif et est inadéquat [4]. L'utilisation d'un réseau *cyber-range* qui reproduit fidèlement un réseau opérationnel pourrait être une solution de remplacement viable pour éviter le processus de reconstruction d'un réseau complet. Wong a également insisté sur le fait que la gestion des sources par lesquelles les logs sont créés sont importantes, les logiciels SIEM sont des systèmes passifs, ils n'ont aucune capacité à détecter et analyser des programmes malicieux. Ils sont dépendants de l'information collectée par les senseurs du réseau. Les opérateurs de SIEM doivent comprendre que la configuration des collecteurs et de leurs senseurs est essentielle [4]. L'utilisation des réseaux *cyber-ranges* pour l'entraînement des opérateurs de logiciel SIEM pourrait potentiellement aider les opérateurs à s'entraîner sur la configuration des senseurs et des collecteurs. Les réseaux *cyber-ranges* implémentent la plupart des senseurs qui sont des sources de logs pour un logiciel SIEM, un exemple évident est un pare-feu. Wong suggère également que des progrès en recherche sont nécessaires afin de pouvoir utiliser divers scénarios avec des connecteurs de reprise pour l'entraînement, spécifiquement en se servant de scénarios d'attaques plus complexes que la propagation d'un ver malicieux [4].

Notre démarche de recherche a pour but de trouver des solutions à des problèmes identifiés par Wong. Particulièrement, les problèmes reliés au besoin d'utiliser l'ingénierie inverse pour reconstruire le réseau; comme suggéré par Wong, cette étape se doit d'être évitée. Une alternative est d'utiliser un réseau véritable, généralement le réseau qui est utilisé par les opérateurs que nous voulons entraîner. Ceci assure un haut niveau de réalisme à l'entraînement sans avoir besoin de simuler les activités des utilisateurs. Une autre alternative est d'utiliser un réseau *cyber-range* pour entrainer les opérateurs, cette option est

parfois la seule acceptable à cause de motifs de sécurités et de nécessités opérationnelles. Cependant, il est important de noter que pour être en mesure de se servir de connecteurs de reprise pour rejouer des séquences de logs dans ces deux environnements. Nous devons être en mesure d'assembler efficacement les séquences de logs qui incluent les scénarios d'entraînement intéressants, avec des séquences de logs qui vont constituer l'arrière-plan requis pour rendre l'entraînement profitable.

2.7 Domaines de recherche connexes

Le but de notre recherche est d'investiguer les outils et les techniques nécessaires pour assembler des séquences de logs d'arrière-plan et des séquences de logs comportant des scénarios intéressants dans un but d'entraînement. Il est primordial de rappeler que notre recherche ne cherche pas directement à reproduire ou générer des séquences de logs reliées à des incidents malicieux. Cependant, certaines idées et techniques utilisées dans le cadre de la génération synthétique ou semi-synthétique de séquences de logs pourraient être utilisées pour solutionner certains des problèmes reliés à notre recherche. Nous allons énumérer quelques éléments de ces domaines de recherche qui s'appliquent à notre situation.

Dans un premier lieu, il important de noter que la génération de logs qui est basée purement sur des moyens synthétiques ne va pas produire des données réalistes [13]. C'est spécifiquement le cas pour la génération de séquences de logs servant d'arrière-plan qui doivent reproduire entre autres le comportement des utilisateurs du réseau. Pour notre situation, le processus de fusion d'une séquence de logs reliée à des activités intéressantes pour l'entraînement à des séquences de logs reliées à des activités courantes qui servent d'arrière-plan est avantageux pour obtenir un résultat viable. En collectant des séquences de logs servant d'arrière-plan d'un réseau opérationnel, nous évitons de récréer complétement la séquence d'arrière-plan qui serait une tâche monumentale et très difficile à implémenter avec des moyens purement synthétiques.

Dans un autre ordre d'idées, S. O'Shaughnessy and G. Gray ont démontré qu'il était possible de générer, à l'aide de moyens purement synthétiques, des séquences de logs reproduisant des attaques de types d'injection sql, de déni de service et de reconnaissance de réseau [14]. Ils ont construit un générateur de logs pour émuler des log produits par des serveurs, des base de données et des pare-feux. Ils ont démontré que les séquences de logs générées par leur outil étaient suffissament viables pour évaluer des techniques de découverte de menaces et des outils d'analyse de sécurité [14]. Ils ont dû pour être en mesure de générer les séquences de logs de manière purement artificielle, comprendre le fonctionnement des différents senseurs. C'est-à-dire être en mesure d'émuler la génération de logs provenant d'un senseur de manière totalement synthétique. Le même type de techniques peut certainement être utilisées pour solutionner certains problèmes reliés à la personnalisation des logs pour le réseau de destination qui va être utilisé pour l'entraînement dans notre situation.

Une autre technique intéressante, surnommée en anglais *blending*, a été utilisée avec succès pour générer des données semi-synthétiques pouvant être utilisés pour tester divers outils de détection. Le laboratoire du *Software Engineering institut* de l'université Carnegie Mellon, a développé plusieurs techniques pour générer des données dans le but d'émuler des scénarios reliés à des menaces internes. Une des techniques utilisées, est le *blending*, qu'on pourrait traduire en français par le mot fusion, qui consiste à transformer un scénario malicieux relié à une menace interne pour le faire paraître comme si un véritable utilisateur avait commis les actions du scénario en l'insérant dans des activités courantes qui servent d'arrière-plan [15]. Autrement dit, un scénario impliquant un incident de sécurité est ajouté à l'arrière-plan tout en préservant sa cohésion. Pour arriver à ce résultat, des altérations sont faites au scénario inséré et à l'arrière-plan dans le but de préserver la cohésion de l'histoire.

Plusieurs techniques de transformation sur les données ont été utilisées avec succès pour implémenter ce processus de fusion, certaines sont intéressantes pour notre recherche[15]:

1. Le changement de temps présumé d'un événement donné pour le rendre cohésif avec les activités de l'arrière-plan.
2. Le changement de certains détails du scénario inséré pour le rendre cohérent avec l'arrière-plan.
3. L'utilisation de certains éléments de l'arrière-plan pour rendre le scénario inséré plus crédible.

Ces techniques ont été utilisées avec succès sur des données plus complexes que des séquences de logs provenant de senseurs de réseaux collectées par un logiciel SIEM. Elles peuvent certainement être utilisées pour assembler des séquences de logs efficaces pour entraîner des opérateurs de logiciel SIEM.

2.8 Conclusion du chapitre

Ce chapitre a couvert les aspects essentiels des logiciels SIEM qui sont nécessaires pour comprendre les problèmes reliés à l'utilisation des connecteurs de reprise pour l'entraînement. Ce chapitre a aussi discuté des démarches de recherche entreprises pour développer de nouvelles méthodes d'entraînement en insistant sur le travail accompli par Wong. Également, nous avons fait mention de diverses méthodes de recherches provenant des domaines de la génération de données synthétiques et semi-synthétiques qui s'appliquent à notre problème d'assemblage de séquences de logs. Suffisamment d'aspects techniques ont été couverts pour être en mesure de comprendre les solutions proposées aux problèmes d'assemblage de séquences de logs utilisables pour entraîner des opérateurs de logiciel SIEM.

Chapitre 3 : Assemblage de séquence de logs

3.1 Introduction

Lors du chapitre 2, nous avons évoqué certains défis à propos de l'assemblage des séquences de log reliées à des scénarios d'entraînement utilisables par des connecteurs de reprise. Ce chapitre présente des solutions pour solutionner ces problèmes. En premier lieu, nous allons discuter des défis que nous avons à surmonter pour être en mesure d'assembler des séquences de logs viables pour l'entraînement à l'aide de méthodes semi-automatiques. En deuxième lieu, nous allons présenter l'architecture d'une solution semi-automatique qui permet d'assembler des séquences de logs convenables pour l'entraînement. Nous allons par la suite décrire chacune des composantes de la solution proposée et conclure avec un bref sommaire sur les avantages et inconvénients de l'architecture proposée.

3.2 Défis posés pour la génération semi-automatique de séquences de logs

Cette section décrit les séquences de logs que nous voulons assembler pour être en mesure de les rejouer pour l'entraînement d'opérateur de logiciel SIEM, pour par la suite, discuter des problèmes reliés à l'assemblage et des composantes d'une solution.

3.2.1 Séquence de logs utilisable pour l'entraînement

Une séquence de logs qui est utilisable pour un connecteur de reprise dans le but d'entrainer des opérateurs de logiciel SIEM est constituée de deux éléments :

1. Une séquence de logs reliée au scénario d'entraînement.
2. Une séquence de logs qui constitue l'arrière-plan.

La Figure 3-1 illustre un simple exemple de séquence de logs qui sera collectée par le SIEM, notre objectif étant de répliquer cette séquence dans le but de la rejouer avec des connecteurs de reprise. Le scénario d'entraînement illustré est une attaque basée sur une injection SQL qui est réussi au premier essai par un attaquant habile. Chacun des senseurs illustrés en haut de la figure, détectent et produisent des logs reliés à cette attaque. Ces logs sont collectés par la suite par le SIEM, il constitue ce que nous appelons les logs reliés au scénario d'entraînement. Ces logs sont aussi collectés en même temps que tous les autres logs provenant de tous les senseurs du réseau. La séquence en bas de la figure illustre la séquence de logs que nous avons besoin d'obtenir pour entraîner des opérateurs de logiciel SIEM.

La séquence de logs reliée au scénario d'entraînement est une série de logs provenant des senseurs qui sont impliqués dans le scénario d'entraînement classés en ordre chronologique. La séquence de logs reliée au scénario d'entraînement se doit d'être insérée en ordre chronologique pour préserver la cohésion du scénario de l'attaque.

Une séquence de logs servant d'arrière-plan qui est efficace, est constituée de l'ensemble des logs de tous les senseurs qui sont surveillés par le SIEM sur un réseau pour une période de temps donné. Souvent, les logs provenant des senseurs impliqués dans le scénario d'entraînement devront être présent dans l'arrière-plan pour obtenir un arrière-plan efficace. Les logs de la séquence d'arrière-plan peuvent être reliés à n'importe quelles activités courantes d'un réseau informatique ayant plusieurs utilisateurs. Ce qui pourrait inclure par exemple, des fausses alertes dues à un mauvais fonctionnement d'un senseur ou des logs reliés à une attaque d'un agent externe hostile. Le travail d'un opérateur de SIEM d'échelon inférieur est de gérer et d'analyser de nombreux logs pour différencier les fausses alertes des événements digne d'intérêt [1], un arrière-plan riche est donc essentiel pour l'entraînement.

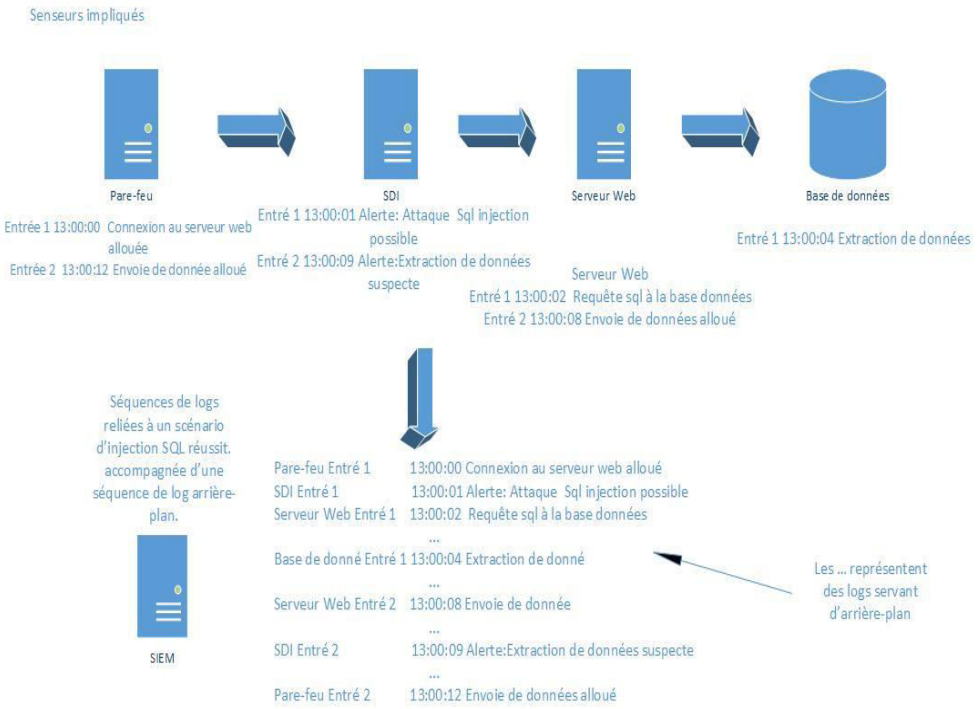

Figure 3-1. Attaque injection SQL

3.2.2 Synthèse du problème d'assemblage de logs

En premier lieu, nous devons acquérir la séquence de logs reliée au scénario d'entraînement. Notre démarche de recherche ne comprend pas la génération de séquence de logs reliées au scénario d'entraînement qui est un problème de recherche en soit. Nous supposons donc que nous avons des séquences reliées à des scénarios d'entraînement en main et qu'elles sont dans une norme compatible avec le SIEM utilisé ou bien qu'il soit possible de les convertir dans une norme utilisable. De plus, cette séquence se doit d'être compatible avec les senseurs de réseau utilisés pour l'entraînement. Cependant, une adaptation de la séquence de logs sera nécessaire pour la rendre utilisable pour notre réseau d'entraînement.

Nous devons avoir la capacité de stocker la séquence du scénario d'entraînement de manière optimale dans un répertoire ou une base de données pour être en mesure de les référencer afin de l'introduire dans une séquence de logs qui va servir d'arrière-plan.

Nous devons être également être en mesure d'acquérir les séquences de logs servant d'arrière-plan et de les stocker. Cependant, ce problème est relativement facile à solutionner car la majorité des implémentations de logiciel SIEM sont munies de ces fonctionnalités. Il est donc très facile de collectionner une séquence de logs d'arrière-plan intéressante si le réseau surveillé par un logiciel SIEM produit suffisamment d'activités courantes pour chacun des senseurs.

Nous désirons fusionner la séquence de logs du scénario que nous voulons utiliser pour l'entraînement à une séquence de logs d'arrière-plan. L'objectif est de réécrire l'histoire en introduisant un nouveau chapitre mais en l'altérant le moins possible. L'étude du problème relié à cet assemblage de logs montre que nous avons trois problèmes à solutionner :

1. La séquence de logs reliée au scénario d'entraînement se doit d'être adaptée au réseau d'entraînement qui sera utilisé. Par exemple, un champ qui constitue une adresse IP d'une station de travail se doit d'être adapté à notre réseau. Ce type de champ se doit être personnalisé pour le réseau d'entraînement utilisé et aussi configuré par l'instructeur selon les besoins du scénario d'entraînement.
2. La séquence de logs reliée au scénario d'entraînement se doit d'être insérée dans la séquence d'arrière-plan de manière à préserver sa cohésion. Premièrement, la séquence du scénario se doit d'être insérée en ordre chronologique, et le temps d'insertion entre chacun des logs se doit d'être respecté pour avoir un scénario crédible. Également, l'instructeur doit avoir la possibilité d'omettre ou de dupliquer certains logs reliés au scénario d'entraînement. Dépendamment de la configuration du SIEM utilisée, certains logs ne pourraient pas être collectés par le SIEM, ou bien une duplication de certains logs est requise pour activer des règles de détection.
3. L'ensemble de la séquence de logs d'entrainement incorporée avec la séquence de logs d'arrière-plan se doit d'être adaptée pour préserver la cohésion du scénario de la séquence. Nous devons rectifier l'information des champs qui est maintenant contradictoires ou qui rentrent en conflit avec la séquence logique dû à l'insertion des logs reliés au scénario.

La Figure 3-2 synthétise les trois étapes nécessaires à la production de séquences de logs utilisables. Premièrement, une séquence de logs reliée à un scénario d'entraînement est extraite d'un répertoire. Cette séquence se doit d'être adaptée au réseau d'entraînement par un processus semi-automatisé, nous avons surnommé cette étape la « personnalisation ». Cette séquence de log est par la suite être incorporée en ordre chronologique avec une séquence de logs d'arrière-plan couvrant une période donnée qui provient d'un réseau opérationnel. Nous avons surnommé cette étape « l'insertion ».

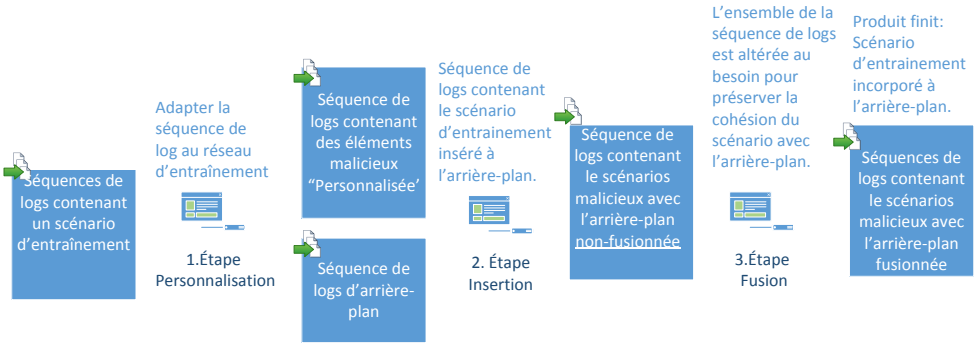

Figure 3-2. Étapes nécessaires à l'assemblage de séquences de logs

La dernière étape consiste à vérifier l'ensemble de séquence de logs pour s'assurer que la cohésion est préservée. Nous avons baptisé cette étape la « fusion ». Après l'exécution de ces étapes, nous obtenons une séquence de logs qui peut être rejouée sur notre réseau d'entraînement. Pour que ce processus d'assemblage de séquence de logs fonctionne, il doit cependant faire partie d'une architecture qui lui fournit les données nécessaires pour être en mesure d'exécuter chacune des étapes de transformation. La prochaine section discute de l'architecture que nous proposons pour l'assemblage de la séquence de logs.

3.3 Architecture proposée

La Figure 3-3 illustre l'architecture de la solution proposée pour produire des séquences de logs utilisables par des connecteurs de reprise pour entraîner des opérateurs. Les trois étapes de l'assemblage de la séquence de log sont illustrées de gauche à droite.

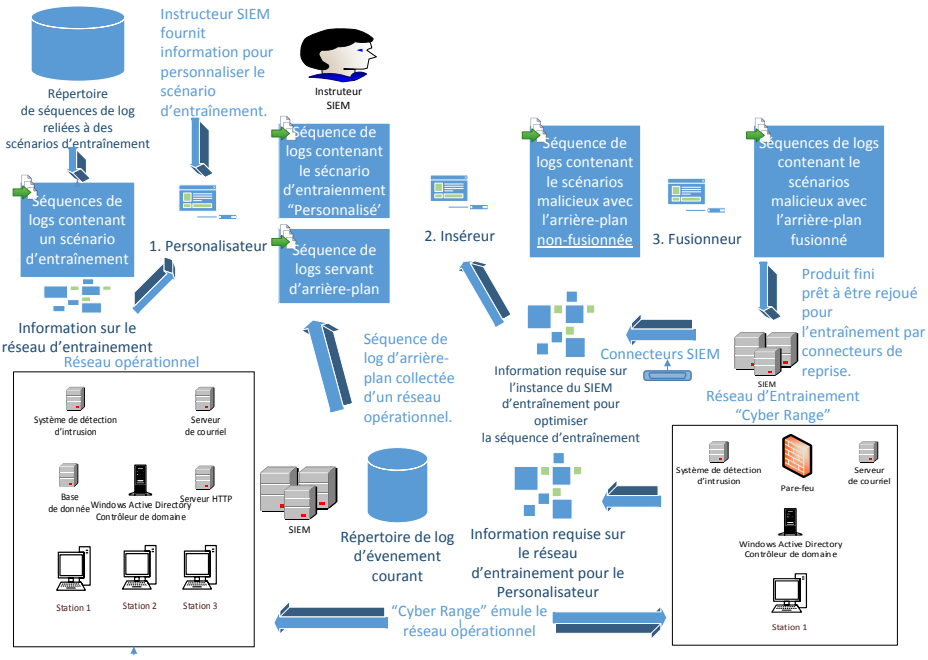

Figure 3-3. Architecture pour assemblage de séquence de logs

Les principales composantes de l'architecture développée sont :

1. Répertoire de séquences de logs reliées à des scenarios d'entraînement : le premier élément de l'architecture est le répertoire contenant les séquences de logs qui sont reliées aux scénarios de l'incident que nous voulons rejouer pour l'entraînement. Ces séquences sont stockées et facilement accessibles pour le logiciel surnommer « *Personnalisateur* ».
2. Le « *Personnalisateur* » : Un programme qui a pour tâche d'adapter les logs reliés aux scénarios d'entraînement pour être en en mesure de les rejouer sans problèmes sur le réseau d'entraînement (cyber-range) qui est situé en bas à droite de la Figure 3-3. Le « *Personnalisateur* » a besoin d'information de deux sources pour être en mesure de personnaliser la séquence de logs servant à l'entraînement. L'information du réseau d'entraînement se doit d'être disponible et extraite de ce réseau. Cette information peut-être par exemple le nom d'un utilisateur ou d'un serveur. Également, l'utilisateur du programme qui est le l'instructeur SIEM, doit fournir au « *Personnalisateur* » l'information nécessaire pour personnaliser le scénario, ceci peut-être par exemple le nom du fichier où un virus a été découvert. Le « *Personnalisateur* » produit une séquence de logs

adaptée au réseau d'entraînement qui inclut en ordre chronologique tous les logs de chacun des senseurs impliqués dans l'incident que nous voulons rejouer pour l'entraînement. Cette séquence de log est par la suite disponible pour le programme que nous avons surnommé l' « inséreur ».

3. L' « *inséreur* » a pour tâche de prendre la séquence de logs « personnalisée » et l'insérer dans la séquence de logs qui sert d'arrière-plan. La séquence de log qui sert d'arrière-plan est extraite de réseau opérationnel, elle est constituée de logs générés par certains senseurs présents sur le réseau d'entraînement sur une période de temps limitée. L'« inséreur » insère les entrées de la séquence de logs dans la séquence d'arrière-plan en respectant l'ordre chronologique et le délai entre chacune des entrées du scénario d'entraînement. Également lors de cette étape, l' « *inséreur* » va vérifier si un log faisant partie de la séquence de scénario peut être rejoué sur le SIEM de destination en se servant de l'information extraite de celui-ci. Nous voulons éviter ainsi qu'un log soit rejoué, lorsque l'instance du SIEM d'entraînement n'est pas configurée pour le détecter. La séquence produite à cette étape est par la suite disponible pour le programme que nous avons surnommé le « *fusionneur* ».
4. La tâche du « *fusionneur* » est de réviser l'ensemble de la nouvelle séquence de logs s'assurer que sa cohésion est préservée. Le « *fusionneur* » altère certains champs de la nouvelle séquence de logs qui causent des conflits. Quand cette étape est complétée, la nouvelle séquence de logs produite est prête à être rejouée par des connecteurs de reprise sur le réseau d'entraînement.

Pour que cette architecture fonctionne de manière optimale, l'utilisation d'un *Cyber Range* pour l'entraînement qui émule le réseau duquel nous collectionnons l'arrière-plan est préférable. Ceci rend la séquence d'arrière-plan collectée du réseau opérationnelle utilisable sur le *Cyber Range* sans avoir besoin d'adaptations majeures pour ces séquences de logs parce que les senseurs et l'instance du SIEM utilisée sur les deux réseaux sont semblables.

Il est primordial de noter que trois éléments importants mais hors de portée de notre démarche de recherche ne sont pas illustrés sur la Figure 3-3 :

1. Un mécanisme pour extraire l'information requise pour le *« Personnalisateur* » du réseau d'entraînement de manière semi-automatique.

2. Développer des interfaces utilisateurs conviviales pour collecter l'information requise de l'utilisateur pour le « Personnalisateur », l'« *inséreur* » et le « *fusionneur* ».
3. Un mécanisme pour extraire l'information requise sur la configuration de l'instance du SIEM d'entraînement qui pourrait avoir une influence sur l'assemblage de la séquence.

3.3.1 Priorités de recherche

Notre démarche de recherche n'a pas couvert l'architecture proposée dans son entièreté. La priorité de travail de notre démarche de recherche s'est située au niveau des processus de « personnalisation », « d'insertion » et de « fusion ».

Le processus de génération de logs reliés à des scénarios d'entraînement ou à des incidents de sécurité n'a pas fait partie de notre démarche de recherche. Il est cependant important de noter que ces séquences de logs peuvent être acquises par diverse méthodes existantes. Nous avons utilisé la manière la plus simple et efficace pour produire les logs reliés à des scénarios d'entraînement en reproduisant les incidents en laboratoire et en collectant les logs produits par les senseurs concernés. L'objectif de notre recherche était de démontré qu'il est possible de stocker ces entrées dans un répertoire rudimentaire dans le but de les utiliser pour les assembler avec une séquence de logs servant d'arrière-plan. Cette activité a été exécutée seulement en support à nos objectifs de recherche principaux.

Également nous n'avons pas entrepris de démarche de recherche à propos du développement d'un processus d'extraction automatique ou semi-automatique de l'information du réseau d'entraînement requise pour personnaliser les séquences de logs. Nous avons tout simplement extrait l'information de manière manuelle pour nos expérimentations. Cependant, rien n'indique qu'une implémentation de processus automatique ou semi-automatique d'extraction de l'information ne pourrait pas être implémentée dans le cadre d'une solution complète.

C'est le cas également pour le développement d'un processus d'extraction automatique ou semi-automatique de l'information requise sur la configuration d'un SIEM pour rendre la séquence d'entraînement viable à être rejoué de ce même SIEM. Nous n'avons pas investigué les moyens pour implémenter cette extraction d'information, cependant rien n'indique que l'implémentation de ce processus n'est pas possible. Aussi, nous n'avons pas mis d'emphase sur le développement d'une interface utilisateur pour collecter l'information requise de l'instructeur, ce processus est un problème de développement technique et a un intérêt de recherche limité.

Les prochaines sections se concentrent sur nos objectifs de recherche, c'est-à-dire le processus de personnalisation, d'insertion et de la fusion.

3.4 Processus de personnalisation

La personnalisation est le processus qui consiste à adapter chacun des champs de la séquence de logs reliée au scénario d'entraînement au réseau d'entraînement *cyber-range*. L'objectif étant de transformer la séquence de log pour qu'elle soit en mesure d'être rejouée avec des connecteurs de reprise sur le réseau d'entraînement. Pour ce faire, nous devons altérer la séquence pour qu'elle soit pratiquement identique à une séquence de logs produite par des senseurs du réseau d'entraînement.

Pour atteindre cet objectif, on doit être en mesure de prendre l'information produite par un senseur sur un réseau et l'adapter comme si elle avait été générée par un même type de senseur mais sur un autre réseau. Nous allons en premier lieu discuter de facteurs déterminants à propos de la personnalisation des logs pour par la suite discuter de son implémentation.

3.4.1 Type de champ à personnaliser

L'altération des logs dans le but de les personnaliser se fait au niveau de leurs champs. Le premier champ d'un log consiste toujours au temps de création du log sur le senseur. Ce champ n'a pas besoin d'être altéré lors de l'étape de personnalisation car le champ du temps d'entrée des logs est altéré pour les connecteurs de reprise lorsque le log est rejoué et sera donc personnalisé à la prochaine étape de processus d'assemblage de « insertion ». Chacun des autres champs de la séquence de log doivent être inspecté pour être altéré au besoin.

Notre recherche a démontré que nous pouvons catégoriser chacun des champs à altérer dans quatre catégories ou types qui sont résumés dans le Tableau 3-1. Cette caractérisation des champs est très utile dans l'application du processus de personnalisation. Nous allons décrire ces quatre types de champ pour par la suite présenter un exemple représentatif d'un log contenant des exemples de chacun des types de champ énumérés.

Le premier type de champ peut être qualifié d'automatique, car pour accomplir l'altération souhaitée, nous avons simplement besoin d'avoir accès à l'information concordante du réseau de destination utilisé pour l'entraînement. Aucune action d'un utilisateur pour personnaliser ce type de champ n'est nécessaire. La plupart du temps, ce type de champs peut être remplacé un pour un avec le champ cible provenant du réseau d'entraînement. Par exemple, le nom de l'interface de réseau surveillée pour un système SDI comme *Suricata*.

Le deuxième type de champ est personnalisable, il est caractérisé par la participation requise de l'utilisateur, c'est-à-dire l'instructeur de logiciel SIEM qui assemble la séquence de log. L'instructeur doit avoir accès aux informations reliées au réseau ciblé pour l'entraînement pour être en mesure choisir des valeurs valables pour ces champs. L'utilisateur peut choisir la valeur du champ selon les options disponibles sur le réseau d'entraînement et selon les besoins du scénario d'entraînement. Deux exemples simples de ce type de champ provenant d'un log généré lors de la détection d'un virus sont :

1. L'adresse IP de la station où le virus a été trouvé.
2. Le chemin du répertoire du fichier où le virus a été trouvé (incluant le nom du fichier).

Tableau 3-1. Type de champs pour la personnalisation

Type de champs	Caractéristiques marquantes
Automatique	Champ personnalisé automatiquement à l'aide de l'information provenant du réseau d'entraînement.
Personnalisable	Champ configurable par l'instructeur à l'aide de l'information provenant du réseau d'entraînement.
Neutre	Champ n'ayant pas besoin d'être personnalisé immédiatement mais qui doit être identifié pour être ajusté avec l'arrière-plan à l'étape de la fusion.
Signature	Champ intrinsèque au scénario d'entraînement, aucune personnalisation n'est nécessaire.

Dans le cas d'un champ contenant l'adresse IP, l'utilisateur doit choisir une adresse IP parmi celles existantes sur le réseau d'entraînement pour une station de travail. Ce choix va lui être offert à l'aide de l'information extraite du réseau d'entraînement. Dans le cas du champ du chemin du répertoire, l'instructeur peut personnaliser l'incident selon les besoins de l'entraînement.

Il est intéressant de noter que le choix fait par l'utilisateur pour un des champs personnalisables pourrait influencer d'autres champs de la séquence de log automatique, par exemple un choix de l'adresse IP pour une station *Windows* rend le champ contenant le nom de la station automatique pour ce log.

Le troisième type de champ est qualifié de champ « neutre ». Ce type de champ n'a pas d'incidence directe sur la personnalisation d'un log et n'a pas besoin d'être personnalisé. Cependant, la plupart du temps, ce type de champ se doit d'être révisé lors de l'étape ultérieure de la « fusion » et nous devons l'identifier à cette étape. Un exemple est le champ *Record Number* provenant d'un log créé par le système d'exploitation *Windows*. Le champ *Record Number* est le numéro d'identification alloué par Windows quand une entrée est faite dans un registre de logs. La méthode de par laquelle les numéros d'identification ou la valeur du champ R*ecord Number* est allouée par *Windows*, est très simple : la première entrée du registre débute à 1 et chacune des nouvelles entrées est augmentée de 1. Lorsque le registre est plein, il est vidé et la valeur *Record Number* repart à 1.

Ce type de champ ne peut être altéré à l'étape de la personnalisation, une vérification de la séquence de logs d'arrière-plan est requise pour allouer une valeur au *Record Number* convenable dans le but de préserver la cohésion de la séquence. Ce type de champ est généralement relié au fonctionnement interne des senseurs.

Le quatrième type de champ, est le plus simple, nous l'avons surnommé de type « signature ». Ce type champ n'a tout simplement pas besoin d'être altéré car il fait partie de la signature de l'attaque ou du scénario d'entraînement. Un exemple simple est le numéro de signature de l'alerte détectée par un SDI *Suricata*, ou bien le champ qui décrit la nature de l'alerte. Les champs de type signature vont demeurer constants durant la personnalisation et les étapes subséquentes. Ce type de champ est souvent spécifique au senseur.

3.4.2 Exemple simple de personnalisation

Le Tableau 3-2 illustre un exemple de log provenant du SDI *Suricata*, cette entrée a été générée suite à une requête faîte pour un téléchargement de fichier provenant d'un site web à partir d'une station de travail. Ce log est sous le format JSON, qui est très facile à lire pour un humain.

La compréhension du fonctionnement des champs est primordiale pour être en mesure d'obtenir des résultats probants pour la personnalisation. Une analyse de chacun des champs nous permet de les catégoriser, le résultat est présenté au Tableau 3-2. Premièrement, le champ '*timestamp*' correspond au temps d'entré du log, ce champ est géré de manière unique à cause que nous devons absolument préserver la logique de la séquence de logs et le personnaliser selon le temps d'insertion demandé par l'utilisateur. Ce champs n'a pas besoin d'être altéré à ce stade-ci, et va être ajusté à l'étape d'assemblage de l'insertion qui est décrite à la prochaine section.

Tableau 3-2. Log de *Suricata* relié au téléchargement d'un fichier


```
{"timestamp":"2016-02-19T12:07:07.509398-0500","flow_id":44793280,"in_iface":"eno16777736","event_type":"http","src_ip":"192.168.1.3","src_port":49265,"dest_ip":"192.168.1.2","dest_port":80,"proto":"TCP","tx_id":0,"http":{"hostname":"192.168.1.2","url":"\/putty.exe","http_user_agent":"Mozilla\/4.0 (compatible; MSIE 8.0; Windows NT 6.1; WOW64; Trident\/4.0; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729; Media Center PC 6.0)","http_method":"GET","protocol":"HTTP\/1.1","length":0}}
```


1. Champs personnalisables en rouge	2. Champs automatiques en vert
3. Champs neutres en bleu.	4. Champs signatures en noir

Le champ '*flow_id*' est un champ neutre, car il correspond à l'identité d'un flot pour *Suricata*. Un flot pour *Suricata* est un tuple constitué de cinq éléments : un protocole de communication réseau, une adresse IP pour la source, une adresse IP pour la destination, un port pour la source et finalement un port pour la destination. Le champ '*flow_id'* identifie une communication bidirectionnel [16]. À l'étape de la personnalisation les champs de type neutre se doivent d'être identifiés pour être traités lors de l'étape de la fusion, leurs valeurs vont être allouées en concordance avec la séquence de logs qui sert d'arrière-plan.

Les champs '*event_type'*, '*http*', '*proto*', '*tx_id*', '*http_method'* et sont des champs de type signature, ils font partis du scénario du téléchargement du fichier et ils n'ont pas besoin d'être altérés pour pouvoir être rejouable sur le réseau d'entraînement.

Le champ '*in_iface'* correspond au nom de l'interface réseau surveillé par l'instance de *Suricata* sur le véritable réseau, le champ sera altéré avec le nom de l'interface réseau d'entraînement directement. C'est un champ de type automatique. C'est le cas également pour le champ '*http_user_agent'* et '*protocol'*, cette information provient du navigateur internet utilisé par l'utilisateur sur sa station de travail. Cette information peut être extraite facilement du réseau d'entraînement et être utilisée pour personnaliser automatiquement ces deux champs.

Les champs '*src_ip'*, '*src_port'* sont respectivement l'adresse IP et le port de la station qui a été utilisée par l'utilisateur pour télécharger le fichier. Les champs '*dest_ip'* et *'dest_port'* sont respectivement l'adresse IP et le port du serveur qui reçoit la requête de téléchargement du ficher. Ces champs peuvent être personnalisés par l'instructeur de SIEM qui connait l'information sur les serveurs web et les stations de travail disponibles sur le réseau d'entraînement. Le champ '*hostname*' est l'adresse IP du serveur et est géré selon la même méthode. Le champ url est aussi de type personnalisable, il représente l'adresse électronique du fichier téléchargé, l'instructeur peut le personnaliser selon les besoins du scénario. Le champ '*timestamp*' est personnalisable par l'utilisateur car celui qui va décider du temps d'insertion du scénario d'entraînement à l'étape ultérieure de l'insertion.

La Figure 3-4 illustre l'étape de la personnalisation de ce log. À la gauche de la figure nous avons une représentation d'une partie d'un réseau opérationnel d'où le log a été collecté. Alice a simplement initié un téléchargement du fichier putty.exe de sa station de travail. Le log généré par *Suricata* lorsqu'Alice a téléchargé son fichier est affiché en haut de la représentation du véritable réseau. À la droite, il une représentation d'une portion du réseau d'entraînement que nous voulons utiliser pour forger le log servant à l'entraînement. En haut de cette représentation, il y a le log personnalisé, dont les champs ont été altérés pour correspondre à la configuration de certains éléments du réseau d'entraînement. C'est-à-dire à la station de travail de Bob qui va télécharger un fichier du serveur web qui réside à l'adresse IP 172.33.23.25.

Le log n'est cependant pas totalement prêt à être rejoué. Il doit être inséré dans la séquence d'arrière-plan, et un autre champ doit être altéré, le 'flow_id' lors de l'étape de la fusion. Le même processus se doit d'être appliqué à chacun des logs de la séquence qui fait partie du scénario d'entraînement.

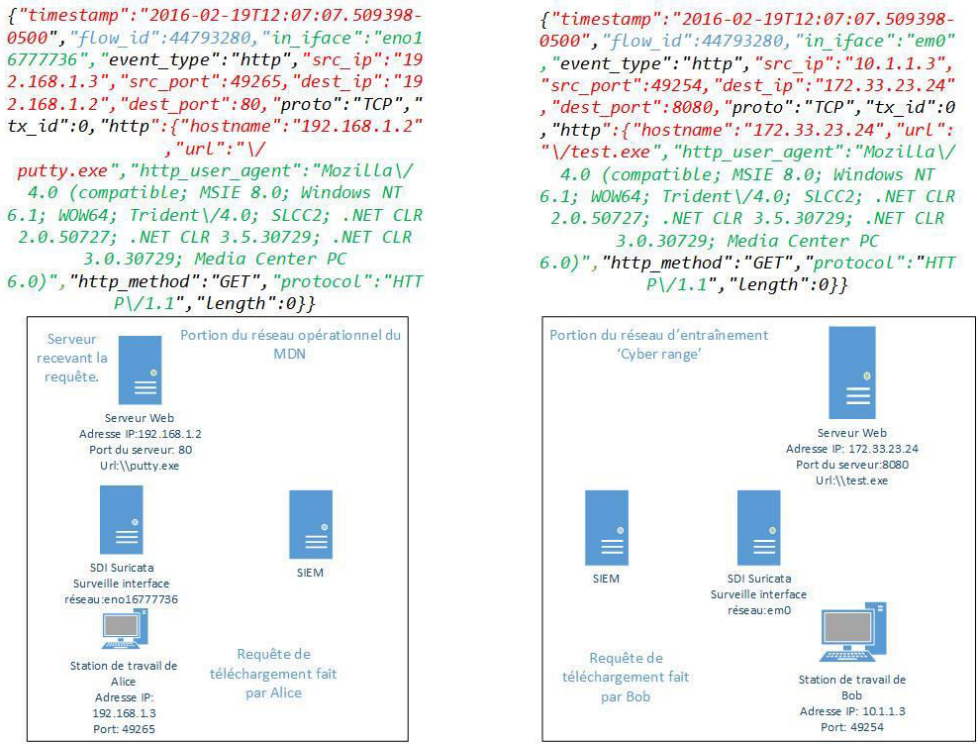

Figure 3-4. Exemple de personnalisation

Un des facteurs qui simplifie la personnalisation est que le senseur du véritable réseau est similaire au senseur sur le réseau d'entraînement. Dans ce cas-ci, nous avons une instance semblable de *Suricata* sur les deux réseaux qui produise le même type de log. Nous avons fait la présomption que les senseurs devraient être semblables pour les deux réseaux, ce qui est le cas quand un cyber-range est utilisé pour l'entraînement.

3.4.3 Considérations pour implémentation du processus de personnalisation

L'algorithme de personnalisation est relativement simple, on doit évaluer tous les champs des logs de la séquence et prendre des décisions d'altération selon son type comme résumé dans le Tableau 3-3. Pour obtenir une solution semi-automatique, un répertoire contenant de l'information sur tous les champs possibles pour les senseurs supportés pourrait être mis-en-place et consultable par le programme de personnalisation. Ce répertoire pourrait contenir de l'information permettant au programme de personnalisation d'assigner une valeur au champ. L'information minimale disponible pour un champ devra obligatoirement inclure son type et un pointeur sur l'information du réseau d'entraînement s'appliquant à lui. Également, les interdépendances entre les champs personnalisables devraient être stockées dans ce répertoire car certains champs personnalisables deviennent automatiques selon les choix faits par l'utilisateur.

Tableau 3-3. Assignation de valeur pour un champ selon son type

Type de champ	Actions
Automatique	Changer la valeur du champ automatiquement selon l'information du réseau d'entraînement.
Neutre	Identifier le champ pour qu'il soit traité lors de l'étape de la fusion et ne faire aucun changement à la valeur du champ.
Personnalisable	L'utilisateur choisit la valeur pour le champ selon les options disponibles sur le réseau d'entraînement.
Signature	Aucun changement à la valeur du champ.

L'automatisation du processus de collecte de l'information du réseau d'entraînement est en dehors du cadre de la démarche de recherche. Cependant, la quantité d'information à collecter n'est pas très grande et l'assemblage manuel pour un scénario donné requiert un effort minimal. Il est à noter que les logs personnalisés reliés à un scénario d'entraînement peuvent être stockés dans un répertoire pour être utilisés sur le même réseau d'entraînement sans avoir besoin d'être repersonnalisés.

3.6 Processus d'insertion des séquences de logs

Lorsque la séquence de logs reliée à un scénario d'entraînement a été personnalisée, la prochaine étape d'assemblage est d'introduire la séquence de logs dans la séquence de logs servant d'arrière-plan. Le respect de la chronologie de la séquence de logs reliée au scénario d'entraînement est primordial dans le processus d'insertion, ce qui implique trois facteurs :

1. L'ordre de la séquence de logs reliée au scénario d'entraînement se doit d'être respecté, car si ce n'est pas le cas, nous risquons d'avoir un manque de cohésion avec le scénario.
2. Le respect du temps d'insertion demandé par l'utilisateur, qui est le temps du début du scénario d'entraînement.
3. Le respect des délais entre chacun des logs de la séquence reliée au scénario d'entraînement.

Nous devons donc insérer le premier log de la séquence reliée au scénario d'entraînement au temps d'insertion demandé par l'utilisateur, et par la suite insérer les logs subséquents de la séquence en respectant les délais d'insertion fournis par le scénario. Le processus d'introduction est donc constitué des étapes suivantes :

1. Étape 1. L'instructeur doit choisir un temps pour débuter le scénario d'entraînement. Ce temps servira à insérer la séquence du scénario dans l'arrière-plan, la première entrée de la séquence reliée au scénario sera insérée au temps du début du scénario dans la séquence l'arrière-plan.
2. Étape 2. Quand la séquence de logs reliée au scénario d'entraînement comporte plusieurs logs, nous devons calculer les délais entre chacun des logs. Les délais entre chacun des logs serviront à calculer le temps de l'insertion dans la séquence d'arrière-plan.
3. Étape 3. Insertion de la séquence de logs selon les temps calculés en étape 1 et 2. Nous débutons l'insertion des séquences au temps choisi par l'instructeur et insérons les autres logs de la séquence à l'aide des temps des différences calculées à l'étape 2.

La Figure 3-5 illustre un exemple simple du processus d'insertion. L'instructeur veut insérer une séquence de logs reliée à un scénario d'entraînement comportant cinq logs à 10 h 42 min 23 s. Premièrement, nous devons calculer les délais entre chacun des logs de la séquence du scénario inséré. Dans notre exemple, nous avons calculé quatre délais qui sont affichées en haut de la figure au centre.

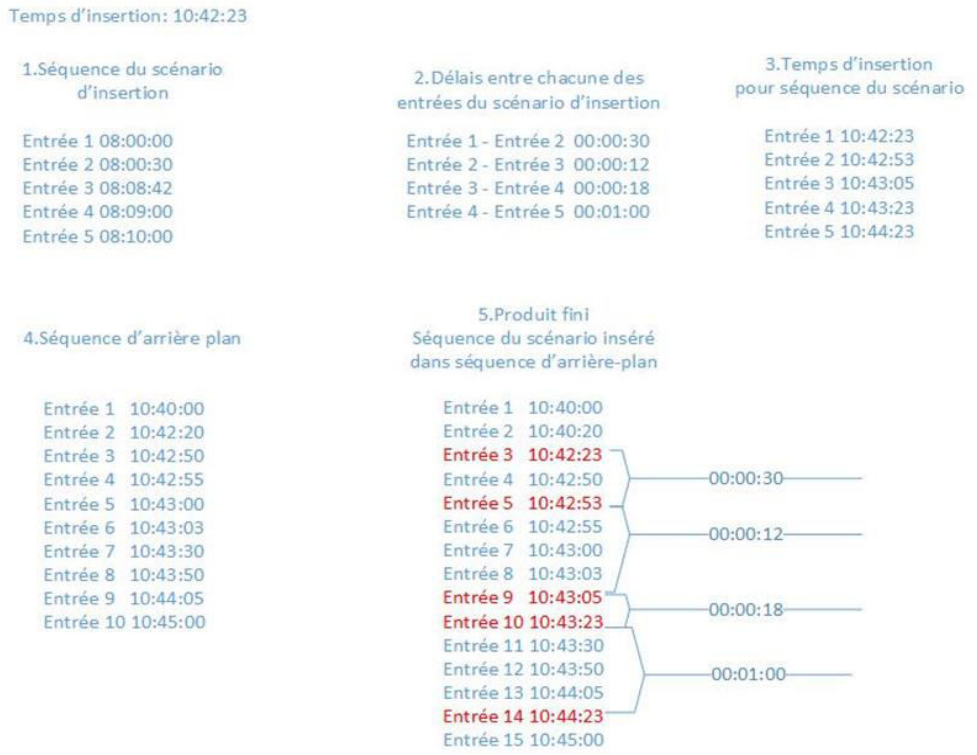

Figure 3-5. Processus d'insertion

En se servant de ces différences de temps entre chacun des logs à insérer et du temps d'insertion demandé. Nous sommes en mesure de calculer les temps d'insertion dans l'arrière-plan de chacune des entrées de la séquence à insérée, les temps insertions de notre exemple sont affichés en haut à droite de la figure.

La dernière étape du processus consiste à insérer la séquence de scénario en respectant les temps d'entrées de chacune des deux séquences. Le produit final est une séquence de logs qui incorpore la séquence de logs du scénario d'entraînement dans la séquence de logs servant d'arrière-plan en respectant les délais de chacune des entrées du scénario.

3.6.1. Considérations à propos du processus d'insertion

Notre approche préserve les délais d'insertion entre chacun des logs de la séquence du scénario inséré. Ces écarts de temps proviennent de processus de génération de la séquence de logs et ils doivent pourvoir être ajustées par l'utilisateur pour deux motifs. Premièrement, ces écarts peuvent varier d'un environnement à un autre. Par exemple, si les logs reliés à un scénario d'entraînement ont été générés dans un laboratoire sur un réseau fermé ayant moins d'activités qu'un réseau opérationnel, les écarts de temps entre les entrées des logs seront probablement moins grands en laboratoire que sur un réseau opérationnel où plusieurs événements sont générés. L'instructeur doit donc être en mesure de personnaliser les délais d'insertion entre les logs pour son environnement. Également, dans le cas d'un scénario complexe, il serait

souhaitable de donner la possibilité à l'instructeur de pouvoir altérer les délais d'insertions pour rendre le scénario adéquat selon ses objectifs d'entraînement. Un exemple trivial pourrait être de changer le temps d'un log relié à l'action d'un utilisateur, qui pourrait être l'ouverture d'un fichier déployant un logiciel malveillant.

L'instructeur devra aussi s'assurer que l'instance de SIEM d'entraînement puisse collecter et détecter les logs de la séquence reliée au scénario d'entraînement. Pour ce faire l'information sur la configuration du SIEM d'entraînement sera utilisée pour lui suggérer des altérations à la séquence de logs. Ceci pourrait inclure, soit d'omettre certains logs de la séquence du scénario (car ces logs ne seraient pas détectés par l'instance SIEM d'entraînement) ou bien de dupliquer certains logs (dans le but d'activer certaines règles de détection de l'instance du SIEM d'entraînement).

Un exemple très simple peut être utilisé pour illustrer ce problème. Un SIEM pouvant détecté des attaques de type force brute sera configuré avec des règles de détections qui sont changeantes. Une attaque de type force brute utilisée pour découvrir un mot de passe va générer plusieurs logs reliés à des tentatives de connexions qui échoueront sur une certaine période de temps limitée. Chaque instance de logiciel SIEM est configurée différemment pour reporter une alerte pour une tentative d'attaque de type force brute. Par exemple, une instance pourrait être configurée pour reporter une alerte quand plus de dix tentatives de connexions sont faîtes dans un délai de dix secondes. C'est pourquoi une solution d'assemblage complète pour assembler des séquences de logs devrait incorporer des fonctionnalités pour dupliquer certains logs reliés au scénario d'entraînement selon la configuration de l'instance du SIEM utilisée.

Lors de notre démarche de recherche, nous n'avons pas insisté sur les altérations qui devraient être réalisées sur la séquence de log assemblée pour qu'elle déclenche une ou plusieurs alertes sur un SIEM. Cependant, une solution complète d'un système d'assemblage de logs devrait inclure des fonctionnalités pour s'assurer que le scénario soit détecté par le SIEM selon les besoins de l'utilisateur. Nous croyons qu'avoir la possibilité d'enlever ou de dupliquer certains logs reliés au scénario d'entraînement dans la séquence serait suffisant dans la majorité des cas afin d'assurer qu'une alerte soit déclenchée par le SIEM. Ces fonctionnalités se doivent d'être incorporées à l'étape d'assemblage du processus d'insertion car le nombre de logs faisant parti du scénario de la séquence insérée influence grandement le processus.

Il est important de noter que deux ou plusieurs logs peuvent avoir exactement le même temps d'entrés dans une séquence, ce qui est courant pour une séquence de logs collectée par un SIEM. Nous pourrions avoir à insérer un log parmi plusieurs logs de l'arrière-plan ayant le même temps comme dans cet exemple :

Entrée 1 10 :00 :00
Entrée 2 10 :00 :00
Entrée 3 10 :00 :00

Entrée 4 10 :00 :00

Dans cette situation, nous pouvons insérer notre log entre n'importe quel des logs de l'arrière-plan sans aucun impact au processus de personnalisation. La position que nous choisirons aura cependant de l'influence sur la prochaine étape d'assemblage qui est la fusion.

Avant de procéder à l'exécution de l'étape de fusion, les positions d'insertion pour la séquence de logs reliée au scénario d'entraînement se doivent d'être décidées et fixées ; si la position d'un log est changée, le processus de fusion se doit d'être refait pour toute la séquence.

3.7 Processus de fusion

Le processus de fusion consiste à altérer la séquence de logs produite par le processus de l'insertion dans le but de solutionner les problèmes de disparités et de cohésions avec les champs. Ces erreurs proviennent des champs que nous avons identifiés comme étant de type neutre à l'étape de la personnalisation. Les logs ayant ces types de champ peuvent entrer en conflit avec des champs similaires existants dans l'arrière-plan lorsqu'ils sont insérés dans une nouvelle séquence de logs servant d'arrière-plan. Nous devons donc analyser tous les champs neutres de la séquence insérée que nous avons identifiés à l'étape de la personnalisation et s'assurer qu'ils sont cohérents pour l'ensemble de la séquence.

Chacun des champs qui sont catégorisé comme neutre ont leur propre fonctionnement. Il n'est donc pas possible de créer un algorithme général pour traiter tous les champs des senseurs qui sont catégorisés comme neutre. Cependant, il est possible de créer des algorithmes en utilisant des méthodes simples pour corriger les erreurs de cohésions existantes pour ces champs dans la séquence assemblée, soit en altérant les logs de la séquence insérée ou bien ceux appartenant à la séquence de l'arrière-plan.

Nos tests avec deux senseurs, le SDI *Suricata* et une station *Windows* nous ont montrés que chacun des champs neutres devait être traités de façon particulière. Cependant, souvent des algorithmes semblables pour différents champs peuvent être utilisés avec seulement de légers ajustements. Les prochaines sections couvrent quelques cas représentatifs des méthodes utilisées pour solutionner les problèmes de cohésion posés par certains champs neutres. Nous discutons du traitement particulier de trois types de champs dans les sections qui suivent.

3.7.1 Problématique du champ neutre '*ProcessID*' de Windows

Le champ '*ProcessID*' est utilisé par une station *Windows* pour identifier un processus qui est relié à un événement reporté par un log sur la station. L'insertion d'un log contenant un champ '*ProcessID*' pourrait causer des conflits potentiels avec la séquence de logs servant d'arrière-plan. Pour la majorité des scénarios d'entraînement, nous devons utiliser un numéro de processus qui n'est pas présent dans l'arrière-plan pour éviter un manque de cohésion. La Figure 3-6 illustre la procédure de base pour gérer les situations problématiques avec le

champ '*ProcessID*' et allouer une valeur à un champ 'ProcessID' que nous insérons. Si la station de travail utilisée durant le scénario d'entraînement n'est pas présente dans l'arrière-plan, nous allons allouer la valeur pour le 'ProcessID' que nous insérons entre 6000 et 9000 et qui est divisible par 4. *Microsoft Windows* alloue le numéro de processus par facteur de 4 et choisir un numéro de 'ProcessID' entre 6000 et 9000 nous empêche de rentrer en conflit avec certains numéros de processus qui sont réservés par *Windows* comme 0 pour le processus appelé '*Idle*' et 4 pour le processus appelé '*System*'.

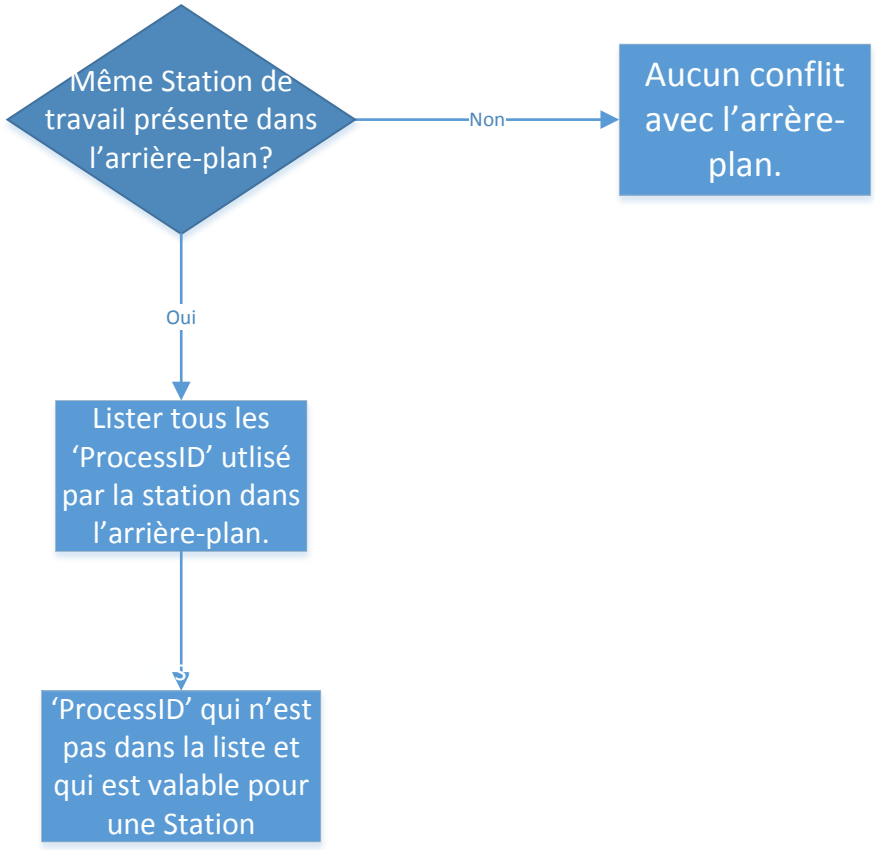

Figure 3-6. Détection et résolution d'un conflit potentiel pour le champ 'Process ID'

3.7.2 Problématique du champ 'Record Number'

Le champ appelé '*Record Number*' est le numéro d'identification alloué par le système d'exploitation *Windows* quand une entrée est faite dans un registre de logs. La méthode utilisée pour allouer le numéro d'identification ou le champ '*Record Number*' est très simple, la première entrée du registre débute à 1 et chacune des nouvelles entrées est augmentée par un. Lorsque le registre est plein, il est archivé et un nouveau registre est ouvert et la valeur de '*Record Number*' repart à 1. Si nous insérons une séquence de logs contenant des champs

'*Record Number*' nous devons suivre l'organigramme de la Figure 3-7 pour détecter un conflit potentiel.

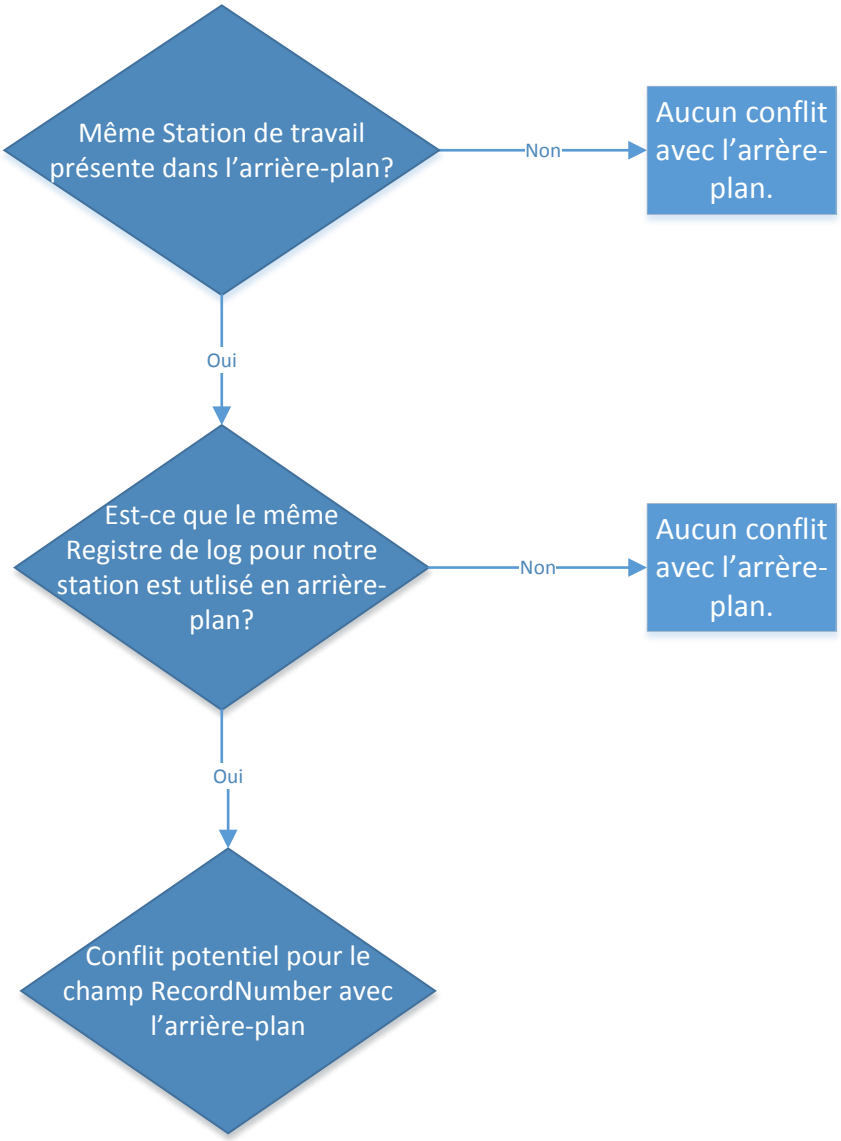

Figure 3-7. Organigramme pour détecter des conflits pour le champ 'Record Number'.

Pour qu'une situation de conflit potentiel existe, il doit y avoir un ou des logs de l'arrière-plan qui concerne la même station de travail *Windows* dont nous voulons insérer un log et qui utilise le même registre de logs. Si nous n'avons pas de conflit avec l'arrière-plan, nous allons allouer pour la valeur du champ '*ProccesID*' de notre log inséré, une valeur plausible, un chiffre au hasard entre 1000 et 10000 serait valable pour la plupart des cas.

Si un conflit potentiel existe, nous aurons à gérer une des situations illustrées à la Figure 3-8 où les logs insérées sont surlignés en rouge pour chacune des situations problématiques.

Situation 1.	Situation 2.
Entrée 1 12:42:32 Record Number 2	Entrée 1 12:40:32 Record Number X
Entrée 2 12:42:45 Record Number 3	Entrée 2 12:42:45 Record Number 3
Entrée 3 12:43:34 Record Number 4	Entrée 3 12:43:34 Record Number 4
Entrée 4 12:44:32 Record Number X	Entrée 4 12:44:32 Record Number 10
Entrée 5 12:46:23 Record Number 12	Entrée 5 12:46:23 Record Number 12

Situation 3.	Situation 4.
Entrée 1 12:42:32 Record Number 2	Entrée 1 12:42:32 Record Number 2
Entrée 2 12:42:45 Record Number 3	Entrée 2 12:42:45 Record Number 3
Entrée 3 12:43:34 Record Number 4	Entrée 3 12:43:34 Record Number 4
Entrée 4 12:44:32 Record Number 10	Entrée 4 12:44:32 Record Number x
Entrée 5 13:00:00 Record Number X	Entrée 5 13:00:00 Record Number 5

Figure 3-8. Situations problématiques pour le champ 'Record Number'

Dans le cas de la situation 1, nous avons inséré un log entre deux logs possédant un champ '*Record Number*'. Nous devons donc choisir pour le log que nous insérons une valeur pour '*Record Number'* qui est située entre la valeur du '*Record Number'* du log précèdent et la valeur du '*Record Number'* du log suivant pour que notre valeur semble légitime. Dans cette situation, nous devons choisir une valeur entre 5 et 11. Comme le log que nous insérons a été entré à un temps qui est plus proche du log précédente, nous devrions choisir un chiffre s'avoisinant à 4, on pourrait choisir 6 ou 7 ou 8, ainsi la valeur du champ '*Record Number'* pour le log inséré serait crédible.

Dans le cas de la situation 2, nous devons choisir une valeur pour le champ '*Record Number'* qui est inférieure à 3, dû au temps rapproché du log inséré avec le prochain log, qui est de 13 secondes, choisir la valeur pour le champ '*Record Number'* de 1 ou 2 serait convenable. Nous pourrions aussi choisir une valeur pour le champ 'Record Number' très grande, car le champ 'Record Number' peut contenir une valeur entre 0 et 2^{64}. Choisir une grande valeur pourrait donner l'illusion que le registre a été vidé et remis à zéro. Cependant, cette situation n'est pas fréquente.

La gestion de la situation 3 est similaire à la situation 2 mais inversé, nous devons choisir un 'Record Number' qui est plus grand que la dernière entrée. Le 'Record Number' se doit d'être plus grand proportionnellement au temps écoulé depuis le dernier log.

La gestion de la situation 4 est plus problématique en ce sens que nous devons changer la valeur du champ '*Record Number'* pour plusieurs des logs de la séquence, car nous devons usurper la valeur du champ 'Record Number' de l'entrée 5 pour le log que nous insérons. Nous allons donc assigner la valeur 5

pour l'entrée 4 et changer la valeur du '*Record Number*' pour l'entrée 5 à 6 comme ceci :

Situation 4. Solutionnée
Entrée 1 12:42:32 Record Number 2
Entrée 2 12:42:45 Record Number 3
Entrée 3 12:43:34 Record Number 4
Entrée 4 12:44:32 Record Number 5
Entrée 5 13:00:00 Record Number 6

Figure 3-9. Situation 4 solutionnée

Nous devons aussi changer la valeur du '*Record Number*' pour les logs après l'entrée 5 au besoin. L'allocation de la valeur pour le champ '*Record Number*' est proportionnelle au temps écoulé, et est très approximative car il n'est pas simple de construire un modèle précis pour prévoir quel sera la valeur du champ '*Record Number*' pour un registre d'une station *Windows* dans un temps donné. Il y a tout simplement trop de variables en jeux, comme par exemple l'activité sur la station de travail, le type de registre utilisé et la configuration de sa taille. Cependant, notre approximation a seulement besoin d'être crédible et normale. Également nous devons noter qu'il serait possible d'avertir l'utilisateur quand un conflit avec le champ '*Record Number*' a lieu, l'utilisateur serait en mesure d'utiliser une autre station de travail dans le cadre du scénario d'entraînement pour éviter le conflit.

3.7.3 Problématique du champ '*Flow_id*'

Premièrement, il est important de rappeler qu'un flot pour Suricata est un tuple constitué de cinq éléments : un protocole de communication réseau, une adresse IP pour la source, une adresse IP pour la destination, un port pour la source et finalement un port pour la destination, et que le champ '*flow_id*' identifie une communication bidirectionnelle entre deux éléments d'un réseau [16]. Nous pouvons présumer que la gestion des conflits causés par les '*flow_id*' sera fréquente car plusieurs logs d'un système SDI sont généralement collectés par un SIEM. Cependant, il existe plusieurs méthodes valables pour solutionner des conflits lorsqu'un log ayant le champ '*flow_id*' est inséré dans un arrière-plan ayant également plusieurs logs avec des champs '*flow_id*' provenant de la même instance d'un SDI Suricata.

L'allocation des valeurs pour le champ '*flow_id*' fait par Suricata n'est pas documentée de manière précise et certaines difficultés semble existées avec son fonctionnement [17]. Nous savons cependant que les valeurs pour le champ '*flow_id*' est un nombre entier de 10 chiffres divisible 4 qui croit à chaque fois qu'un nouveau '*flow_id*' est alloué. Le champ '*flow_id*', peut-être réalloué quand la communication liée à un champ '*flow_id*' est terminée. Également, le nombre maximum de flot que Suricata peut gérer par défaut avant de rentrer en mode d'urgence est de 10000.

Si par chance nous avons inséré un champ '*flow_id*' dans un arrière-plan ne contenant aucun champ '*flow_id*', conserver la valeur du '*flow_id*' de log inséré

est suffisant, cette situation est facile à gérer et n'est pas réaliste parce qu'elle n'est pas fréquente. La Figure 3-10 illustre les autres situations que nous devons gérer pour allouer des valeurs au champ '*flow_id*' pour les logs reliés au scénario d'entraînement que nous devons insérer. Il est à noter que la figure 3-10 représente une portion de la séquence qui a lieu longtemps après que les *'flow_id'* individuels ont été introduits. Il existe trois situations différentes, soit que nous insérons un log ayant un champ '*flow_id*' quand nous avons aucun champ '*flow_id*' qui le précède dans la séquence d'arrière-plan, le cas illustré par la situation 2. La situation 3 est l'inverse, nous n'avons aucun champ '*flow_id*' qui suit dans l'arrière-plan. La situation 1 est une insertion au milieu de plusieurs logs d'arrière-plan qui contiennent des champs '*flow_id*'.

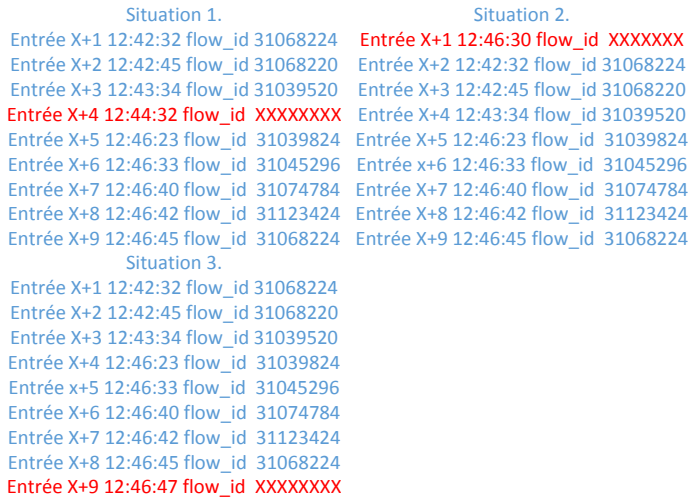

Figure 3-10. Situations problématiques avec le champ 'Flow_id'

La Figure 3-11 illustre l'organigramme pour assigner une valeur à un champ '*flow_id*' pour un log inséré pour les trois situations décrites précédemment. En premier lieu, dans tous les cas, nous devons construire une liste de tous les valeurs pour les champs '*flow_id*' utilisés dans l'arrière-plan, cette liste va nous aider à décider de la valeur choisie pour le champ 'flow_id' que nous insérons. La liste des *'flow_id*' en ordre croissant pour les 3 situations est :

31039520, 31039824, 31045296, 31068220, 31068224, 31074784, 31123424

Vous noterez que *Suricata* insère une entrée à chaque fois qu'une communication dans le flux est observée, ce qui explique pourquoi on voit deux entrés avec le *'flow_id*' 31068224 à la figure 3-10. Bien que les *'flow_id*' sont créés en ordre croissant, notre séquence est capturé plus tard dans l'arrière-plan, et les entrées les plus vielles dans la figure 3-10 n'ont pas nécessairement les *'flow_id'* les plus petits.

Les situations 2 et 3 sont les plus simples à solutionner. Dans le cas de la situation 2, la valeur pour le champ '*flow_id*' sera égale à la valeur du plus petit '*flow_id*' trouvé en arrière-plan, auquel nous soustrairons quatre, donc 31039516. Pour le

cas de la situation 3, nous devons faire le contraire, notre valeur sera la plus grande valeur trouvée en arrière-plan et auquel nous allons additionner quatre, donc 31123428. Si nous avons le cas de la situation 1, nous devons trouver une valeur de *'flow_id'* qui est plus vielle que le *'flow_id'* précédent le plus vieux, tout en étant la plus proche possible de celui-ci. Pour ce faire, nous énumérons tous les valeurs des champs '*flow_id*' précèdent le plus vieux avant le point d'insertion en ordre décroissant, et on choisit la prochaine valeur de '*flow id*' disponible parmi la liste triée. Par exemple, nous avons les valeurs pour les champs '*flow_id*' qui précède notre position d'insertion pour la situation 1:

31039520, 31068224, 31068220

Nous choisissons 31039524 puisque c'est la prochaine valeur disponible. Cette valeur sera aussi utilisée pour les autres logs insérés appartenant au même flot. Si aucun « trou » n'est disponible parmi la séquence nous allons simplement assigner une valeur plus grande comme à la situation 3.

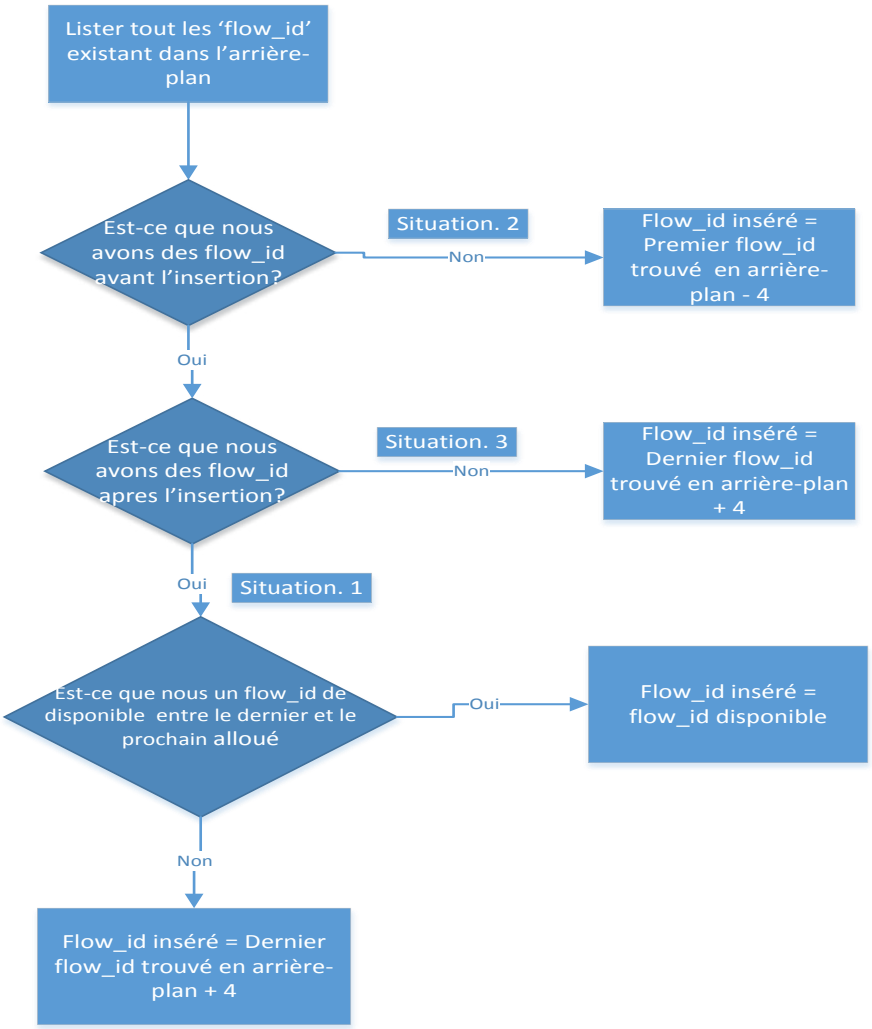

Figure 3-11. Organigramme de décision pour traitement du champ 'flow_id'

Une autre situation exceptionnelle pourrait avoir lieu, dans le cas que plus de 10000 champs 'flow_id' seraient présents dans la séquence, nous aurions à dupliquer une des entrées de l'arrière-plan. Suricata entre en mode d'urgence quand plus de 10000 flots sont ouverts, le programme va fermer le flot qui été le moins actif pour le réassigner à un nouveau flot. Nous pourrions donc assigner une valeur à notre 'flow_id' inséré, en choisissant le 'flow_id' qui a le plus petit nombre de logs dans l'arrière-plan pour simuler la réassignation faite par Suricata. Il est cependant important de noter que cette situation est exceptionnelle.

3.7.4 Limites du processus de fusion

Le processus de fusion ne corrige parfois pas totalement les problèmes de cohérence dans certaines situations. Le champ '*RecordNumber*' de *Windows* en est un exemple, il est difficile de prévoir précisément quel sera le numéro du

'*RecordNumber*' sur une station après une certaine période de temps. Il y a tout simplement trop de facteurs qui entrent en jeu pour être en mesure de faire une estimation précise. Cependant, si nous sommes en mesure d'assigner une valeur qui est crédible, ceci est suffisant pour atteindre l'effet désiré pour l'entraînement d'opérateurs de logiciel SIEM.

La correction des champs neutres de la séquence de log reliée au scénario inséré dépend de la séquence d'arrière-plan utilisée. Il est souvent possible d'éviter les situations conflictuelles avec l'arrière-plan simplement en altérant le scénario d'entraînement. Par exemple, dans le cas d'une station *Windows*, seulement utiliser une autre station qui n'est pas présente dans la séquence d'arrière-plan peut éviter un conflit. Il est possible d'analyser la séquence d'arrière-plan pour suggérer à l'utilisateur des éléments d'un scénario qui seraient optimaux pour éviter les conflits avec les champs neutres.

Il est important cependant de noter que l'instructeur devra vérifier la séquence fusionnée car des problèmes de cohérence avec le scénario pourraient être causés par un manque d'information dans la séquence servant d'arrière-plan. Ce manque d'information dans l'arrière-plan pourrait causer des problèmes de contexte dans la nouvelle séquence de logs fusionnée. Pour illustrer ce problème, prenons l'exemple simple d'un scénario dont nous voulons assembler la séquence de logs à une séquence de logs servant d'arrière-plan. Bob ouvre un fichier sur sa station de travail qui contient un virus. Nous devons insérer seulement un log dans une séquence d'arrière-plan, ce log rapporte une détection de virus sur la station de Bob. Cependant, il pourrait y avoir un problème si le SIEM est configuré pour reporter qu'un utilisateur se connecte et est présent sur une station de travail. Les logs reportant que Bob s'est connecté à la station de travail devraient être présents dans la séquence servant d'arrière-plan pour préserver la cohésion du scénario. Si ces logs ne sont pas présents, nous aurons une disparité qui va causer un manque de cohésion.

Ces erreurs se doivent d'être détectées par l'instructeur de SIEM car le processus de fusion qui nous proposons ne le détectera pas. Il existe cependant des solutions possibles pour solutionner ces erreurs. On pourrait utiliser un utilisateur autre que Bob, qui serait présent dans la séquence d'arrière-plan, des logs faisant partis de l'arrière-plan montreraient que l'utilisateur utilisé pour le scénario est connecté à une station de travail. Notre implémentation pourrait suggérer ces options à l'instructeur. Une autre solution valable serait d'ajouter dans la séquence de logs reliée au scénario d'entraînement, les logs qui montreraient que Bob est connecté à sa station. Ces logs feraient maintenant partis du scénario à assembler, cependant l'ajout de ces logs pourrait causer d'autre erreur de cohésion dans la séquence et nous devions les ajouter avec prudence. La participation de l'instructeur est essentielle pour solutionner les erreurs de cohésion et vérifier la fiabilité de la nouvelle séquence créée.

3.7.5 Considérations pour l'implémentation du processus de fusion

L'implémentation du processus de fusion passe par l'identification des champs de type neutre pour chacun des logs insérés qui sont reliés au scénario. Pour chacune des séquences de logs reliées à un scénario d'entraînement que nous voulons assembler, nous devrons donc modifier notre programme pour incorporer la gestion de champs neutres qui ne sont pas implémentée. La présence de nombreux champs neutres pourrait possiblement compliquer l'implémentation du processus de fusion en exacerbant la complexité du développement.

Nous avons donc étudié brièvement les champs de quelques autres types de senseurs pour vérifier sur les champs neutres sont fréquents. Nous avons vérifié les champs des senseurs suivants : serveur http *Apache* et *Microsoft IIS*, serveur courriel *Postfix*, et le serveur ftp *Microsoft*. Les champs neutres ne semblent pas fréquents et sont souvent reliés au fonctionnement interne pour ces senseurs. Par exemple, l'étude des logs produits par le serveur web Apache nous a montré que seulement un champ était de type de neutre et que ce champ était relié à l'identité du processus qui traite la demande du client web. La méthode pour gérer ce champ serait similaire à la gestion du champ '*Process ID'* d'une station *Windows*. L'échantillon vérifié est bien sûr limité et rien de ne garantit que certains senseurs aient un grand nombre de champs neutres. Nous croyons cependant que les problèmes causés par les champs neutres sont la plupart du temps gérables. Ils peuvent être solutionnés en implémentant des algorithmes relativement simples comme nous avons démontré avec notre approche. Il est aussi possible de détecter et d'avertir l'utilisateur pour les situations qui sont plus problématiques, et de lui suggérer des options de simple altération au scénario inséré pour simplement éviter un problème causé par un champ neutre. Les difficultés causées par les champs neutres ne sont pas des obstacles insurmontables pour l'implémentation d'une solution semi-automatisée d'assemblage de séquence de logs.

3.8 Avantages de la solution proposée

L'architecture proposée pour assembler des séquences de logs utilisables pour entraîner des opérateurs de logiciels SIEM fournit les avantages suivants :

1. La séquence de logs reliée au scénario d'entraînement est incorporée dans un arrière-plan qui est réaliste et efficace.
2. La capacité de réutiliser des séquences de logs reliées à des scénarios d'entraînement dans un contexte différents grâce à un arrière-plan changeant.
3. Évite d'avoir à construire une séquence de logs servant d'arrière-plan qui est un processus complexe.

Enregistrer des séquences de logs produites par les senseurs d'un réseau qui est sous l'attaque d'une équipe *« Red Team »* va produire une séquence de logs qui

est plus efficace comparativement à notre approche. Cependant, l'arrière-plan dans ces séquences ne pourra pas être altéré facilement. L'entraînement sera donc redondant quand ces séquences seront utilisées et elles auront une utilité limitée lorsqu'elles seront utilisées sur un réseau d'entraînement comme un cyber-range.

3.9 Conclusion du chapitre.

Ce chapitre a proposé une architecture pour assembler des séquences de logs dans le but d'entraîner des opérateurs de logiciel SIEM en utilisant des connecteurs de reprise. Nous avons présenté les processus de personnalisation, d'insertion et de fusion qui composent les étapes essentielles à l'incorporation d'une séquence de logs reliée à un scénario d'entraînement à une séquence de logs reliée à un arrière-plan. Pour être en mesure d'évaluer si l'approche proposée est utilisable pour entraîner des opérateurs, nous devons démontrer qu'il est possible d'assembler des scénarios d'entraînements ayant une certaine complexité. Le prochain chapitre présente la démarche que nous avons utilisée pour valider notre approche.

Chapitre 4 : Implémentation et validation

4.1 Introduction

L'objectif de ce chapitre est de valider l'efficacité des méthodes que nous avons proposées pour assembler des séquences de logs utilisables pour entraîner des opérateurs de logiciel SIEM. Pour ce faire, nous avons développé comme preuve de concept un prototype implémentant les diverses fonctionnalités de personnalisation, d'insertion et de fusion pour un nombre limité de senseurs. Nous avons utilisé notre prototype pour assembler de manière semi-automatique une séquence de logs reliée à un scénario typique d'incident de sécurité à différentes séquences de logs servant d'arrière-plan dans le but de tester son efficacité. En premier lieu, nous allons discuter brièvement des exigences et détails de l'environnement d'implémentation où nous avons développé le prototype. En deuxième lieu, nous allons décrire le scénario d'entraînement utilisé en détail pour valider notre approche, pour par la suite décrire les résultats que nous avons obtenus et conclure le chapitre avec une discussion à propos de ceux-ci.

4.2 Exigences de la preuve de concept

L'implémentation d'une solution complète implémentant les concepts de personnalisation, d'insertion et de fusion supportant de nombreuses fonctionnalités est hors de la portée de cette recherche. Cependant, l'implémentation d'un prototype étant capable d'assemblé une séquence de logs reliée à un scénario d'entraînement représentatif et ayant une certaine complexité, est intéressante pour démontrer que notre concept d'assemblage de séquences de logs est possible. Ce prototype implémente un nombre limité de senseurs et de fonctionnalités mais est suffisamment représentatif pour démontrer la validité de notre approche d'assemblage de logs.

Les exigences pour chacune des fonctionnalités de prototype sont :

1. Personnalisation : le prototype se doit être en mesure de personnaliser une séquence de logs reliée à un scénario d'entraînement en acceptant les entrants suivants :
    a. Une séquence de logs reliée à un scénario d'entraînement.
    b. Les informations reliées à un réseau de destination utilisé pour l'entraînement.
    c. Les préférences de l'utilisateur pour le scénario d'entraînement.

    Les informations reliées au réseau de destination et aux préférences des utilisateurs peuvent être fournies au prototype à l'aide de moyens manuels, c'est-à-dire que nous avons assigné des valeurs plausibles à notre prototype directement en assignant des valeurs à des constantes dans le code. Le développement de méthodes semi-automatiques pour extraire ces informations ne fait pas partie de notre démarche de

recherche. Le produit de la personnalisation sera une séquence de logs reliée à un scénario d'entraînement avec tous les champs personnalisés à part ceux de type neutre. Ces champs neutres se doivent être identifiés, pour être en mesure d'être ajustés durant l'étape de la fusion.

2. Insertion : le prototype se doit d'être en mesure d'insérer une séquence de logs dans un arrière-plan donné en prenant les entrants suivants :
    a. Un temps d'insertion pour le scénario.
    b. Une séquence de logs personnalisée liée à un scénario d'entraînement.
    c. Une séquence de logs servant d'arrière-plan.
    d. Informations sur la configuration du SIEM utilisé sur le réseau d'entraînement.
    e. Préférences des utilisateurs pour les délais d'insertion entre chacun des logs reliés au scénario d'entraînement et le choix de l'utilisateur d'omettre des logs reliés aux scénarios.

    Les informations sur la configuration du SIEM et les préférences des utilisateurs peuvent être fournies au prototype en assignant directement des valeurs à des constantes dans le code. Le produit de l'insertion sera une séquence de logs qui va incorporer les logs de la séquence personnalisée avec la séquence de logs servant d'arrière-plan dans un ordre chronologique.
3. Fusion : le prototype se doit être en mesure d'altérer une séquence de logs selon les champs neutres identifiés durant l'étape de la personnalisation en acceptant une séquence de logs reliée à un scénario d'entraînement et qui a été inséré dans une séquence de logs d'arrière-plan. Le produit final de la fusion sera une séquence de logs rejouable sur un réseau d'entraînement.

4.3 Description de l'environnement

Cette section discute de divers aspects techniques de notre prototype.

4.3.1 Langage de développement et environnement de développement intégré

Nous avons choisi pour développer notre prototype le langage de programmation Python[18]. Plusieurs motifs nous ont conduits à faire ce choix :

1. Capacité de développement rapide, un avantage important pour développer un prototype.
2. Une capacité de programmation orientée objet.
3. La manipulation facile et efficace de chaînes de caractères.

4. Python est munit de structures de données intégrées permettant une manipulation aisée de l'information contenue dans un log.
5. Existence de plusieurs modules ayant des fonctionnalités permettant la gestion efficace de diverses normes de log dont la norme JSON.
6. La documentation sur Python est abondante et facile à consulter.

Nous avons utilisé l'environnement de développement intégré à code source ouvert PyCharm distribué par la compagnie JetBrains [19] pour développer notre prototype.

4.3.2 Norme de log utilisée pour l'implémentation du prototype

La norme de log JSON offre plusieurs avantages pour l'implémentation de notre prototype. La norme JSON inclut la présence du nom des champs avec sa valeur, ce qui rend ce format très facile à lire pour un humain; un facteur qui a facilité grandement le développement et le débogage de notre prototype. Les logs qui sont dans un format JSON peuvent également être convertis relativement facilement dans une autre norme et peuvent être directement utilisés par plusieurs implémentations de logiciel SIEM dont Splunk [20]. Une autre motif pour lequel nous avons choisi la norme JSON, c'est qu'il permet facilement d'être représenté dans des objets et des tableaux qui permettent une représentation directe dans les structures de données du langage Python, comme par exemple les type dictionnaires et les listes référencées [21]. Cette caractéristique du format JSON facilite grandement le traitement de séquences de logs par la plupart des langages de programmation particulièrement le langage de programmation Python.

Nous avons choisi le format de logs JSON pour implémenter notre prototype, cependant il aurait été très certainement possible d'utiliser une norme de log fortement structuré comme par exemple la norme CEF utilisé par le SIEM *Arcsight*. L'effort de développement aurait probablement été plus important mais quand même tout à fait possible.

4.3.3 Architecture du prototype implémenté

L'architecture de notre prototype est basée sur deux modules Python, le premier module implémente la fonction de personnalisation et le deuxième module implémente les fonctions d'insertion et de fusion. Pour stocker les diverses séquences de logs requises par le prototype, nous avons utilisé des fichiers de format log de type JSON conservés dans des répertoires de fichier plat (en anglais « flat directory »), qui sont simplement un répertoire classique de fichier implémenté par le système d'opération *Unix*. Ce type de format de stockage était suffisant pour notre implémentation et nous l'avons utilisé pour chacun des répertoires de notre architecture illustrée à la Figure 4-1. Le module 1 qui implémente la personnalisation lit des séquences de logs reliées à un scénario d'entraînement qui ont été collectées ou produites d'un fichier enregistré dans le répertoire de scénario d'entraînement. Il faut noter que le module de personnalisation a besoin également de données liées au réseau qui sera utilisé

pour l'entraînement et des préférences de l'utilisateur pour personnaliser les logs. Nous avons stocké ces informations dans des constantes du module 1 pour simuler ces entrées de donnés pour le prototype. Nous avons utilisé le même moyen pour les informations requises pour le module 2 à propos de la configuration du SIEM utilisé pour l'entraînement et des préférences de l'utilisateur.

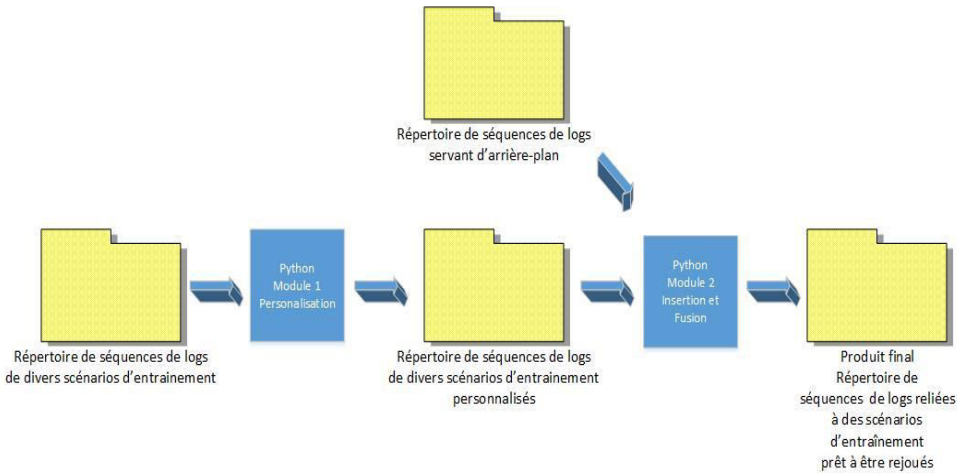

Figure 4-1. Architecture du prototype

La séquence de log personnalisée produite par le module 1 est par la suite stockée dans un répertoire qui est disponible pour le module 2. Le module 2 va insérer ces séquences personnalisées dans les séquences de logs servant d'arrière-plan. Les différentes séquences d'arrière-plan qui ont été collectées de notre réseau sont également stockées dans un répertoire qui est accessible pour le module 2. Finalement, le résultat final produit par le module 2, est stocké dans un répertoire et prêt à être rejoué par des connecteurs de reprise sur un réseau d'entraînement.

4.4 Scénario de validation

Pour valider notre système, nous avons utilisé un scénario d'entraînement représentatif et ayant une certaine complexité, cette section décrit ce scénario. La Figure 4-2 illustre les différentes étapes du scénario de validation qui implique le membre de Département de la défense du pays Pineland appelé Bob qui a été ciblé par une opération de la division cybernétique des forces d'oppositions. L'objectif des forces oppositions est extraire des documents confidentiels qui sont présents sur la station de travail de Bob.

Pour ce faire, Natacha, une agente des forces de l'opposition, ayant un grand talent de persuasion, a été envoyée pour convaincre Bob de télécharger un fichier intéressant pour son travail. Bob s'est laissé convaincre de télécharger un fichier de format de type `Adobe Portable Format (PDF)`, en apparence inoffensif, sous la suggestion de la convaincante Natacha. Bob sans le savoir a téléchargé un

fichier qui contient un logiciel malicieux de type cheval de Troie qui va s'activer lorsque Bob ouvrira le fichier pour le lire.

Le cheval de Troie est un logiciel malveillant qui va installer sur la station de travail de Bob. Ce type de logiciel malveillant est appelé en anglais *backdoor*, ou une porte dérobée en français. Cette porte dérobée consiste à un programme d'invite de commande qui communique à l'aide d'un tunnel http inversé au serveur de contrôle des forces de l'opposition. L'agent des forces de l'opposition sera ainsi en mesure de recevoir une demande de connexion de la porte dérobée installée sur la station de Bob à partir de son serveur web factice. Un tunnel de communication inversé se servant du protocole http sera ainsi ouvert. Il permettra à l'agent ennemi d'avoir accès à la station de Bob à l'aide du logiciel d'invite malicieux installé et d'en d'extraire les documents confidentiels de celle-ci.

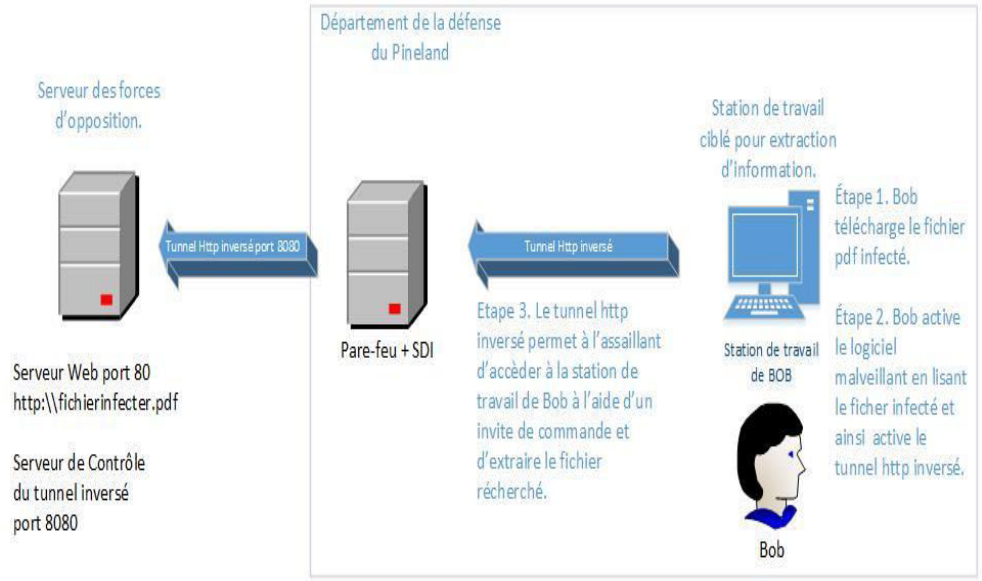

Figure 4-2. Aperçu du scénario de validation

La méthode de communication d'un tunnel http inversé peut être utilisée par les logiciels malveillants pour circonvenir les dispositifs de sécurité des pare-feux installés sur un réseau. Le tunnel http inversé utilise les ports reliés au protocole http pour communiquer de l'intérieur vers l'extérieur. Ces ports se doivent d'être ouverts sur le pare-feu pour permettre aux utilisateurs du réseau de naviguer sur les serveurs web qui sont à l'extérieur du réseau, ce qui fait qu'un tunnel HTPP inversé est une méthode efficace pour extraire de l'information d'un réseau de manière furtive. Il est important de noter que l'information transmise dans un tunnel http inversé est très souvent encryptée et camouflée dans les url des requêtes http exécutées par le logiciel malicieux.

Le scénario de validation couvre toutes les étapes du modèle appelé en anglais « *kill chain* » qui décrit le processus systématique de ciblage et d'engagement qu'un

adversaire accompli pour être en mesure de s'introduire dans un réseau protégé et de l'exploiter [22]. La Tableau 4-1 présente des caractéristiques de scénario selon les diverses étapes du modèle « *kill chain* ». Notre scénario de validation couvre toutes les étapes d'une opération d'attaque cybernétique complète et est intéressant dans le cadre d'entraînement d'opérateur de logiciel SIEM.

Tableau 4-1. Scénario selon modèle du 'kill chain'

Étape du modèle '*kill chain*'	**Caractéristique du scénario de validation**
1. Reconnaissance	Bob est identifié comme une cible intéressante à l'aide de méthodes de reconnaissance utilisées par les services de renseignements de la force d'opposition.
2. Arsenalisation	Le fichier PDF est muni d'une charge utile exploitable.
3. Livraison	La charge utile est livrée à l'aide des actions de Natacha et du serveur web de la force d'opposition.
4. Exploitation	Bob ouvre le fichier PDF infecté et exécute la charge utile sur sa station de travail.
5. Installation	Un programme d'invite de commande contrôlable à distance est installé sur la station de travail de Bob et exploitable à l'aide d'un tunnel http inversé.
6. Commande et contrôle (C2)	La station de travail infectée de Bob émet un signal au serveur de contrôle de la force d'opposition qui accepte la connexion. L'agent ennemi peut ainsi contrôler à distance la recherche des fichiers sur la station de travail de Bob.
7. Actions sur l'objectif	Les fichiers confidentiels sur la station de Bob sont trouvés et l'extraction est faîte par l'agent ennemi à partir de son serveur de contrôle. La connexion est par la suite interrompue.

4.4 Environnement test utilisé pour simuler le scénario de validation

Cette section décrit l'environnement qui a été utilisé pour collecter les logs qui sont reliés au scénario décrit à la section précédente. Nous avons bâti un réseau dans un environnement virtuel pour être en mesure de répliquer notre scénario

de validation. L'objectif étant de récolter les séquences de logs qui sont reliées au scénario pour être en mesure de les réutiliser pour valider notre prototype.

La Figure 4-3 illustre le réseau que nous avons conçu pour faire nos tests. La portion de l'environnement externe émule le rôle de l'internet ou de l'environnement externe d'un réseau protégé. L'environnement externe est constitué d'une seule station dont le rôle est de servir d'aide à la construction du scénario d'entraînement, la plupart du temps en attaquant le réseau interne. Nous avons utilisé le système d'exploitation *Kali 2.0* qui est muni d'une riche suite d'outils servant à tester la configuration de sécurité d'un réseau dont le serveur web *Apache 2.4.10* que nous avons utilisé pour notre scénario [21].

La portion du réseau interne émule une partie d'un réseau d'une entreprise, elle est constituée d'une station de travail, un serveur servant de pare-feu et qui surveille les entrées et sorties à l'aide d'un SDI, et d'une station collectant les logs des différents senseurs à l'aide du logiciel SIEM Splunk.

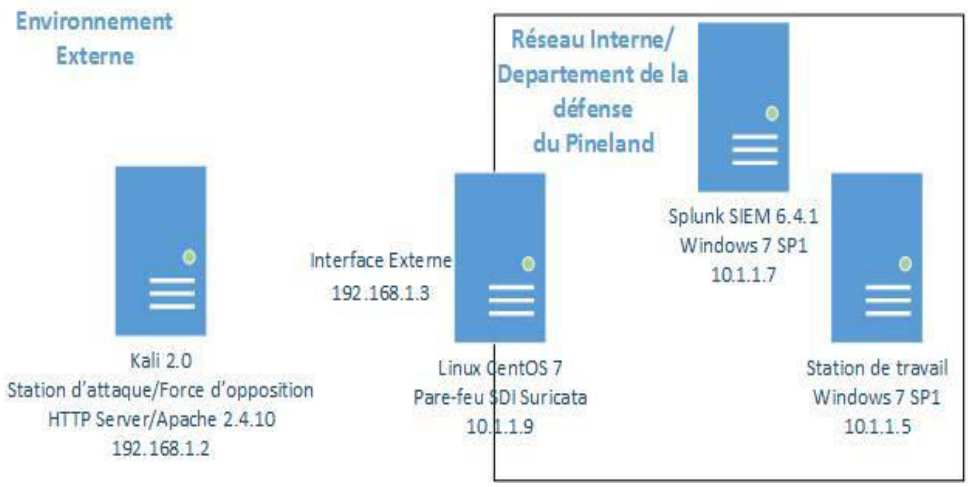

Figure 4-3. Environnement test

Le serveur de pare-feu est muni de deux interfaces de réseau et sert de pont entre le réseau externe et le réseau interne. Pour implémenter ce serveur nous avons utilisé le système d'exploitation *CentOS 7*, en conjonction avec le logiciel pare-feu *Iptables*. La configuration du pare-feu est très simple, elle comprend la mascarade IP, et permet aux utilisateurs de la station interne de pouvoir naviguer sur le serveur web du réseau externe. Pour surveiller le trafic entre le réseau interne et le réseau externe, nous avons installé le SDI *Suricata* configuré en série (en anglais '*inline mode*'). Cette configuration est typique d'un réseau courant d'entreprise et convient pour notre scénario de validation.

La partie du réseau interne comprend une station de travail munie du système d'exploitation *Windows 7 Service Pack 1*. Nous avons choisi la version de ce système d'exploitation parce qu'il est vulnérable et sous-susceptible d'être exploité relativement facilement avec les outils disponibles sur la station d'attaque.

Nous avons installé le SIEM *Splunk* sur une station *Window 7*. Nous avons utilisé cette station pour faire nos tests avec le SIEM *Splunk* au besoin. Nous avons capturé les logs de deux capteurs, le SDI *Suricata* installé sur le serveur servant de pare-feu, et la station de travail *Windows 7* du réseau interne.

4.4.1 Implémentation du scénario d'attaque de validation

Pour nous aider à générer l'attaque requise pour le scénario de validation, nous avons utilisé l'outil *Metasploit* qui est installé par défaut sur la station d'attaque de notre réseau externe. L'outil *Metasploit* possède un module permettant d'assembler un fichier PDF avec un programme malicieux [23]. Il est important de noter que nous avons utilisé une version de système d'exploitation *Windows* et du logiciel *Adobe Reader* périmée pour s'assurer que la charge malicieuse s'exécute correctement.

L'outil *Metasploit* permet de personnaliser le programme malicieux camouflé dans le fichier PDF. Pour les besoins de notre scénario, nous avons choisi d'implémenter un interpréteur de commande de type *Metasploit,* ce logiciel de type d'invite de commande est muni de plusieurs fonctionnalités dont entre autres la capacité de chercher des fichiers sur une station Windows et de les télécharger.

Le logiciel *Metasploit* permet aussi de personnaliser le moyen de communication que l'interpréteur de commande va utiliser pour communiquer avec son contrôleur et nous avons choisi d'utiliser un tunnel http inversé pour ce faire. Nous avons utilisé cette fonctionnalité pour ouvrir une connexion entre la station d'attaque et la station de travail interne, ainsi l'attaquant, à partir de la station d'attaque, était en mesure d'accéder à un interpréteur de commande sur la station de travail du réseau interne. Le système d'exploitation *Kali 2.0* est muni d'un serveur web, *Apache 2.4.10*, que nous avons utilisé pour distribuer notre fichier infecté, à Bob pour notre le besoin de notre scénario.

4.5 Séquence de logs collectée

Cette section décrit comment les logs reliés au scénario de l'incident ont été collectés sur notre réseau test. Nous avons décidé de limiter notre collecte de logs à deux senseurs soit le SDI *Suricata* et la station de travail *Windows* de Bob. Nous avons modifié les configurations par défaut de ces deux senseurs comme décrit ici-bas. Dans le cas du SDI Suricata, la configuration par défaut reporte le téléchargement de fichiers. Cependant Suricata ne détecte pas l'activité d'un interpréteur *Metasploit* qui communique à l'aide d'un tunnel inverse HTTP, pour ce faire des règles de détections doivent d'être ajoutées à la configuration par défaut [24]. Nous avons choisi de ne pas ajouter ces règles de détection pour fin de simplicité. Une station *Windows* avec sa configuration par défaut ne génère aucun log, nous avons configuré la station pour générer des logs manière selon les meilleurs pratiques [25], [26]. Une station *Windows* génère une quantité importante de logs, seule une quantité minimale de logs est habituellement transférée au SIEM, nous avons choisis le log qui est intéressant pour notre scénario.

Nous allons décrire chacune des activités du scénario qui génèrent des logs sur ces deux senseurs. La première activité du scénario qui peut être détectée par les senseurs est le téléchargement par Bob du fichier *PDF* infecté. La Figure 4-4 illustre cette première étape.

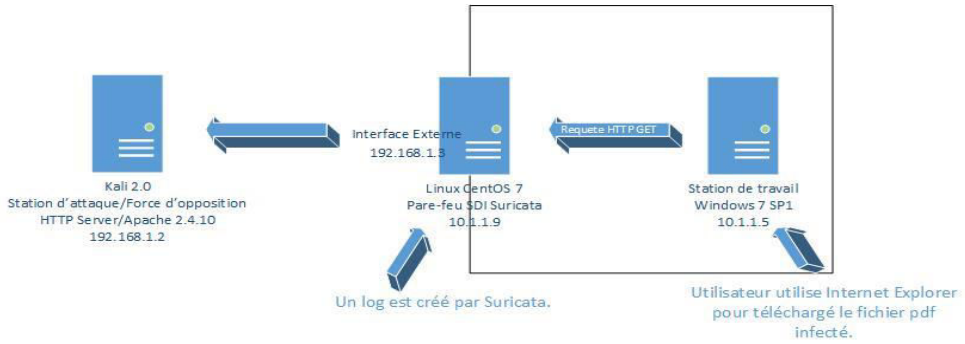

Figure 4-4. Étape 1 Téléchargement du fichier PDF infecté

Le log suivant a été créé par Suricata suite à la requête http faite par Bob pour télécharger le fichier PDF infecté :

Tableau 4-2. Log lié à un téléchargement de fichier


```
{"timestamp":"2016-05-30T12:54:32.365020-
0400","flow_id":37154240,"in_iface":"eno16777736","event_typ
e":"http","src_ip":"192.168.1.3","src_port":49186,"dest_ip":
"192.168.1.2","dest_port":80,"proto":"TCP","tx_id":0,"http":
{"hostname":"192.168.1.2","url":"\/template.pdf","http_user_
agent":"Mozilla\/4.0 (compatible; MSIE 8.0; Windows NT 6.1;
WOW64; Trident\/4.0; SLCC2; .NET CLR 2.0.50727; .NET CLR
3.5.30729; .NET CLR 3.0.30729; Media Center PC
6.0)","http_method":"GET","protocol":"HTTP\/1.1","length":0}
}
```


La deuxième activité du scénario détectée par les senseurs a lieu lorsque Bob ouvre le fichier *PDF* infecté afin de le lire, Bob déclenche ainsi le programme malicieux qui ouvre le tunnel http inversé. Le SIEM collecte certains logs, la Figure 4-5 illustre le nombre de logs collectés.

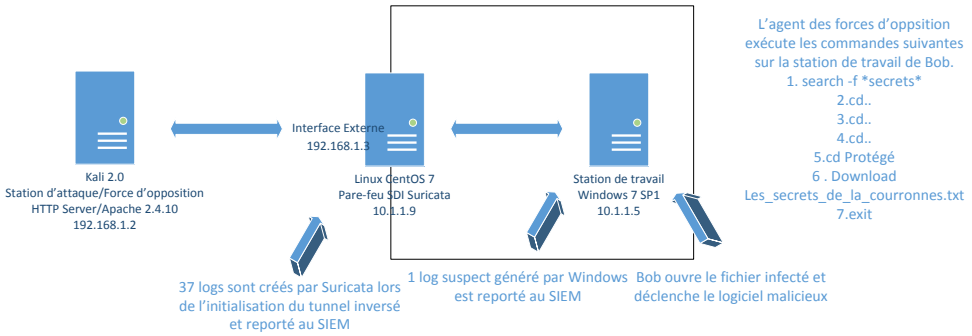

Figure 4-5. Étape 2 Ouverture du programme.

Le log suivant est reporté au SIEM parce qu'une hausse de privilèges est détectée et également l'exécution d'une invite de commande suspecte est faîte sur la station *Windows* de Bob :

Tableau 4-3. Log lié à détection d'une hausse de privilèges


```
{"timestamp":"2016-05-30 12:55:05","Hostname":"WIN-RL1FM3ULMHA","Keywords":-
9214364837600034816,"EventType":"AUDIT_SUCCESS","SeverityValue":2,"Severity":"INFO"
,"EventID":4688,"SourceName":"Microsoft-Windows-Security-
Auditing","ProviderGuid":"{54849625-5478-4994-A5BA-
3E3B0328C30D}","Version":1,"Task":13312,"OpcodeValue":0,"RecordNumber":19545,"Proce
ssID":4,"ThreadID":48,"Channel":"Security","Message":"A new process has been
created.\r\n\r\nSubject:\r\n\tSecurity ID:\t\tS-1-5-21-75874118-1364727433-
2505763305-1000\r\n\tAccount Name:\t\tSEB\r\n\tAccount Domain:\t\tWIN-
RL1FM3ULMHA\r\n\tLogon ID:\t\t0x5a60e\r\n\r\nProcess Information:\r\n\tNew Process
ID:\t\t0x9b8\r\n\tNew Process Name:\tC:\\Windows\\SysWOW64\\cmd.exe\r\n\tToken
Elevation Type:\tTokenElevationTypeLimited (3)\r\n\tCreator Process
ID:\t0x550\r\n\tProcess Command Line:\t\"C:\\Windows\\System32\\cmd.exe\" /Q /C
%HOMEDRIVE%&cd %HOMEPATH%&(if exist \"Desktop\\template.pdf\" (cd \"Desktop\"))&(if
exist \"My Documents\\template.pdf\" (cd \"My Documents\"))&(if exist
\"Documents\\template.pdf\" (cd \"Documents\"))&(if exist
\"Escritorio\\template.pdf\" (cd \"Escritorio\"))&(if exist \"Mis
Documentos\\template.pdf\" (cd \"Mis Documentos\"))&(start
template.pdf)\n\n\n\n\n\n\n\n\nTo view the encrypted content please tick the \"Do
not show this message again\" box and press Open.\r\n\r\nToken Elevation Type
indicates the type of token that was assigned to the new process in accordance with
User Account Control policy.\r\n\r\nType 1 is a full token with no privileges
removed or groups disabled.  A full token is only used if User Account Control is
disabled or if the user is the built-in Administrator account or a service
account.\r\n\r\nType 2 is an elevated token with no privileges removed or groups
disabled.  An elevated token is used when User Account ContAccount Control is
disabled or if the user is the built-in Administrator account or a service
account.\r\n\r\nType 2 is an elevated token with no privileges removed or groups
disabled.  An elevated token is used when User Account Control is enabled and the
user chooses to start the program using Run as administrator.  An elevated token is
also used when an application is configured to always require administrative
privilege or to always require maximum privilege, and the user is a member of the
Administrators group.\r\n\r\nType 3 is a limited token with administrative
privileges removed and administrative groups disabled.  The limited token is used
when User Account Control is enabled, the application does not require
administrative privilege, and the user does not choose to start the program using
Run as administrator.","Category":"Process
Creation","Opcode":"Info","SubjectUserSid":"S-1-5-21-75874118-1364727433-
2505763305-1000","SubjectUserName":"SEB","SubjectDomainName":"WIN-
RL1FM3ULMHA","SubjectLogonId":"0x5a60e","NewProcessId":"0x9b8","NewProcessName":"C:
\\Windows\\SysWOW64\\cmd.exe","TokenElevationType":"%%1938","CommandLine":"\"C:\\Wi
ndows\\System32\\cmd.exe\" /Q /C %HOMEDRIVE%&amp;cd %HOMEPATH%&amp;(if exist
```

```
\"Desktop\\template.pdf\" (cd \"Desktop\"))&amp;amp;(if exist \"My
Documents\\template.pdf\" (cd \"My Documents\"))&amp;amp;(if exist
\"Documents\\template.pdf\" (cd \"Documents\"))&amp;amp;(if exist
\"Escritorio\\template.pdf\" (cd \"Escritorio\"))&amp;amp;(if exist \"Mis
Documentos\\template.pdf\" (cd \"Mis Documentos\"))&amp;amp;(start
template.pdf)\n\n\n\n\n\n\n\n\nTo view the encrypted content please tick the \"Do
not show this message again\" box and press
Open.","EventReceivedTime":1464627411,"SourceModuleName":"in","SourceModuleType":"i
m_msvistalog"}
```


Les quatre logs suivants sont un échantillon des 37 logs qui sont créés par *Suricata*, ils sont créés à cause des activités du tunnel http inversé de son ouverture à sa fermeture:

Tableau 4-4. Tunnel http inversé échantillon 1


```
{"timestamp":"2016-05-30T12:55:05.894917-
0400","flow_id":37154544,"in_iface":"eno16777736","event_typ
e":"http","src_ip":"192.168.1.3","src_port":49187,"dest_ip":
"192.168.1.2","dest_port":8080,"proto":"TCP","tx_id":0,"http
":{"hostname":"192.168.1.2","url":"\/h5CPgExZ1e3C9sP3lb3-
FwjsxxjY6imMc4BzU3WN0ru3asqzl20_aASzam8mjunoRmmP-uhB-ma49ZA-
suOOTzF3UWEuaR19mHPsV52qutbxXNv48GfcbnhKodHZ1kcDTyHIUrlMwx5j
brTwA7bwexJ3pRxCIY90701UFOB5USXU5H5iYSi6352TZBK","http_metho
d":"GET","protocol":"HTTP\/1.1","length":0}}
```


Tableau 4-5. Tunnel http inversé échantillon 2


```
{"timestamp":"2016-05-30T12:55:05.962234-
0400","flow_id":37154848,"in_iface":"eno16777736","event_typ
e":"http","src_ip":"192.168.1.3","src_port":49188,"dest_ip":
"192.168.1.2","dest_port":8080,"proto":"TCP","tx_id":0,"http
":{"hostname":"192.168.1.2","url":"\/h5CPgExZ1e3C9sP3lb3-
FwrLkwjmI6tqHaNq74uwVGqtUQfr6TmrMhNlxhwyUfO3zTC2R2fxdbcojbw1
X0PfDZhiX2fucidvrLNiSotUjspxdlX5n_zlXrZ7MKcd4ps\/","http_use
r_agent":"Mozilla\/4.0 (compatible; MSIE 6.1; Windows
NT)","http_content_type":"application\/octet-
stream","http_method":"POST","protocol":"HTTP\/1.1","status"
:200,"length":73}}
```

Tableau 4-6. Tunnel http inversé échantillon 3


```
{"timestamp":"2016-05-30T12:55:05.962234-
0400","flow_id":37154848,"in_iface":"eno16777736","event_typ
e":"fileinfo","src_ip":"192.168.1.3","src_port":49188,"dest_
ip":"192.168.1.2","dest_port":8080,"proto":"TCP","http":{"ho
stname":"192.168.1.2","url":"\/h5CPgExZ1e3C9sP3lb3-
FwrLkwjmI6tqHaNq74uwVGqtUQfr6TmrMhNlxhwyUfO3zTC2R2fxdbcojbw1
X0PfDZhiX2fucidvrLNiSotUjspxdlX5n_zlXrZ7MKcd4ps\/","http_use
r_agent":"Mozilla\/4.0 (compatible; MSIE 6.1; Windows
NT)","http_content_type":"application\/octet-
stream","http_method":"POST","protocol":"HTTP\/1.1","status"
:200,"length":73},"app_proto":"http","fileinfo":{"filename":
"\/h5CPgExZ1e3C9sP3lb3-
FwrLkwjmI6tqHaNq74uwVGqtUQfr6TmrMhNlxhwyUfO3zTC2R2fxdbcojbw1
X0PfDZhiX2fucidvrLNiSotUjspxdlX5n_zlXrZ7MKcd4ps\/","state":"
CLOSED","stored":false,"size":4,"tx_id":0}}
```


Tableau 4-7. Tunnel http inversé échantillon 4


```
{"timestamp":"2016-05-30T12:55:05.962844-
0400","flow_id":37154848,"in_iface":"eno16777736","event_typ
e":"fileinfo","src_ip":"192.168.1.2","src_port":8080,"dest_i
p":"192.168.1.3","dest_port":49188,"proto":"TCP","http":{"ho
stname":"192.168.1.2","url":"\/h5CPgExZ1e3C9sP3lb3-
FwrLkwjmI6tqHaNq74uwVGqtUQfr6TmrMhNlxhwyUfO3zTC2R2fxdbcojbw1
X0PfDZhiX2fucidvrLNiSotUjspxdlX5n_zlXrZ7MKcd4ps\/","http_use
r_agent":"Mozilla\/4.0 (compatible; MSIE 6.1; Windows
NT)","http_content_type":"application\/octet-
stream","http_method":"POST","protocol":"HTTP\/1.1","status"
:200,"length":73},"app_proto":"http","fileinfo":{"filename":
"\/h5CPgExZ1e3C9sP3lb3-
FwrLkwjmI6tqHaNq74uwVGqtUQfr6TmrMhNlxhwyUfO3zTC2R2fxdbcojbw1
X0PfDZhiX2fucidvrLNiSotUjspxdlX5n_zlXrZ7MKcd4ps\/","state":"
CLOSED","stored":false,"size":73,"tx_id":0}}
```


Maintenant que le tunnel http inversé est ouvert, l'attaquant est en mesure d'exécuter des commandes sur la station de travail de Bob. Il est en mesure de rechercher et télécharger le fichier confidentiel. Ces activités ne sont cependant pas détectées par les senseurs. Le scénario d'entraînement comprend au total 39 logs, 38 provenant du SDI *Suricata* et un seul provenant de la station *Windows*. Le Tableau 4-8 résume le nombre de logs produits par chacun des senseurs durant chacune des étapes du scénario.

Tableau 4-8. Logs produits et colletés par les senseurs

Étapes du scénario	Suricata	Windows 7
Téléchargement du fichier infecté par Bob.	1	0
Lecture du fichier infecté par Bob qui provoque l'ouverture du tunnel http inversé.	37	1

4.6 Test de prototype

Après avoir collecté la séquence de logs reliée à notre scénario nous avons utilisé notre prototype pour assembler la séquence de logs à des différentes séquences de logs servant d'arrière-plan. Nous avons utilisé différentes séquences d'arrière-plan pour tester différentes exceptions reliées au processus de fusion. Nous avons généré des activités d'utilisateurs manuellement sur notre environnement test dans le but de générer des séquences de logs comportant des activités normal d'un réseau. Pour être en mesure de généré des séquences de logs valables pour l'entraînement d'opérateurs SIEM, une séquence réaliste servant d'arrière-plan doit être utilisée. L'objectif étant de reproduire une séquence de log 'fusionnée' qui est pratiquement semblable à la séquence de log qui serait générée par les senseurs qui surveilleraient l'activité du réseau pendant l'exécution du scénario d'entraînement. La séquence d'arrière-plan est constituée de logs liés à des activités courantes d'un réseau sur une certaine période de temps suffisante pour insérée les séquences de logs reliés à des scénarios d'entraînement.

Pour obtenir des séquences d'arrière-plan réaliste, nous avons donc généré des activités sur la station de travail *Windows* dont certaines généraient également des activités sur le pare-feu *Suricata.* Nous avons également généré des activités à partir de la station d'attaque vers le pare-feu *Suricata* dans le but de générer des logs d'activité courante d'un réseau. Ces activités incluaient par exemple des balayages de réseau qui sont détecté par *Suricata*, qui sont détecté couramment par un pare-feu. Également, nous avons généré différentes séquences de logs d'arrière-plan comprenant des champs de neutres qui sont problématiques pour nos différents tests. Nous allons discuter des résultats obtenus pour chacune des étapes de l'assemblage. L'objectif était de rendre la séquence de logs rejouable pour les deux senseurs utilisés sur un réseau différents.

4.6.1 Processus de personnalisation

Nous devions personnaliser 39 logs au total, 1 log provenant pour la station *Windows*, et 38 logs provenant du SDI *Suricata*. Nous avons choisi pour décrire les résultats pour trois logs représentatifs et qui incluent une instance de tous les champs que nous avions à personnaliser. Le Tableau 4-9 illustre l'analyse d'un log

provenant de *Suricata*, et reliée au téléchargement du fichier infecté. Le champ *flow_id* est un champ neutre que nous devons identifier pour le gérer à l'étape de fusion. Les autres champs sont automatiques ou personnalisables, nous avions pour ces champs simplement à substituer les valeurs provenant de l'utilisateur ou du réseau d'entraînement.

Tableau 4-9. Log original provenant de *Suricata* relié au téléchargement de fichier template.pdf


```
{"timestamp":"2016-05-30T12:54:32.365020-
0400","flow_id":37154240,"in_iface":"eno16777736","event_typ
e":"http","src_ip":"192.168.1.3","src_port":49186,"dest_ip":
"192.168.1.2","dest_port":80,"proto":"TCP","tx_id":0,"http":
{"hostname":"192.168.1.2","url":"\/template.pdf","http_user_
agent":"Mozilla\/4.0 (compatible; MSIE 8.0; Windows NT 6.1;
WOW64; Trident\/4.0; SLCC2; .NET CLR 2.0.50727; .NET CLR
3.5.30729; .NET CLR 3.0.30729; Media Center PC
6.0)","http_method":"GET","protocol":"HTTP\/1.1","length":0}
}
```


1. Champs personnalisables en rouge	2. Champs automatiques en vert
3. Champs neutres en bleu.	4. Champs signatures en noir

Le Tableau 4-10 décrit l'analyse faîte pour un log provenant de *Suricata* qui est relié aux communications entre le serveur de contrôle malicieux et la station de travail. L'information entre les deux stations est passée à l'aide de fichiers qui sont encryptés. Lors de l'implémentation de notre prototype, nous devions nous assurer que ces champs demeurent inchangés et qu'ils étaient gérés comme des champs de type signature et non comme des champs de type personnalisable. Notre prototype devait donc être en mesure d'identifier ces logs reliés à l'ouverture du tunnel http inversé et gérer les champs de ces logs en conséquence.

Tableau 4-10. Log original provenant de *Suricata* relié au tunnel http inversé


```
{"timestamp":"2016-05-30T12:55:05.963720-
0400","flow_id":37154848,"in_iface":"eno16777736","event_typ
e":"fileinfo","src_ip":"192.168.1.3","src_port":49188,"dest_
ip":"192.168.1.2","dest_port":8080,"proto":"TCP","http":{"ho
stname":"192.168.1.2","url":"\/h5CPgExZ1e3C9sP3lb3-
FwrLkwjmI6tqHaNq74uwVGqtUQfr6TmrMhNlxhwyUfO3zTC2R2fxdbcojbw1
X0PfDZhiX2fucidvrLNiSotUjspxdlX5n_zlXrZ7MKcd4ps\/","http_use
r_agent":"Mozilla\/4.0 (compatible; MSIE 6.1; Windows
NT)","http_content_type":"application\/octet-
stream","http_method":"POST","protocol":"HTTP\/1.1","status"
:200,"length":0},"app_proto":"http","fileinfo":{"filename":"
\/h5CPgExZ1e3C9sP3lb3-
FwrLkwjmI6tqHaNq74uwVGqtUQfr6TmrMhNlxhwyUfO3zTC2R2fxdbcojbw1
X0PfDZhiX2fucidvrLNiSotUjspxdlX5n_zlXrZ7MKcd4ps\/","state":"
CLOSED","stored":false,"size":4,"tx_id":1}}
```


1. Champs personnalisables en rouge	2. Champs automatiques en vert
3. Champs neutres en bleu.	4. Champs signatures en noir

Le Tableau 4-11 analyse la personnalisation du log provenant de la station *Windows* détecté et reporté au SIEM quand le logiciel malicieux est ouvert sur la station de travail. La personnalisation inclut l'indentification de plusieurs champs neutres qui devront être ajustés lors de la fusion. La personnalisation du champ *message* a demandé une manipulation de chaîne de caractères un peu plus complexe qu'un champ habituel. Le champ *LogonID* et *SubjectLogonId* sont deux cas intéressants à personnaliser, il représente un numéro d'identification d'une session qui est assigné à un utilisateur lorsqu'il se connecte sur une station. Ce champ se doit être personnalisé par l'instructeur à l'aide de l'information disponible sur le réseau. L'utilisateur devra choisir un utilisateur qui est présentement connecté sur la station, ou bien l'autre option que nous avons est de laisser la valeur par défaut qui est présente dans les logs collectés.

Tableau 4-11. Log original reporté par Windows


```
{"timestamp":"2016-05-30 12:55:05","Hostname":"WIN-RL1FM3ULMHA","Keywords":-
9214364837600034816,"EventType":"AUDIT_SUCCESS","SeverityValue":2,"Severity":"INFO"
,"EventID":4688,"SourceName":"Microsoft-Windows-Security-
Auditing","ProviderGuid":"{54849625-5478-4994-A5BA-
3E3B0328C30D}","Version":1,"Task":13312,"OpcodeValue":0,"RecordNumber":19545,"Proce
ssID":4,"ThreadID":48,"Channel":"Security","Message":"A new process has been
created.\r\n\r\nSubject:\r\n\tSecurity ID:\t\tS-1-5-21-75874118-1364727433-
2505763305-1000\r\n\tAccount Name:\t\tSEB\r\n\tAccount Domain:\t\tWIN-
RL1FM3ULMHA\r\n\tLogon ID:\t\t0x5a60e\r\n\r\nProcess Information:\r\n\tNew Process
ID:\t\t0x9b8\r\n\tNew Process Name:\tC:\\Windows\\SysWOW64\\cmd.exe\r\n\tToken
Elevation Type:\tTokenElevationTypeLimited (3)\r\n\tCreator Process
ID:\t0x550\r\n\tProcess Command Line:\t\"C:\\Windows\\System32\\cmd.exe\" /Q /C
%HOMEDRIVE%&cd %HOMEPATH%&(if exist \"Desktop\\template.pdf\" (cd \"Desktop\"))&(if
exist \"My Documents\\template.pdf\" (cd \"My Documents\"))&(if exist
\"Documents\\template.pdf\" (cd \"Documents\"))&(if exist
\"Escritorio\\template.pdf\" (cd \"Escritorio\"))&(if exist \"Mis
Documentos\\template.pdf\" (cd \"Mis Documentos\"))&(start
template.pdf)\n\n\n\n\n\n\n\n\nTo view the encrypted content please tick the \"Do
not show this message again\" box and press Open.\r\n\r\nToken Elevation Type
indicates the type of token that was assigned to the new process in accordance with
User Account Control policy.\r\n\r\nType 1 is a full token with no privileges
removed or groups disabled.  A full token is only used if User Account Control is
disabled or if the user is the built-in Administrator account or a service
account.\r\n\r\nType 2 is an elevated token with no privileges removed or groups
disabled.  An elevated token is used when User Account ContControl is disabled or
if the user is the built-in Administrator account or a service account.\r\n\r\nType
2 is an elevated token with no privileges removed or groups disabled.  An elevated
token is used when User Account Control is enabled and the user chooses to start
the program using Run as administrator.  An elevated token is also used when an
application is configured to always require administrative privilege or to always
require maximum privilege, and the user is a member of the Administrators
group.\r\n\r\nType 3 is a limited token with administrative privileges removed and
administrative groups disabled.  The limited token is used when User Account
Control is enabled, the application does not require administrative privilege, and
the user does not choose to start the program using Run as
administrator.","Category":"Process Creation","Opcode":"Info","SubjectUserSid":"S-
1-5-21-75874118-1364727433-2505763305-
1000","SubjectUserName":"SEB","SubjectDomainName":"WIN-
RL1FM3ULMHA","SubjectLogonId":"0x5a60e","NewProcessId":"0x9b8","NewProcessName":"C:
\\Windows\\SysWOW64\\cmd.exe","TokenElevationType":"%%1938","CommandLine":"\"C:\\Wi
ndows\\System32\\cmd.exe\" /Q /C %HOMEDRIVE%&amp;cd %HOMEPATH%&amp;(if exist
\"Desktop\\template.pdf\" (cd \"Desktop\"))&amp;(if exist \"My
Documents\\template.pdf\" (cd \"My Documents\"))&amp;(if exist
\"Documents\\template.pdf\" (cd \"Documents\"))&amp;(if exist
\"Escritorio\\template.pdf\" (cd \"Escritorio\"))&amp;(if exist \"Mis
Documentos\\template.pdf\" (cd \"Mis Documentos\"))&amp;(start
template.pdf)\n\n\n\n\n\n\n\n\nTo view the encrypted content please tick the \"Do
not show this message again\" box and press Open."}
```


1. Champs personnalisables en rouge	2. Champs automatiques en vert
3. Champs neutres en bleu.	4. Champs signatures en noir

Le Tableau 4-12 est un sommaire de tous les champs qui ont eu besoin d'être personnalisés pour les 39 logs reliés au scénario. Nous n'avons pas assigné de valeur pour les champs neutres car ces champs sont ajustés à l'étape de la fusion.

Tableau 4-12. Sommaire des champs personnalisés

Champs	Senseur	Valeur provenant du réseau de collection	Valeur adapté au réseau d'entraînement
IP Serveur Http	*Suricata*	192.168.1.3	185.165.12.15
Port Serveur Http	*Suricata*	80	80
IP Interface Externe	*Suricata*	10.1.1.9	175.154.43.42
Port Serveur de contrôle	*Suricata*	8080	8080
Suricata Hostname	*Suricata*	192.168.1.2	185.165.12.15
In_iface Interface réseau	*Suricata*	eno16777736	eth0
Port éphémère	*Suricata*	49196	49198
Flow_id	*Suricata*	Neutre	Neutre
Url	*Suricata*	template.pdf et url multiples incluant information encodé	template.pdf + url multiples incluant information encodé
Hostname	Windows	WIN-RL1FM3ULMBA	WINDOWS-DND1.dwan.com
Account Name	Windows	SEB	Bob
Account Domain	Windows	WIN-RL1FM3ULMHA	DWANDOMAIN
SubjectUserSID	Windows	*"S-1-5-21-75874118-1364727433-2505763305-1000"*	*"S-1-5-21-75874118-1364727433-2505763305-1000"*
SubjectUserName	Windows	SEB	Bob
SubjectDomainName	Windows	WIN-RL1FM3ULMHA	DWANDOMAIN
SubjectLogonid	Windows	*0x5a60e*	*0x5a60e*
NewProcessID	Windows	Neutre	Neutre
LogonID	Windows	*0x5a60e*	*0x5a60e*
ThreadID	Windows	Neutre	Neutre
ProcessID	Windows	Neure	Neutre
RecordNumber	Windows	Neutre	Neutre

4.6.2 Discussion de l'implémentation de la personnalisation

Les résultats que nous avons obtenus pour la personnalisation montrent que le processus fonctionne et qu'il peut être implémenté pour un nombre de senseurs donné. Les résultats obtenus montrent qu'il est possible de personnaliser des logs provenant d'un senseur pour donner l'impression qu'ils proviennent d'une autre

instance d'un senseur qui peut être installée sur un réseau différent. La seule limitation que nous avons est au niveau des champs neutres dont souvent la valeur dépend de l'instance du senseur utilisé. L'implémentation de la personnalisation demande cependant un certain effort de configuration pour l'ajout de nouveaux senseurs et de nouveaux scénarios d'entraînement.

L'effort principal d'implémentation pour le processus de personnalisation est relié à la compréhension du fonctionnement des senseurs dans le but d'être en mesure de catégoriser les champs des logs qui sont produits selon leur type. Pour certains senseurs, comme par exemple une station de travail Windows, le fonctionnement de certains champs est plus difficile à comprendre car la documentation n'est pas disponible ou incomplète. Pour gérer ces cas moins fréquents, l'observation d'échantillons de logs est très souvent suffisante pour comprendre le fonctionnement du champ.

Également, l'ajout d'un scénario d'entraînement demande une analyse; puisque certains champs doivent être personnalisés selon le contexte du scénario ; par exemple le champ *url* qui correspond à une adresse internet pour *Suricata*, peut-être selon le contexte d'un scénario, soit un champ de type personnalisable, ou bien un champ de type signature qui fait partie du scénario. Dans notre scénario de validation, le champ *url* du premier log du scénario est relié au téléchargement du fichier malicieux, le champ est une véritable adresse de site web, un champ totalement personnalisable par l'utilisateur. Les champs *url* des logs subséquents sont reliés à l'installation du tunnel http inversé, ces champs font partis du scénario et sont donc de type signature. L'implémentation du prototype doit gérer ces exceptions pour chacun des différents scénarios d'entraînement.

Nous avons démontré que l'implémentation de la personnalisation pour des senseurs et des scénarios différents est possible, cependant un effort de développement et de design est nécessaire pour qu'une solution complète soit en mesure d'intégrer facilement de nouveaux senseurs et de nouvelles séquences de log reliées à des scénarios. Bien que cet effort soit hors de la portée de notre recherche, il est probable que pour solutionner ce problème, la solution optimale passerait par la construction d'un répertoire contenant l'information nécessaire pour personnaliser chacun des champs des logs reliés aux scénarios d'entraînement. Ainsi, il serait probablement plus facile de maintenir l'application et de lui ajouter des fonctionnalités.

4.7 Discussion de l'implémentation du processus d'insertion

L'implémentation du processus d'insertion a été relativement triviale. Nous n'avons pas eu de problèmes majeurs à implémenter ce processus. Il est à noter cependant qu'une implémentation complète devrait inclure une option pour permettre de changer les délais entre chacun des logs insérés reliées au scénario d'entraînement. Nous avons implémenté facilement cette fonctionnalité à notre prototype. L'expérimentation nous a démontré qu'être en mesure de changer les délais entre les logs insérés aide dans certaines situations à simplifier le processus de fusion. Dépendamment de la séquence de logs d'arrière-plan utilisée,

simplement changer la position d'un log inséré dans la séquence fusionnée peut dans certain cas simplifier la solution d'un conflit occasionné par un champ neutre et rendre la cohésion de la séquence de log assemblée meilleure.

4.8 Discussion sur l'implémentation du processus de fusion

Le Tableau 4-13 liste tous les champs neutres que nous avions à gérer lors de de l'implémentation du processus de fusion pour la séquence de logs reliée aux scénarios d'entraînement. Dépendamment de la séquence d'arrière-plan utilisée, les problèmes reliés aux champs neutres vont varier au niveau de leurs difficultés à être gérés.

Pour notre situation, le champ '*flow_id*' de *Suricata* demande toujours à être géré car des logs d'arrière-plan contenant des champs '*flow_id*' sont toujours présents. Nous avons implémenté l'algorithme pour gérer ces champs dont nous avons discuté à la section 3.7.3. L'algorithme s'est révélé efficace pour la plupart des situations car la majorité des séquences d'arrière-plan que nous avons utilisé avait des 'trous' dans la séquence des valeurs efficaces et crédibles pour le 'flow_id' était disponibles.

Tableau 4-13. Champs neutres à gérer

Champs	Senseur
Flow_id	*Suricata*
NewProcessID	Windows
ThreadID	Windows
ProcessID	Windows
RecordNumber	Windows

Nous avions besoin d'allouer trois nouvelles valeurs pour les champs '*flow_id*' pour notre séquence de 38 logs. Les valeurs pour les champs '*flow id*' qui ont été assignées et semblent crédibles et assurent la cohésion de la séquence. Nous n'avons pas testé le cas limites de *Suricata* quand 10000 flots existent et que le système entre en mode d'urgence puisque ce cas est exceptionnel et difficile à reproduire. Allouer la valeur pour un champ '*flow_id*' inséré avec l'algorithme que nous avons développé pour la plupart des situations rencontrées est efficace.

Les autres champs neutres proviennent du senseur *Windows*, nous devons gérer des situations problématiques pour tous les champs neutres seulement quand un des logs d'arrière-plan provient de la même station que nous avons utilisée dans le scénario. Généralement, sur un réseau à grande échelle, le SIEM est configuré pour restreindre la collection de logs sur les stations de travail à cause du trop grand volume de logs générés par les nombreuses stations de travail. Cependant, ces situations conflictuelles pourraient certainement avoir lieu quand la séquence d'arrière-plan utilisée est échelonnée sur une longue période de temps. Pour tester ces cas qui sont plus rares, nous avons modifié un arrière-plan pour inclure

des logs qui sont générés par la même station de travail *Windows* que nous avons utilisées dans le scénario.

Pour les champs '*ThreadID'*,'*ProcessID',*'*NewProcessID'* et '*RecordNumber'* nous avons implémenté les algorithmes décrites aux sections 3.7.1 et 3.7.2 pour solutionner les conflits occasionnés par ces champs. Dans le cas des champs '*ThreadID'*, '*ProcessID'* et le '*NewProcessID'* nous avons obtenu d'excellent résultat, les valeurs des champs alloués étant crédibles et tout à fait possible. Il est relativement facile de trouver des valeurs disponibles pour ces champs qui ne sont pas déjà utilisées pour les champs dans les logs d'arrière-plan.

Pour le champ *RecordNumber*, le résultat est acceptable mais imparfait. Pour les motifs expliqués à la section 3.7.2, il est difficile de prévoir quel sera la véritable valeur du champ *RecordNumber* sur la station de travail après certains délais, il y a simplement trop de facteurs en jeux. Cependant, nous sommes en mesure de fournir une valeur pour le champ qui semble crédible et qui est acceptable dans la majorité des situations.

L'implémentation du processus de fusion démontre la faisabilité du concept de fusion. Nous pensons que le processus est viable et qu'il est possible de l'appliquer à grande échelle sur plusieurs senseurs. Cependant, pour chacun des champs neutres que nous voudrions fusionner, un effort de développement devra être accompli pour les incorporer à une solution complète, bien qu'un tel effort soit hors de la portée de la présente recherche. Également, le processus ne peut pas être parfaitement viable comme l'a montré l'exemple du champ *RecordNumber* du senseur *Windows*. Nous pensons que ces erreurs sont acceptables dans la majorité des circonstances pour entraîner un opérateur de logiciel SIEM, ce qui ne serait pas nécessairement le cas si notre approche développait des séquences de logs pour tester des outils appelés en anglais de type *forensic* où une plus grande précision pour l'assignation des valeurs est requise. Aussi, il serait possible dans une implémentation à grande échelle d'avertir l'utilisateur qu'un champ neutre cause des conflits avec la séquence l'arrière-plan. Ainsi, l'utilisateur pourrait décider par exemple de changer un élément de son scénario d'entraînement ou bien de changer le délai d'insertion pour le log qui cause problème, ce qui permettrait d'éviter le problème de cohésion.

4.10 Activités de validation.

Nous avons utilisé un processus itératif pour supporter le développement et la validation de notre prototype. Nous avons débuté le développement de notre prototype en supportant des scénarios simples d'assemblage qui incorporaient un seul senseur et une présence restreinte de champs neutres qui sont plus problématiques à implémenter. Par exemple, nous avons débuté le développement de notre prototype en émulant la détection d'un téléchargement d'un fichier fait par un utilisateur qui a généré un log sur le senseur *Suricata*. Nous avons également développé le scénario d'une alerte de détection de virus sur une

station de travail *Windows*. Nous avons par la suite développé notre prototype pour être en mesure de supporter le scénario de validation décrit à la section 4.4.

Nous avons utilisé, pour valider notre prototype, diverses méthodes de test selon les différents processus d'assemblage. Pour tester l'implémentation du processus de personnalisation, nous avons varié les différentes constantes du code contenant les valeurs qui émulaient l'information provenant des utilisateurs et l'information extraite sur la configuration du réseau de destination. Pour s'assurer que chaque étape de développement fonctionnait, nous avons simplement vérifié manuellement les champs des logs personnalisés pour s'assurer que les valeurs étaient altérées selon le résultat attendu.

Le processus d'insertion est relativement simple à implémenter et valider, cependant le temps d'insertion du scénario d'entraînement choisit à une incidence marquée sur le processus de fusion. Nous avons donc validé le processus de fusion en utilisant différents temps d'insertion en conjonction avec différentes séquences de logs servant d'arrière-plan. Le Tableau 4-14 décrit chacune des configurations des séquences d'arrière-plan utilisées.

Tableau 4-14. Configuration des différentes séquences d'arrière-plan testées

No. de test	Logs SDI	Logs Station de travail	SDI : champs neutres	Station de travail : champs neutres
1.	Non	Non	Aucun	Aucun
2.	Oui	Non	Aucun	Aucun
3.	Non	Oui	Aucun	Aucun
4.	Oui	Oui	'Flow_id'	Aucun
5.	Oui	Oui	Aucun	Flow_id, NewprocessID, ThreadID
6.	Oui	Oui	Aucun	RecordNumber
7.	Oui	Oui	Aucun	RecordNumber,Flow_id,NewprocessID, ThreadID
8.	Oui	Oui	'Flow_id'	RecordNumber,Flow_id,NewprocessID, ThreadID

Les différents tests d'arrière-plan incluent des séquences ayant au moins 100 logs incorporant des difficultés de plus en plus complexes. Le test 1 est simple, nous insérons des logs provenant d'un senseur dans un arrière-plan contenant aucun log provenant de ce senseur, le processus de fusion n'a pas besoin d'être complété pour terminer l'assemblage. Les tests 2 et 3 amènent une certaine

complexité car une vérification doit être accomplie pour détecter si nous avons des conflits potentiels avec les champs neutres, ce qui n'est pas le cas pour ces deux tests. Dans le cas des tests 5 à 8, des conflits avec des champs neutres étaient présents. Le test 8 est le plus complexe. La séquence d'arrière-plan incorpore de nombreux logs provenant du SDI, en plus de logs de la station de travail *Windows,* qui proviennent du même journal d'événements *Windows* que les logs insérés du scénario.

Le temps d'insertion a une incidence directe sur le processus de fusion comme nous avons démontré à la section 3.7. Pour chacun des tests du tableau 4-14, nous avons donc utilisé différents temps d'insertion pour tester extensivement l'implémentation du processus de fusion. Nous avons, lorsqu' applicable, reproduit les différentes situations problématiques liées aux champs neutres décrites aux sections 3.7.1, 3.7.2, et 3.7.3.

Les nouvelles séquences de logs assemblées étaient parfaitement prêtes à être rejouées par les connecteurs de reprise du SIEM installé sur le réseau d'entraînement de destination sans devoir faire d'altérations supplémentaires.

4.11 Résultats

L'évaluation extensive de la qualité des séquences de logs produites par l'architecture que nous avons proposée représente un effort important et est hors de la portée de notre démarche de recherche. Cependant, dans le but de vérifier si le processus d'assemblage était viable, nous avons accompli plusieurs tests en rejouant différentes séquences de logs assemblées par notre prototype à l'aide du connecteur de reprise du logiciel SIEM *Splunk*. Nous avons utilisé diverses règles de détection du SIEM qui sont activées par la séquence de logs originales liées au scénario malicieux inséré, pour évaluer la nouvelle séquence assemblée. Le Tableau 4-15 décrit les règles de détection écrite à l'aide du langage de requête appelé en anglais '*Splunk Query Language*' (SPL).

Nous avons testé les séquences les plus complexes que nous avions assemblées selon les séquences d'arrière-plans décrient au tableau 4-14, c'est-à-dire celles incorporant des logs des deux senseurs avec différents temps d'insertion. L'alerte 1 est simple et détecte si un processus de commande est créé sur une station de travail. L'alerte 2 détecte si un processus de commande a été créé sur un poste de travail et qu'une hausse de privilège a été demandée pour celui-ci. L'alerte 3 est similaire, mais elle est seulement activée si le processus qui a initié le processus de la console de commande est un fichier de type pdf. L'alerte 4 détecte si un fichier contient dans son nom la chaîne de caractère 'template.pdf'. Cette alerte n'est pas réaliste en soit, mais elle est basée sur une requête que l'opérateur accomplirait pour investiguer l'incident de sécurité. L'alerte 5 est déclenchée quand un nombre important de téléchargement provenant d'une destination est fait par une source unique dans un délais très court, ce qui pourrait indiquer un tunnel http inversé ou d'autres opérations suspectes. Les alertes 1, 2, 3 et 4 se déclenchent sans problème quelle que soit la séquence de logs assemblée. L'alerte 5 devait parfois être ajustée dépendamment du délai de

temps utilisé par la règle de détection et des délais d'insertion de la séquence insérées, mais celle-ci se déclenche lorsque les conditions sont propices.

Tableau 4-15. Règles d'alertes

No.	Règle SPL	Description
1.	search NewProcessName="*cmd.exe"	Activé quand un nouveau processus 'cmd.exe' est créé.
2.	search NewProcessName="*cmd.exe*" and Message="*Elevation*"	Activé quand une hausse de privilège sur une station de travail est détectée provenant d'un processus de console de commande.
3.	search NewProcessName="*cmd.exe*" and Message="*Elevation*" and Message="*.pdf*"	Activé quand une console de commande est créé à partir d'un fichier pdf et qu'une hausse de privilèges a lieu.
4.	search http.url="*template.pdf*"	Reporter un log provenant de *SDI* qui détecte que le fichier 'template' a été téléchargé.
5.	search dest_ip="185.165.12.15" and ip_source="175.154.43.42" |sort +_time |earliest_time ='X' and lastiment='X'|stats count as Total|where Total > 10	Activé quand plusieurs téléchargements de fichier provenant du même site ont lieu en un temps limité.

Ces tests démontrent que le processus d'assemblage est à première vue viable pour des règles de détection SIEM simple. Cependant, pour être en mesure de démontré l'efficacité du processus d'assemblage, un important travail de validation devrait être accompli, ce qui est hors de la portée de notre recherche. Nous suspectons que des problèmes pourraient être causés par les champs neutres lors de certains scénarios. Les valeurs assignées pour ces champs par le processus d'assemblage pourraient potentiellement causées des problèmes avec certaines règles de détection. Cependant, une étude plus poussée devrait être effectuée pour découvrir ces problèmes et les évaluer.

4.12 Conclusion du chapitre

L'implémentation de notre prototype et sa validation en assemblant les séquences de logs incorporant un scénario représentatif d'entraînement s'est montrée concluante. Le produit de la personnalisation et de l'insertion ont démontré d'excellents résultats.

La séquence de logs produits par le processus de la fusion a démontré de bons résultats dans la plupart des situations et le produit obtenu est rejouable sur le réseau d'entraînement de destination. Certains éléments de la séquence de logs servant d'arrière-plan peuvent occasionnés des problèmes avec la correction des champs neutres. Cependant, nous pensons que ces problèmes peuvent être solutionnés ou contournés à l'aide de moyens simples, comme discuté auparavant.

Chapitre 5 : Conclusions et recommandations

5.1 Introduction du chapitre

Ce chapitre discute des conclusions de notre recherche. En premier lieu, nous allons élaborer sur les raisons pour lesquelles nous croyons que nos objectifs de recherche ont été rencontrés. Également, nous allons discuter de travaux futurs qui incluent des idées d'améliorations pour l'approche que nous avons développée pour assembler des séquences de logs. Finalement, nous allons présenter un nouveau concept d'assemblage de log qui serait intéressant d'explorer et conclure notre dissertation.

5.2 Sommaire de la recherche

La motivation de notre démarche de recherche était basée sur la nécessité de développer des méthodes d'entraînement pour les opérateurs de logiciel SIEM qui ne reposent pas sur l'utilisation d'équipes de type *« Red Team »*.L'utilisation de connecteurs de reprise permettant de rejouer des séquences de logs reliées à des scénarios d'entraînement est une méthode valable pour entraîner des opérateurs[4]. Cependant, l'assemblage de séquences de logs utilisables pour l'entraînement d'opérateurs n'est pas un problème trivial à solutionner.

Comme décrit à la section 1.3, l'objective de notre recherche était d'investiguer et de développer des techniques pour assembler des séquences de logs comportant des scénarios d'entraînement et des activités courantes servant d'arrière-plan de manière semi-automatique qui sont utiles pour l'entraînement des opérateurs de logiciel SIEM. Pour ce faire, nous avons proposé une architecture pour être en mesure d'assembler des séquences de logs pour l'entraînement de manière efficace, et nous avons restreint notre démarche de recherche à certains éléments de cette architecture.

La contribution de notre recherche comprend les éléments principaux suivants :

1. Développement de méthodes et de techniques pour adapter une séquence de logs reliées à des scénarios intéressants pour l'entraînement. L'objectif étant de rendre ces séquences de logs entreposées dans un répertoire rejouable sur un réseau d'entraînement à l'aide de moyens semi-automatiques.
2. Développement de méthodes semi-automatiques pour insérer et fusionner une séquence de logs reliée à un scénario d'entraînement, qui a été personnalisée pour le réseau d'entraînement utilisé, avec une séquence de logs reliée à des activités courantes d'un réseau qui sert d'arrière-plan.

À l'aide du développement de notre prototype, nous avons démontré qu'il était possible de stocker des séquences de log reliées à un scénario d'entraînement dans un répertoire pour les utiliser dans un processus d'assemblage dans le but

de les rejouer. Cet objectif était en support à notre démarche de recherche sans être essentiel à celui-ci.

Nous avons aussi démontré qu'il était possible de personnaliser de manière semi-automatique des séquences de logs provenant d'un senseur dans le but de pouvoir les rejouer sur un réseau d'entraînement sous certaines conditions. Nous avons appris qu'il est possible d'implémenter le processus de personnalisation quand l'information sur le réseau d'entraînement est disponible et que le fonctionnement du senseur est bien compris. Nous notons également que la séquence de logs qui est reliée au scénario d'entraînement doit provenir de senseurs utilisés sur le réseau d'entraînement pour que la personnalisation soit possible.

Nous avons confirmé que le déploiement à grande échelle d'un système de personnalisation capable de supporter plusieurs senseurs demanderait un certain effort de développement et de maintenance. L'implémentation d'une solution complète inclurait un répertoire contenant l'information essentiel pour personnaliser chacun des champs de la séquence de logs de chacun des scénarios d'entraînement.

Nous avons démontré, qu'il possible d'insérer et de fusionner une séquence de logs reliée à un scénario d'entraînement de manière semi-automatique avec un résultat suffisamment réaliste pour qu'elle soit rejouable sur un réseau d'entraînement de manière cohésive. L'état final étant que la séquence de logs puisse être utilisée pour entraîner des opérateurs de logiciel SIEM.

Cependant, pour certaines situations, la solution proposée pourrait ne pas être parfaitement fiable dû à certains conflits causés par les champs neutres de la séquence de logs fusionnée avec la séquence de logs servant d'arrière l'arrière-plan. Il faut souligner que la présence de champs neutres semble plus rare comparativement aux autres types de champs pour la plupart des senseurs ce qui contribue à la faisabilité de notre approche.

Néanmoins, il existe des méthodes pour mitiger les problèmes reliés aux champs neutres. Il est possible d'identifier les conflits quand ils ont lieu pour avertir l'utilisateur qui assemble la séquence de logs. Ainsi, nous pouvons suggérer à l'utilisateur d'altérer certains éléments du scénario d'entraînement pour éviter ces conflits, ou bien simplement de lui suggérer une valeur convenable pour le champ qui est problématique. Nous croyons qu'à l'aide de ces moyens de mitigations, il est possible de gérer les conflits qui ne sont pas très fréquents.

Nous pensons que l'architecture proposée pourrait contribuer à assembler une séquence de logs utilisables pour entraîner des opérateurs de logiciel SIEM de manière efficace. Le développement de notre prototype a démontré qu'il est possible d'assembler une séquence de logs pour un scénario représentatif à une séquence de logs servant d'arrière-plan convenable pour un nombre limité de senseurs. Les résultats obtenus pour un nombre limité de senseurs sont encourageants et démontrent qu'il serait possible d'incorporer plusieurs senseurs

dans le processus, ce qui permettrait d'assembler des séquences de logs pour des scénarios d'entraînements complexes pour entraîner des opérateurs de logiciel SIEM.

Pour être en mesure de développer une solution complète qui sera utilisable de manière pratique pour un instructeur, un effort de développement plus important que notre prototype devrait être accompli pour implémenter tous les éléments de l'architecture proposée. Nous avons démontré cependant que l'implémentation de cette architecture est tout à fait possible et que l'effort développement ne serait pas substantiel.

5.3 Travaux futurs

Durant notre démarche de recherche, nous avons réfléchi à certaines idées qui seraient intéressantes et qui pourraient permettre de pousser cette recherche plus loin. Cette section décrit brièvement ces idées qui ont pour but d'améliorer le processus d'assemblage de logs proposé et certains autres projets connexes qui mériteraient d'être explorés.

5.3.1 Idées d'amélioration du processus d'assemblage de logs

Il serait bien sûr intéressant de continuer le développement de notre prototype d'assemblage de logs. On pourrait l'intégrer dans un environnement opérationnel semblable à l'architecture que nous avons proposé à la section 3. Ainsi, il serait possible de développer notre prototype pour être en mesure de supporter plusieurs senseurs et plusieurs scénarios d'entraînement réalistes pour entraîner des opérateurs. En plus, nous pourront être en excellente position pour évaluer formellement la qualité des séquences de logs assemblées par l'architecture que nous proposons. Nous pourrions également implémenter et tester certains concepts que nous allons expliquer brièvement.

Notre recherche a démontré que, pour le processus d'insertion et particulièrement pour celui de la fusion, la séquence de logs servant d'arrière-plan influence grandement le processus d'assemblage. Une approche intéressante pour améliorer le processus d'assemblage de log serait d'implémenter de manière semi-automatique l'analyse et l'extraction d'information de la séquence de logs servant d'arrière-plan. Cette information pourrait particulièrement être utile pour élaborer les scénarios d'entraînement et optimiser le processus d'assemblage. Par exemple, en analysant la séquence d'arrière-plan, nous pourrions détecter quel utilisateur est connecté au réseau, ou bien quel site web a été visité par un utilisateur. Il serait possible d'extraire cette information et de la stocker temporairement dans une structuré de données. Nous pourrions ainsi utiliser cette information pour construire nos scénarios de manière plus efficace, en suggérant des idées pour certains éléments des scénarios à l'instructeur en utilisant des éléments déjà existants dans l'arrière-plan. L'utilisation des éléments de l'arrière-plan améliorerait le processus de fusion et fournirait potentiellement une meilleure cohésion à la séquence de logs assemblée.

Lorsque nous avons testé notre prototype en assemblant les séquences de logs reliées au tunnel http inversé de notre scénario de validation, nous avons réalisé qu'il serait peut-être possible de changer le scénario de manière limitée en ajoutant des logs. Dans le cas de notre scénario de tunnel inversé, il serait possible d'ajouter des logs pour ajouter ou altérer la série de commandes exécutées par l'utilisateur hostile sur la machine de Bob seulement en altérant la séquence de logs reliée scénario. Altérer le scénario semble possible car chacun des logs qui sont produits après l'ouverture du tunnel sont reliés aux commandes exécutées qui sont stockées dans des champs avec le résultat de l'exécution de ces mêmes commandes sur la machine de Bob. On pourrait donc forger de nouveaux logs pour émuler l'exécution d'une séquence de commandes différentes sur la machine de Bob. Cette approche pourrait être utile pour optimiser certains scénarios d'entraînement qui sont complexes, particulièrement lorsque nous voulons émuler le plus fidèlement possible le comportement d'un attaquant dans certaines situations comme par exemple lors d'un scénario de menace persistante avancée. Nous avons développé notre prototype en utilisant la norme de log JSON, ce qui a été très efficace et a facilité grandement son développement. Il serait intéressant d'investiguer les difficultés d'implémentations pour les processus d'assemblages que nous avons développés avec d'autres normes de log. L'objectif serait d'étudier les difficultés d'implémentations avec des normes ne sont pas aussi structurés que le format JSON, par exemple la norme CEF, pour en apprendre les difficultés; la conversion de logs d'une norme à une autre est fréquente lors de l'utilisation d'un SIEM.

5.3.2 Implémentation de l'assemblage en temps quasi-réel

L'implémentation de notre processus d'assemblage fonctionne seulement en utilisant une séquence d'arrière-plan qui été enregistrée et stockée généralement par un SIEM. Il serait intéressant d'être en mesure d'assembler des scénarios en temps quasi réel pour entrainer des opérateurs de logiciel SIEM à l'aide de trafic provenant d'un réseau opérationnel. Il est possible de dupliquer tout le trafic parvenant à une instance SIEM vers une autre instance qui serait installée sur un réseau d'entraînement. Ainsi, avec un certain délai, nous pourrions utiliser la séquence d'arrière-plan dupliquée sur le réseau d'entraînement, pour l'assembler à notre séquence de logs impliquant un scénario d'entraînement. Cette architecture comportant l'utilisation de trafic temps quasi réel serait en mesure de fournir un entraînement très réaliste aux opérateurs.

5.4 Conclusion

Développer des méthodes d'entraînement pour le personnel qui défend ses réseaux opérationnels est primordial pour des organisations militaires comme les FC. Le développement de programme d'entraînement de cyberdéfense est maintenant vital pour les organisations militaires. Il est particulièrement important pour ces organisations d'être en mesure de livrer de l'entraînement de cyberdéfense régulièrement et à grande échelle. Notre démarche de recherche à contribuer à développer des moyens pour atteindre ces buts. Notre objectif de recherche était d'investiguer des méthodes et des techniques nécessaires pour

assembler des séquences de logs, contenant des scénarios d'entraînement et des séquences d'activités courantes servant d'arrière-plan, qui sont utiles pour l'entraînement des opérateurs de logiciel SIEM. Nous avons proposé une architecture qui est en mesure de générer de manière semi-automatique des séquences de logs utilisables pour l'entraînement d'opérateur SIEM. Cette architecture comprend trois éléments :

1. Répertoire contenant les séquences de logs qui sont reliées à des scénarios d'entraînement pour les opérateurs de SIEM. Nous avons démontré qu'il est possible d'utiliser un répertoire pour stocker les séquences de logs intéressantes pour l'entraînement et les utiliser pour construire des séquences de logs utilisables par des connecteurs de reprise.
2. Développement de méthodes et de techniques pour adapter une séquence de logs reliée à des scénarios intéressants pour l'entraînement. Nous avons démontré qu'il possible de rendre ces séquences de logs entreposées dans un répertoire rejouable sur un *Cyber Range* à l'aide de moyens semi-automatiques.
3. Développement de méthodes semi-automatiques pour insérer et fusionner une séquence de logs reliée à un scénario d'entraînement. Nous avons démontré qu'il est possible de fusionner des séquences de logs qui ont été personnalisées pour le réseau d'entraînement utilisé, avec une séquence de logs reliée à des activités courantes d'un réseau qui sert d'arrière-plan de manière semi-automatique.

Nous avons atteint nos objectifs de recherche en démontrant qu'il est possible d'assembler des séquences de logs utilisables pour l'entraînement d'opérateurs de logiciel SIEM de manière semi-automatique. L'architecture d'assemblage de logs que nous proposons est utilisable pour entraîner des opérateurs de logiciels SIEM à l'aide d'un réseau 'Cyber Range' et nous considérons notre démarche de recherche comme un succès.

RÉFÉRENCES

[1] S. Bhatt, P. K. Manadhata, and L. Zomlot, “The operational role of security information and event management systems,” *Security & Privacy, IEEE*, vol. 12, no. 5, pp. 35–41, 2014.

[2] J. M. Neff, “Verification and validation of the Malicious Activity Simulation Tool (MAST) for network administrator training and evaluation,”M.A thesis, NAVAL POSTGRADUATE SCHOOL MONTEREY CA, 2012.

[3] W. R. Taff Jr and P. M. Salevski, “Malware mimics for network security assessment,”M.A. thesis, NAVAL POSTGRADUATE SCHOOL MONTEREY CA, 2011.

[4] W. K. Wong, “Employing replay connectors for SIEM operator education,”M.A. thesis, NAVAL POSTGRADUATE SCHOOL MONTEREY CA, 2013.

[5] H.-P. D. Company, “Concepts for ArcSight ESM 6.8c.” Dec-2015 [Online]. Available: http://www8.hp.com/us/en/software-solutions/siem-security-information-event-management/index.html,2014

[6] A. Madani, S. Rezayi, and H. Gharaee, “Log management comprehensive architecture in Security Operation Center (SOC),” in *Computational Aspects of Social Networks (CASoN), 2011 International Conference on*, 2011, pp. 284–289.

[7] J. Kühnel, “Centralized and structured log file analysis with Open Source and Free Software tools,” Frankfurt University of Applied Sciences, 2013.

[8] R. Gerhards, “Log Normalization Systems and CEE Profiles.” 2015 [Online]. Available: http://www.gerhards.net/download/LogNormalizationV2.pdf,2011

[9] HP, “ArcSight Connectors - Supported Products.” 2015 [Online]. Available: http://www.hpenterprisesecurity.com/collateral/HP-ArcSight-SmartConnectors-Supported-Products-Jan-2013.pdf,2013

[10] R. Longoria Jr, “Scalability Assessments for the Malicious Activity Simulation Tool (MAST),” M.A thesis,NAVAL POSTGRADUATE SCHOOL MONTEREY CA, 2012.

[11] C. Sharp, “The Splunk Event Generator.” 2016 [Online]. Available: https://splunkbase.splunk.com/app/1924/#/details,2012

[12] L. P. 2015 Hewlett-Packard Development Company, *ESM 101 Concepts for ArcSight ESM 6.8c*, 6.8c ed. 2015 Hewlett-Packard Development Company, L.P.

[13] F. Skopik, G. Settanni, R. Fiedler, and I. Friedberg, “Semi-synthetic data set generation for security software evaluation,” in *Privacy, Security and Trust (PST), 2014 Twelfth Annual International Conference on*, 2014, pp. 156–163.

[14] S. O’Shaughnessy and G. Gray, “Development and evaluation of a dataset generator tool for generating synthetic log files containing computer attack signatures,” *Pervasive and Ubiquitous Technology Innovations for Ambient Intelligence Environments*, p. 116, 2012.

[15] K. Wallnau, B. Lindauer, M. Theis, R. Durst, T. Champion, E. Renouf, and C. Petersen, “Simulating malicious insiders in real host-monitored user data,” in *7th Workshop on Cyber Security Experimentation and Test (CSET 14)*, 2014.
[16] O. I. S. Foundation, “Suricata User Guide.” Open Information Security Foundation 2016 [Online]. Available: https://redmine.openinfosecfoundation.org/projects/suricata/wiki/Suricatayaml,2016
[17] O. I. S. Foundation, “Feature1696 improve logged flow_id.” Open Information Security Foundation, 2016 [Online]. Available: https://redmine.openinfosecfoundation.org/issues/1696,2016
[18] *https://www.python.org*. [Online]. Available: https://www.python.org
[19] Pychain., “PyCharm Community.” 2016 [Online]. Available: https://www.jetbrains.com/pycharm/download/#section=windows,2016
[20] S. Enterprise, “Splunk Documentation.” 2016 [Online]. Available: http://docs.splunk.com/Documentation/Splunk/6.3.3/SearchReference/Spath,2016
[21] E. International, *ECMA-404 The JSON Data Interchange Standard.* http://www.json.org: ECMA International, 2016 [Online]. Available: http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-404.pdf,2013
[22] E. M. Hutchins, M. J. Cloppert, and R. M. Amin, “Intelligence-driven computer network defense informed by analysis of adversary campaigns and intrusion kill chains,” *Leading Issues in Information Warfare & Security Research*, vol. 1, p. 80, 2011.
[23] Rapid7, “Metaploit.” 2016 [Online]. Available: https://www.metasploit.com, 2016
[24] D. Stevens, “Detecting Network Traffic from Metasploit’s Meterpreter Reverse HTTP Module.” May-2015 [Online]. Available: https://blog.didierstevens.com/2015/05/11/detecting-network-traffic-from-metasploits-meterpreter-reverse-http-module/,2015
[25] M. Archeology, “The Windows Logging Cheat Sheet.” Jan-2016 [Online]. Available: http://www.malwarearchaeology.com/cheat-sheets/,2016
[26] NSA, “Spotting the Adversary with Windows Event Log Monitoring.” Dec-2016 [Online]. Available: https://www.iad.gov/iad/library/reports/spotting-the-adversary-with-windows-event-log-monitoring.cfm,2013