Laboratoire d'analyse forensique

Alexandre Compastié

Le laboratoire d'analyse forensique

Alexandre Compastié
Inria,
Centre d'opérations de sécurité (SOC)
2 Rue Simone IFF
75012 Paris

Résumé

L'analyse forensique consiste à rechercher des preuves sur des supports numériques pour comprendre un incident informatique passé ou en cours, de manière à y remédier et pouvoir prendre des décisions éclairées.

Ces preuves sont des traces, des artefacts numériques qui fournissent des informations sur l'incident. Une fois agrégés, ils permettent de dégager un scénario factuel d'évènements et d'apporter des réponses à l'équipe en charge des investigations.

Au travers de cet article, nous exposerons l'architecture technique ainsi que les composants sur lesquels repose le laboratoire forensique du centre d'opérations de sécurité d'Inria :

– les moyens de collecte d'artefacts, qu'ils soient en mémoire, dans des traces de flux réseau, ou présents sur disque ;

– les capacités de transfert et de stockage des artefacts précédemment récoltés ;

– le traitement des artefacts et leur restitution dans un format facilitant les investigations.

Ce dernier point comprend notamment l'analyse syntaxique des journaux, l'exploitation de règles de détection et l'intégration de l'information dans l'outil de recherche, en automatisant au mieux toute cette procédure.

Enfin, nous ferons un retour d'expérience sur l'utilisation de cette solution par le SOC d'Inria lors de la résolution d'incidents de sécurité, en présentant les avantages en matière de capacité de traitement et les limites de cet outil.

Mots-clefs

Forensique, analyse, artefacts, collecte, traitement, stockage, compromission

1 Introduction

L'adaptation continue de notre niveau de maturité en matière de sécurité des systèmes d'information (SSI) est essentielle pour faire face aux menaces auxquelles sont confrontés nos systèmes d'information. Cela implique la mobilisation de multiples acteurs, notamment les équipes opérationnelles de sécurité. Ainsi, pour assurer l'évolution de sa posture défensive, la compréhension de l'origine d'un incident, qui repose sur la capacité d'investigation de ces équipes,

est primordiale. Malheureusement, mener une levée de doute ou réaliser une analyse post-mortem, aussi appelée « analyse forensique » présente un coût humain important qui peut limiter la capacité à y recourir ; d'autant plus que l'interconnexion toujours croissante de nos infrastructures impose d'approfondir son enquête tout en élargissant ses moyens d'intervention.

Afin de prolonger ses capacités d'analyse tout en préservant les ressources du service, l'emploi d'un outillage spécifique répondant au besoin de l'analyste et s'adaptant aux contraintes opérationnelles s'avère être une solution efficace dont nous présenterons les caractéristiques.

2 Un contexte d'investigation contraint

Le SOC d'Inria opère avec trois contraintes majeures qui ont influé sur les besoins de l'équipe.

Premièrement, le périmètre d'intervention est très large. En effet, on recense plus de 7000 postes clients et environ 850 machines virtualisées au sein du centre de données. Ces machines peuvent fournir des services socles de la direction des systèmes d'information (DSI), tels que le service messagerie électronique, de visioconférence, ou encore le service d'authentification d'Inria. À cela, il faut ajouter les machines de chacun des centres d'Inria servant aux différentes équipes dans le cadre de leurs projets de recherche.

Deuxièmement, le contexte technique est très riche, et cela s'illustre par la prise en charge de trois systèmes d'exploitation (*Windows*, *macOS* et *Linux*) par la DSI. L'équipe de cybersécurité doit donc être en mesure de mener ses investigations, quel que soit le système employé, impactant naturellement le choix des outils concernant la collecte d'artefacts forensiques.

Troisièmement, Inria étant réparti en neuf centres sur le territoire national et ses agents pouvant télétravailler, cela ajoute des difficultés supplémentaires à la collecte des données au cours des investigations. Par ailleurs, les membres du SOC étant eux-mêmes répartis sur différents centres, il est important que l'outil propose des fonctions collaboratives adaptées.

3 Présentation de la solution technique

3.1 Architecture simplifiée

Le laboratoire forensique est composé de deux machines virtuelles. La première est utilisée pour le dépôt d'*artefacts*. Elle peut être contactée par n'importe quelle machine connectée au réseau interne d'Inria ou par des machines au travers du VPN d'Inria afin d'y déposer, depuis la machine analysée ou une machine intermédiaire, les artefacts collectés. La seconde machine est le centre d'analyses d'artefacts ; elle soumet les artefacts à une chaîne de traitements et aboutit à l'affichage d'une chronologie complète.

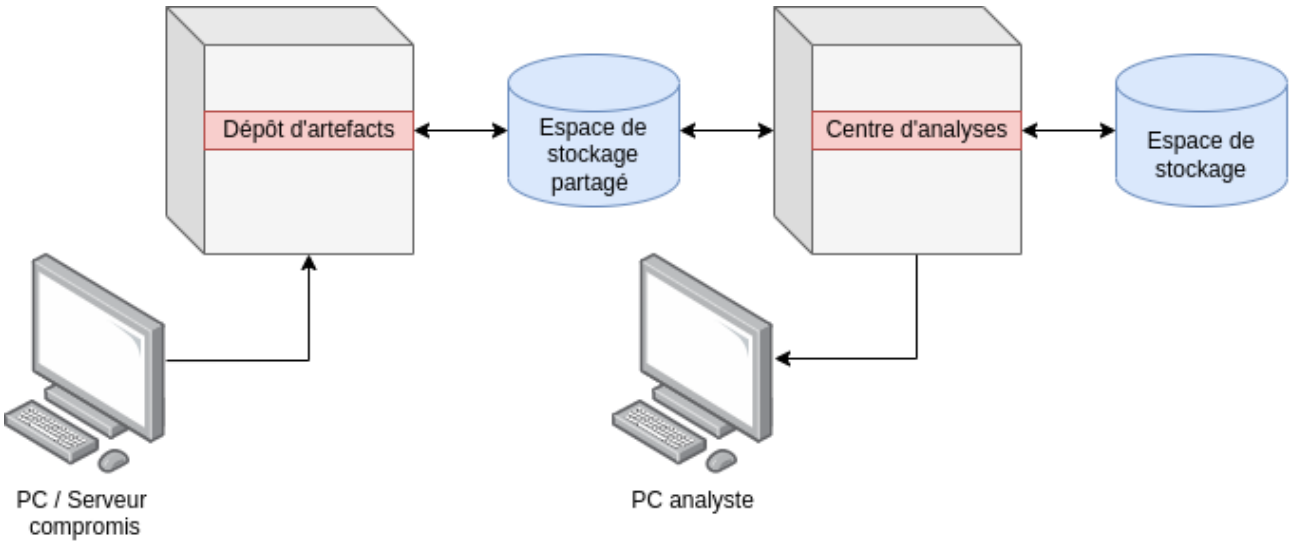

Figure 1 – Schéma d'architecture simplifié du laboratoire forensique

Ces deux machines disposent de deux espaces de stockages séparés : l'un est partagé entre les machines tandis que l'autre est dédié au centre d'analyses. En effet, les données du centre d'analyses étant enrichies par les analystes au travers de leur investigation, elles nécessitent de fait un niveau de sensibilité supérieur.

3.2 Architecture logicielle

Le laboratoire forensique est découpé en trois groupes de composants logiciels open source distincts :

- les outils dédiés à la collecte d'artefacts,
- les outils pour leur traitement,
- les outils de restitution et d'analyse.

a. Phase de collecte

Le premier groupe contient les outils à déployer sur la station de travail ou le serveur pour lequel une levée de doute ou une analyse forensique est requise. Leurs fonctions sont d'extraire les traces laissées par l'attaquant. Nous récupérons jusqu'à quatre types d'artefacts différents selon le contexte d'exploitation. Les journaux systèmes et applicatifs, qui constituent un premier type d'artefacts, sont systématiquement extraits. Pour automatiser cette phase, nous utilisons l'outil *FastIR Artifacts[1]* de *SekoiaLab* qui propose plusieurs fonctionnalités s'intégrant au mieux à notre contexte. Premièrement, il s'agit d'un exécutable nécessitant peu de configuration pour fonctionner, ce qui est idéal pour un déploiement rapide. En effet, la liste des artefacts collectés est régulièrement mise à jour et repose sur la connaissance de la communauté au moyen du *Digital Forensics Artifacts Repository*[2]. Deuxièmement, l'outil s'exécute sur les trois systèmes d'exploitation pris en charge par l'institut, limitant donc le recours à des outils et procédures spécifiques selon le système étudié.

Une capture des flux réseau peut constituer un autre type d'artefact. Elle s'effectue au moyen de

l'outil *tcpdump*. Cette capture réseau peut ensuite être analysée sur le poste d'investigation au moyen de l'outil d'analyse de flux *Wireshark*. Les règles de détection proposées par un IDS tel que *Suricata* peuvent aussi servir dans le cadre d'une levée de doute.

Plus exceptionnellement, une capture mémoire avec *LiME[3]* peut être effectuée et rejoindre notre liste d'artefacts. Le traitement de celle-ci demande un investissement important et n'est pas automatisé. Notons qu'il s'agit d'une action intrusive sur le système puisqu'elle consiste à ajouter un module au noyau au système puis à accéder au contenu de la mémoire. Cette collecte est donc motivée par la réalisation d'une analyse forensique et non d'une levée de doute.

Enfin, en cas de compromission avérée, il est possible de soumettre au laboratoire un dernier type d'artefact : une capture de disque obtenue par l'intermédiaire de l'outil *dd* ou par les outils de *VMware* s'il s'agit d'un serveur virtualisé. L'étape de collecte est plus longue, ainsi que le temps de traitement. Cependant, c'est l'option à privilégier pour disposer d'une flexibilité accrue au cours de l'investigation.

Tous les artefacts sont exportés vers le dépôt forensique qui expose un serveur *transfer.sh[4]* offrant une solution clé en main de téléversement de fichier par *curl*. Ce dernier fait partie des paquetages par défaut, il est donc régulièrement installé sur les machines étudiées. Le transfert est sécurisé par HTTPS afin de limiter le risque d'exposition ou d'altération des éléments. Une fois l'échange terminé, le traitement automatique des artefacts s'initie.

b. Phase de traitement

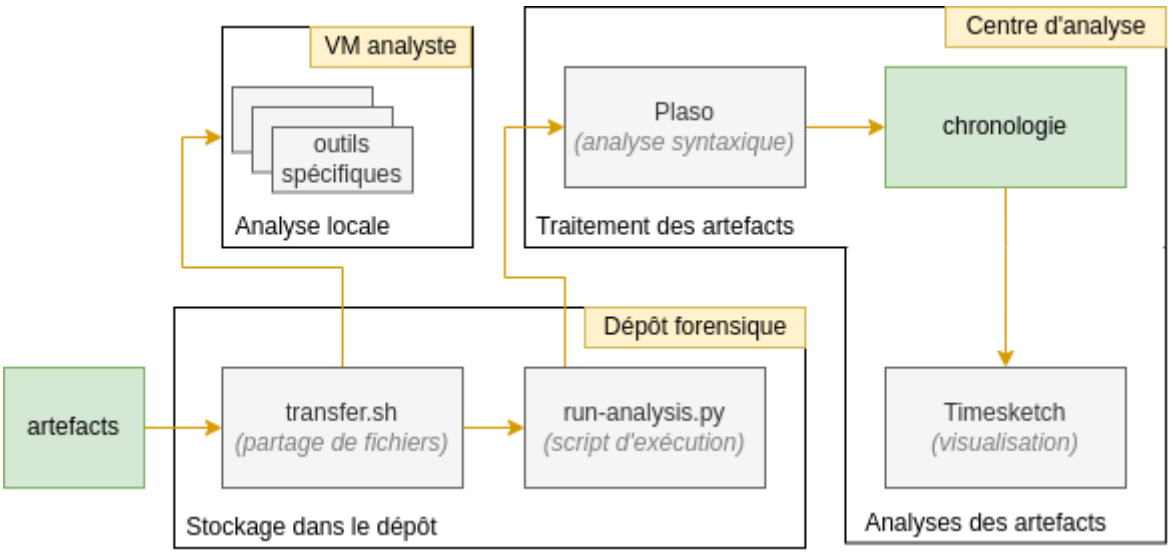

Figure 2 – Processus général de la phase de traitement

Le traitement des artefacts repose sur un outil, lui aussi open source, appelé *Plaso[5]*. Sa fonction est d'extraire les évènements de sécurité provenant de sources variées ; il peut s'agir d'un simple journal d'évènements, de répertoires entiers jusqu'à la copie d'un support de stockage d'une machine. Cela aboutit à la production d'une chronologie des différents évènements intervenus sur le système. Il s'agit d'un processus pouvant être long et dépendant des caractéristiques techniques du centre d'analyse et des données collectées. En effet, le traitement de l'outil est fortement parallélisé

et la vitesse d'exécution est directement impactée par le nombre de CPU et le type de stockage mis à disposition. Facteur évident, la taille des journaux ou la taille du support traité jouent un rôle essentiel.

À la réception des artefacts sur le dépôt forensique, un script python développé en interne contacte le démon *Docker* du centre d'analyse. La communication est initiée une fois la reconnaissance mutuelle des certificats TLS réalisée. Un conteneur est créé à partir d'une image personnalisée de *log2timeline/plaso* sur le centre d'analyse et entame le traitement des artefacts présents sur le stockage partagé. Le temps de traitement est variable ; il oscille entre la demi-heure dans le cas de journaux de quelques gigaoctets et dépasse la journée dans le cas de supports de stockage importants. Certaines options peuvent être ajoutées, comme l'emploi de règles de détection *Yara[6]* ou encore le traitement des archives, allongeant encore un peu plus la durée cette étape.

En pratique, ce temps de traitement n'est pas une perte sèche pour l'analyste, qui saura le mettre à profit en étudiant en parallèle d'autres preuves nécessitant un travail manuel, comme la copie de mémoire. De cette façon, les premières conclusions serviront de cadre à l'investigation de la chronologie produite.

En fin de processus, un fichier est généré puis transféré à *Timesketch[7]* pour ingestion. Il s'agit de l'outil avec lequel l'analyste va interagir pour explorer la chronologie.

c. Phase d'analyse

L'une des forces de l'outil est de permettre l'étude de plusieurs chronologies en parallèle. En effet, il est fréquent que l'investigation porte sur un ensemble de machines. *Timesketch* offre une vision complète des interactions entre les différentes machines en facilitant la corrélation d'évènements entre eux. Par exemple, il est possible d'interroger les journaux de *NetFlow*, des différents composants d'infrastructure (*Active Directory*, DNS...) et de l'ensemble des machines présentes sur le VLAN de la machine compromise depuis une unique interface. Suivre l'avancée de l'attaquant au sein du système d'information est donc grandement facilitée.

En sous-main, *Timesketch* indexe les évènements collectés au sein d'une base de données *Elasticsearch*. Cela permet d'interroger les éléments récupérés au moyen du *Query string query*, ne nécessitant que peu de temps d'adaptation à l'analyste pour en exploiter son plein potentiel. En cas de besoin, l'*Elasticsearch query language* peut être utilisé pour les requêtes plus complexes. De plus, l'interrogation des évènements s'avère très rapide, surtout en comparaison de méthodes plus traditionnelles. Cela qui permet à l'analyste d'exécuter davantage de requêtes dans un même temps, en se concentrant donc sur son métier. De plus, les requêtes peuvent être préenregistrées et servir pour de futures analyses en limitant le recours aux expressions régulières dont le maintien peut s'avérer hasardeux. L'analyste peut donc faire l'économie de requêtes redondantes, par exemple des requêtes de bots lors de l'inspection de journaux de serveur web. Avec ces différentes caractéristiques, on obtient un moteur de recherche redoutablement efficace et pratique à utiliser.

Timesketch embarque un ensemble de plug-ins facilitant l'analyse des évènements. Ces plug-ins ajoutent des étiquettes aux évènements et peuvent, dans certains cas, générer des rapports synthétiques. Certains plug-ins mettront en valeur des journaux particuliers tels que les activités de

connexion et déconnexion à une session, les tentatives d'accès par force brute, les accès SSH, le plantage d'application ou l'absence de journalisation. Mais, celui qui attirera l'attention de l'analyste est le plug-in *Sigma[8]*. Ce dernier permet l'analyse de la chronologie d'évènements au moyen de règles de détections. En d'autres termes, et pour reprendre ceux de ses concepteurs « *Sigma est aux journaux ce que sont Snort au trafic réseau et Yara aux fichiers* ». De cette façon, ce sont plus de quatre cents règles de détection qui sont intégrées à l'outil, et que l'on ne manquera pas de mettre à jour régulièrement et d'enrichir en fin d'analyse.

Enfin, l'outil propose des fonctions favorisant le travail collaboratif. En effet, le partage de l'information pendant l'analyse permet à l'équipe d'organiser ses recherches plus efficacement. Pour cela, il propose un système de commentaire à ajouter sur les différents évènements étudiés. Il est aussi possible de mettre en emphase certains évènements et d'en extraire des indicateurs de compromission, qu'il s'agisse d'adresses IP ou d'empreintes. Enfin, le système de *stories* permet de fournir une vue synthétique de l'analyse et de son déroulé.

4 Le laboratoire forensique au cœur de l'investigation

4.1 Bénéfice de l'utilisation du laboratoire forensique

Par son architecture, le laboratoire forensique répond au besoin de l'analyste, que ce soit en matière de polyvalence, de modularité ou de performance. En effet, le laboratoire permet de traiter des incidents de tailles variables. L'analyste pourra l'utiliser en phase de levée de doute, en exploitant les différents outils de collectes portables sur le poste compromis, mais aussi dans le cas d'un incident majeur impliquant un nombre important d'actifs interconnectés en utilisant la fonctionnalité de chronologie offerte par l'outil. Cette caractéristique permet d'étendre la portée d'intervention des équipes de sécurité, puisque les procédures et les outils d'intervention sont homogènes, quelle que soit la situation appréhendée.

Les outils de traitement et d'analyse peuvent être facilement adaptés au cadre d'emploi. En effet, en intégrant des moteurs de détection et en proposant des schémas de données d'artefacts éditables, l'outil s'avère très modulaire et peut être modifié pendant le traitement de l'incident pour prendre en charge des situations imprévisibles. En fin de processus, il est possible d'enrichir le moteur de détection avec ses propres règles, et ainsi capitaliser les connaissances tout juste acquises au cours de l'analyse.

Enfin, l'indexation des données, l'automatisation d'actions redondantes, le travail collaboratif et le dimensionnement adapté de la plateforme permettent d'assurer un bon niveau de performance, ce qui contribue à limiter les dommages causés par l'attaquant (p. ex. latéralisation de l'attaque) tout en accélérant la résolution de l'incident.

4.2 Intégration à la phase de réponse aux incidents en pratique

Le laboratoire d'analyse forensique a été utilisé très récemment par le SOC d'Inria dans le cadre du traitement de la vulnérabilité *log4shell* (CVE-2021-44228). Après avoir identifié une première liste

d’actifs vulnérables de notre système d’information, nous avons mené une campagne de levée de doute sur certains systèmes dont le comportement était jugé anormal. C’est notamment le cas d’une instance de test employant une solution de *mobile device management*, pour laquelle des flux caractéristiques d’un mineur de cryptomonnaie ont été détectés par Renater.

La levée de doute sur cette machine a été faite en s’appuyant sur l’outil de collecte *FastIR*. Le SOC a extrait les journaux systèmes et applicatifs de la machine et les a transmis au laboratoire forensique. L’investigation précédemment réalisée en direct sur la machine n’avait pas permis d’identifier la présence d’un tiers malveillant. En effet, aucune connexion anormale n’avait été identifiée au moyen de *lsof*, de *netstat* ou de *tcpdump*. Il en était de même pour l’IP signalée par l’opérateur. La charge de la machine était nominale et aucun processus suspect n’était en exécution. Cependant, la solution hébergée utilisait *log4j* en version 2.13 et était donc vulnérable.

L’analyse prend alors une autre perspective lorsque le module *Sigma* de *Timesketch* rapporte un dépassement de la capacité mémoire ayant abouti à l’arrêt de *Java* et de *MySQL*. Nous prenons la décision d’isoler la machine, jugeant le risque de compromission élevé. Le support de stockage virtualisé est extrait puis soumis au traitement du laboratoire forensique avant que la machine ne soit arrêtée.

La chronologie a finalement pu être générée par *Timesketch* puis complétée par les données de *NetFlow* ainsi que les journaux applicatifs de la machine, dont le format non standard n’avait pas permis le traitement par *Plaso*. Nous disposions alors d’une visibilité complète sur les actions réalisées sur la machine, que ce soit au niveau matériel, noyau, applicatif ou réseau. C’était un peu plus de 2,3 millions d’évènements collectés.

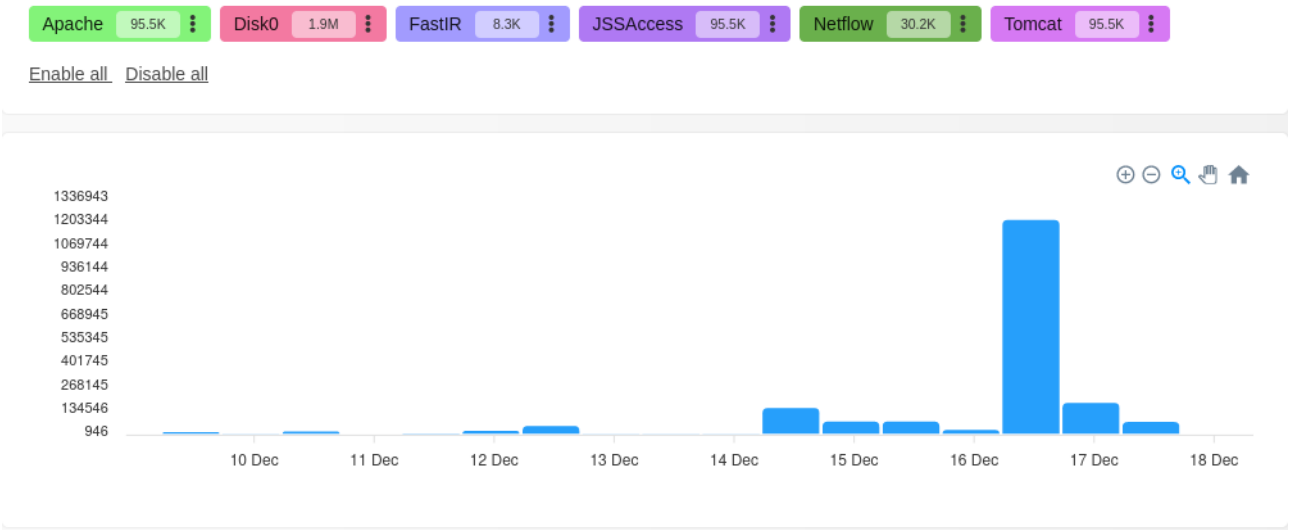

Figure 3 – V*olumétrie des évènements selon le temps visualisé avec Timesketch*

Nous nous sommes ensuite appuyés sur les règles de détection de *Yara* qui ont relevé la présence de deux exécutables. Ceux-ci avaient été déposés puis accédés dans */tmp* à 3 h 41 du matin et semblaient compressé avec UPX. Nous avons employé l’outil *image_export.py* de *Plaso* pour extraire le fichier suspect et l’avons soumis à *VirusTotal* qui en a confirmé le caractère malveillant. L’empreinte du second fichier était celle du binaire *nohup*, dont le nom a été modifié pour ne pas

éveiller de soupçon. Pourtant, son exécution n'était pas discrète : nous avons constaté que le mineur de cryptomonnaie a chargé un ensemble de librairies cryptographique de *python*, puis a contraint le système à libérer de la mémoire au moyen d'*oom-kill*, arrêtant par la même occasion le serveur *Tomcat* et le programme malveillant.

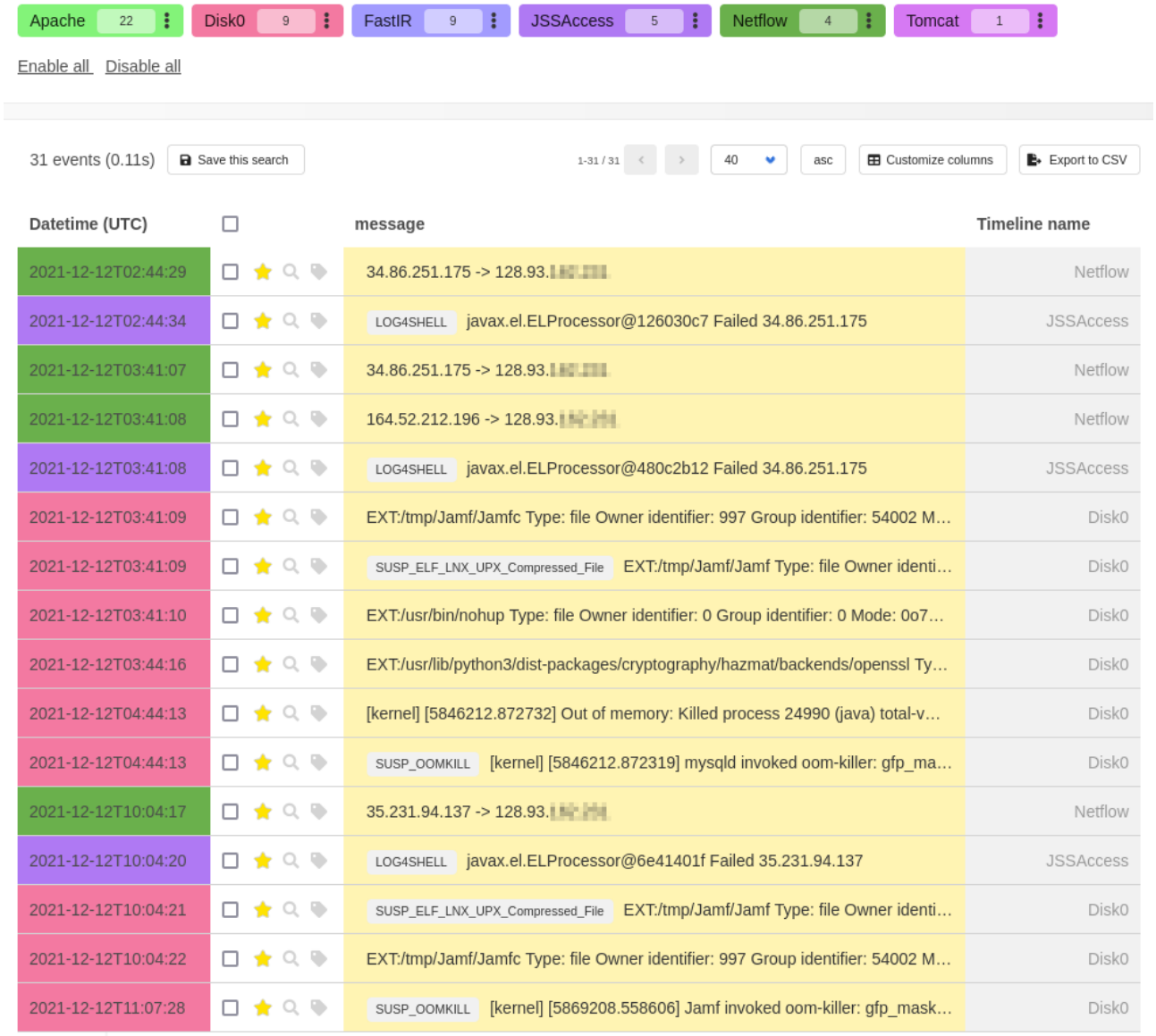

Figure 4 – *Chronologie des actions ayant mené à la compromission*

Enfin, nous avons découvert le moyen utilisé par l'attaquant pour s'introduire dans le système en remontant la chronologie. C'est ainsi que nous avons constaté de multiples échanges entre le serveur et une adresse IP à l'exact moment du dépôt du fichier malveillant. L'analyse des requêtes d'authentification au service hébergé nous a indiqué que la vulnérabilité *log4shell* avait été exploitée en détournant la classe *javax.el.ELProcessor*. C'est une méthode qui permet d'esquiver les contre-mesures avec certaines versions de Java.

4.3 Limites de la plateforme

Bien que le laboratoire forensique s'avère un outil très efficace dans son contexte opérationnel, il présente plusieurs inconvénients lors de son utilisation. Tout d'abord, *Plaso* ne peut traiter les artefacts dont il ne connaît pas la syntaxe. Ainsi, les journaux non standards peuvent être plus difficiles à traiter. Effet corollaire, l'automatisation du traitement est loin d'être parfaite et peut nécessiter des actions manuelles. Notons par exemple la récupération des artefacts depuis le dépôt de stockage afin d'adapter leur format avant ingestion dans *Timesketch*.

Aussi les outils de collecte ne sont pas les plus discrets, surtout en les comparant à d'autres outils concurrents. Certains problèmes sont liés à la jeunesse des outils, comme la présence de quelques bogues éparses peut être constatée.

On regrettera de ne pas disposer d'une interconnexion avec un service de bac à sable local, permettant d'assurer la confidentialité des binaires collectés et l'étude comportementale de ceux-ci.

Une perspective d'évolution à étudier est la surveillance du parc informatique au moyen d'agents présents sur les postes, accroissant davantage la capacité d'intervention des équipes de cybersécurité.

5 Conclusion

Pour conclure, le laboratoire forensique propose un ensemble d'outils permettant la collecte, le traitement et l'analyse d'artefacts. Il s'agit d'outils libres, adaptés à un parc hétérogène et aux contraintes du travail en équipe et en distanciel. Leur utilisation permet d'accroître les capacités d'intervention des équipes de cybersécurité, en facilitant la phase de levée de doute. Elle permet aussi d'approfondir les investigations en favorisant au moyen de chronologies l'étude de systèmes interconnectés complexes. Les différents modules du centre d'analyses permettent de facilement réinvestir les connaissances acquises au cours de l'analyse tout en accélérant la vitesse de traitement. En pratique, l'outil s'est avéré très efficace lors des multiples investigations menées par le SOC d'Inria. Cependant, l'automatisation du traitement reste un défi compte tenu des incertitudes liées au contexte d'investigation et à l'évolution rapide des outils disponibles.

Lexique

Termes	Définitions
artefact	Information générée ou modifiée par l'activité d'un tiers malveillant
DSI	Direction des systèmes d'information
IDS	Intrusion Detection System
SOC	Security Operations Center
SSI	Sécurité des Systèmes d'Information

Bibliographie

[1] SekoiaLab. 2019. FastIR Artifacts. San Francisco (CA) : GitHub ; [accessed 2022 February 14]. https://github.com/SekoiaLab/fastir_artifacts.

[2] ForensicArtifacts. 2014. Digital Forensics Artifacts Repository. San Francisco (CA) : GitHub ; [accessed 2022 February 14]. https://github.com/ForensicArtifacts/artifacts.

[3] 504ensicsLabs. 2014. LiME ~ Linux Memory Extractor. San Francisco (CA) : GitHub ; [accessed 2022 February 14]. https://github.com/504ensicsLabs/LiME.

[4] dutchcoders. 2014. transfer.sh. San Francisco (CA) : GitHub ; [accessed 2022 February 14]. https://github.com/dutchcoders/transfer.sh.

[5] log2timeline. 2014. Plaso (Plaso Langar Að Safna Öllu) - super timeline all the things. San Francisco (CA) : GitHub ; [accessed 2022 February 14]. https://github.com/log2timeline/plaso.

[6] VirusTotal. 2012. YARA in a nutshell. San Francisco (CA) : GitHub ; [accessed 2022 February 14]. https://github.com/VirusTotal/yara.

[7] Google. 2014. Timesketch. San Francisco (CA) : GitHub ; [accessed 2022 February 14]. https://github.com/google/timesketch.

[8] SigmaHQ. 2016. Timesketch. San Francisco (CA) : GitHub ; [accessed 2022 February 14]. https://github.com/SigmaHQ/sigma.