ADMINISTRATION SYSTÈME & RÉSEAU

GNU LINUX PRATIQUE

HORS-SÉRIE N°49

NOV. DÉC. 2020

FRANCE MÉTRO. : 14,90 €
BELUX : 15.90 €
CH : 23.90 CHF
ESP/IT/PORT-CONT : 14,90 €
DOM/S : 15,90 €
TUN : 45 TND
MAR : 165 MAD
CAN : 24,99 $CAD

SUPERVISION SYSTÈME

Utilisez Monit, une solution de supervision décentralisée pour garder un œil sur vos systèmes

LES INDISPENSABLES

SÉCURISEZ VOS SERVEURS

ET VOTRE RÉSEAU LOCAL

▸PREMIERS PAS
Définissez l'architecture de votre infrastructure et déployez vos serveurs

▸CONFIGURATION
Paramétrez un firewall, un système de prévention d'intrusion et un VPN pour protéger votre réseau

▸ADMINISTRATION
Répondez aux problématiques de sécurité d'accès distants en exploitant pleinement OpenSSH

▸MAINTENANCE
Sauvegardez vos données, centralisez vos logs et supervisez votre sécurité

L 12225 - 49 H - F: 14,90 € - RD

DEVOPS
Automatisez le déploiement et la configuration d'une instance JBoss/Wildfly avec Ansible et JCliff

À SAVOIR
Sachez tirer parti des outils grep et find pour rechercher des fichiers ou du texte

handicap international

À Noël, pas de Sapin, sans Sac à Sapin

....ni pochettes cadeaux !

PAPIER 100% RECYCLÉ

3 FORMATS DIFFÉRENTS

NOUVEAU !

LE COFFRET DE 5 POCHETTES POUR EMBELLIR VOS CADEAUX !

handicap-international.fr

ÉCOLOGIQUE · SOLIDAIRE · CONÇU PAR DES TRAVAILLEURS HANDICAPÉS · PRATIQUE

Retrouvez toutes nos publications
LES ÉDITIONS DIAMOND
sur **www.ed-diamond.com**

Linux Pratique Hors-Série
est édité par **Les Éditions Diamond**

12, Place du Capitaine Dreyfus, 68000 Colmar, France

Tél. : 03 67 10 00 20 / **Fax :** 03 67 10 00 21

E-mail : cial@ed-diamond.com
lecteurs@linux-pratique.com

Service commercial : abo@linux-pratique.com

Sites : www.linux-pratique.com
www.ed-diamond.com

Directeur de publication : Arnaud Metzler

Chef des rédactions : Denis Bodor

Rédactrice en chef : Aline Hof

Conception graphique : Kathrin Scali

Responsable publicité : Tél. : 03 67 10 00 27

Service abonnement : Tél. : 03 67 10 00 20

Impression : pva, Druck und Medien-Dienstleistungen GmbH, Landau, Allemagne

Illustrations/photos : https://stock.adobe.com/fr/

Distribution France :
(uniquement pour les dépositaires de presse)

MLP Réassort :
Plate-forme de Saint-Barthélemy-d'Anjou.
Tél. : 02 41 27 53 12
Plate-forme de Saint-Quentin-Fallavier.
Tél. : 04 74 82 63 04

Service des ventes :
Distri-médias : Tél. : 05 34 52 34 01

IMPRIMÉ en Allemagne - PRINTED in Germany

Dépôt légal : À parution

N° ISSN : 2101-6836

Commission Paritaire : K78 990

Périodicité : Bimestrielle

Prix de vente : 14,90 Euros

Membre April www.april.org

Retrouvez-nous sur :

@linuxpratique
@editionsdiamond

@linuxpratique

https://connect.ed-diamond.com/Linux-Pratique

www.ed-diamond.com

ÉDITO

Comme chacun le sait, cette année 2020 aura été largement marquée par la crise sanitaire liée à l'épidémie de Covid-19 qui a et aura bon nombre de répercussions sur notre société. L'accélération de la transformation digitale est l'une d'entre elles. Les mesures mises en place pour limiter les contacts entre les individus et réduire le risque de propagation du virus ont servi son développement. Cette transformation numérique « accélérée » s'est ressentie à diverses échelles. La généralisation des paiements sans contact et électroniques, le passage au télétravail qui, de temporaire durant la période de confinement a fini par perdurer au sein de certaines organisations, le télé-enseignement qui a suivi le même chemin, la poursuite de la dématérialisation des tâches administratives (dans le cadre du plan Action publique 2022)... sont autant de témoins des changements opérés durant cette année.

Au milieu de tout cela, l'utilisateur, qui a dû faire face à toutes ces évolutions et s'adapter tant bien que mal sans pour autant avoir été formé ou a minima sensibilisé aux nouvelles pratiques qui ont pu émerger. Travailler chez soi n'est pas travailler dans son bureau où une infrastructure aura été pensée et mise en place pour garantir un accès à un environnement informatique sécurisé. Un cadre dans lequel chacun aura en principe été sensibilisé aux bonnes pratiques à suivre pour se prémunir de tout risque qui pourrait nuire au SI de l'entreprise.

Il ne faut pas perdre de vue le fait que le risque humain figure en bonne place parmi les menaces informatiques auxquels toute organisation ou société doit faire face. Cela, les fraudeurs l'ont bien compris, voilà en partie pourquoi les attaques de phishing se sont multipliées depuis le début de l'année. Pour beaucoup, prendre le temps de former un minimum tout un chacun aux risques encourus en matière de cybersécurité et aux moyens de s'en prémunir serait l'une des solutions les plus efficaces pour limiter les menaces. Hélas, les formations de ce type sont loin d'être légion dans les entreprises et se limitent pour certaines à 2-3 recommandations échangées brièvement. Faute de mieux, « l'erreur humaine » est généralement en partie contrebalancée par la mise en place d'un système qui permette de limiter les menaces potentielles.

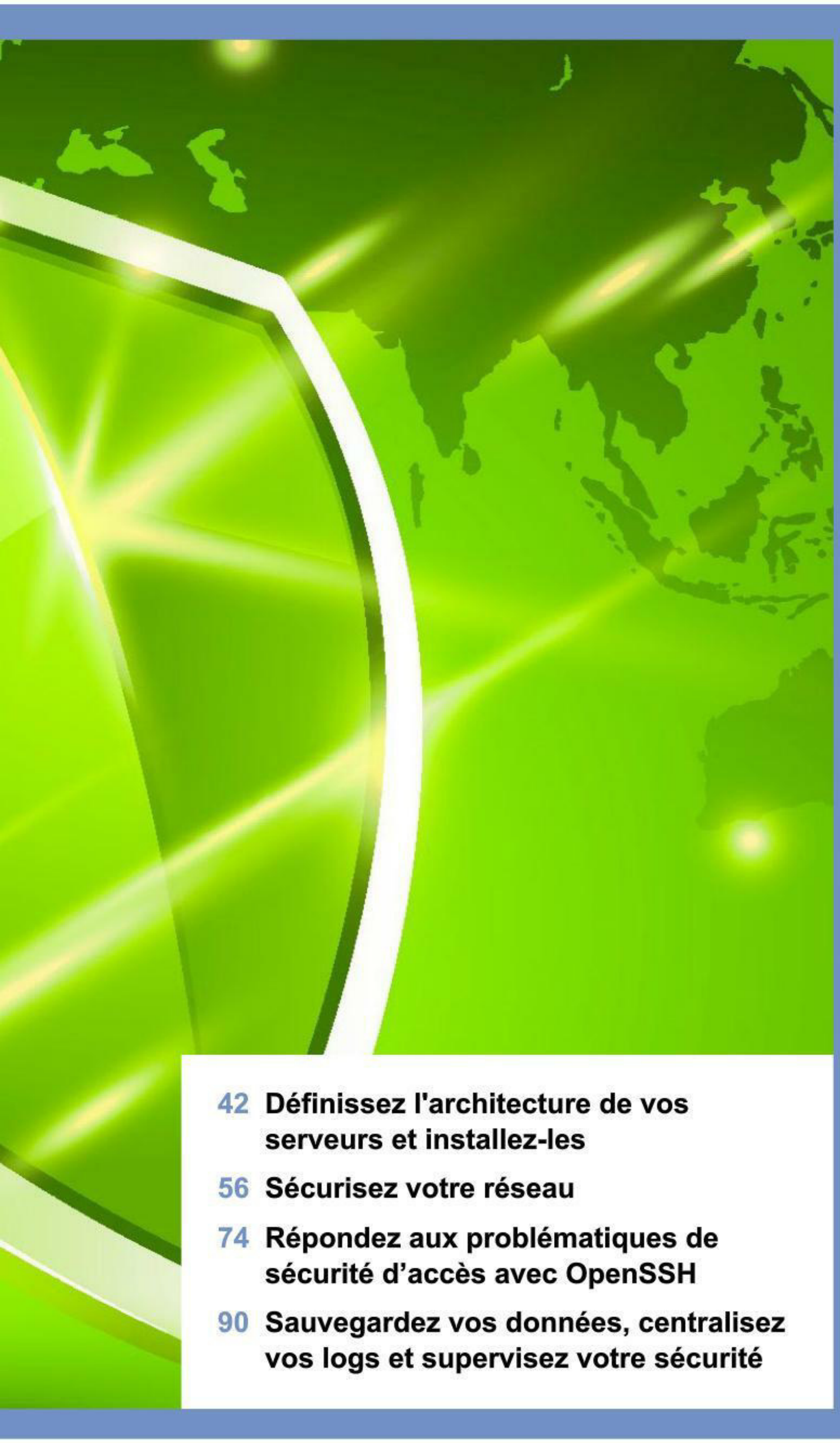
Cela tombe bien, le dossier du présent numéro s'attachera à vous accompagner dans la sécurisation de vos serveurs. Il ne s'agit pas là évidemment de la solution ultime pour prévenir l'ensemble des menaces, mais d'un excellent point de départ pour mettre en place une infrastructure évolutive et profiter de nombreuses pistes pour améliorer la sécurité de vos serveurs personnels ou d'entreprise.

Je vous laisse désormais découvrir plus en détail le contenu de ce numéro spécial en espérant qu'il vous apportera de quoi mieux faire face à d'éventuels imprévus (attaques comme situations exceptionnelles) que vous pourrez rencontrer dans le futur. ▌

Aline HOF

SOMMAIRE

LINUX PRATIQUE HORS-SÉRIE N° 49



DÉCOUVREZ LA NOUVELLE VERSION !

CONNECT

LA DOCUMENTATION TECHNIQUE DES PROS DE L'IT

LES 6 NOUVEAUTÉS :

1. Nouvelle ergonomie
2. Articles premiums jamais publiés
3. Parcours thématiques de la rédaction
4. Listes de lecture personnalisées
5. Création d'alertes de publication
6. Moteur de recherche amélioré

ET TOUJOURS...

- ✓ Plus de 2500 articles disponibles
- ✓ Plus de 130 nouveaux articles chaque année
- ✓ Magazines et hors-séries d'hier et d'aujourd'hui
- ✓ Accès illimité 24h/24 - 7j/7
- ✓ Abonnements multi-connexions

CONTACTEZ-NOUS :

par téléphone :
03 67 10 00 28

par e-mail :
connect@ed-diamond.com

connect.ed-diamond.com

LINUX PRATIQUE

Lecture en avant-première des magazines en ligne !

LA DOCUMENTATION TECHNIQUE DES PROS DE L'IT

OUI, JE M'ABONNE

Abonnement à retourner avec votre règlement à :
Les Éditions Diamond Service des Abonnements 12 Place du Capitaine Dreyfus, 68000 Colmar, France

Réf. : LP+3	Réf. : LP+3/5
☐ **1 LECTEUR : 189 € HT* / AN** soit : 15,75 € HT / MOIS / LECTEUR	☐ **5 LECTEURS : 259 € HT* / AN** soit : 4,32 € HT / MOIS / LECTEUR

(France) TVA 20% = ...

Total = ...

* Tarifs France Métro.

Merci de consulter les tarifs hors France Métro. sur : www.ed-diamond.com

Société : ..... Nom : ..... Prénom : .....

Adresse : .....

Code Postal : ..... Ville : .....

Pays : ..... Téléphone : .....

Veuillez indiquer svp l'adresse e-mail du référent : .....

Veuillez adresser les règlements par **chèque bancaire** ou **postal** à l'ordre des Éditions Diamond (uniquement France et DOM TOM). Pour les règlements par **virement**, merci de nous contacter **par e-mail : cial@ed-diamond.com ou par téléphone : +33 (0)3 67 10 00 20**

EN BREF...

» UN MOOC POUR SE FORMER AU NUMÉRIQUE RESPONSABLE

30 minutes c'est le temps qu'il vous faut environ pour suivre cette formation qui vous permettra de mieux cerner les enjeux d'un numérique plus responsable et vous fournira des ressources utiles pour aller plus loin sur le sujet. Mis au point par l'Institut du Numérique Responsable et La Rochelle Université, ce MOOC est accessible à tous ceux cherchant à mieux comprendre les impacts du numérique sur l'environnement.

Ce premier module – qui sera complété par un programme de formation plus complet qui devrait être disponible à l'heure à vous lirez ces lignes – vous fournira un état des lieux sur le contexte actuel et des pistes pour agir. Il comprend 11 leçons qui pourront être suivies dans l'ordre de votre choix. Pour suivre cette formation gratuite, rendez-vous sur **https://www.academie-nr.org/sensibilisation/#/**. ▌

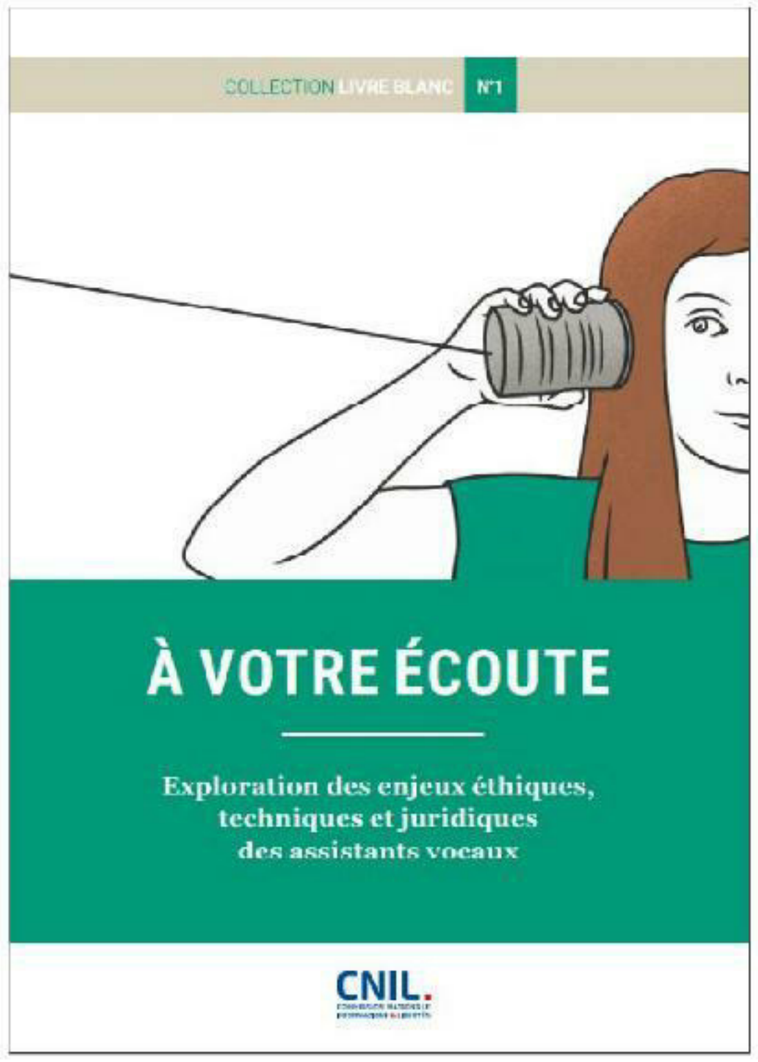
» FACILITER LE RÉEMPLOI D'ORDINATEURS USAGÉS AVEC EMMABUNTÜS

Le collectif Emmabuntüs œuvre en faveur du reconditionnement d'ordinateurs à destination des associations humanitaires (en particulier aux communautés Emmaüs). Ses initiatives sont multiples comme vous pourrez le découvrir sur son nouveau site internet à l'adresse : **https://emmabuntus.org/**.

Elle a notamment développé une distribution Linux (Emmabuntüs) pour faciliter le réemploi d'ordinateurs usagés. L'une de ses dernières campagnes, réalisée conjointement avec ses partenaires BlaBlaLinux.be, Tugaleres.com et Linux-Live-CD.org vise à montrer comment créer facilement une clé USB de réemploi

» UN LIVRE BLANC CONSACRÉ AUX ASSISTANTS VOCAUX

Les assistants vocaux sont de plus en plus présents dans nos vies, or ces derniers peuvent être perçus comme étant particulièrement intrusifs et leur traitement de nos données personnelles peut également être problématique. C'est pourquoi la CNIL travaille depuis 2016 sur le sujet et vient de publier les fruits de cette investigation dans un livre blanc qui permet de mieux cerner les enjeux de cette nouvelle technologie.



Ce livre blanc – que l'on pourra se procurer sur **https://www.cnil.fr/fr/votre-ecoute-la-cnil-publie-son-livre-blanc-sur-les-assistants-vocaux** – s'adresse aussi bien aux concepteurs/développeurs/intégrateurs de cette technologie, qu'aux organismes qui souhaite la déployer (dans le respect des dispositions du RGPD) sans oublier les utilisateurs finaux qui souhaitent y recourir.

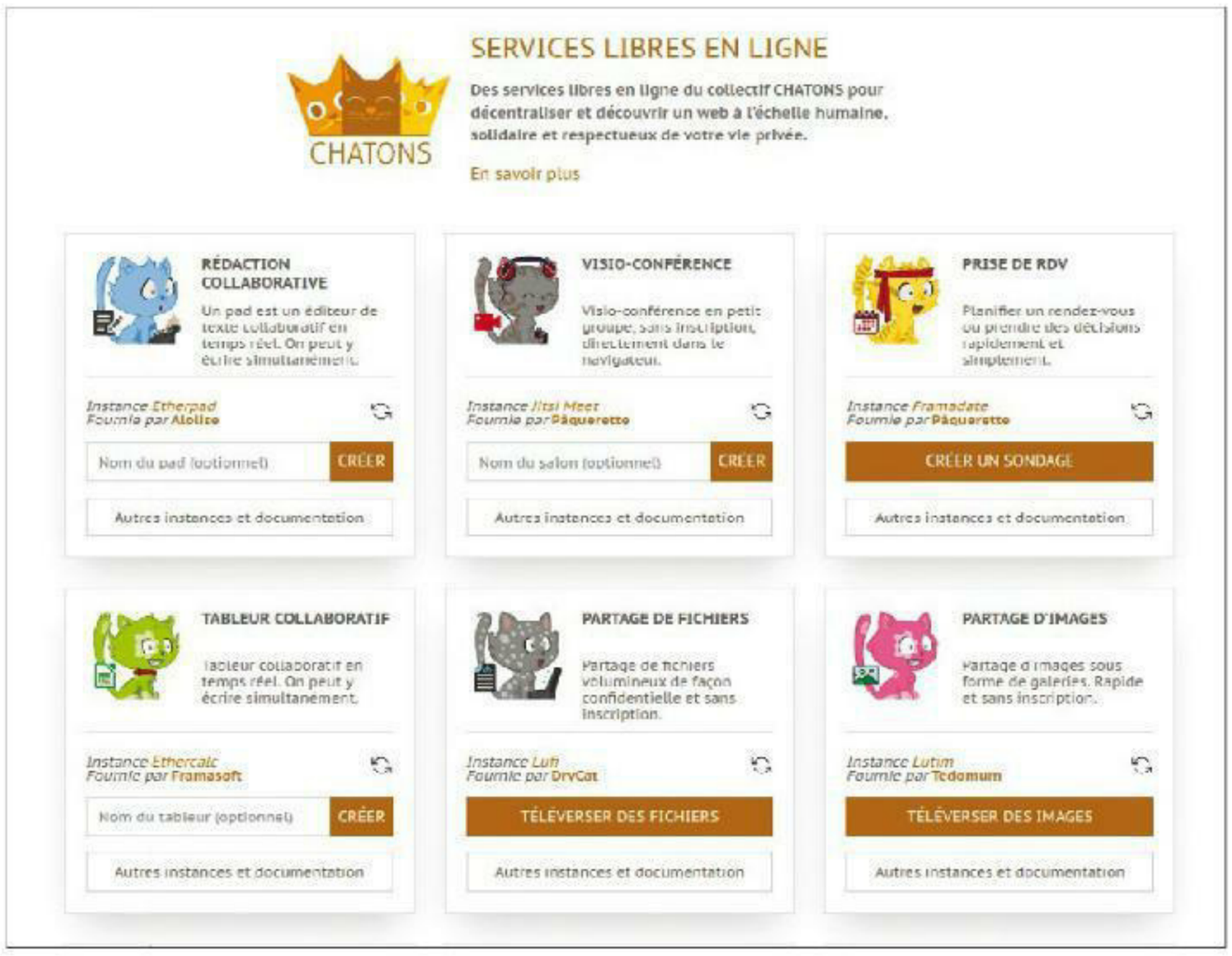
Vous y trouverez tout d'abord un premier chapitre permettant de mieux appréhender cette technologie, puis une seconde partie détaillant les mythes et enjeux des assistants vocaux. Le troisième chapitre est dédié au RGPD avec des exemples de cas d'usage propres à l'utilisation de cette technologie. Et pour finir, la dernière section fournira des conseils à tout un chacun autour de ces assistants.

Ainsi, si vous cherchez prochainement à acquérir un produit de ce type, la lecture de ce document vous éclairera tout particulièrement sur les points auxquels il faudra être vigilant (la confidentialité de vos échanges, l'enrichissement de votre profil publicitaire, l'usage qui peut en être fait par des enfants, etc.). ▌

afin de simplifier toujours davantage la réutilisation de vieilles machines. Le mode d'emploi est disponible sur **https://emmabuntus.org/methode-de-realisation-de-la-cle-de-reemploi-emmabuntus/** et s'accompagne de vidéos de démonstration permettant ainsi à tout un chacun de se lancer. ▌

» LES SERVICES LIBRES EN LIGNE DU COLLECTIF CHATONS

Profiter de services libres et décentralisés et en prime respectueux de votre vie privée, le tout en quelques clics c'est possible et cela se passe du côté du collectif d'hébergeurs alternatifs CHATONS à l'adresse **https://entraide.chatons.org/fr/**. Il sera possible de bénéficier ainsi rapidement :



- d'un éditeur de texte collaboratif ;
- d'un outil de visioconférence ;
- d'un tableur collaboratif ;
- d'un outil de création de sondages pour planifier de futurs rendez-vous ou recueillir des avis sur un sujet ;
- d'un outil de partage de fichiers ;
- d'un outil de partage d'images sous forme de galeries ;
- d'un outil de partage de textes chiffrés ;
- d'un éditeur de notes sous la forme de post-its ;
- d'un raccourcisseur de liens.

On pourra se lancer en un clic, changer d'instance et/ou s'informer en jetant un œil à la documentation de chaque outil. Notez que les habitués des services en ligne proposés par l'association Framasoft ne devraient pas être dépaysés puisqu'il s'agit d'une partie des services en question que l'association a souhaité décentraliser. ▌

» ÉVALUER LA MATURITÉ NUMÉRIQUE DE SON ENTREPRISE

Le CNAM Grand Est a établi un questionnaire en ligne, réalisé par un collectif interdisciplinaire de professionnels à partir de travaux universitaires, permettant de mesurer la transition d'une entreprise ou d'une organisation au numérique.

Les questions portent sur divers pans de l'activité professionnelle (le personnel, la communauté, le cadre juridique et règlementaire, les ressources, les activités et la stratégie) et permettront d'attribuer un pourcentage

» UN GUIDE POUR ANTICIPER ET RÉAGIR FACE AUX RANSOMWARES

Les ransomwares ou rançongiciels ont connu une forte hausse depuis le début de l'année. Pour permettre aux entreprises et collectivités de faire face à ce constat inquiétant, l'Agence Nationale de la Sécurité des Systèmes d'Information (l'ANSSI) a publié un guide qui leur est dédié en partenariat avec la Direction des Affaires criminelles et des grâces (DACG) du ministère de la Justice.

Ce guide, intitulé « Attaques par rançongiciels, tous concernés – Comment les anticiper et réagir en cas d'incident ? » fournit des recommandations tirées d'expériences de plusieurs acteurs (la CNIL, la Direction centrale de la Police judiciaire, le dispositif cybermalveillance.gouv.fr ou encore la Brigade de lutte contre la cybercriminalité) et met en lumière les témoignages de trois organisations victimes d'un ransomware en 2019. Le document, qui pourra être téléchargé depuis **https://www.ssi.gouv.fr/guide/attaques-par-rancongiciels-tous-concernes-comment-les-anticiper-et-reagir-en-cas-dincident/**, évoque le sujet sous la forme de deux grands chapitres : réduire le risque d'attaque et réagir en cas d'attaque.

Concernant le premier volet, diverses bonnes pratiques sont soulignées parmi lesquelles : la sauvegarde de ses données, l'importance de maintenir à jour ses logiciels (antivirus compris) et systèmes, le cloisonnement de son système d'information, une meilleure maîtrise des accès Internet, la sensibilisation de tous ses collaborateurs, la mise en œuvre d'un plan de réponse aux cyberattaques, etc.

Le second volet traite des actions à mettre en place pour réduire les pertes liées à une attaque par rançongiciel. Plusieurs points sont à cet effet mis en avant : la mise en œuvre d'une gestion de crise, la nécessité de trouver une assistance technique, la mise en place d'une communication spécifique, l'importance de ne pas payer de rançon, le dépôt d'une plainte...

Le tout se complète de ressources utiles qui fourniront des éléments supplémentaires aux entreprises et organisations cherchant à se sensibiliser au maximum à ce sujet. ▌

de maturité numérique pour chacun d'entre eux. Dirigeants et responsables auront ainsi un meilleur aperçu des secteurs sur lesquels mettre l'accent dans le cadre de leur transformation numérique.
Le test pourra être effectué sur **http://maturite-numerique.cnam-grandest.fr/**. ▌

RECHERCHEZ EFFICACEMENT DES FICHIERS OU DU TEXTE AVEC FIND ET GREP

Tristan COLOMBO

Dans la vie de tous les jours, lorsque l'on perd un objet, on se retrouve la plupart du temps seul face à sa mémoire, sans personne pour vous indiquer où vous avez bien pu poser ces #$*!@& de clés ! Sous Linux, il existe des outils bien pratiques qui vous permettront de gagner un temps appréciable...

Lorsque l'on travaille sur un ordinateur, on y stocke nécessairement des données qui vont être organisées dans des fichiers à l'intérieur d'une arborescence de répertoires. Ça, c'est si vous êtes un minimum rigoureux... et même avec la meilleure volonté du monde il arrive, par flemme, urgence ou autre, que l'on enregistre un fichier un peu n'importe où. Sur le moment cela n'a pas trop d'importance, on sait très bien que le fichier se trouve dans un répertoire improbable dans lequel il ne devrait pas être, et on se souvient même du nom de ce répertoire. Mais au bout de quelques jours, voire quelques mois ?

Nous allons partir dans cet article d'un exemple pratique, un fichier que nous allons déposer dans un répertoire étrange et nous allons étudier les différentes façons de le retrouver. Ce fichier se nommera `fichier.txt` et sera placé dans votre répertoire personnel (nous utiliserons ici une Raspberry Pi et ce répertoire est en général `/home/pi`) dans `Public/Rep/SousRep`. Pour créer le répertoire, nous utiliserons les commandes suivantes :

```
pi@raspberry:~ $ cd Public
pi@raspberry:~/Public $ mkdir Rep
pi@raspberry:~/Public $ cd Rep
pi@raspberry:~/Public/Rep $ mkdir SousRep
pi@raspberry:~/Public/Rep $ cd SousRep
pi@raspberry:~/Public/Rep/SousRep $
```

Notez que pour créer un sous-répertoire (ici `SousRep`), il faut que le répertoire parent existe (ici `Rep`), ce qui implique des opérations de création, de déplacement, de création, etc. Cela peut s'avérer fastidieux si vous devez créer un sous-répertoire dans une arborescence profonde... Heureusement que `mkdir` dispose de l'option `-p` pour créer les répertoires parents s'ils n'existent pas. En utilisant cette option, nos lignes de commandes ne seraient plus que :

```
pi@raspberry:~ $ mkdir -p Public/Rep/SousRep
pi@raspberry:~ $ cd Public/Rep/SousRep
pi@raspberry:~/Public/Rep/SousRep $
```

Pour créer le fichier `fichier.txt`, vous pouvez utiliser l'éditeur nano (vous verrez plus loin dans ce hors-série d'autres éditeurs plus performants) :

```
pi@raspberry:~/Public/Rep/SousRep $ nano
fichier.txt
```

Le contenu du fichier est le suivant :

```
Ceci est un petit fichier de test.
Il contient un peu de code:
import random
print('Tirage aléatoire :', random.randint(1, 10))
```

Appuyez ensuite sur [Ctrl] + [X], ensuite [O] pour confirmer l'écriture du fichier, et [Return] pour valider le nom du fichier.

Retournons maintenant dans le répertoire utilisateur (par `cd ~`) et considérons ce fichier comme « perdu ».

1. PREMIER CAS : ON SE SOUVIENT DU NOM OU D'UNE PARTIE DU NOM

En connaissant le nom du fichier recherché ou éventuellement seulement une partie, on se trouve dans la situation la plus confortable, car nous possédons une information qui va nous permettre d'éliminer beaucoup de candidats.

locate

La commande `locate` permet d'afficher instantanément la localisation d'un fichier. Pour cela, elle a besoin d'indexer l'ensemble des fichiers du disque au sein d'une base de données (il s'agit du fichier `/var/lib/mlocate/mlocate.db`). Si vous recherchez le fichier que nous venons de créer... vous ne le trouverez pas :

```
pi@raspberry:~ $ locate
fichier.txt
pi@raspberry:~ $
```

Vous n'obtiendrez pas de message d'erreur, seulement une absence de réponse.

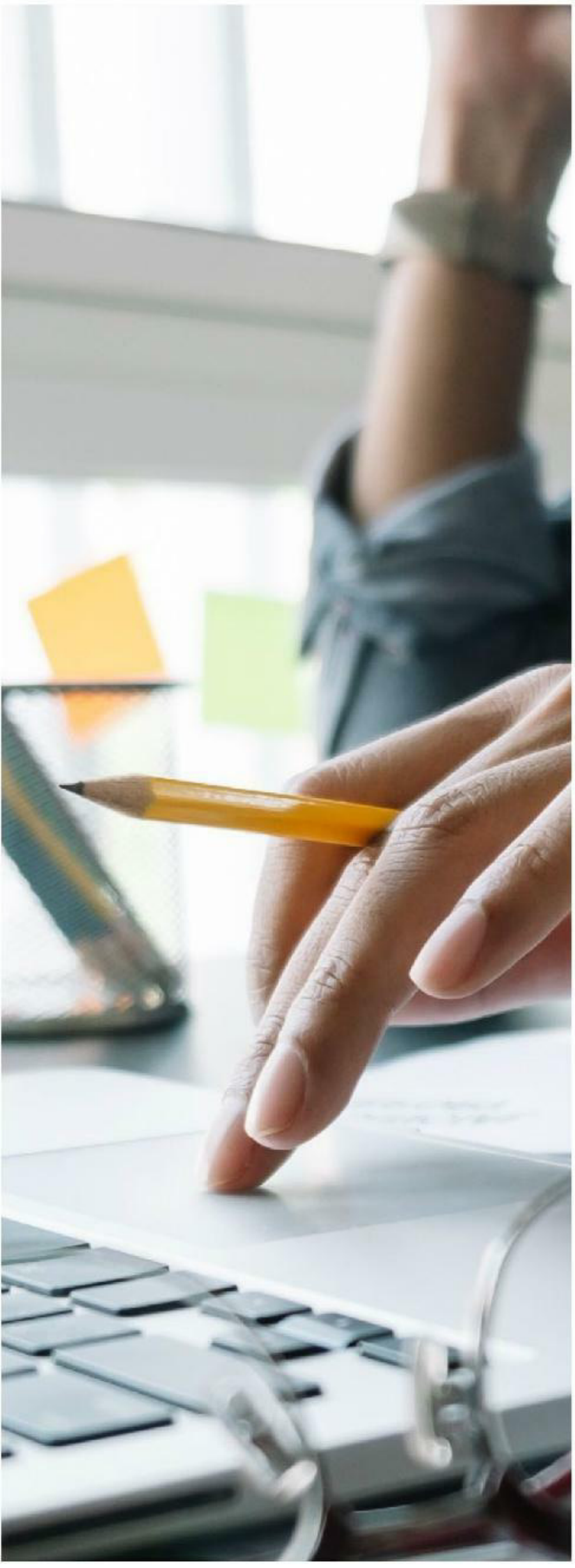
ARBORESCENCE DES FICHIERS

Pour classer les fichiers, on utilise des répertoires qui peuvent donc contenir des fichiers, mais également d'autres répertoires (que l'on nommera alors sous-répertoires). La commande `tree` permet d'avoir une vision graphique de cette arborescence :

```
pi@raspberry:~ $ sudo apt-get install tree
pi@raspberry:~ $ tree
.
├── affiche.py
├── Desktop
├── display16x32
│   ├── hackable
│   │   └── test_display.py
│   └── rpi-rgb-led-matrix
├── Documents
│   └── Scratch Projects
│       ├── Asteroid Blaster.sb
│       ├── Pacman for Scratch.sb
│       └── Scratch Invaders.sb
├── Downloads
├── HK_display8x8
├── Music
├── oldconffiles
├── Pictures
│   ├── affiche_serpent.png
│   ├── jeu_avec_murs.png
│   ├── serpent_avance_1.png
│   └── WS2812 Matrix Emulator_022.png
├── Public
│   └── Rep
│       └── SousRep
│           └── fichier.txt
├── python_games
├── Templates
└── Videos
```

Pour se déplacer dans l'arborescence, on utilise la commande `cd` (abréviation de *change directory* - changer de répertoire). `cd nom_repertoire` permet de se déplacer dans le répertoire `nom_repertoire` à condition que celui-ci soit un enfant du répertoire courant (visible par la commande `ls`). Par exemple, comme `Rep` est un répertoire enfant de `Public`, alors, depuis le répertoire `Public`, `cd Rep` sera valide.

Il faut également savoir que `..` est un raccourci pour désigner le répertoire parent. En étant dans `Rep`, `cd ..` nous déplace dans `Public`.



Note !

Pour afficher la valeur de retour d'une commande du shell, on peut afficher la valeur de la variable $?. Cette variable vaut 0 si tout s'est bien déroulé et un code d'erreur compris entre 1 et 255 sinon. Dans le cas d'un appel à `locate` sur un nom de fichier absent de la base, on obtient la valeur 1 :

```
pi@raspberry:~ $ locate fichier.txt
pi@raspberry:~ $ echo $?
1
```

Que l'on ne trouve pas un fichier absent du disque dur, cela paraît logique... mais pourquoi le fichier que nous venons de créer n'est-il pas détecté ? Comme expliqué précédemment, la recherche est en fait exécutée dans une base de données, ce qui permet d'obtenir des résultats très rapidement. Mais pour pouvoir trouver un fichier, il faut que la base de données soit à jour ! L'opération de mise à jour de la base peut être très longue si elle n'a jamais été exécutée. La commande à lancer est **`updatedb`**. Si nous l'exécutons, nous aurons cette fois un résultat :

```
pi@raspberry:~ $ updatedb
pi@raspberry:~ $ locate fichier.txt
/home/pi/Public/Rep/SousRep/fichier.txt
```

Cette commande fonctionne également si l'on ne connaît qu'une partie du nom du fichier :

```
pi@raspberry:~ $ locate hier.t
/home/pi/Public/Rep/SousRep/fichier.txt
```

Extrêmement rapide pour retrouver des fichiers indexés dans la base, c'est la commande à privilégier pour retrouver des fichiers anciens pour lesquels on est certain qu'une commande **`updatedb`** a été lancée. En effet, si vous devez lancer une mise à jour de la base pour trouver votre fichier, autant utiliser **`find`** !

find

`find` est une commande « couteau suisse » de la recherche de fichiers. Elle dispose en effet de nombreuses options dont nous ne pourrons voir ici qu'une partie (voir **`man find`** et l'encadré sur les pages de manuel).

Commençons par l'appel le plus simple : nous connaissons le nom du fichier puisqu'il s'agit de **`fichier.txt`** et nous allons supposer que l'on

sait également qu'il se trouve dans notre répertoire personnel (donc **/home/pi**). Pour effectuer la recherche, nous allons indiquer à **find** à partir de quel répertoire chercher (il explorera ensuite tous les sous-répertoires jusqu'à parcourir toute l'arborescence) et quel est le nom (*name* en anglais) du fichier à retrouver :

```
pi@raspberry:~ $ find /home/pi -name fichier.txt
/home/pi/Public/Rep/SousRep/fichier.txt
```

La réponse a été plus ou moins rapide, dépendant du nombre de fichiers et de répertoires présents dans votre répertoire personnel. Bien entendu, la même recherche lancée sur tout le disque (donc **/**) sera beaucoup plus lente !

Si vous êtes en possession d'autres informations, permettant de cibler plus précisément le fichier recherché (plusieurs fichiers peuvent porter le même nom), vous pourrez les ajouter. Par exemple, si la taille du fichier est supérieure à 1 Mio on ajoutera **-size +1M**, si le fichier a été modifié il y a moins d'une semaine on ajoutera **-mtime -7**, etc. Certaines options pourront être cumulées. Ainsi pour un fichier de plus d'1 Mio modifié il y a moins d'une semaine, on exécutera :

```
pi@raspberry:~ $ find /home/pi -size +1M -mtime -7 -name fichier.txt
```

Voici un tableau récapitulant les principales options :

Option	Description
-size n	Recherche des fichiers de taille **n** : avec **+n** c'est supérieur à **n**, avec **-n** c'est inférieur à **n** et avec **n** sans signe c'est exactement égal à **n**. Cet usage des signes **+** et **-** est valable pour toutes les options nécessitant un paramètre de taille, durée, etc. Les suffixes de taille suivant sont autorisés (et les plus employés), à placer après la taille **n** : ■ **c** pour des octets ; ■ **k** pour des kibi-octets ; ■ **M** pour des mébi-octets ; ■ et **G** pour des gibi-octets.
-ctime n	Recherche des fichiers modifiés il y a **n** jours (**+n** pour plus de **n** jours, **-n** pour moins de **n** jours et **n** pour **n** jours exactement). Ne détecte que les modifications de contenu des fichiers.
-mtime n	Recherche des fichiers modifiés il y a **n** jours. Détecte les modifications de contenu de fichiers et de leurs attributs (lecture, écriture, exécution).
-type	Type des fichiers : ■ **f** pour un fichier régulier ; ■ **d** pour un répertoire ; ■ **l** pour un lien symbolique ; ■ etc.
-amin	Fichier auquel on a accédé il y a **n** minutes.

Pour récapituler, si nous recherchons le fichier de nom commence par « fic », d'extension .txt, qui est un fichier régulier, de taille inférieure à 500 octets et dont le contenu a été modifié il y a plus de 3 jours :

```
pi@raspberry:~ $ find /home/pi -name "fic*.txt" -type f -size -500c
-ctime +3
```

Par extension, bien évidemment, on peut également rechercher les fichiers correspondant à certains critères sans connaître le nom du fichier ! Pour tous les fichiers d'extension .txt :

```
pi@raspberry:~ $ find /home/pi -name "*.txt"
```

Remarque

L'utilisation des guillemets pour encadrer une expression contenant des caractères *joker* telle que `*.txt` n'est pas obligatoire... jusqu'au jour où vous lancerez cette recherche depuis un répertoire contenant déjà un fichier .txt. Si ce fichier se nomme `poubelle.txt`, votre commande sera transformée en `find /home/pi -name poubelle.txt`, ce qui n'a pas la même signification que notre commande initiale. Il est donc recommandé de toujours employer des guillemets.

On peut encore rechercher tous les fichiers de plus de 5 Mio, sans en préciser le nom (l'option `-name` n'est alors plus utile) :

```
pi@raspberry:~ $ find /home/pi -size +5M
```

Il faut également savoir que `find` dispose d'une option redoutable : `-exec`. Grâce à cette option, il est possible d'exécuter une commande shell sur le ou les fichiers correspondants à la recherche. Le plus courant est par exemple de rechercher des fichiers pour les effacer. Supposons que nous souhaitions supprimer tous les fichiers .iso de notre répertoire personnel. Nous n'aurons qu'à taper :

```
pi@raspberry:~ $ find ~ -name "*.iso" -exec rm {} \;
```

> **Rappel**
>
> **Le caractère ~ est un raccourci qui désigne le répertoire utilisateur (donc `/home/pi` si votre utilisateur est `pi`).**

Lors de l'exécution de la commande, les caractères `{}` sont remplacés itérativement par le nom des fichiers trouvés, créant pas à pas une nouvelle commande. Il est très important de ne pas oublier de préfixer le point-virgule par `\` de manière à indiquer qu'il fait partie de la commande `find` et n'est pas là pour séparer deux instructions du shell. Si vous l'oubliez, vous obtiendrez un message d'erreur peu signifiant si vous n'êtes pas au courant de ce fonctionnement :

```
pi@raspberry:~ $ find ~ -name "*.iso" -exec rm {} ; # Attention ligne erronée
find: Paramètre manquant pour " -exec "
```

Notez pour finir que l'option `-ok` est équivalente à `-exec` à la différence près qu'elle demande confirmation avant l'exécution de chaque commande.

> **Rappel**
>
> **Il est possible d'écrire plusieurs commandes sur une seule ligne en les séparant par des points-virgules. Par exemple :**

```
pi@raspberry:~ $ ls ; echo "Hello" ; find . -name "*.txt"
```

> **Rappel**
>
> **Le caractère # permet de définir des commentaires en ligne : tout ce qui suit ne sera pas considéré comme une commande :**

```
pi@raspberry:~ $ ls # Ceci est un commentaire
```

Un cas à part : retrouver le nom d'un paquet contenant un fichier spécifique

Parfois, lorsque l'on veut compiler des sources, on se heurte à des messages d'erreur indiquant que tel ou tel fichier n'a pas été trouvé. Il s'agit généralement de bibliothèques qui peuvent être installées simplement avec le système apt... à condition de connaître le nom du paquet ! Heureusement que la commande `apt-file` permet de réaliser simplement cette recherche. Pour commencer, il faut l'installer :

```
pi@raspberry:~ $ sudo apt install apt-file
```

Ensuite, comme avec **locate**, il faut mettre à jour sa base de données :

```
pi@raspberry:~ $ apt-file update
```

Il suffit de lui passer en paramètre le nom du fichier manquant :

```
pi@raspberry:~ $ apt-file search libvlc.so.5
libvlc5: /usr/lib/libvlc.so.5
libvlc5: /usr/lib/libvlc.so.5.5.0
```

On sait donc maintenant qu'en installant le paquet **libvlc5**, nous aurons deux fichiers **libvlc.so.5** et **libvlc.so.5.5.0** (correspondants à notre recherche) qui seront installés dans **/usr/lib**.

2. DEUXIÈME CAS : ON A OUBLIÉ LE NOM, MAIS ON SAIT CE QUE DOIT CONTENIR LE FICHIER

Dans le cas de figure où l'on a oublié le nom du fichier, il faudra partir sur une recherche basée sur des éléments se trouvant dans le fichier.

grep

Le nom de la commande **grep** provient de *Global Regular Expression Print* et, comme son nom le laisse supposer, elle utilise les expressions régulières (voir encadré). Elle permet de rechercher une chaîne de caractères dans des fichiers et peut être utilisée comme un filtre (i.e. traiter la sortie d'une autre commande à l'aide du symbole *pipe* : **|**).

Effectuons une première recherche simple. Nous savons que notre fichier contient « import random » et nous allons donc voir ce que **grep** peut trouver :

```
pi@raspberry:~ $ grep "import random" *
pi@raspberry:~ $
```

Nous avons demandé à **grep** d'examiner le contenu de tous les fichiers (caractère *joker* *****), et pourtant nous n'avons aucun résultat... En fait la commande n'a effectué sa recherche que dans le répertoire courant ! Si l'on souhaite une recherche dans toute l'arborescence il faut passer en mode récursif en spécifiant l'option **-r** :

```
pi@raspberry:~ $ grep -r "import random" *
Public/Rep/SousRep/fichier.txt:import random
python_games/memorypuzzle.py:import random, pygame, sys
...
```

Rappel

Bien entendu, si l'on veut vérifier la présence d'une chaîne de caractères au sein d'un fichier dont le nom est connu, c'est tout à fait possible :

```
pi@raspberry:~ $ grep "import random" Public/Rep/SousRep/fichier.txt
import random
```

Nous avons commencé par une recherche très simple... mais il serait dommage de ne pas exploiter les expressions régulières ! Supposons maintenant que nous ayons un vague souvenir du contenu du fichier recherché : une des lignes fait appel à `randint()` en lui passant deux entiers en paramètres. Cela se traduit par :

```
pi@raspberry:~ $ grep "randint([0-9]*, [0-9]*)" Public/Rep/SousRep/
fichier.txt
print('Tirage aléatoire :', random.randint(1, 10))
```

LES PAGES DE MANUEL

Pour obtenir de l'aide sur une commande et notamment la liste des paramètres attendus et des options possibles, on utilise `man` (abréviation de *manual*, le manuel d'utilisation). Pour savoir comment utiliser cette commande, il suffit de taper... `man man` !

Dans son utilisation la plus simple, on tape `man` suivi du nom de la commande sur laquelle on s'interroge. Par exemple, dans le cas de `find` on exécute :

```
pi@raspberry:~ $ man find
FIND(1)                         General Commands Manual

NOM
       find - Rechercher des fichiers dans une hiérarchie de
répertoires

SYNOPSIS
       find [-H] [-L] [-P] [-D option-debogage] [-Oniveau]
[chemin...] [expression]

DESCRIPTION
       Cette  page de manuel documente la version GNU de find.
...
```

Le `(1)` que vous pouvez voir en première ligne, juste après `FIND`, signifie qu'il s'agit de la page « Programmes exécutables ou commandes de l'interpréteur de commandes (shell) ». En effet, les commandes sont décrites dans des sections. Il y a au total 9 sections :

- 1 Programmes exécutables ou commandes de l'interpréteur de commandes (shell) ;
- 2 Appels système (fonctions fournies par le noyau) ;
- 3 Appels de bibliothèque (fonctions fournies par les bibliothèques des programmes) ;
- 4 Fichiers spéciaux (situés généralement dans `/dev`) ;
- 5 Formats des fichiers et conventions. Par exemple, `/etc/passwd` ;
- 6 Jeux ;
- 7 Divers (y compris les macropaquets et les conventions), par exemple man(7), groff(7) ;
- 8 Commandes de gestion du système (généralement réservées au super-utilisateur) ;
- 9 Sous-programmes du noyau [hors standard].

Attention !

Une expression régulière recherche précisément le motif qui lui est transmis. Ainsi, nous avons recherché (et trouvé) un appel à `randint()` dans lequel les paramètres entiers sont séparés par une virgule suivie d'un espace... si nous n'avions pas indiqué l'espace dans le motif nous n'aurions rien trouvé.

Certaines commandes ne possèdent des informations que dans une seule section, comme c'est le cas pour `find` alors que d'autres peuvent par exemple être utilisées dans le shell (section 1), mais aussi dans un programme C (section 3). Pour faire référence à une section particulière d'une page de `man`, il suffit de l'indiquer sur la ligne de commandes. Voici un exemple pour `printf` en section 1 puis en section 3 :

```
pi@raspberry:~ $ man 1 printf
PRINTF(1)                  Commandes

NOM
       Printf - Formater et afficher des données

SYNOPSIS
       printf FORMAT [PARAMÈTRE] ...
       printf OPTION
...
pi@raspberry:~ $ man 3 printf
PRINTF(3)                  Linux Programmer's Manual

NAME
       printf, fprintf, sprintf, snprintf, vprintf, vfprintf,
vsprintf, vsnprintf - formatted output conversion

SYNOPSIS
       #include <stdio.h>

       int printf(const char *format, ...);
       int fprintf(FILE *stream, const char *format, ...);
       int sprintf(char *str, const char *format, ...);
...
```

Rappel

Pour que vos pages soient en français, vous devez ajouter un paquet supplémentaire :

```
pi@raspberry:~ $ sudo apt install manpages-fr
```

Nous avons vu les bases du fonctionnement de **grep**. Maintenant, il n'y a plus qu'à l'agrémenter de quelques options bien choisies et vous obtiendrez un outil surpuissant pour toutes vos recherches :

Option	Description

-c	Affiche le nombre de lignes qui contiennent le motif de recherche. Exemple :
	`pi@raspberry:~ $ grep -r -c "import random" *`
	`Pictures/food_1.png : 0`
	`python_games/simulate.py: 1`
	`...`

--color	Met en valeur (couleur) le motif trouvé sur chaque ligne. Exemple :
	`pi@raspberry:~ $ grep --color "random" Public/Rep/SousRep/fichier.txt`
	`import random`
	`print('Tirage aléatoire :', random.randint(1, 10))`

-n	Préfixe les lignes correspondant au motif de recherche par leur numéro. Exemple :
	`pi@raspberry:~ $ grep -r -n "random" *`
	`python_games/simulate.py:193:    newBgColor = (random.randint(0, 255), random.randint(0, 255), random.randint(0, 255))`
	`...`

-f	Lit les motifs à rechercher dans un fichier. Par exemple avec le fichier **liste_recherche** :
	`randint([0-9]*, [0-9]*)`
	`^import`
	On peut appeler :
	`pi@raspberry:~ $ grep -r -f liste_recherche *`
	`import random`
	`print('Tirage aléatoire :', random.randint(1, 10))`

-i	Ignore la casse (majuscules ou minuscules) dans le motif de recherche. Exemple :
	`pi@raspberry:~ $ grep -i "Ti" Public/Rep/SousRep/fichier.txt`
	`Ceci est un petit fichier de test.`
	`Il contient un peu de code:`
	`print('Tirage aléatoire :', random.randint(1, 10))`

Bien évidemment, il existe d'autres options (voir la page de **man**) et ces options sont cumulables :

```
pi@raspberry:~ $ grep -r --color -n -i "import random" *
Public/Rep/SousRep/fichier.txt:3:import random
python_games/memorypuzzle.py:6:import random, pygame, sys
...
```

Chez votre marchand de journaux

et sur **www.ed-diamond.com**

Toujours disponible sur

www.ed-diamond.com

PAPIER

FLIPBOOK HTML5

sur www.ed-diamond.com

CONNECT

LA DOCUMENTATION TECHNIQUE DES PROS DE L'IT

sur connect.ed-diamond.com

LES EXPRESSIONS RÉGULIÈRES

Une expression régulière permet de définir un motif, un modèle qui va permettre d'identifier des chaînes de caractères. On utilise pour cela des caractères ayant une signification spéciale :

Caractère	Signification
^	Début de ligne.
$	Fin de ligne.
.	Un caractère quelconque.
*	Répétition de 0 à n fois du caractère précédent.
[]	Un des caractères présent entre les crochets.
[^]	Un caractère différent de ceux entre les crochets.
\	Protège un caractère (par exemple, pour désigner le caractère * qui a une signification pour les expressions régulières, on utilise *).

Les expressions régulières étendues (qui ne sont pas acceptées par défaut par `grep` à moins d'utiliser l'option `-E`) offrent un certain nombre de raccourcis. En voici quelques-uns :

Motif	Signification
+	Répétition de 1 à n fois du caractère précédent.
|	Union. (a|b) signifie a ou b.
[:digit:]	Un chiffre ([0-9]).
[:lower:]	Une lettre minuscule (équivalent à [a-z]).
[:upper:]	Une lettre majuscule (équivalent à [A-Z]).

Les variantes de grep

Nous avons précédemment utilisé l'option `-r` de `grep` pour effectuer une recherche récursive dans toute l'arborescence. La commande `rgrep` est équivalente à `grep -r` (donc inutile de lui ajouter l'option `-r`).

L'option `-F` de `grep` permet d'indiquer que l'on recherche une chaîne de caractères et non pas une expression régulière, ce qui se traduit par un temps de traitement bien moins important. La commande `fgrep` aura un comportement identique à `grep -F`.

Avec l'option `-E` de `grep`, les expressions régulières étendues (voir encadré) sont reconnues. La commande `egrep` sera équivalente à `grep -E` et vous pourrez ainsi cibler un chiffre par `[:digit:]` au lieu de `[0-9]`.
Par exemple :

```
pi@raspberry:~ $ grep -r -E "([[:digit:]]\+, [[:digit:]]\+) *
Public/Rep/SousRep: print('Tirage aléatoire :', random.randint(1, 10))
python_games/memorypuzzle_obfuscated.py:    b = pygame.display.set_mode((640, 480))
...
```

CONCLUSION

Nous avons vu dans cet article les outils proposés par Linux pour retrouver des informations sur votre disque dur via le shell. Que ceci ne soit pas un encouragement à ne pas organiser scrupuleusement vos fichiers, mais au moins, si vous devez rechercher des données, vous saurez par où commencer... ▌

SURVEILLER SON SYSTÈME AVEC MONIT

Romain PELISSE

La supervision d'un système en production demeure un enjeu aussi complexe qu'essentiel. Il existe de nombreuses solutions, très complètes, de supervision, mais la plupart adoptent une approche centralisée, qui demande l'utilisation de ressources dédiées. Aujourd'hui, nous étudierons une approche alternative, une solution de supervision décentralisée, nommée Monit.

La problématique de la supervision de systèmes informatiques est un thème récurrent dans l'industrie. En effet, une absence ou une supervision trop faible peut se révéler néfaste, voire dangereuse pour le système d'information. Mais, à l'inverse une supervision trop exhaustive remonte de nombreuses informations peu pertinentes qui, par leurs présences, noient les véritables signaux d'incidents.

À ceci s'ajoute le fait que la supervision est elle-même une application qui a besoin d'être maintenue, installée, et elle aussi supervisée ! Et surtout, elle doit être construite et déployée de manière à tenir la montée en charge du système d'information, sans compromettre ses performances. Ce qui se révèle souvent problématique, car ces solutions sont pour la plupart centralisées et ce centre névralgique est progressivement mis sous pression par l'augmentation de nombre de systèmes à surveiller et d'informations à collecter.

C'est avec tout ceci en tête que nous allons aujourd'hui aborder l'outil Monit, qui est une solution de supervision légère, construite pour être déployée directement sur le système à superviser.

Cette stratégie peut sembler de prime abord contraire à l'idée même que l'on se fait de la supervision, mais se révèle extrêmement adaptée aux besoins d'un environnement distribué comme celui d'un « *cloud* ». Le système est ainsi déployé avec son propre système de surveillance qui lui est dédié. Il est donc par essence très facile de tenir la montée en charge, puisque l'ajout d'instance n'augmente pas la charge sur le système de supervision. Tout nouveau système déployé au sein du système d'information vient avec sa propre instance de Monit, et n'ajoute donc pas de coût supplémentaire en ressources sur l'infrastructure logicielle.

En outre, dans le contexte de cet article, l'utilisation de Monit va nous permettre d'évoquer de nombreux aspects de la supervision, de manière pratique, sans nécessiter le déploiement, la configuration et l'installation de nombreuses machines (virtuelles ou non) ou d'autres artifices. On peut simplement utiliser Monit sur son système local ou même au sein d'un simple conteneur Docker. C'est cette dernière solution que nous allons utiliser ici afin de disposer d'un environnement isolé.

1. MISE EN PLACE

Démarrage d'un conteneur Docker

Afin de nous donner un cadre de travail propre, isolé du reste du système et reproduisant un environnement de production, nous allons utiliser un conteneur Docker. Si le lecteur n'est pas familier de cette technologie, il peut simplement ignorer cette section et passer à la suivante. En effet, toutes les opérations effectuées par la suite au sein du conteneur Docker peuvent très bien s'effectuer directement sur le système utilisé. L'utilisation de Docker ici est essentiellement à des fins de simplicité de reproduction et d'isolation par rapport au reste du système.

L'image Docker que nous allons utiliser ici est l'image centos/systemd. Cette image nous permet en effet de bénéficier d'un système équivalent à Red Hat Enterprise Linux (RHEL) dans sa version 7. Ceci nous permettra d'installer et de tester Monit comme sur un système de production utilisant RHEL 7 comme système d'exploitation.

Avant de lancer un conteneur à partir de cette image, il est important de noter que nous allons avoir besoin, au sein de celui-ci, d'un gestionnaire de services afin de démarrer celui associé à Monit. Dans le cas de RHEL7, il s'agit naturellement de Systemd. Cependant, par défaut, Docker n'autorise pas l'exécution d'un tel processus en son sein.

Afin d'autoriser le conteneur à exécuter Systemd, nous allons ajouter deux options supplémentaires au lancement du conteneur :

- l'option `--privileged` qui indique que le conteneur peut effectuer ce type d'opération ;

- l'option `-v /sys/fs/cgroup:/sys/fs/cgroup:ro` qui donne accès à `cgroup` au conteneur Docker.

```
$ sudo docker run -ti --rm -v /sys/fs/cgroup:/ sys/fs/cgroup:ro --privileged
-d centos/systemd /usr/sbin/init
4244b8fad756eeeb51d3ba74cf5b7a14dcd2a106a25ac52980de89b79c8c9958
```

Le lecteur attentif notera aussi l'utilisation de l'option `--rm` qui indique que le conteneur devra être immédiatement effacé une fois son exécution terminée.

Une fois le conteneur ainsi démarré, il ne reste plus qu'à nous y connecter afin de disposer de notre environnement de travail, équivalent à RHEL 7 et isolé du reste du système :

```
$ sudo docker exec -ti
4244b8fad756eeeb51d3ba74cf5b7a14dcd2a106a25ac52980de89b79c8c9958 /bin/
bash
[root@4244b8fad756 /]#
```

Installation de Monit

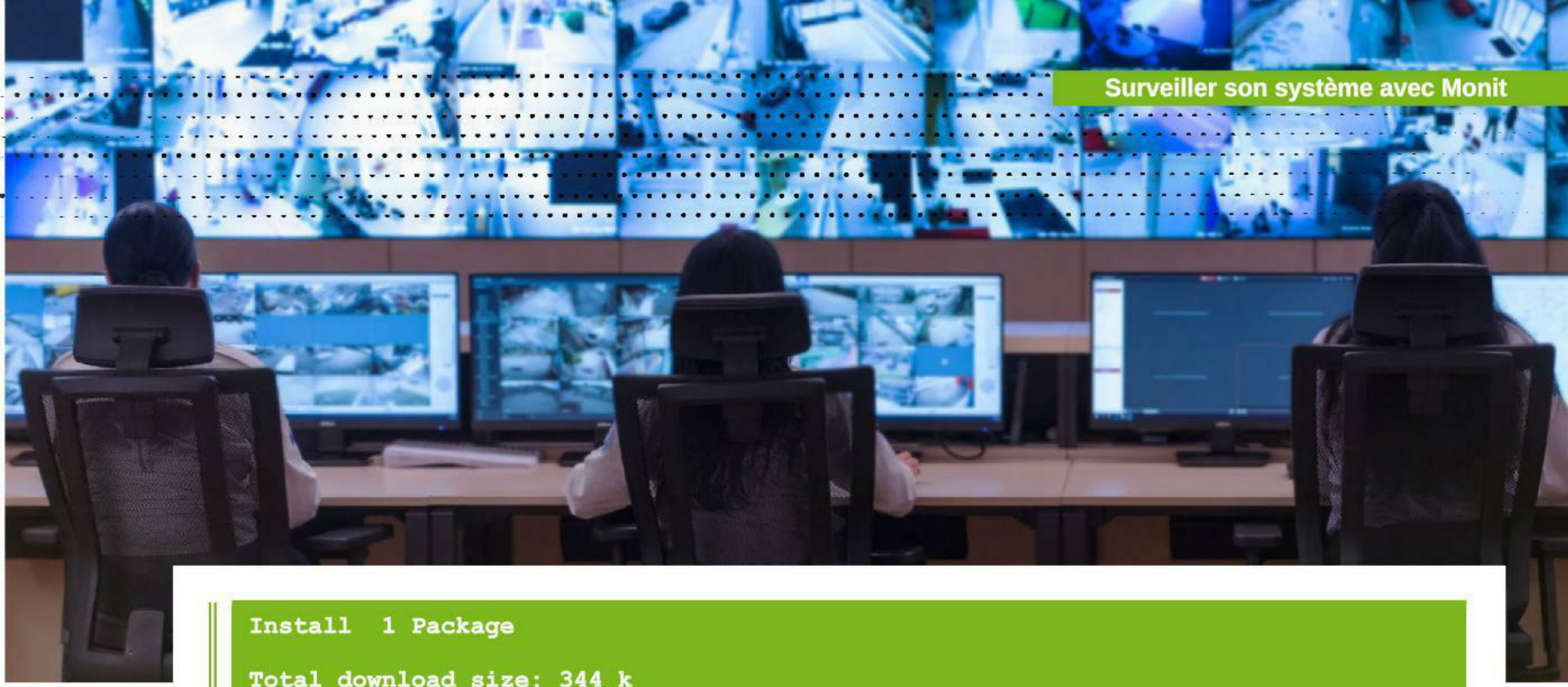
Pour installer Monit sur le système, nous avons besoin d'installer des dépôts logiciels supplémentaires. En effet, Monit n'est pas un logiciel fourni par CentOS. Il nous donc installer les dépôts supplémentaires « *Extra Packages for Enterprise Linux* » (EPEL) , ce qui se fait sans difficulté à l'aide de la commande suivante :

```
# yum install https://dl.fedoraproject.org/pub/epel/epel-release-
latest-8.noarch.rpm
```

Ces dépôts supplémentaires installés, il ne nous reste plus qu'à lancer l'installation de Monit :

```
# yum install monit
Updating Subscription Management repositories.
Unable to read consumer identity
This system is not registered to Red Hat Subscription Management. You can use
subscription-manager to register.
Last metadata expiration check: 0:00:15 ago on Tue Aug 25 16:09:29 2020.
Dependencies resolved.
================================================================================
================================================================================
 Package                                                Architecture
Version                                                       Repository
Size
================================================================================
================================================================================
Installing:
 monit                                                  x86_64
5.26.0-1.el8                                                  epel
344 k

Transaction Summary
================================================================================
================================================================================
```



```
Install  1 Package

Total download size: 344 k
Installed size: 1.1 M
Is this ok [y/N]: y
Downloading Packages:
monit-5.26.0-1.el8.x86_64.rpm
                                    1.0 MB/s | 344 kB     00:00
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
Total
                                                 337 kB/s | 344 kB
00:01
warning: /var/cache/dnf/epel-fafd94c310c51e1e/packages/monit-5.26.0-1.el8.x86_64.
rpm: Header V3 RSA/SHA256 Signature, key ID 2f86d6a1: NOKEY
Extra Packages for Enterprise Linux 8 - x86_64
                      1.6 MB/s | 1.6 kB     00:00
Importing GPG key 0x2F86D6A1:
 Userid     : "Fedora EPEL (8) <epel@fedoraproject.org>"
 Fingerprint: 94E2 79EB 8D8F 25B2 1810 ADF1 21EA 45AB 2F86 D6A1
 From       : /etc/pki/rpm-gpg/RPM-GPG-KEY-EPEL-8
Is this ok [y/N]: y
Key imported successfully
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
  Preparing        :                                              1/1
  Installing       : monit-5.26.0-1.el8.x86_64                    1/1
  Running scriptlet: monit-5.26.0-1.el8.x86_64                    1/1
  Verifying        : monit-5.26.0-1.el8.x86_64                    1/1

Installed products updated.

Installed:
  monit-5.26.0-1.el8.x86_64


Complete!
```

Avant d'aller plus loin, vérifions que Monit a en effet été bien installé :

```
# monit -V
This is Monit version 5.26.0
Built with ssl, with ipv6, with compression, with pam and with large
files
Copyright (C) 2001-2019 Tildeslash Ltd. All Rights Reserved.
```

Il ne nous reste plus qu'à démarrer le service qui lui est associé :

```
# systemctl start monit
[root@5efaf598c2df /]# systemctl status monit
● monit.service - Pro-active monitoring utility for unix systems
   Loaded: loaded (/usr/lib/systemd/system/monit.service; disabled;
vendor preset: disabled)
   Active: active (running) since Tue 2020-08-25 17:51:44 UTC; 3s ago
 Main PID: 204 (monit)
   CGroup: /docker/5efaf598c2df96374ef2e5e2843e49b9a95733b06f42d6836e12
790b42e38af3/system.slice/monit.service
           └─204 /usr/bin/monit -I

Aug 25 17:51:44 5efaf598c2df systemd[1]: Started Pro-active monitoring
utility for unix systems.
Aug 25 17:51:44 5efaf598c2df monit[204]: New Monit id:
2a9ea7b4e56ad9eea17653d029c20013
Aug 25 17:51:44 5efaf598c2df monit[204]: Stored in '/root/.monit.id'
Aug 25 17:51:44 5efaf598c2df monit[204]: Starting Monit 5.26.0 daemon
with http interface at [localhost]:2812
Aug 25 17:51:44 5efaf598c2df monit[204]: '5efaf598c2df' Monit 5.26.0
started
```

On peut déjà demander à Monit de nous faire un rapport sur l'état du système :

```
# monit status
Monit 5.26.0 uptime: 0m

System 'a22f12bb6c71'
  status                            OK
  monitoring status                 Monitored
  monitoring mode                   active
  on reboot                         start
  load average                      [0.81] [0.69] [0.52]
  cpu                               1.3%us 1.1%sy 0.1%wa
  memory usage                      6.0 GB [19.2%]
  swap usage                        22.2 MB [0.1%]
  uptime                            2d 1h 37m
  boot time                         Mon, 24 Aug 2020 07:23:13
  data collected                    Wed, 26 Aug 2020 09:00:02
```

Passons maintenant à la configuration d'alertes afin que Monit puisse nous donner un peu plus d'informations.

2. CONFIGURER UNE PREMIÈRE ALERTE

Surveiller l'utilisation d'une partition

Pour définir une alerte, il suffit d'ajouter un fichier de configuration au sein du répertoire `/etc/monit.d/`. La syntaxe utilisée par Monit est très explicite et transparente et se passe donc d'explications détaillées. Le lecteur prendra juste soin de noter dans le fichier présenté ci-dessous que les parties « variables » de la configuration ont été soulignées afin de les distinguer des instructions spécifiques à Monit :

```
# cat /etc/monit.d/disk
check device disk with path /
    if SPACE usage > 80 then alert
```

Une fois ce fichier proprement déployé, il suffit de recharger Monit pour que la surveillance de la ressource soit mise en place :

```
# monit reload
Reinitializing monit daemon
```

Ceci fait, on peut vérifier immédiatement que l'alerte fonctionne, en invoquant à nouveau la commande `monit status` :

```
# monit status
Monit 5.26.0 uptime: 4m

Filesystem 'disk'
  status                            Resource limit matched
  monitoring status                 Monitored
  monitoring mode                   active
  on reboot                         start
  filesystem type                   overlay
  filesystem flags                  rw,seclabel,relatime,lowerdir=/var/
lib/docker/overlay2/l/MCA3UGADX35RJDTXUTDHOJOBLQ:/var/lib/docker/
overlay2/l/G4XLJJIGLWUPIESJP6W5ZWM2HK:/var/lib/docker/overlay2/l/VIYOF2
IIW2W72IAEWTB4WROWG4,upperdir=
  permission                        755
  uid                               0
  gid                               0
  block size                        4 kB
  space total                       53.8 GB (of which 5.1% is reserved for
root user)
  space free for non superuser      6.2 GB [11.6%]
  space free total                  9.0 GB [16.7%]
  inodes total                      3604480
  inodes free                       2947624 [81.8%]
  data collected                    Wed, 26 Aug 2020 09:04:32

System 'a22f12bb6c71'
  status                            OK
  monitoring status                 Monitored
  monitoring mode                   active
```

```
on reboot                       start
load average                    [0.18] [0.37] [0.42]
cpu                             1.0%us 1.0%sy 0.0%wa
memory usage                    6.0 GB [19.3%]
swap usage                      22.2 MB [0.1%]
uptime                          2d 1h 41m
boot time                       Mon, 24 Aug 2020 07:23:13
data collected                  Wed, 26 Aug 2020 09:04:32
```

Surveiller l'accessibilité d'un service externe

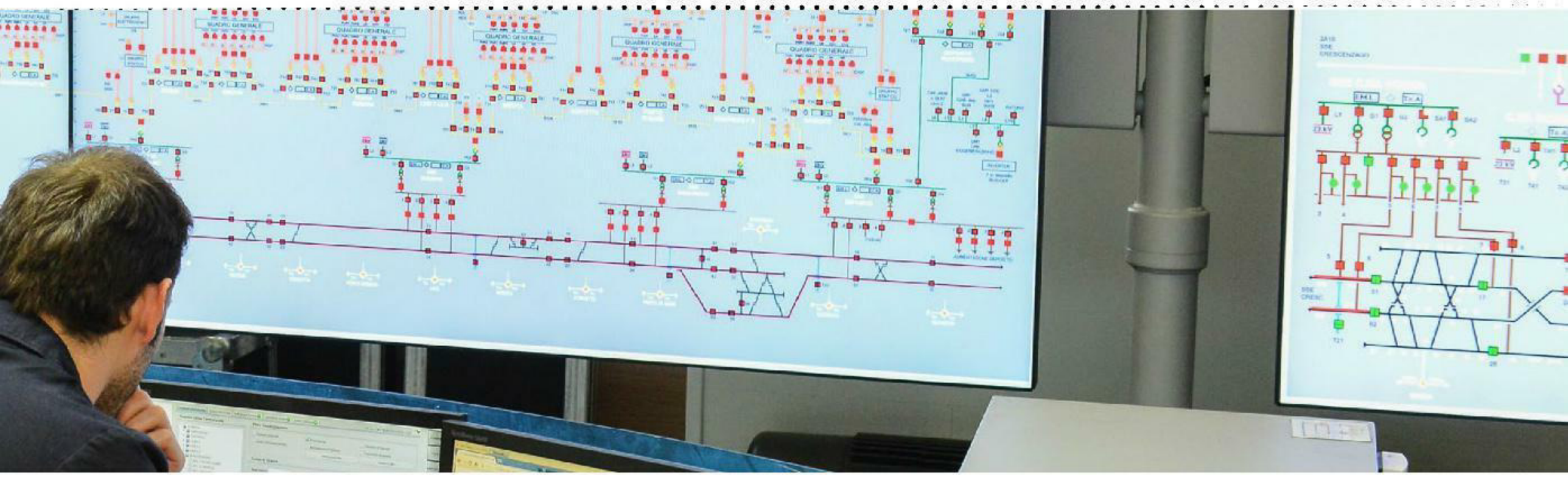
Si Monit permet déjà de surveiller le système en lui-même, comme nous l'avons illustré ci-dessus (en surveillant une partition), l'outil permet aussi de surveiller l'accessibilité à des systèmes distants. De nos jours, c'est un point critique, car il est rare qu'un service déployé sur un système n'ait pas de dépendances vers d'autres.

À titre d'exemple, imaginons que le système supervisé soit destiné à l'intégration continue de logiciels Java. La machine va donc régulièrement accéder aux miroirs du projet Maven afin de télécharger les dépendances (sous forme d'archive Jar) de ces logiciels.

Ainsi, pour surveiller et garantir le bon fonctionnement de la plateforme, il est essentiel de vérifier que le système soit toujours en mesure d'atteindre les serveurs associés au nom DNS `repo.maven.apache.org`. Monit va nous permettre de surveiller cette connectivité à deux niveaux. Le premier se situe en bas de la couche réseau et utilise le protocole ICMP pour effectuer régulièrement des « *ping* » vers l'hôte afin de vérifier qu'il soit toujours accessible. Le second se situe au sein de la couche applicative (HTTP) et opère en vérifiant qu'une URL de test est bien accessible. Ceci nous permettra de disposer d'une stratégie fine de surveillance du service distant — à la fois d'un point de vue réseau et du point de vue service.

Ci-dessous une première configuration qui va nous permettre de mettre en place la surveillance de la connectivité au système distant. Comme précédemment, les parties variables de la configuration ont été soulignées afin de bien les distinguer de la syntaxe de Monit :

```
# cat /etc/monit.d/repo.maven.apache.org
check host repo1.maven.org with address repo1.maven.org
    if failed ping with timeout 15 seconds then alert
```



Après avoir rechargé la configuration de Monit, on peut vérifier que la surveillance de la ressource a en effet été mise en place :

```
# monit status
Monit 5.26.0 uptime: 29m

Remote Host 'repo1.maven.org'
  status                            OK
  monitoring status                 Monitored
  monitoring mode                   active
  on reboot                         start
  ping response time                135.281 ms
  data collected                    Wed, 26 Aug 2020 09:29:16
...
```

Allons maintenant un peu plus loin et vérifions le bon fonctionnement du service associé, soit la disponibilité d'un dépôt Maven distant. Ajoutons le fichier de configuration suivant au répertoire **/etc/monit.d/** :

```
# cat /etc/monit.d/https_repo.maven.apache.org
check host maven-central with address repo.maven.apache.org
  if failed
 port 443
 protocol HTTPS request /maven2/ with timeout 12 seconds then alert
```

Une fois la configuration de Monit à nouveau rechargée, on peut vérifier que la surveillance est bien mise en place :

```
# monit status
...
Remote Host 'maven-central'
  status                            OK
  monitoring status                 Monitored
  monitoring mode                   active
  on reboot                         start
  port response time                447.337 ms to repo.maven.apache.org:443/
maven2/ type TCP/IP using TLS (certificate valid for 394 days) protocol
HTTP
  data collected                    Wed, 26 Aug 2020 09:41:33
...
```

ATTENTION !

Le lecteur attentif doit se poser la question depuis le début de cet article : si la solution de supervision est déployée sur le système surveillé lui-même, comment peut-on être notifié en cas d'incident aboutissant à l'arrêt complet de ce système ? La réponse à cette question vient avec la capacité de Monit à surveiller les systèmes distants dont sa cible dépend. Si l'instance Monit déployée sur le système défaillant ne sera bien sûr pas en mesure de signaler l'indicent, l'ensemble des services qui dépend de ce système indiqueront sa soudaine indisponibilité.

Cette approche a aussi l'avantage de permettre de différencier aisément un système indisponible d'un système temporairement impossible à atteindre (par exemple, suite à une avarie réseau).

3. SUPERVISER ET REDÉMARRER AUTOMATIQUEMENT LES SERVICES

Monit peut aller encore plus loin que la simple surveillance. En effet, le logiciel peut même effectuer des opérations de redémarrage si un service local devient indisponible. Nous allons en faire la démonstration en mettant en place un service, en l'occurrence le serveur Nginx, que nous allons placer sous le contrôle de Monit.

Commençons par installer le logiciel nécessaire :

```
# yum install -y nginx
```

Ceci fait, démarrons le service :

```
# systemctl start nginx
```

Vérifions qu'il a démarré sans erreur avant d'aller plus loin :

```
# systemctl status nginx
● nginx.service - The nginx HTTP and reverse proxy server
   Loaded: loaded (/usr/lib/systemd/system/nginx.service; disabled; vendor
preset: disabled)
   Active: active (running) since Wed 2020-08-26 09:11:50 UTC; 43min ago
  Process: 377 ExecStart=/usr/sbin/nginx (code=exited, status=0/SUCCESS)
  Process: 376 ExecStartPre=/usr/sbin/nginx -t (code=exited, status=0/SUCCESS)
  Process: 375 ExecStartPre=/usr/bin/rm -f /run/nginx.pid (code=exited,
status=0/SUCCESS)
 Main PID: 378 (nginx)
   CGroup: /docker/
a22f12bb6c710767048ebb6f81addd6d7f88da974bd1eac4de2facc28a29a499/system.slice/
nginx.service
           ├─378 nginx: master process /usr/sbin/nginx
           ├─379 nginx: worker process
           ├─380 nginx: worker process
           ├─381 nginx: worker process
           ├─382 nginx: worker process
```

```
             ├─383 nginx: worker process
             ├─384 nginx: worker process
             ├─385 nginx: worker process
             └─386 nginx: worker process

Aug 26 09:11:50 a22f12bb6c71 systemd[1]: Starting The nginx HTTP and reverse
proxy server...
Aug 26 09:11:50 a22f12bb6c71 nginx[376]: nginx: the configuration file /etc/
nginx/nginx.conf syntax is ok
Aug 26 09:11:50 a22f12bb6c71 nginx[376]: nginx: configuration file /etc/nginx/
nginx.conf test is successful
Aug 26 09:11:50 a22f12bb6c71 systemd[1]: Started The nginx HTTP and reverse
proxy server.
```

Si le serveur est démarré et s'exécute en mémoire, rien ne garantit qu'il soit fonctionnel. Ainsi, nous allons vérifier que Nginx répond bien aux requêtes qu'on lui envoie à l'aide d'une simple requête **curl** :

```
# curl -I http://127.0.0.1/
HTTP/1.1 403 Forbidden
Server: nginx/1.16.1
Date: Wed, 26 Aug 2020 09:55:46 GMT
Content-Type: text/html
Content-Length: 153
Connection: keep-alive
```

Mettons en place une surveillance de ce service comme nous l'avons fait plus haut avec Maven Central :

```
# cat /etc/monit.d/nginx
check host nginx with address localhost
  if failed
 port 80
 protocol HTTP request /404.html with timeout 12 seconds then alert
```

Voilà Nginx est installé, démarré et surveillé par Monit. Nous allons maintenant aller plus loin en indiquant à Monit de redémarrer le serveur s'il devient indisponible !

Modifions ainsi la configuration de notre ressource supervisée afin d'y ajouter les instructions suivantes :

```
check process nginx with pidfile /run/nginx.pid
    start program = "/bin/systemctl start nginx start"
    stop program  = "/bin/systemctl stop nginx stop"

if failed host localhost port 80
    protocol HTTP request /404.html with timeout 12 seconds then alert
```

Après avoir rechargé la configuration de Monit, l'outil est désormais capable de démarrer et d'éteindre le service :

```
# monit stop nginx
# monit status nginx
Monit 5.26.0 uptime: 1h 17m
```

```
Process 'nginx'
  status                            Not monitored
  monitoring status                 Not monitored
  monitoring mode                   active
  on reboot                         start
  data collected                    Wed, 26 Aug 2020 10:16:25

# monit start nginx
# monit status nginx
Monit 5.26.0 uptime: 1h 17m

Process 'nginx'
  status                            OK
  monitoring status                 Monitored
  monitoring mode                   active
  on reboot                         start
  pid                               445
  parent pid                        1
  uid                               0
  effective uid                     0
  gid                               0
  uptime                            0m
  threads                           1
  children                          8
  cpu                               -
  cpu total                         -
  memory                            0.0% [3.0 MB]
  memory total                      0.2% [56.1 MB]
  security attribute                system_u:system_r:spc_t:s0
  port response time                -
  data collected                    Wed, 26 Aug 2020 10:17:00
...
```

Monit va encore plus loin, car si l'on interrompt le service, l'outil de supervision constate qu'il est devenu indisponible et le lance à nouveau !

```
# systemctl stop nginx
# monit status
Monit 5.26.0 uptime: 1h 19m

...

Process 'nginx'
  status                            OK
  monitoring status                 Monitored
  monitoring mode                   active
  on reboot                         start
  pid                               468
  parent pid                        1
  uid                               0
  effective uid                     0
  gid                               0
```

```
  uptime                        0m
  threads                       1
  children                      8
  cpu                           0.0%
  cpu total                     0.0%
  memory                        0.0% [3.1 MB]
  memory total                  0.2% [56.1 MB]
  security attribute            system_u:system_r:spc_t:s0
  port response time            0.312 ms to localhost:80/404.html type
TCP/IP protocol HTTP
  data collected                Wed, 26 Aug 2020 10:19:02
```

ATTENTION !

Le lecteur prendra soin de noter que valider le bon fonctionnement de Nginx en vérifiant la disponibilité de la page 404 (dédiée aux erreurs de type ressources demandées absentes) n'est pas suffisant. Dans le cadre d'un déploiement en production, il est fortement recommandé de compléter cette surveillance en vérifiant aussi la disponibilité d'URL correspondant à des applications déployées au sein du serveur.

Grâce à cette approche, non seulement notre serveur Nginx est supervisé, mais notre système d'information est capable de se remettre immédiatement, sans intervention humaine, d'un incident aboutissant à la mise en indisponibilité de ce service !

4. DÉPLOYER UN SCRIPT DE SUPERVISION

Si Monit offre déjà de nombreuses fonctionnalités prêtes à l'emploi, il est aussi fort d'appréciable que l'outil permette de déployer son propre type d'alerte sous forme de scripts. Ainsi, si aucun des mécanismes mis à disposition par Monit ne convient à son cas d'utilisation, on peut simplement concevoir un script et l'intégrer à l'outil de supervision.

Supposons que l'on héberge un service sur le système sous forme de conteneur Docker (par exemple pour héberger, de manière isolée du reste du système, une base de données PostgreSQL) :

```
$ docker ps
CONTAINER ID        IMAGE                                                COMMAND
CREATED             STATUS              PORTS               NAMES
c4a1e1beebc1        registry.access.redhat.com/
rhel7.8   "/usr/sbin/init"    8 hours ago         Up 8 hours
postgresqldb
```

On souhaite disposer d'une alerte si ce conteneur n'est plus présent parmi les conteneurs en cours d'exécution. Voici un exemple de script qui permet de détecter un tel incident :

```
#!/bin/bash
set -euo pipefail

readonly CONTAINER_NAME=${CONTAINER_NAME:-'postgresqldb'}

docker inspect -f '{%raw%}{{.State.Running}}{%endraw%}' "${CONTAINER_
NAME}" 2>/dev/null | grep -e 'true' -q
```

S'il n'y a aucun conteneur au nom indiqué en cours d'exécution, le script termine avec une valeur de statut différente de zéro, indiquant ainsi à l'appelant qu'il a constaté une avarie. C'est grâce à ceci que Monit sera en mesure de déterminer que la ressource supervisée est, en effet, dans un état invalide.

Voyons maintenant comment configurer cette alerte au sein de Monit :

```
$ cat /etc/monit.d/pgsql-docker
check program pgsql-docker with path /etc/monit/scripts/pgsql-docker
    with timeout 20 seconds
    if status != 0 then alert
```

5. POUR ALLER PLUS LOIN

Cette brève introduction à Monit nous a déjà permis de faire un tour exhaustif des fonctionnalités de base de Monit. Cependant, il dispose de nombreuses autres fonctionnalités que lecteur peut explorer une fois les éléments décrits dans cet article acquis.

Par exemple, en configurant un serveur SMTP, il est possible d'être notifié par e-mail si une ressource supervisée devient indisponible ou en état d'erreur :

```
# Set the mail server to us to send alerts
set mailserver smtp.example.com port 25

# Define Monit's mail structure
set mail-format {
  from: belaran@example.com
  subject: Monit - state of service $SERVICE on $HOST changed
  message:
  $DESCRIPTION
}
set alert belaran@example.com
```

Le lecteur prendra soin de noter que Monit définit des variables telles que `$SERVICE` ou `$HOST`, utilisées ci-dessus, afin de permettre de facilement configurer le contenu du message.

Une autre fonctionnalité très appréciable de Monit est son interface web. En fait, une fois le service démarré, Monit peut aussi être accédé via une petite application web. Attention néanmoins à ne pas laisser l'accès à ce site ouvert, car n'importe quelle personne qui s'y connecte peut arrêter ou redémarrer les services placés sous le contrôle de Monit !

Le lecteur prendra soin de noter aussi que les fichiers de configurations de Monit sont très faciles à transformer en fichier patron (« *template* » pour des outils tels qu'Ansible, ce qui permet aisément d'automatiser son installation.

En outre, la configuration passant intégralement par une arborescence de fichiers plats, il est aussi très facile de la placer au sein d'un dépôt Git.

CONCLUSION

Monit est une solution de supervision simple, légère et élégante. Son exécution, sur un serveur dédié au déploiement de l'application web, ne consomme que peu de ressources. Avec une supervision déportée sur le système cible, sa montée en charge n'est plus un réel problème : on dispose d'autant d'instances de Monit que de systèmes déployés. En outre, la richesse des alertes prédéfinies, ajoutées à la capacité de déployer des scripts de supervision personnalisés assure que Monit sera à même de surveiller tout aspect critique d'un système.

Les capacités « réactives » de Monit, qui lui permettent de redémarrer, au besoin, un service en panne, apportent une réelle plus-value en comparaison à une solution de supervision « classique ». En effet, le système sous le contrôle de Monit peut ainsi rester fonctionnel, avec une période d'interruption de service minimum, même après un incident. ▌

DOSSIER

LES INDISPENSABLES

SÉCURISEZ VOS SERVEURS

ET VOTRE RÉSEAU LOCAL

AU SOMMAIRE :

2020 aura été une année marquante pour nos vies et nos sociétés. Il aura fallu se réinventer, trouver des solutions à des situations exceptionnelles. Dans les entreprises, l'Éducation ou la Santé, la mobilisation des ressources informatiques aura été maximale. Nos infrastructures auront ployé, tangué, parfois presque craqué, mais au final, cela aura tenu.

Au moment du bilan, les entreprises, les associations, mais aussi les citoyens auront pu constater que la prévoyance, l'anticipation et surtout l'adaptabilité de leurs ressources informatiques auront été des qualités nécessaires pour faire face à des situations inédites, voire extraordinaires.

Aussi, quand on m'a demandé de concevoir ce hors-série sur la sécurisation d'un serveur, j'ai souhaité porter une attention particulière à la résilience et l'évolutivité de notre infrastructure, des sujets pas toujours abordés quand on parle de sécurité informatique.

Comme nous le verrons au fil des pages de ce hors-série, nous parlerons plus souvent d'infrastructure et de services plutôt que de serveur individuel. Ne vous alarmez cependant pas : l'idée n'est bien entendu pas de vous emmener sur la conception et la sécurisation d'une infrastructure hautement disponible multi centres de données ;-)

Nous allons simplement travailler ensemble à bâtir des services sécurisés. Cela nous amènera notamment à mettre en place des composants autour de notre serveur. Ils permettront d'ajouter des fonctionnalités de réseau privé entre serveurs, de surveillance, de gestion de logs ou de sauvegarde.

Ces fonctions support sont autant d'atouts indispensables dans un contexte de sécurité informatique. Elles améliorent la détection préventive, facilitent l'investigation en cas de suspicion d'attaque ou la restauration à un état sain après une attaque réussie par un rançongiciel.

De son côté, la conception évolutive de notre architecture consistera principalement à disposer d'un serveur d'administration en plus du serveur principal. Les fonctions de sécurité comme les sauvegardes, le VPN ou la centralisation des logs seront localisées sur ce serveur d'administration tandis que le serveur principal n'hébergera que des services applicatifs. Nous pourrons donc ajouter d'autres serveurs applicatifs à l'avenir sans avoir aucune adhérence avec les fonctions sécurité.

Côté implémentation, nous nous reposerons bien entendu sur des logiciels libres tant serveurs que réseaux. Ils nous donneront une souplesse de paramétrage et une capacité de réaction inégalable. Ils permettent notamment de ne pas être soumis à des services en ligne payants qui, le plus souvent, exposent nos données d'entreprises ou privées à des risques supplémentaires.

Bien entendu, nous nous concentrerons aussi sur la sécurisation classique de nos serveurs. Nous couvrirons ainsi le choix du fournisseur de cloud, la sécurisation de l'installation et du socle système du serveur ou la mise en place de la configuration réseau et des services associés. Nous ferons aussi un focus sur la supervision sécurité de nos services et une plongée dans la configuration d'OpenSSH.

Vous ne trouverez pas par contre de services applicatifs s'exécutant dans des conteneurs. C'est un choix volontaire. Je ne souhaitais pas introduire un niveau de complexité supplémentaire avec ces technologies qui aurait nuit à la facilité d'assimilation de l'ensemble de cette thématique de sécurisation de serveurs. En effet, la mise en œuvre de conteneurs de manière bien imbriquée avec la gestion de logs, les possibilités d'investigation numérique ou de VPN est un sujet en soi qui mérite une couverture dédiée.

J'espère que cette vision élargie de la sécurité, mêlant réflexion sur l'architecture, sécurisation classique, supervision et capacité de réponse à incident, vous amènera les connaissances nécessaires à l'amélioration de la sécurité de vos serveurs personnels ou d'entreprise. ▌

Christophe BROCAS

DÉFINISSEZ L'ARCHITECTURE DE VOS SERVEURS ET INSTALLEZ-LES

Christophe BROCAS

Dans cet article, nous réfléchirons aux besoins de sécurité auxquels nos serveurs devront répondre. Il sera d'ailleurs plus question d'architecture que de serveur personnel. Pourquoi cela ? Car nos besoins vont à coup sûr évoluer dans le temps. L'approche la plus pérenne sera donc de mener une réflexion basée sur des services et non sur un serveur unique. Nous allons aussi nous attacher à assurer la résilience de nos services de base. Nos choix d'architecture auront pour objectif de pouvoir mieux détecter, contrer et éventuellement réparer les dommages causés par une attaque informatique. Nous pourrons par exemple restaurer nos services si un attaquant réussissait à prendre le contrôle du serveur. Notre plan de bataille commencera par la définition des grandes lignes de notre infrastructure, puis par la sélection de nos fournisseurs. Nous déploierons ensuite le serveur avec un premier palier de sécurisation système.

1. ARCHITECTURE PHASE 1 : LE DNS

Choix de localisation du service DNS hébergeant notre domaine

Lorsque vous souhaitez mettre en place un serveur personnel ou d'une petite à moyenne entreprise (PME), l'une des premières actions est de lui assigner un nom DNS. Pour cela, nous aurons besoin d'un nom de domaine DNS. Pour cet article, nous choisirons **mondomaine.fr**.

Nous allons donc gérer **mondomaine.fr** au travers d'un service DNS.

Tout d'abord, quelques rappels sur le service DNS :

- Son rôle principal est de publier des noms de machines ou de services sous le domaine **mondomaine.fr** ainsi que l'adresse IP associée à leur nom.
- Pour assurer sa continuité de service, il est généralement associé à un service DNS secondaire qui a pour mission de pouvoir répondre aux requêtes DNS des clients de la même manière que le ferait le service DNS primaire. La résolution DNS de **mondomaine.fr** est ainsi toujours assurée dans le cas où le service primaire est incapable de répondre. Les raisons de cette indisponibilité peuvent être très diverses : perte d'accessibilité réseau suite à une panne ou à une attaque par déni de service, incident sur le serveur hébergeant le service primaire, etc.

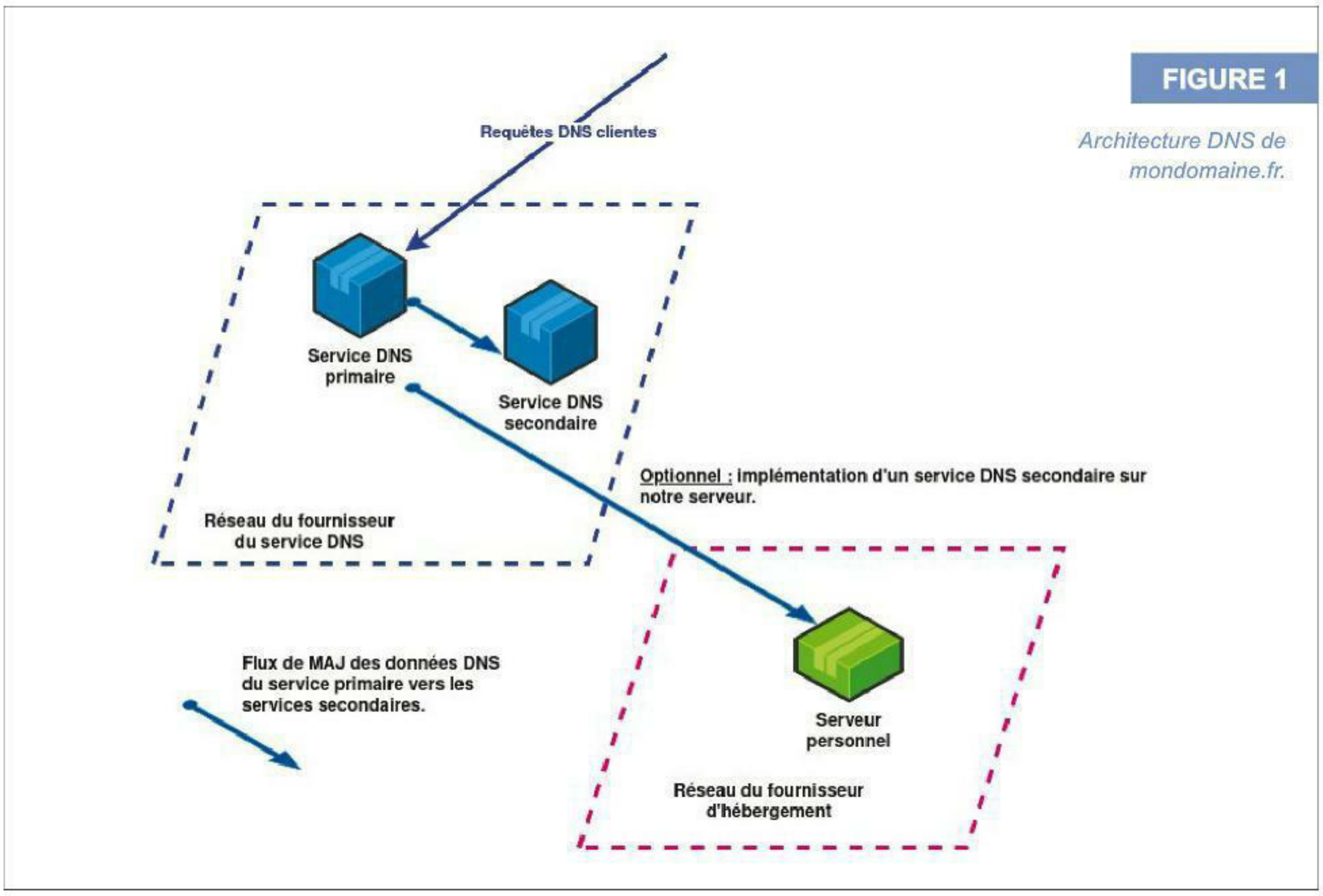


FIGURE 1

Architecture DNS de mondomaine.fr.

Ainsi, pour préserver notre service DNS primaire, nous n'installerons pas ce service sur notre serveur. De même, nous ne l'installerons pas non plus sur une infrastructure dépendant du prestataire d'hébergement de notre serveur.

Les bonnes pratiques que nous allons appliquer seront les suivantes :

- Achat : nous achèterons le nom de domaine **mondomaine.fr** chez un bureau d'enregistrement qui n'est pas notre prestataire d'hébergement. De même, notons que nous avons choisi un nom de domaine en .fr, car les litiges juridiques le concernant sont arbitrés devant la justice française.
- Administration et architecture : nous administrerons **mondomaine.fr** depuis l'infrastructure de ce bureau d'enregistrement. Une fois que nous nous sommes assurés de sa résilience, on peut valider que le DNS secondaire est aussi assuré par cette infrastructure.
- Option : nous pouvons mettre en place sur notre serveur un second service secondaire de DNS qui aura ainsi une copie des données du serveur primaire.

Notre architecture DNS ressemble désormais à la figure 1, page précédente.

Quelques bonnes pratiques DNS, mais pas que …

Supposons que nous enregistrons **mondomaine.fr** chez Gandi, un des plus anciens bureaux d'enregistrement de noms de domaines français. Quand nous interrogeons les services DNS associés à notre nom de domaine, nous obtenons les informations suivantes :

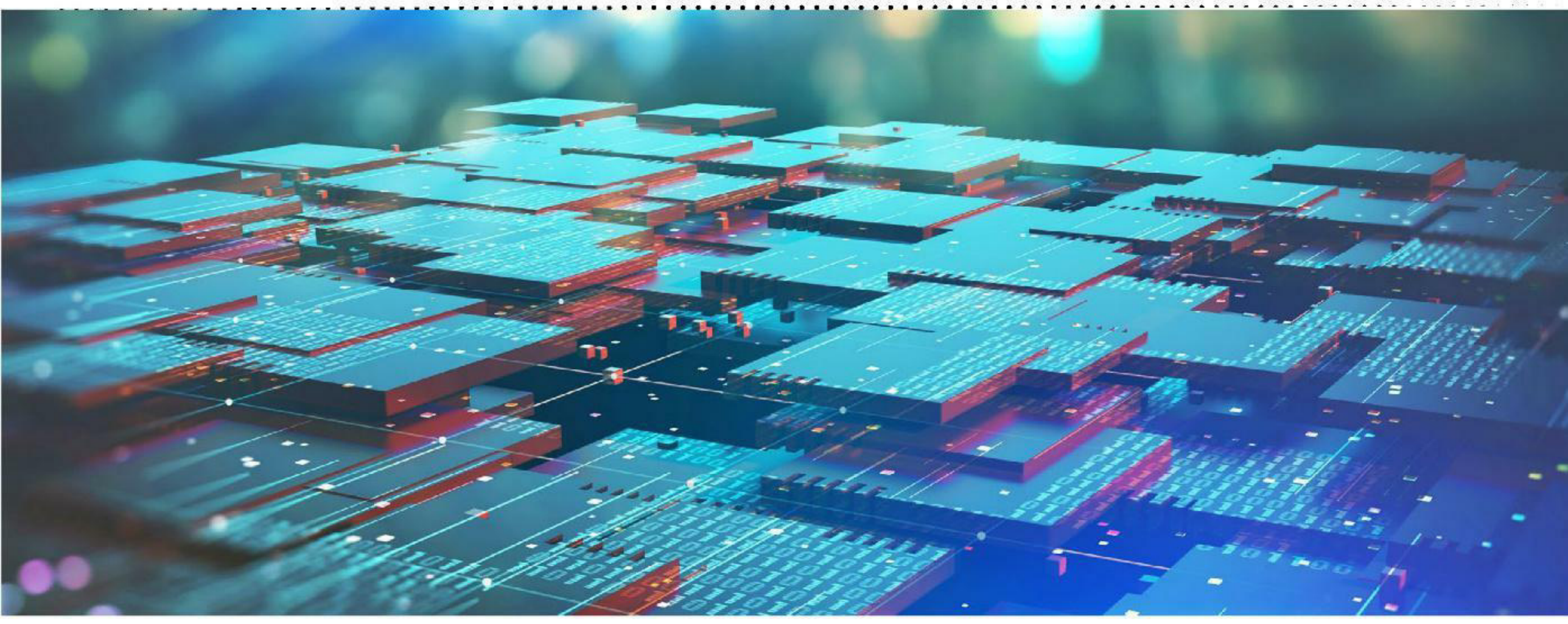
```
$ dig ns mondomaine.fr


;; QUESTION SECTION:
;mondomaine.fr.   IN NS

;; ANSWER SECTION:
mondomaine.fr.  10800 IN NS ns-90-a.gandi.net.
mondomaine.fr.  10800 IN NS ns-59-c.gandi.net.
mondomaine.fr.  10800 IN NS ns-148-b.gandi.net.
```

FIGURE 2

Zone.



Nous avons déjà trois serveurs de noms secondaires différents (enregistrement DNS de type NS), nommés ici `ns-90-a`, `ns-59-c` et `ns-148-b`. Ils assurent la résilience de notre résolution de nom. Il est important de comprendre que derrière ces 3 noms de serveurs se cachent des architectures réparties et redondantes.

Pour ajouter un autre service secondaire DNS sur notre serveur personnel ou d'entreprise, il faudra rajouter un enregistrement de type NS dans la configuration chez Gandi.

La console d'administration web du service DNS chez Gandi est visible en mode texte dans la figure 2.

L'enregistrement SOA est un enregistrement donnant des informations générales concernant les données présentes dans la zone DNS ainsi que sur leur gestion.

Pour ce qui est de la résilience du service DNS, le troisième paramètre, valant ici 604 800 secondes soit 7 jours, indique la durée pendant laquelle les serveurs DNS secondaires continueront à répondre aux requêtes des clients si le serveur DNS primaire est injoignable par les services secondaires.

Nous n'allons pas nous étendre plus sur la sécurisation du DNS, mais sachez simplement que ce protocole doit faire l'objet d'une attention particulière de votre part. Il est en effet sollicité au début de la quasi-totalité des connexions réseau sur Internet.

Concernant le choix du fournisseur de nom de domaine, pensez à vérifier les points suivants :

- la réputation de fiabilité, résilience (caractère réparti de ses services DNS), mais aussi performance de son infrastructure ;
- le bon niveau de son interface web de gestion DNS :
  - → délai de propagation des modifications DNS (cela peut être très important en cas d'attaques sur un de vos services) ;
  - → ergonomie et fonctionnalités (ex. : sauvegarde et restauration de zones DNS) ;
  - → possibilité de protéger son interface de gestion par une authentification à multifacteurs (voir par ailleurs).

L'ensemble de ces éléments vous aidera à faire le bon choix. La France a d'ailleurs la chance d'avoir de bons opérateurs de services DNS comme OVH ou Gandi.

INTERFACE D'ADMINISTRATION : SYSTÉMATISER L'AUTHENTIFICATION À MULTIPLES FACTEURS

Tout au long de cet article, nous parlerons d'interfaces d'administration permettant de gérer de manière aisée et néanmoins efficace vos services d'infrastructures : DNS, hébergement, messagerie, logs, etc.

L'actualité récente en cybersécurité sur les problématiques d'infrastructure (voir les attaques DNSpionage et SeaTurtle [1] de piratage d'infrastructures DNS) me pousse à vous recommander la mise en place systématique d'authentification à multiples facteurs sur vos comptes d'administration web d'infrastructures ou de messagerie :

- **Comptes de messagerie : le vol de votre nom d'utilisateur et de votre mot de passe de messagerie vous mènerait tout droit à être expulsé très rapidement de votre infrastructure par le pirate. En effet, sitôt connecté à votre adresse de messagerie, il va essayer de réinitialiser les mots de passe de tous vos comptes d'administration via la fonction « mot de passe perdu ou oublié » généralement présente sur les portails de connexion.**
- **Comptes d'administration d'infrastructures : de la même manière, si un vol de nom d'utilisateur et de mot de passe arrive pour ces comptes, une part notable voire la totalité de votre infrastructure peut être impactée. Si on accède à votre zone DNS ou à l'interface d'administration de votre serveur, les dégâts (fuite ou destruction de données personnelles/sensibles, abus de vos utilisateurs, etc.) peuvent être majeurs.**

Vous comprenez désormais les risques d'une simple connexion utilisateur/mot de passe.

Pensez donc à activer l'authentification à plusieurs facteurs. Souvent, il s'agit d'ajouter la génération d'un jeton TOTP (jeton à usage unique basé sur temps) au workflow classique « utilisateur / mot de passe ».

Ainsi, l'interface d'administration va effectuer les étapes suivantes :

- **Elle vous demande votre utilisateur et votre mot de passe.**

2. ARCHITECTURE PHASE 2 : QUAND UN SERVEUR SUPPLÉMENTAIRE POURRAIT NOUS RENDRE BIEN DES SERVICES

Services indispensables non hébergeables sur notre serveur principal

Vous avez décidé de mettre en place un serveur personnel vous permettant d'héberger un certain nombre de services comme par exemple un cloud, une messagerie ou un service de collaboration vidéo. Cependant, comme nous l'avons déjà abordé précédemment, certains services ne pourront pas être opérés sur ce serveur, notamment pour des raisons de sécurité.

Si nous n'avons pas l'ambition de mettre en place de suite des services hautement disponibles grâce à ce serveur tiers, on peut tout à fait utiliser un NAS présent à votre domicile ou sur le LAN de votre entreprise pour

- **Si vos identifiants sont acceptés, vous devrez générer un jeton à usage unique valide pendant un court laps de temps (ex. : 30 secondes). Vous le ferez en général avec une application sur smartphone, mais cela peut être aussi via une application sur ordinateur. Ces applications auront été préalablement enrôlées comme décrit ci-après. Quelques exemples d'applications smartphones permettant la génération de jetons de type OTP : FreeOTP, Authy ou Google Authenticator, etc.**
- **Si le jeton est accepté, vous êtes connecté.**

L'enrôlement de l'application est relativement simple :

- **Vous vous connectez en saisissant votre utilisateur et votre mot de passe sur l'interface d'administration voulue (messagerie, DNS, serveur, etc.).**
- **Vous activez l'authentification à base de TOTP.**
- **L'interface vous propose des codes de récupération que vous devez conserver soigneusement. Par exemple, dans votre gestionnaire de mots de passe (vous possédez et vous utilisez un gestionnaire de mots de passe, n'est-ce pas ? ;-)).**

Rappel sur le cloud et la sauvegarde : pensez à bien sauvegarder de manière régulière votre base de mots de passe. Il est important de noter qu'un cloud synchronisé n'est pas une sauvegarde. En effet, si vous supprimez par erreur votre base de mots de passe, la synchronisation va propager la suppression de votre base de mots de passe sur tous vos autres périphériques avant même que vous réalisiez l'erreur irréversible que vous venez de commettre. Une sauvegarde automatisée de votre cloud sur un NAS avec une vraie politique de rétention peut être une des solutions possibles à ces soucis.

- **L'interface d'administration affiche ensuite un QR code que vous allez scanner avec votre application TOTP. Le QR code contient le secret partagé qui permettra à l'application de générer le jeton TOTP que l'interface pourra ensuite valider comme ayant été généré par vous.**

Et voilà, l'accès à vos comptes critiques est désormais sécurisé.

jouer ce rôle. La seule exigence sera alors de disposer d'un accès root dessus et que le système de base soit un Linux récent. En effet, nous devons pouvoir installer des logiciels à jour sur ce NAS.

Voyons quelques exemples :

- La sauvegarde : vos services sur votre serveur primaire vont manipuler une certaine quantité de données. Or, ces données peuvent être altérées, voire totalement perdues. Les causes de cette perte peuvent être variées : souci de fonctionnement de votre stockage de données, erreur de manipulation de vos utilisateurs ou de l'administrateur, un ransomware qui arrive à s'exécuter sur votre serveur. Avoir un serveur externe qui assure une sauvegarde régulière des données de votre serveur primaire est une bonne pratique. Il faudra qu'il implémente sur ces sauvegardes une rétention en cohérence avec vos besoins de restauration. Idéalement, avoir une sauvegarde quotidienne avec 7 jours de rétention, hebdomadaire avec 4 semaines de rétention et mensuelle avec 3 ou 6 mois de rétention vous permettra de dormir de manière apaisée.
- La supervision sécurité : lorsque vous suspecterez un incident de sécurité sur votre serveur principal, il sera peut être déjà trop tard. La première action qu'effectue en général un attaquant est de masquer son intrusion

et d'assurer la persistance de son accès illicite sur votre serveur. Il va donc s'assurer d'effacer les lignes correspondant à son intrusion dans les traces du serveur. Si les traces de votre serveur primaire ne sont pas envoyées en continu vers un serveur tiers, vous ne pourrez pas investiguer sur un éventuel accès illicite à votre serveur primaire. Ainsi, vous disposez d'une plateforme d'investigation saine et disponible en cas d'attaque.

- L'évolutivité : dans le cas d'un serveur d'entreprise, il y a fort à parier que votre serveur primaire aura à évoluer au point de se voir adjoindre d'autres serveurs pour assurer d'autres services ou pour sécuriser les services existants sur ce serveur tant en termes de disponibilité que de performances. Dans ce cas-là, le serveur tiers pourrait par exemple jouer le rôle de chef d'orchestre d'un service d'administration centralisé type Ansible.

Architecture de nos serveurs

Désormais convaincu de l'intérêt de posséder deux serveurs, voici à quoi notre architecture peut donc ressembler :

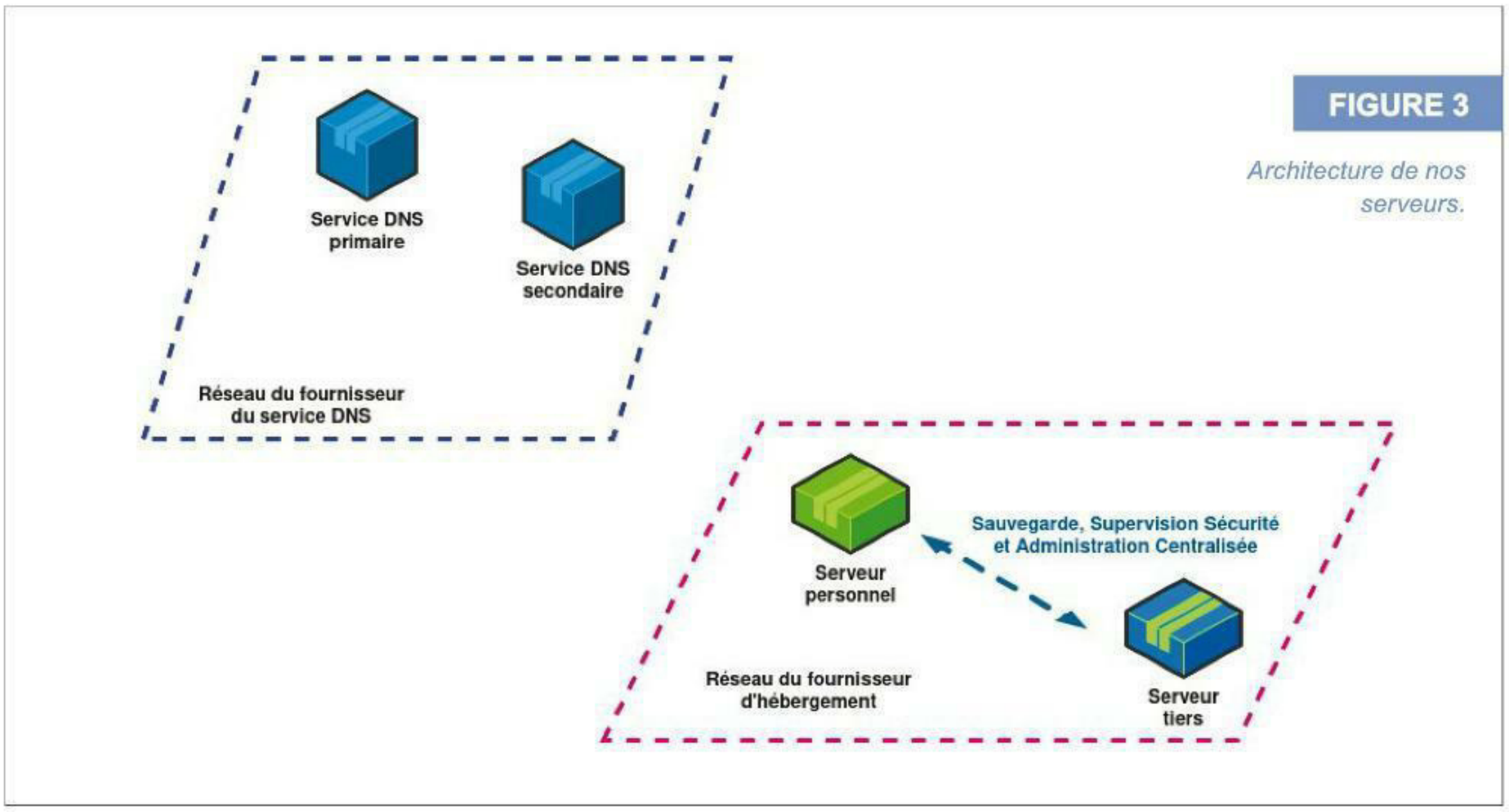


FIGURE 3

Architecture de nos serveurs.

Bien entendu, cette architecture a un coût. Nous pouvons tout à fait investir sur notre serveur primaire et nous contenter d'un serveur tiers modeste en termes de mémoire et de processeur. Disposer d'un espace disque raisonnable est par contre nécessaire pour assurer la sauvegarde. Si votre hébergeur est à même de vous fournir cette fonction de sauvegarde par un service tiers, vous pouvez même acheter un serveur avec relativement peu d'espace disque.

3. CRITÈRES DE CHOIX DU SERVEUR

Le choix du serveur primaire, mais aussi du serveur tiers va donc dépendre des éléments suivants :

- Système d'exploitation et distribution : un serveur virtuel ou physique de type Linux. Nous choisirons une distribution Debian (Debian 10.4 à la rédaction de cet article) afin de bénéficier d'une bonne stabilité ainsi que d'une logithèque de qualité.

- Type d'accès : un accès SSH est impératif afin de pouvoir administrer son serveur de partout de manière sécurisée.
- Droits : vous devez avoir un droit d'accès root possible. C'est impératif afin d'avoir la réelle maîtrise de votre serveur.
- Sécurité des données : idéalement, votre serveur doit bénéficier d'une protection de type RAID contre les pannes disques. En général, vous pourrez trouver des configurations de type RAID 1 (un disque est miroir de l'autre) à prix raisonnable. De même, un espace disque chiffré lors de l'installation du système doit être possible si votre analyse de risque vous le préconise. Le risque à couvrir serait ici de se prémunir d'un accès physique au disque du serveur lors de son exploitation ou lors de sa réutilisation ultérieure (client suivant, mise au rebut du serveur).
- Intervention de secours : il faut que vous puissiez a minima redémarrer votre serveur depuis l'interface d'administration web. Idéalement, un KVM ou commutateur qui vous permet d'avoir un accès clavier et écran à un terminal ou même au BIOS de votre serveur. En général, sur les offres de premier prix, cet accès est remplacé par la possibilité de solliciter le support pour une intervention. En tous les cas, avoir accès à un support technique est une nécessité.

Maintenant que vous avez les éléments de choix, à vous de retenir l'offre qui vous convient le mieux.

4. PREMIERS ÉLÉMENTS DE SÉCURISATION DE VOTRE SERVEUR POST-INSTALLATION

Voilà, le grand saut est effectué : vous avez sélectionné votre fournisseur et avez installé la distribution Debian sur votre serveur. Vous disposez de surcroît d'un accès en mode administrateur sur ce serveur.

Comme évoqué dans la section précédente, c'est lors de l'installation que vous pouvez faire le choix d'installer votre distribution Debian sur un espace disque chiffré. Le choix de chiffrer son espace disque s'effectue lors de la définition du partitionnement disque du serveur. Sachez cependant que cela implique une limitation côté exploitation de notre serveur : si votre serveur redémarre, il ne pourra pas le faire sans la saisie d'un mot de passe au cours des premiers instants de sa relance, ce qui peut être problématique dans le cadre d'une administration à distance.

Voyons maintenant quelques bonnes pratiques de sécurité système à appliquer en sortie d'installation de votre serveur. Ces tâches d'administration système peuvent et doivent être déclinées tant sur le serveur primaire que le serveur tiers.

Paramétrer l'envoi de mails pour rendre possibles les notifications de sécurité

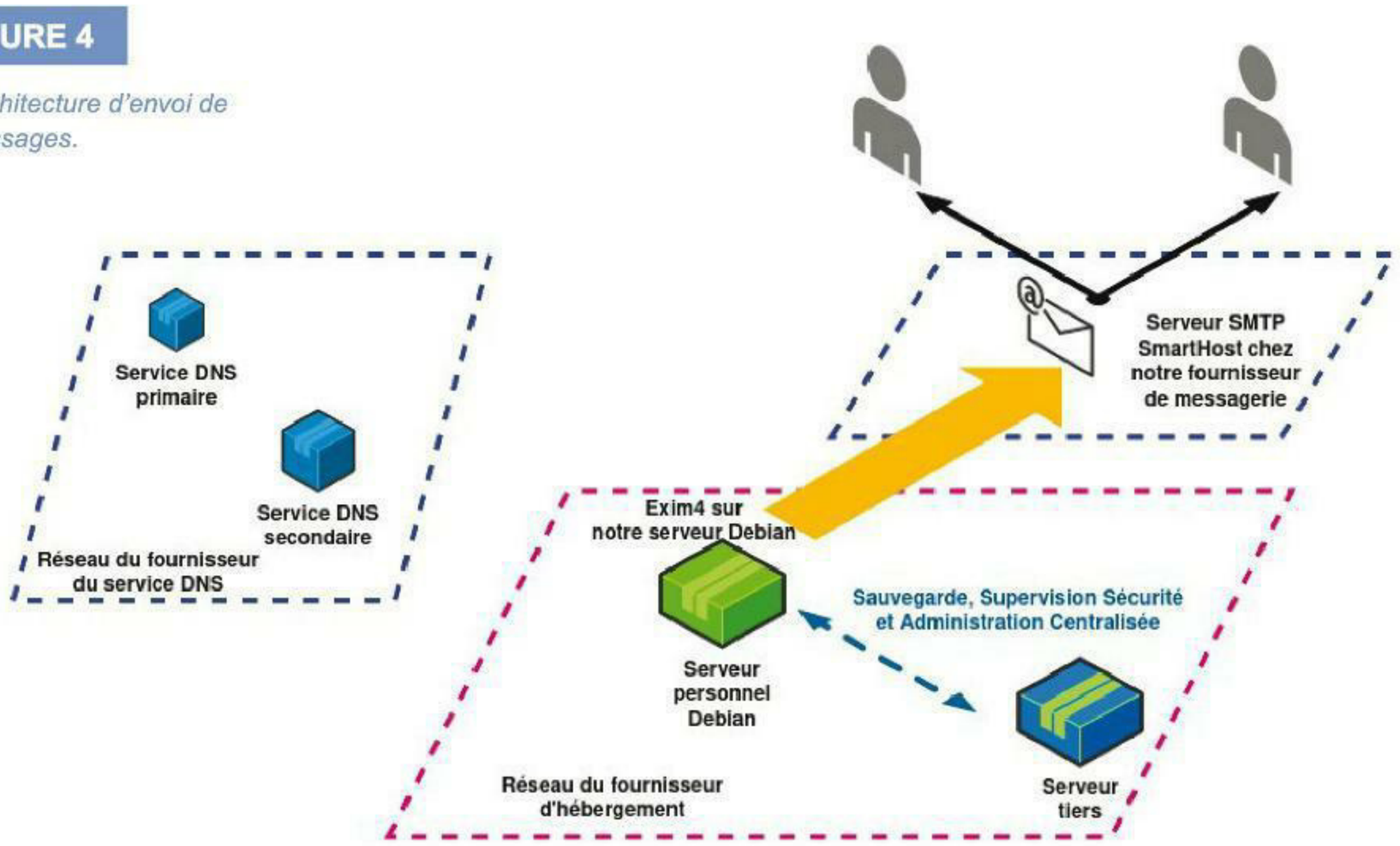
Quand on pense à la sécurité d'un serveur ou d'un service, on pense immédiatement à la détection d'une menace ou d'une attaque. Et qui dit détection, dit être prévenu afin de pouvoir réagir.

En général, l'un des moyens les plus fiables et les moins coûteux d'être prévenu, c'est le message électronique. Par défaut, vos serveurs ne sont pas correctement paramétrés pour en envoyer. Remédions rapidement à ce souci.

Exim4 sera le logiciel que nous utiliserons comme service de courrier sortant. Plus précisément, vu qu'administrer un serveur de courriers est une tâche complexe et qui peut être harassante, Exim4 va envoyer tous les messages sortants vers un serveur SMTP d'un de nos fournisseurs de messagerie. On appelle ce type de serveur SMTP distant, un *smarthost*.

FIGURE 4

Architecture d'envoi de messages.



La mise en œuvre est simple :

- On installe mailx et Exim4 s'ils ne sont pas déjà installés.

```
$ sudo apt install heirloom-mailx
$ sudo apt install exim4
```

- On configure ensuite Exim4 pour qu'il fonctionne en mode *smarthost*.

```
# sudo dpkg-reconfigure exim4-config
```

Les choix que vous allez renseigner sont les suivants :

```
> mail sent by smarthost; no local mail
> System mail name:
> IP-addresses to listen on for incoming SMTP connections:  127.0.0.1 ; ::1
> Other destinations for which mail is accepted:
> Visible domain name for local users:
> IP address or host name of the outgoing smarthost: mail.gandi.net::587
> Keep number of DNS-queries minimal (Dial-on-Demand) ? No
> Split configuration into small files? No
```

Quand il n'y a rien de marqué après un choix, c'est qu'il ne faut rien saisir et valider le choix tel quel.

Pour l'exemple, nous avons utilisé les services du serveur SMTP de Gandi (**mail.gandi.net** sur le port 587 correspondant au SMTP encapsulé dans une session chiffrée TLS). À vous d'utiliser le service SMTP sur lequel vous disposez de votre boite aux lettres d'administration système (OVH, Gmail, etc.).

Dernier point : en général, l'envoi de messages SMTP est authentifié. Il faut donc renseigner un nom d'utilisateur, reconnu par le serveur SMTP *smarthost*, et son mot de passe dans le fichier **passwd.client**.

```
sudo vim /etc/exim4/passwd.client
```

Et ajouter la ligne :

```
mail.gandi.net:user:password
```

Où **mail.gandi.net** est à changer par le serveur SMTP *smarthost* que vous utilisez et le couple user/password par votre utilisateur de messagerie et son mot de passe.

Voilà, vous pouvez envoyer des messages d'alerte depuis votre serveur.

Gestion de l'utilisateur root

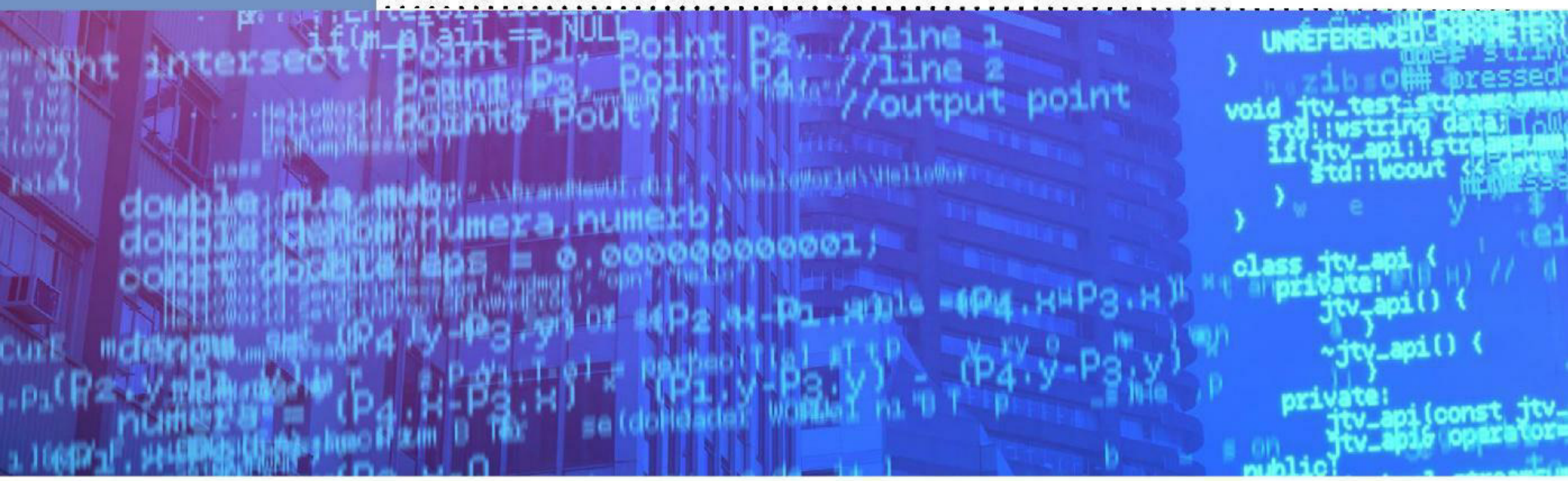
» root : pourquoi en désactiver la connexion

Une bonne pratique est de ne pas rendre possible la connexion sur le compte root. Je ne parle pas ici de la connexion SSH, dont nous détaillerons en profondeur l'usage dans un prochain article, mais de la connexion sur le serveur même.

En effet, on a tendance à se connecter au compte root pour pouvoir enchaîner plusieurs commandes privilégiées. Or, bien que l'on comprenne l'avantage immédiat de cette approche en termes de productivité, plusieurs inconvénients tant d'exploitation que de sécurité pointent rapidement le bout de leur nez.

Tout d'abord, vous pouvez déclencher une action avec des conséquences importantes sans que vous ne le réalisiez immédiatement comme la suppression d'un répertoire système (**/var** par exemple). En étant connecté sous votre utilisateur standard, vous auriez reçu un refus de suppression dudit répertoire. Vous auriez été obligé de saisir la commande **sudo** en amont de votre commande ce qui aurait pu attirer votre attention sur le caractère inapproprié de votre commande. Notez que j'ai utilisé le conditionnel, car ce n'est absolument pas une assurance tout risque contre les fausses manipulations. Mais c'est une sécurité supplémentaire.

L'autre raison est la traçabilité.



Quand vous travaillez directement sous root, voici la trace que vous obtenez dans le fichier de trace **/var/log/auth.log** :

```
Jul 16 09:34:53 hostname sudo: pam_unix(sudo:session): session opened for user root
by (uid=0)
Jul 16 09:36:57 hostname sudo: pam_unix(sudo:session): session closed for user root
```

Aucune information n'est disponible sur les actions effectuées par l'utilisateur connecté hormis sa date et heure de connexion/déconnexion.

Par contre, quand vous exécutez une commande privilégiée avec la commande **sudo** depuis votre utilisateur standard (ici **cb**) vous disposerez dans ce même fichier de la trace de la commande exécutée ainsi que de l'utilisateur de départ. Lors d'une investigation numérique suite à la suspicion d'un incident de sécurité, cela change la donne :

```
Jul 16 13:12:53 hostname sudo: cb : TTY=pts/1 ; PWD=/home/cb ; USER=root ;
COMMAND=/usr/bin/apt update
Jul 16 13:12:53 hostname sudo: pam_unix(sudo:session): session opened for user root
by (uid=0)
Jul 16 13:12:57 hostname sudo: pam_unix(sudo:session): session closed for user root
```

» Inactivation de la connexion au compte root

Prenons une précaution primordiale : ajoutons, si ce n'est déjà fait, les droits administrateur à votre utilisateur standard (appelé ici **user1**) via la commande suivante, qui rajoute l'utilisateur **user1** au groupe **sudo** :

```
$ /sbin/adduser user1 sudo
```

Inactivons ensuite la connexion au compte root en changeant la ligne suivante dans le fichier **/etc/passwd** :

```
root:x:0:0:root:/root:/bin/bash
```

par la ligne :

```
root:x:0:0:root:/root:/sbin/nologin
```

Mise à jour automatisée du système

» Pourquoi faire ce choix

Le principal danger auquel est soumis un serveur sur Internet est le fait qu'il n'ait pas été mis à jour en temps et heure par rapport aux vulnérabilités corrigées.

Par exemple, les deux dernières vulnérabilités touchant l'outil d'administration centralisé SaltStack **[2]** ont été exploitées à grande échelle sur Internet trois jours seulement après la communication des vulnérabilités CVE-2020-11651 et CVE-2020-2020-11652 ainsi que la mise à disposition des correctifs.

Les conséquences ont été immédiates et radicales pour les entreprises impactées :

- prise de main à distance ;
- exécution de code arbitraire sur tous les serveurs dépendant des serveurs SaltStack primaires.

Dernier point à noter : ce genre de mésaventure n'arrive pas qu'aux administrateurs système « en herbe ». Cette vulnérabilité a touché de très grosses sociétés comme Algolia, Cisco ou DigiCert. Elles ont été touchées, car elles exposaient en direct sur Internet leur interface d'administration SaltStack (sujet que nous traiterons dans l'article dédié aux protections réseaux), mais aussi, car leur système de déploiement de correctifs n'a pas été assez réactif.

Donc, en tant qu'administrateur de nos serveurs, la dernière chose que nous avons envie de vivre, c'est qu'une telle publication sorte pendant nos vacances et que les correctifs bien que disponibles ne soient pas appliqués.

» Mise en œuvre

Installons donc les deux paquets Debian dédiés à la gestion de ces mises à jour non effectuées manuellement :

```
$ sudo apt install unattended-upgrades apt-listchanges
```

Ensuite, nous mettons en place l'exécution d'`unattended upgrades` :

```
$ sudo dpkg-reconfigure -plow unattended-upgrades
```

Enfin, nous activons les notifications mail et l'installation automatisée des mises à jour de sécurité. Pour cela, nous modifions comme suit les lignes suivantes dans le fichier **/etc/apt/apt.conf.d/50unattended-upgrades** :

- Activation de la notification e-mail en enlevant le **#** en début de ligne et en mettant votre adresse e-mail entre les guillemets :

```
Unattended-Upgrade::Mail "votre-adresse-email";
```

- Ensuite, on active (toujours en enlevant le **#** au début des lignes) la ligne concernant les mises à jour de sécurité :

```
"origin=Debian,codename=${distro_codename},label=Debian-Security";
```

Si vous disposez de packages non officiels fournis par exemple au travers du PPA du projet en question, vous pouvez tout de même les gérer par **unattend-upgrades**.

Pour cela, nous allons :

- Déterminer le fichier contenant les informations du PPA en question : il faut trouver un fichier terminé par InRelease relatif à ce dépôt dans le répertoire **/var/lib/apt/lists/**.
- Trouver dans ce fichier la mention **Origin**.
- La reporter dans la même section **Unattended-Upgrade::Allowed-Origins** que ci-dessus dans le fichier **/etc/apt/apt.conf.d/50unattended-upgrades** sous la forme de la ligne suivante :

```
"origin=Origin-trouvée-dans-le-fichier-InRelease-du-PPA";
```

Enfin, vous pouvez saisir la commande suivante pour tester votre configuration :

```
$ sudo unattended-upgrades --dry-run --debug
```

CONCLUSION

La première étape de notre voyage dans la sécurisation de notre infrastructure est achevée. Nous avons :

- dessiné les contours de notre infrastructure ;
- choisi nos fournisseurs de DNS et d'hébergement de serveurs ;
- déployé ces serveurs ;
- et mis en œuvre les mesures de sécurisation systèmes indispensables à la protection de ces serveurs.

Nous pouvons désormais nous occuper sereinement de la sécurité réseau de nos systèmes. Et je ne vous cache pas que je considère cette étape comme une étape clé de notre chemin vers la relative sérénité à laquelle nous pouvons aspirer en tant qu'administrateurs système :-) ▌

RÉFÉRENCES

[1] **https://blog.talosintelligence.com/2018/11/dnspionage-campaign-targets-middle-east.html** et **https://blog.talosintelligence.com/2019/04/seaturtle.html**

[2] **https://blog.algolia.com/salt-incident-may-3rd-2020-retrospective-and-update/**

LISEZ DÈS À PRÉSENT TOUS NOS MAGAZINES EN LIGNE !

Simple. Rapide. Pratique.

FLIPBOOK HTML5

Tous nos magazines ont leur version Flipbook :

LINUX MAGAZINE / FRANCE · MISC · LINUX PRATIQUE · HACKABLE MAGAZINE

Rendez-vous sur www.ed-diamond.com

DOSSIER

SÉCURISEZ VOTRE RÉSEAU

Christophe BROCAS

Maintenant que notre serveur principal est déployé et que nous y avons appliqué un premier niveau de sécurisation système, occupons-nous de sa sécurisation réseau. Nous allons détailler en quoi les attaques réseau sont primordiales dans notre modèle de menace. Comme nous le verrons, l'accès distant est le risque principal qui guette nos serveurs. Nous allons mettre en œuvre une sécurité en profondeur et les mesures de protection réseau en seront une de ses dimensions importantes.

1. MODÈLE DE MENACE DE NOTRE INFRASTRUCTURE

La menace de l'accès physique à nos serveurs

À moins que vous n'ayez des adversaires étatiques (sic) ou commerciaux extrêmement agressifs, le principal risque encouru par votre infrastructure n'est pas un accès physique à vos serveurs. En effet, cela impliquerait un accès privilégié au centre de données qui héberge vos serveurs ou à défaut, à un KVM connecté dessus. Un KVM est un combiné regroupant un clavier, un écran et une souris mutualisés entre plusieurs serveurs. Nous pouvons décider dans une relative sérénité de nous concentrer sur d'autres types de menaces.

Le réseau, la porte d'entrée universelle menant à nos données

Nos différents serveurs doivent impérativement « écouter » sur le réseau pour rendre les services dont nos utilisateurs ont besoin. Par « écouter », nous entendons qu'un processus de votre serveur reçoit des informations transmises en TCP ou en UDP vers le port sur lequel il écoute.

Ces informations peuvent être envoyées sur le port en question par n'importe qui et notamment par des personnes malveillantes. Toutes les attaques distantes consistent à envoyer des données sur de tels logiciels en écoute en vue d'en exploiter des vulnérabilités. Bien entendu, l'objectif de ces attaques est d'accéder à des données, de gagner des droits d'administration ou d'exécuter des actions normalement interdites.

Nous comprenons donc rapidement que la sécurité réseau est une priorité dans notre démarche de sécurisation de notre infrastructure.

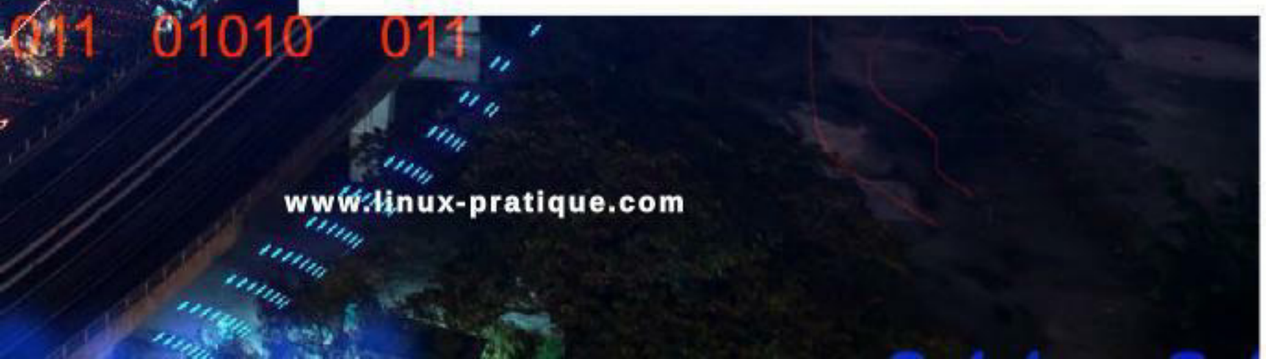
Plan d'action

Voici quelques-uns des risques pesant sur l'aspect réseau de notre infrastructure :

- Des applications sensibles (ex. : interfaces d'administration, moteurs de bases de données) sont en écoute sur un port ouvert sur Internet.
- Des tentatives de connexion sont déclenchées en masse pour deviner un mot de passe d'accès à un de nos services.
- Des envois de données massifs sont effectués en vue d'obtenir un déni de services sur un serveur ou un service.
- Une vulnérabilité sur la partie authentification ou contrôle d'habilitation d'un service ouvert sur Internet est divulguée.

Nous allons donc déployer les mesures de sécurité suivantes afin de mitiger ces risques :

1. Contrôler les applications actuellement en écoute sur Internet sur nos serveurs et inactiver celles ne le devant pas.
2. Mettre en place un pare-feu afin de n'accepter les connexions entrantes que sur les ports autorisés.
3. Déployer un mécanisme de bannissement des adresses IP générant des tentatives de connexion en masse sur nos services.
4. Mettre en place un VPN d'administration qui interconnecte nos deux serveurs de manière sécurisée. Cela nous permettra notamment de sécuriser l'accès aux applications sensibles comme les interfaces d'administration.



2. RÉDUCTION DE NOTRE SURFACE D'ATTAQUE

Problématique de la surface d'attaque

Nos serveurs ont vocation à fournir des services à nos utilisateurs. Pour cela, un certain nombre d'applications doivent être ouvertes sur Internet. La liste de ces applications ainsi ouvertes constitue notre surface d'attaque. Toute vulnérabilité sur une de ces applications sera une menace directe pour l'intégrité du serveur et des données qu'il héberge.

Notre objectif sécurité sera donc d'assurer les services attendus par nos utilisateurs tout en réduisant au maximum notre surface d'attaque. Autrement dit, minimisons rapidement le nombre d'applications en écoute sur Internet sur nos serveurs !

Vérification des applications en écoute

Nous allons donc tout d'abord utiliser la commande **netstat** pour obtenir la liste des applications en écoute.

```
$ netstat -ltnp
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address
State       PID/Program name
tcp        0      0 127.0.0.1:3306          0.0.0.0:*
LISTEN      561/mysqld
tcp        0      0 127.0.0.1:53            0.0.0.0:*
LISTEN      467/named
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN
459/sshd
tcp        0      0 127.0.0.1:25            0.0.0.0:*
LISTEN      9297/exim4
tcp        0      0 127.0.0.1:953           0.0.0.0:*
LISTEN      467/named
tcp6       0      0 :::443                  :::*                    LISTEN
579/apache2
tcp6       0      0 :::80                   :::*                    LISTEN
579/apache2
tcp6       0      0 ::1:53                  :::*
LISTEN      467/named
tcp6       0      0 :::22                   :::*
LISTEN      459/sshd
tcp6       0      0 ::1:25                  :::*
LISTEN      9297/exim4
tcp6       0      0 ::1:953                 :::*
LISTEN      467/named
```

Comme mis en évidence ici, nous avons deux applications sur notre serveur qui sont en écoute sur Internet. Les autres applications écoutent sur l'interface locale (**127.0.0.1** en IPv4 et **::1** en IPv6) qui n'est accessible que... localement sur le serveur ;-)

À noter : ici, le serveur web Apache n'est marqué comme n'écoutant qu'en IPv6, mais il écoute bien en IPv4 aussi. Il ouvre en effet par défaut des sockets sur toutes les interfaces réseaux disponibles sur la machine.

Gérer l'écoute sur Internet de nos applications

Pour illustrer comment on peut gérer l'accès réseau à nos applications, nous prendrons l'exemple du moteur de bases de données.

Cas n°1 : nous n'avons pas besoin d'ouvrir la base de données à des serveurs tiers.

Sur la capture de terminal ci-dessus, MariaDB (fork de MySQL **[2]**) est repérable par le nom de son processus `mysqld`.

Nous voyons qu'elle écoute sur l'interface locale `127.0.0.1`. C'est une bonne pratique sécurité. Elle permet de ne pas exposer inutilement nos bases de données à des tentatives d'accès externes. Ce paramétrage est le paramétrage par défaut de MariaDB, ce qui en fait un paramètre *Secured by default* (voir par ailleurs).

SÉCURITÉ PAR DÉFAUT : CE QUI DOIT ÊTRE LA NORME SUR INTERNET

La sécurité par défaut est une règle de gestion des paramètres des applications et des langages de développement qui s'est développée face à la montée des incidents de sécurité sur Internet.

En appliquant cette méthode de conception, chaque paramètre de configuration est positionné en sortie d'installation du produit ou du langage sur le réglage le plus sécurisé.

En effet, certaines applications ou langages de développement ont eu et ont encore parfois de nos jours des paramètres en sortie d'installation qui les laissent dans un état vulnérable à une attaque.

Reprenons l'exemple de MariaDB. Par le passé, ce moteur de base de données sortait de son installation standard en écoutant par défaut sur toutes les interfaces d'un serveur. Ainsi, une installation à peine déployée écoutait déjà sur Internet avant même que la personne en assurant l'administration en ait vérifié sa sécurité.

Un autre exemple de configuration par défaut non sécurisé est décrit dans la fiche d'évolution du langage Python nommée PEP 0476 [1] : quand une fonction d'un module client de la bibliothèque standard (comme `urllib`, `urllib2`, `http`, ou `httplib`) appelait une URL en https, la bibliothèque standard s'occupait notamment du *handshake* TLS (séquence d'initialisation de la connexion sécurisée TLS) durant lequel, entre autres choses, le certificat du serveur était envoyé au client.

Or, à l'époque, la bibliothèque standard ne vérifiait pas par défaut :

- **si le certificat avait été signé par une des autorités de certification auxquels le système sur lequel le code Python s'exécute fait confiance ;**
- **ni si le nom d'hôte du certificat (champ `CN` ou `SubjectAltNames` du certificat) correspondait au nom d'hôte demandé par la fonction.**

Il y avait alors une possibilité d'interception transparente du trafic (attaque de type homme du milieu) sans que l'utilisateur ne soit jamais notifié du risque par une alerte de sécurité.

Cas n°2 : bases de données devant être accédées depuis le réseau.

Imaginons maintenant que vous deviez rendre disponible votre base de données auprès d'un ou de plusieurs serveurs distants. Par exemple, quand vous avez une application de type N tiers où le serveur d'application exécutant le code et le serveur de base de données ne sont pas hébergés sur le même serveur physique.

Les mesures à prendre sont les suivantes :

- MariaDB : faire écouter MariaDB sur une adresse IP normale et non locale (paramètre `bind-address` du fichier de configuration de MariaDB).

  Important : ouvrir un accès distant ne veut pas dire obligatoirement faire écouter MariaDB sur l'adresse IP publique. En effet, comme nous le verrons dans le paragraphe dédié au VPN Wireguard, les 2 serveurs peuvent être reliés par un réseau local privé sur lequel les services des deux serveurs peuvent écouter sans être exposés sur Internet.

- MariaDB : créer un utilisateur sous MariaDB dédié au serveur devant accéder à la base de données. Important : vous pouvez très bien limiter cet utilisateur à un accès depuis le serveur en question ou depuis un sous-réseau.

  Exemple : la commande ci-dessous crée un utilisateur `User1` ayant tous les droits sur la base `Database1` depuis le sous réseau 192.168.100.0/24.

```
GRANT ALL PRIVILEGES ON Database1 TO 'user1'@'192.168.100.%'
IDENTIFIED BY 'my-new-password' WITH GRANT OPTION;
```

- Pare-feu : mettre en place une règle de pare-feu (voir plus en détail dans un chapitre à venir dans l'article) ne permettant l'accès au port écouté par MariaDB que par les serveurs devant y accéder.

3. SÉCURITÉ EN PROFONDEUR

Comme la sécurité par défaut, la sécurité en profondeur est un principe clé pour lequel vous devez vous battre au quotidien pour qu'il soit mis en place et surtout maintenu dans le temps au sein de vos infrastructures.

Nous avons illustré dans le dernier exemple en quoi consiste la sécurité en profondeur : nous mettons en place plusieurs barrières de nature différente et à des niveaux complémentaires de l'infrastructure pour qu'une connexion distante à la base de données se fasse de manière sécurisée :

Sur le plan réseau :

- grâce au pare-feu : nous n'ouvrons les capacités de connexion à la base de données qu'à partir de certains serveurs ;
- grâce à l'interface réseau sur laquelle on fait écouter sur MariaDB : on peut n'écouter que depuis une adresse IP joignable que par VPN ;
- grâce à la création de l'utilisateur MariaDB : on n'autorise sa connexion que si elle est faite depuis une machine dont l'adresse IP est autorisée.

Sur le plan authentification :

- grâce à l'authentification MariaDB : on n'autorise qu'un utilisateur authentifié à accéder à une base de données ciblée avec des droits maîtrisés.

Comme vous pouvez le constater sur ce cas d'usage, la sécurité en profondeur nous permet de poser des verrous qui limitent les risques engendrés soit par une vulnérabilité récemment publiée soit par un défaut d'application d'une des autres mesures de sécurité.

Exemple : imaginons qu'une vulnérabilité de contournement de l'authentification MariaDB soit publiée. Il s'agirait alors du cauchemar absolu de tous vos collègues administrateurs de bases de données et par ricochet de vos RSSI, DSI et PDG.

Dans notre cas, pour que cela nous impacte réellement, il faudrait qu'un attaquant puisse réaliser l'ensemble des tâches suivantes :

- prise de main sur un de vos serveurs tiers afin de pouvoir s'adresser au serveur de base de données ;
- rebondir depuis ce serveur pour tenter d'exploiter cette vulnérabilité critique.

Si vous aviez directement ouvert votre MariaDB sur Internet sans positionner de règle sur le pare-feu du serveur l'hébergeant, en vous reposant uniquement sur l'authentification MariaDB, la vulnérabilité aurait exposé directement vos données aux attaquants.

La sécurité en profondeur n'est pas une solution parfaite et présente des difficultés de mise en œuvre, notamment au sein des grandes entreprises :

- Elle exige une mise en œuvre coordonnée des différentes mesures de sécurité.
- Ces mesures sont difficiles à maintenir dans le temps vis-à-vis d'une infrastructure qui évolue de plus en plus rapidement.

Cependant, à notre échelle individuelle ou de petite équipe au sein d'une PME, appliquer les principes de la sécurité en profondeur est la quasi-assurance de beaucoup mieux dormir la nuit et pendant nos vacances !

4. PARE-FEU, LE BIEN NOMMÉ

Le principe de fonctionnement d'un pare-feu est qu'il contrôle les connexions réseau à l'entrée et la sortie de chaque interface réseau d'une machine. Comme démontré dans les paragraphes précédents, son usage est une nécessité et nous allons le mettre en œuvre immédiatement.

Le noyau Linux possède grâce à Netfilter un framework de filtrage et de NAT. Les interfaces de paramétrage de ce pare-feu Netfilter ont été Iptables et désormais, Nftables.

Cependant, j'ai retenu une autre interface, particulièrement simple d'usage, Uncomplicated FireWall ou UFW. Voyons comme on peut en tirer le meilleur parti.

Installation et premiers pas avec UFW

L'installation est triviale :

```
$ sudo apt install ufw
```

et le contrôle de l'état du pare-feu l'est tout autant :

```
$ sudo ufw status
Status: inactive
```

Ici, le pare-feu est inactif. Pour le passer en actif, nous utiliserons la commande suivante :

```
$ sudo ufw enable
```

Cependant bien qu'aucune règle n'ait été saisie, si nous l'activions dès maintenant, il appliquerait un certain nombre de paramètres comme les politiques d'accès par défaut.

Règles par défaut

UFW possède comme tout pare-feu des politiques par défaut en fonction de l'endroit où le paquet est émis.

Une politique par défaut est la règle qui s'applique (**ACCEPT** ou **DROP**) si aucune règle créée par l'utilisateur ne s'applique :

```
$ more /etc/default/ufw
[...]
DEFAULT_INPUT_POLICY="DROP"
DEFAULT_OUTPUT_POLICY="ACCEPT"
DEFAULT_FORWARD_POLICY="ACCEPT"
[...]
```

Ici, les paquets provenant de l'extérieur sont rejetés. Ceux émis depuis le serveur vers l'extérieur ou transférés sont autorisés.

Mise en œuvre basique

» Ajout d'autorisation

L'autorisation de différents services que nous souhaitons ouvrir se fait via la commande suivante :

```
$ sudo ufw allow <service name>
```

Vous pourrez trouver le nom de tous les services connus de votre système dans le fichier **/etc/services**.
Ce fichier contient aussi le protocole (TCP et/ou UDP) et le numéro de port correspondant au service.

Du coup, la commande :

```
$ sudo ufw allow ssh
```

se traduira par l'ouverture du port 22 en entrée en TCP.

Et si vous saisissez :

```
$ sudo ufw allow domain
```

cela ouvre les accès au service DNS sur le port 53 en entrée en TCP et en UDP.

Note : si le paramètre `IPV6=yes` est positionné dans le fichier de valeurs par défaut `/etc/default/ufw`, ces accès seront ouverts tant en IPv4 qu'en IPv6.

Vous pouvez tout aussi bien ouvrir l'accès SSH en ne saisissant que le port et le protocole comme suit :

```
$ sudo ufw allow 22/tcp
```

» Configuration type

Un exemple de configuration simple autorisant en entrée SSH, HTTP, HTTPS, DNS (TCP et UDP) et le VPN Wireguard (en UDP sur le port par défaut 51820) tant en IPv4 qu'en IPv6 :

```
$ sudo ufw status
Status: active


To                         Action      From
--                         ------      ----
22/tcp                     ALLOW       Anywhere
80/tcp                     ALLOW       Anywhere
443/tcp                    ALLOW       Anywhere
51820/udp                  ALLOW       Anywhere
53/udp                     ALLOW       Anywhere
53/tcp                     ALLOW       Anywhere
22/tcp (v6)                ALLOW       Anywhere (v6)
80/tcp (v6)                ALLOW       Anywhere (v6)
443/tcp (v6)               ALLOW       Anywhere (v6)
51820/udp (v6)             ALLOW       Anywhere (v6)
53/udp (v6)                ALLOW       Anywhere (v6)
53/tcp (v6)                ALLOW       Anywhere (v6)
```

Configuration plus affinée

» Autorisation d'un sous-réseau sur un service

Nous avons indiqué tout à l'heure que nous souhaiterions n'ouvrir le moteur de bases de données MariaDB sur le port 3306 qu'aux machines présentes sur un sous-réseau privé. Imaginons que l'adresse de ce sous-réseau est la suivante : 192.168.100.0/24.

Il faudra saisir la commande suivante :

```
$ sudo ufw allow from 192.168.100.0/24 to any port 3306
```

Blocage d'un sous-réseau sur un service

À l'inverse, pour bloquer un sous-réseau spécifique qui par exemple tente un DDoS sur votre serveur web cela donne :

```
$ sudo ufw deny from 45.134.5.0/24 to any port 80
$ sudo ufw deny from 45.134.5.0/24 to any port 443
```

Point d'attention : ces deux dernières règles ne vont pas fonctionner si on a déjà mis en place les règles d'acceptation sur les ports 80 et 443. En effet, les règles sont évaluées dans l'ordre descendant.

Du coup, nous allons :

- afficher de manière numérotée les règles ;
- insérer les règles de rejet avant la règle d'acceptation sur les ports 80 et 443 ;
- refaire l'affichage.

```
$ sudo ufw status numbered
Status: active

     To                         Action      From
     --                         ------      ----
[ 1] 22/tcp                     ALLOW IN    Anywhere
[ 2] 80/tcp                     ALLOW IN    Anywhere
[ 3] 443/tcp                    ALLOW IN    Anywhere
[ 4] 51820/udp                  ALLOW IN    Anywhere
[ 5] 53/udp                     ALLOW IN    Anywhere
[ 6] 53/tcp                     ALLOW IN    Anywhere
[ 7] 3306                       ALLOW IN    192.168.100.0/24
[ 8] 22/tcp (v6)                ALLOW IN    Anywhere (v6)
[ 9] 80/tcp (v6)                ALLOW IN    Anywhere (v6)
[10] 443/tcp (v6)               ALLOW IN    Anywhere (v6)
[11] 51820/udp (v6)             ALLOW IN    Anywhere (v6)
[12] 53/udp (v6)                ALLOW IN    Anywhere (v6)
[13] 53/tcp (v6)                ALLOW IN    Anywhere (v6)

$ sudo ufw insert 2 deny from 45.134.5.0/24 to any port 80
Rule inserted
```

```
$ sudo ufw insert 3 deny from 45.134.5.0/24 to any port 443
Rule inserted
$ sudo ufw status numbered
Status: active

     To                         Action      From
     --                         ------      ----
[ 1] 22/tcp                     ALLOW IN    Anywhere
[ 2] 80                         DENY IN     45.134.5.0/24
[ 3] 443                        DENY IN     45.134.5.0/24
[ 4] 80/tcp                     ALLOW IN    Anywhere
[ 5] 443/tcp                    ALLOW IN    Anywhere
[ 6] 51820/udp                  ALLOW IN    Anywhere
[ 7] 53/udp                     ALLOW IN    Anywhere
[ 8] 53/tcp                     ALLOW IN    Anywhere
[ 9] 3306                       ALLOW IN    192.168.100.0/24
```

Voilà vous savez protéger votre serveur au niveau réseau !

Dernier point à ne pas oublier, vérifiez que le pare-feu soit lancé au démarrage de la machine :

```
$ more /etc/ufw/ufw.conf
# /etc/ufw/ufw.conf
#

# Set to yes to start on boot. If setting this remotely, be sure to add a rule
# to allow your remote connection before starting ufw. Eg: 'ufw allow 22/tcp'
ENABLED=yes
```

5. BLOCAGE DES ATTAQUES PAR FORCE BRUTE AVEC FAIL2BAN

Principe de fonctionnement

Maintenant que nous avons doté nos serveurs d'un pare-feu, voyons si nous pouvons en automatiser le blocage pour bloquer des attaquants qui tenteraient de forcer l'accès à un de nos services.

Fail2ban est un logiciel de blocage au niveau réseau qui va nous permettre de réaliser cette tâche de manière transparente.

Son fonctionnement est basé sur le principe suivant :

- Chaque tentative d'accès infructueuse réalisée par les attaquants génère le plus souvent une ligne dans les logs de l'application visée.
- Fail2ban scanne en permanence ces logs à la recherche de ces tentatives d'accès.
- Quand il en trouve une, il incrémente un compteur et le rattache à l'adresse IP en question.
- Si ce compteur d'erreurs dépasse un certain niveau pour une période de temps donnée (les 2 indicateurs sont paramétrables), Fail2ban bloque l'adresse IP en question via le pare-feu. La durée de blocage est aussi paramétrable.

Mise en œuvre

Installation

Installons tout d'abord Fail2ban :

```
$ sudo apt install fail2ban
```

Contrôle de bon fonctionnement

Via l'outil en ligne de commandes de Fail2ban, il nous est immédiatement possible de voir son état :

```
$ sudo fail2ban-client status
Status
|- Number of jail: 1
`- Jail list: sshd
```

Fail2ban est donc actif et dispose d'un service protégé par défaut, notre serveur SSH.

Valeurs par défaut

Les fichiers et répertoires par défaut sont les suivants sous Debian :

- **/etc/fail2ban/jail.conf** : fichier de configuration par défaut.
- **/etc/fail2ban/jail.d/defaults-debian.conf** : sous une distribution Debian, c'est dans ce fichier que le lancement automatique de la protection du serveur SSH est activé.
- **/etc/fail2ban/action.d** : ce dossier contient toutes les actions déjà définies avec notamment les différentes interfaces de pare-feu comme Iptables, Pf, mais aussi Cloudflare pour les serveurs dans le cloud protégés par ce fournisseur ;
- **/etc/fail2ban/filter.d** : ce dossier contient de la même manière la définition des expressions régulières permettant de trouver les échecs de connexions pour tous les logiciels déjà définis.

Première configuration

Notre première action est de créer un fichier de configuration local qui ne sera pas écrasé lors de la prochaine mise à jour. De même, sa présence fait que ses directives prendront le pas sur la configuration par défaut :

```
$ cp /etc/fail2ban/jail.conf /etc/fail2ban/jail.local
```

Fail2ban utilisant par défaut Iptables comme interface de gestion de pare-feu, il convient de modifier notre fichier pour que Fail2ban utilise UFW.

Pour cela, nous effectuons les actions suivantes :

- ouverture du fichier **jail.local** nouvellement créé ;
- recherche des deux directives **banaction** et **banaction_allports = ufw** ;
- saisie des valeurs suivantes à la place : **banaction = ufw** et **banaction_allports = ufw**.

Dans ce fichier, vous trouverez aussi des valeurs par défaut que vous pourrez ajuster à votre besoin :

```
[...]
ignoreip = 127.0.0.1/8 ::1
bantime  = 10m
findtime  = 10m
Maxretry = 5
[...]
```

Ces valeurs fixent respectivement :

- les adresses à ne pas interdire malgré les erreurs de connexions (veillez à y mettre par exemple l'adresse IP publique de votre box ou de votre pare-feu d'entreprise ;
- la durée de l'exclusion par défaut ;
- la durée de la plage de temps pendant laquelle on compte les essais infructueux de connexion ;
- et le nombre d'échecs avant d'exclure l'adresse IP.

Ces valeurs peuvent être ensuite aussi modifiées dans les sections de chaque logiciel supervisé plus loin dans le fichier. Ces sections, appelées *jail*, sont repérées par la chaîne de caractères du logiciel surveillé (**[roundcube-auth]** dans l'exemple ci-dessous).

Exemple de la *jail* pour le webmail Roundcube :

```
[roundcube-auth]
port     = http,https
logpath  = %(roundcube_errors_log)s
```

» Contrôle de bon fonctionnement

Après avoir fait vos modifications et relancé le service, vérifions que Fail2ban fonctionne pour sa *jail* **sshd** :

```
$ sudo fail2ban-client status sshd
Status for the jail: sshd
|- Filter
|  |- Currently failed: 27
|  |- Total failed: 454
|  `- File list:/var/log/auth.log
`- Actions
   |- Currently banned: 15
   |- Total banned: 38
   `- Banned IP list: 64.227.36.108 61.177.172.13 212.64.5.28 218.92.0.249
175.24.17.53 222.186.169.194 51.158.111.168 31.125.100.24 167.99.166.195 51.68.190.223
112.35.62.225 222.186.173.154 159.89.199.229 82.196.15.195 134.209.12.115
```

Nous avons donc 15 adresses qui ont été bannies par Fail2ban.

Pour savoir si ce blocage est réellement effectif et cohérent, contrôlons si ces adresses sont bien bloquées aussi dans UFW :

```
$ sudo ufw status
Status: active

To                         Action      From
--                         ------      ----
Anywhere                   REJECT      87.138.254.133
Anywhere                   REJECT      58.246.187.102
Anywhere                   REJECT      134.209.12.115
Anywhere                   REJECT      82.196.15.195
Anywhere                   REJECT      159.89.199.229
Anywhere                   REJECT      222.186.173.154
Anywhere                   REJECT      112.35.62.225
Anywhere                   REJECT      51.68.190.223
Anywhere                   REJECT      167.99.166.195
Anywhere                   REJECT      31.125.100.24
Anywhere                   REJECT      51.158.111.168
Anywhere                   REJECT      222.186.169.194
Anywhere                   REJECT      175.24.17.53
22/tcp                     ALLOW       Anywhere
80/tcp                     ALLOW       Anywhere
443/tcp                    ALLOW       Anywhere
51820/udp                  ALLOW       Anywhere
53/udp                     ALLOW       Anywhere
53/tcp                     ALLOW       Anywhere
3306                       ALLOW       192.168.100.0/24
```

Le blocage est effectif et cohérent avec les données fournies par Fail2ban.

» Aller plus loin

Vous pouvez ensuite faire quelques actions plus poussées :

- Gérer les notifications : l'action de notification par défaut (**action = %(action_)s**) est de ne rien signaler lors du bannissement l'adresse IP détectée. Les autres actions de notification possibles sont la notification par mail, avec ou sans extraction du Whois, avec ou sans extraction des lignes de logs incriminées) avec respectivement les syntaxes suivantes :

→ **action = %(action_m)s** ;

→ **action = %(action_mw)s** ;

→ **action = %(action_mwl)s**.

- Créer un filtre pour une application pas encore supportée :

→ Vous pourrez prendre modèle sur un des fichiers présents dans **/etc/fail2ban/filter.d** pour créer une surveillance pour une application qui vous intéresse. Commencez par copier un des fichiers en le nommant du nom de votre application et modifiez-le.

→ La ligne clé est la ligne commençant par **failregex =**. Elle permet de fixer l'expression régulière assurant la détection de la ligne de log identifiant l'échec de connexion.

→ Pour tester le bon fonctionnement de votre expression régulière, vous pouvez utiliser cette ligne de commandes :

```
$ fail2ban-regex /chemin/du/fichier-de-log /etc/fail2ban/monapplication.conf
```

→ Une fois que votre filtre est opérationnel, vous pouvez l'activer de cette manière :

- Dans le fichier **jail.local**, créez une *jail* du nom du fichier de votre filtre.
- Si notre fichier de filtre est **/filter.d/monapplication.conf**, vous pourrez y faire référence au sein du fichier **jail.local** en utilisant le nom de *jail* suivant : **[monapplication]**.
- Vous pourrez activer cette *jail* en ajoutant dans sa définition la ligne **enabled = true**.

Juste pour vous donner la quantité d'applications déjà définies que vous pouvez protéger des tentatives d'accès massives grâce à Fail2ban, en voici la liste dans le dossier **filter.d** :

```
$ ls filter.d/
3proxy. botsearch-common.conf exim-spam.conf mysqld-auth.conf portsentry.
conf sogo-auth.conf apache-auth.conf common.conf freeswitch.conf nagios.
conf postfix.conf solid-pop3d.conf apache-badbots.conf counter-strike.
conf froxlor-auth.conf named-refused.conf proftpd.conf squid.conf apache-
botsearch.conf courier-auth.conf  groupoffice.conf nginx-botsearch.
conf pure-ftpd.conf squirrelmail.conf apache-common.conf courier-smtp.
conf gssftpd.conf nginx-http-auth.conf qmail.conf sshd.conf apache-
fakegooglebot.conf cyrus-imap.conf guacamole.conf nginx-limit-req.
conf recidive.conf stunnel.conf apache-modsecurity.conf directadmin.
conf haproxy-http-auth.conf  nsd.conf                roundcube-auth.conf
suhosin.conf apache-nohome.conf domino-smtp.conf horde.conf openhab.conf
screensharingd.conf   tine20.conf apache-noscript.conf dovecot.conf
ignorecommands          openwebmail.conf  selinux-common.conf   uwimap-
auth.conf apache-overflows.conf dropbear.conf          kerio.conf
oracleims.conf selinux-ssh.conf  vsftpd.conf apache-pass.conf drupal-auth.
conf       lighttpd-auth.conf pam-generic.conf sendmail-auth.conf webmin-
auth.conf
apache-shellshock.conf  ejabberd-auth.conf mongodb-auth.conf  perdition.
conf sendmail-reject.conf  wuftpd.conf assp.conf exim-common.conf monit.
conf phpmyadmin-syslog.conf sieve.conf xinetd-fail.conf
asterisk.conf exim.conf murmur.conf php-url-fopen.conf slapd.conf
zoneminder.conf
```

5. MISE EN PLACE D'UN RÉSEAU PRIVÉ ENTRE NOS SERVEURS AVEC WIREGUARD

Comme nous l'avons abordé lors du chapitre sur la sécurité en profondeur, nous allons installer un VPN permettant de mettre en place un réseau privé ouvert entre nos serveurs. Les services déployés sur ces serveurs pourront donc communiquer de manière sécurisée et libre tout en étant protégés d'Internet. Ainsi, en tant qu'administrateur, nous pourrons nous connecter sur ces serveurs depuis notre ordinateur.

Voici, en figure 1, le schéma d'architecture mis à jour avec cet ajout :

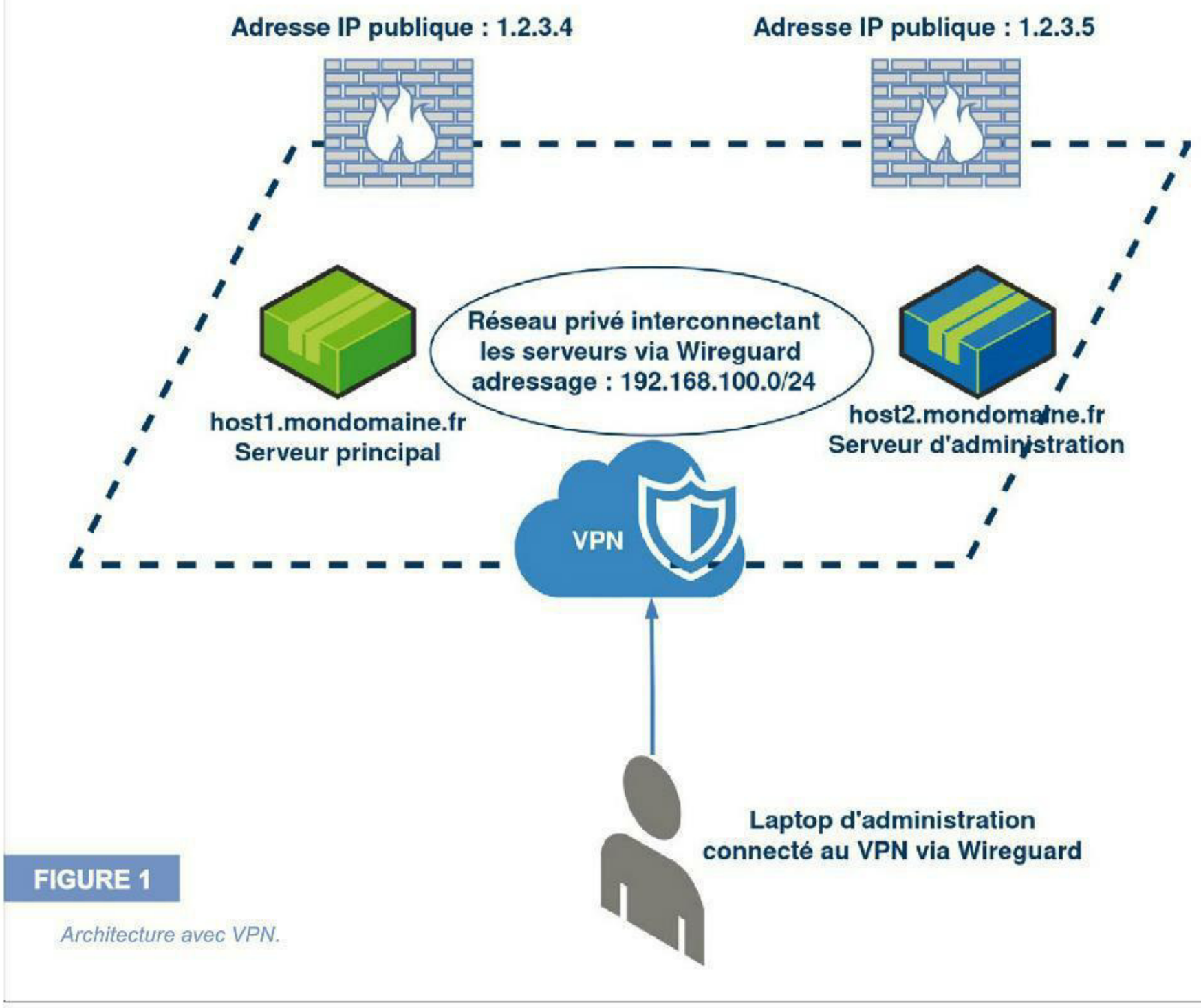


FIGURE 1

Architecture avec VPN.

Mise en place

Nous allons installer Wireguard de la manière suivante :

- La machine serveur de notre VPN Wireguard sera le serveur **host2.mondomaine.fr** de notre infrastructure. Il s'agit là d'un choix de répartition des rôles totalement arbitraire pouvant être remis en cause.

- Les adresses des machines sur ce réseau privé seront :
  - → 192.168.100.1 : le serveur d'administration, assurant le rôle de serveur Wireguard ;
  - → 192.168.100.2 : le serveur primaire de notre infrastructure ;
  - → 192.168.100.3 : l'ordinateur portable de l'administrateur.

» Installation

Nous installons le logiciel sur toutes les machines (serveurs et poste client de l'administrateur).

```
$ sudo apt install wireguard
```

Tous les fichiers de configuration sont créés sous **/etc/wireguard**.

Générons ensuite les clés nécessaires au bon fonctionnement sur chacun des serveurs et de l'ordinateur portable de l'administrateur :

```
$ sudo cd /etc/wireguard/
$ umask 077; wg genkey | tee privatekey | wg pubkey > publickey
```

» Configuration de la partie serveur de Wireguard

Nous activons tout d'abord le routage sur le serveur **host2.mondomaine.fr** :

```
$ sysctl -w net.ipv4.ip_forward=1
$ sysctl -w net.ipv6.conf.all.forwarding=1
$ sysctl -p
```

Ensuite, nous créons la configuration Wireguard côté serveur.

Après avoir récupéré le nom de votre carte réseau portant votre adresse IP publique grâce à la commande **ifconfig** (dans mon cas, il s'agit de l'interface **eno1**), nous allons créer le fichier de configuration **/etc/wireguard/wg0.conf** et le compléter avec une première partie de la configuration du serveur :

```
[Interface]
Address =  192.168.100.1/24
ListenPort = 51820
PostUp = iptables -A FORWARD -i wg0 -j ACCEPT; iptables -t nat -A
POSTROUTING -o eno1 -j MASQUERADE
PostDown = iptables -D FORWARD -i wg0 -j ACCEPT; iptables -t nat -D
POSTROUTING -o eno1 -j MASQUERADE
PrivateKey = COPIER_ICI_LA_CLE_PRIVEE_DU_SERVEUR
SaveConfig = true
```

Détaillons quelques-unes des options :

- **Address** : cela permet de fixer l'adresse IP du serveur au sein du VPN. Ici ce sera 192.168.100.1. Point important : le masque **/24** permet de dire à Wireguard que les adresses du réseau VPN vont de 192.168.100.0 à 192.168.100.255.
- **PostUp/Down** : ces lignes permettent de mettre en place des règles iptables de translation d'adresses à l'activation du VPN et de les inactiver à son arrêt. Cela permet de router les paquets réseau venant des clients vers Internet quand c'est nécessaire.

Note : veillez à bien valoriser la configuration ci-dessus avec :

- le nom de votre interface réseau (comme dit plus haut, dans mon cas **eno1**) ;
- le contenu de votre fichier **privatekey** généré à l'étape précédente.

On modifie ensuite les droits du fichier afin d'en assurer la confidentialité (lecture possible uniquement par l'utilisateur root) :

```
$ sudo chmod 600 /etc/wireguard/wg0.conf
```

Puis nous activons le VPN via les commandes **wg-quick up wg0** et **wg show**.

» Configuration sur les clients Wireguard

Ici, cela concerne donc la machine **host1.mondomaine.fr** et l'ordinateur portable de l'administrateur.

Créons et remplissons le fichier de configuration **/etc/wireguard/wg0.conf** :

```
[Interface]
Address = 192.168.100.2 ou 3 (respectivement adresse IP VPN du serveur
host1.mondomaine.fr et celle de laptop de l'administrateur)
PrivateKey = COPIER_ICI_LA_CLE_PRIVEE_DU_CLIENT

[Peer]
PublicKey = COPIER_ICI_LA_CLE_PUBLIQUE_DU_SERVEUR
AllowedIPs = 192.168.100.0/24
Endpoint = COPIER_ICI_L_ADRESSE_IP_PUBLIQUE_DU_SERVEUR:51820
```

Corrigeons enfin les permissions du fichier de configuration :

```
# chmod 600 /etc/wireguard/wg0.conf
```

La dernière étape de la configuration est de reporter les informations de ces 2 machines (**host1** et le laptop de l'administrateur) dans le fichier de configuration du serveur **/etc/wireguard/wg0.conf**. Nous ajoutons à la fin du fichier le paragraphe suivant qui décrit la configuration du client avec notamment sa clé publique :

```
[Peer]
PublicKey =  COPIER_ICI_LA_CLE_PUBLIQUE_DU_CLIENT
AllowedIPs = ADRESSE_IP_DU_CLIENT/32  (ici ce sera donc 192.168.100.2
ou 3)
```

Important : notez bien qu'il faut avoir stoppé le VPN avant de modifier le fichier si on ne veut pas voir sa modification écrasée par l'arrêt du VPN.

Si vous souhaitez ne pas générer de coupure du serveur VPN, vous pouvez utiliser la ligne de commandes suivante :

```
# sudo wg set wg0 peer COPIER_ICI_LA_CLE_PUBLIQUE_DU_CLIENT allowed-ips
ADRESSE_IP_DU_CLIENT/32
```

Il suffit de saisir la même commande tant du côté serveur que côté client :

```
# sudo wg-quick up wg0
```

Nous avons désormais un réseau privé opérationnel entre nos 3 machines permettant aux différents services réseaux des 3 machines de dialoguer entre eux sans augmenter la surface d'attaque de notre infrastructure. En effet, tous ces échanges sont totalement inaccessibles aux machines localisées sur Internet.

CONCLUSION

Nous pouvons maintenant déployer et administrer des services au sein d'une infrastructure sécurisée au sens réseau :

- Nous avons réduit le nombre de services en écoute sur Internet au strict nécessaire.
- Les flux réseaux sont contrôlés par un pare-feu.
- Les tentatives de connexion par force brute sont détectées et rejetées par le logiciel Fail2ban.
- Les machines de notre infrastructure sont interconnectées entre elles par un réseau privé virtuel qui sécurise leur communication interne.

Nous allons pouvoir nous concentrer sur la mise en place d'outils nous permettant de détecter et d'investiguer en cas d'attaques informatiques, réussies ou non. Mais avant, intéressons-nous un peu à un des couteaux suisses de la sécurité et de l'administration système, OpenSSH. ▌

RÉFÉRENCES

[1] **https://www.python.org/dev/peps/pep-0476/**

[2] **https://en.wikipedia.org/wiki/MariaDB**

DOSSIER

RÉPONDEZ AUX PROBLÉMATIQUES DE SÉCURITÉ D'ACCÈS AVEC OPENSSH

Christophe BROCAS

Notre infrastructure est désormais stable et sécurisée tant au niveau système que réseau. Nous allons pouvoir étudier de manière un peu approfondie un logiciel particulier : OpenSSH. Ce démon réseau nous permet de nous connecter en toute sécurité sur nos serveurs via le protocole SSH. Son développement a commencé il y a plus de 20 ans chez nos amis d'OpenBSD. La liste de ses fonctionnalités est d'une longueur impressionnante. Nous allons en parcourir ensemble quelques-unes qui, je l'espère, nous permettront d'améliorer tant notre sécurité que notre productivité quotidienne.

1. SSH : LA SÉCURITÉ D'ACCÈS PAR LA CRYPTOGRAPHIE

Les plus vieux d'entre nous (ou les administrateurs réseau ;-)) ont connu les affres des protocoles d'accès en clair comme Telnet ou Rlogin. Grâce à OpenSSH, il n'a pas fallu attendre les révélations d'Edward Snowden pour que les administrateurs de systèmes d'exploitation libres fassent confiance à la cryptographie pour accéder à leurs serveurs en toute sécurité.

OpenSSH a implémenté la version 2 du protocole SSH. Ce protocole utilise notamment la cryptographie asymétrique pour authentifier et chiffrer le trafic entre un client et un serveur. De nouvelles primitives cryptographiques sont régulièrement implémentées par OpenSSH afin de se prémunir des vulnérabilités trouvées au fil des ans au sein de ces fonctions par la communauté de la recherche en sécurité offensive.

L'utilisation de SSH s'est diffusée bien au-delà du simple cas d'usage de la connexion à des serveurs : il est aussi utilisé lors de transfert de fichiers (SFTP, SCP), d'encapsulation de trafic tiers (X11) ou d'implémentation de système de fichiers distants (SSHFS). Dans tous ces cas d'usage, OpenSSH apporte son organisation en 3 couches :

- la couche de transport : elle assure l'intégrité et la confidentialité du transport des données entre le client et le serveur. C'est dans cette partie que sont échangées les clés et que sont négociés les algorithmes de cryptographie asymétrique, de chiffrement symétrique, d'authentification de messages et de hash.
  C'est aussi lors de cette phase de la connexion que le client découvre le serveur. Lors de la première connexion SSH d'un client à un serveur, le principe de confiance appliqué est le TOFU (*Trust On First Use*). Le client reçoit l'empreinte de la clé du serveur. L'utilisateur doit vérifier de manière manuelle que cette clé est bien celle du serveur. Une fois cette vérification effectuée, le client fera confiance à cette clé lors des connexions suivantes à ce serveur. La clé sera stockée dans le fichier `/.ssh/known_hosts`. L'utilisateur ne recevra une alerte de sécurité que si cette clé change de valeur.
- la couche d'authentification : une fois la couche de transport sécurisée mise en place, la couche d'authentification permet à l'utilisateur de s'authentifier sur le serveur avec les méthodes que ce dernier supporte : par mot de passe, avec un couple clé privée/clé publique, etc.
- la couche de connexion : le client étant authentifié, les échanges applicatifs peuvent désormais s'effectuer. Ils se font au travers de multiples tunnels qui sont multiplexés au sein de la connexion unique SSH désormais ouverte.

Voyons ensemble comment tirer le meilleur parti des fonctionnalités d'OpenSSH !

2. CONNEXION : CLÉ VS MOT DE PASSE

Mot de passe un jour, mot de passe toujours ?

C'est en effet le risque qui nous guette tous ! Lorsque nous installons un serveur, la première chose que nous demande le système en général, c'est d'initialiser le mot de passe de notre utilisateur. Cela a tendance à pousser l'administrateur à se connecter avec ce simple moyen d'authentification.

Or, même si vous le générez et le stockez au sein de votre gestionnaire de mots de passe, n'utiliser qu'un simple mot de passe affaiblit très fortement la sécurité d'accès à vos serveurs.

En effet, un rapide espionnage au-dessus de notre épaule lors de l'établissement d'une connexion permettra à un attaquant de nous subtiliser notre mot de passe. Ce vol aura eu raison de toute la sécurité que nous aurons pu déployer par ailleurs sur nos serveurs. Il s'agit d'une parfaite illustration du fait que la sécurité d'une chaîne se limite à celle de son maillon le plus faible.

Que faire donc pour améliorer la situation ?

Connexion par clé publique/clé privée

Commençons par comprendre rapidement l'intérêt de la cryptographie asymétrique que nous allons utiliser pour nous connecter en SSH à nos serveurs.

Chaque membre d'un échange chiffré par cryptographie asymétrique possède deux clés prenant la forme de deux fichiers différents :

1. une clé privée : le contenu de ce fichier ne doit être connu que de vous. Son accès est protégé par un mot de passe. Il sert à signer des messages ou à déchiffrer des messages qui ont été chiffrés par votre interlocuteur en utilisant votre clé publique.
2. une clé publique : sa valeur est connue de tous vos interlocuteurs. Elle leur sert notamment à chiffrer des messages à votre destination ou à vérifier que c'est bien vous qui avez signé un message.

Le principal intérêt sécurité d'utiliser ces clés publique et privée est que l'attaquant doit nous voler désormais 2 choses :

- une chose que l'on possède : le fichier contenant notre clé privée ;
- une chose que l'on connaît : le mot de passe protégeant l'accès à cette clé privée.

Nous sommes donc dans un schéma de connexion à multiples facteurs.

Création d'une clé SSH à l'état de l'art en 2020

En 2020, la cryptographie asymétrique pour SSH peut utiliser 2 grandes familles d'algorithmes :

- ceux factorisant de grands nombres premiers comme RSA ou DSA ;
- ceux reposant sur des courbes elliptiques comme ECDSA ou Ed25519.

L'Agence Nationale de Sécurité des Systèmes d'Information recommande l'usage des courbes elliptiques et plus particulièrement de ECDSA dans un document datant déjà de 2015. Cette courbe elliptique sert dans l'algorithme de signature électronique à base de clé privée.

Cependant, l'état de l'art actuel nous pousserait plutôt à utiliser Ed25519 du fait d'un doute pesant sur les entrants fournis par le NIST pour faire fonctionner ECDSA **[1]**.

Le principal avantage de ces algorithmes à base de courbes elliptiques est qu'ils permettent l'usage de clés de très petites tailles par rapport à celles nécessaires pour RSA (respectivement 256 bits et 3072 bits pour un même niveau de sécurité).

» Création d'une clé Ed25519

Voici donc comment créer une clé de type Ed25519 de longueur de 256 bits. Nous l'associerons à notre adresse e-mail afin de pouvoir l'identifier plus aisément :

```
$ ssh-keygen -t ed25519 -b 256 -C christophe@brocas.org
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/cb/.ssh/id_ed25519):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/cb/.ssh/id_ed25519
Your public key has been saved in /home/cb/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:U6pTLGHC0EZt6w11NGeZeAJtESG5ncsS0kC+RJ037tA christophe@brocas.
org
The key's randomart image is:
+--[ED25519 256]--+
|   .o.oo.o**=oo   |
|    ooo+ =.O++    |
|    .o.== O.=     |
|      +++=oE      |
|      .o+S= .     |
|       .+o.+      |
|       o  .       |
|          .       |
|                  |
+----[SHA256]-----+
```

Par facilité de langage, nous dirons donc désormais que nous sommes en possession d'une clé SSH de type Ed25519.

Clés pour accéder à des distributions plus anciennes

Le seul inconvénient à l'usage de clés basées sur des courbes elliptiques est leur manque de compatibilité avec des plateformes anciennes. Si vous devez accéder à ce type de plateformes (ex. : inférieures à Debian 8), la recommandation sera d'utiliser des clés de type RSA d'une longueur de 4096 bits :

```
$ ssh-keygen -t rsa -b 4096
```

Sécurité d'accès de nos clés

1. Mot de passe : lors de la création de la clé, veillez bien à saisir un mot de passe. Il sera nécessaire pour tout accès à votre clé privée. Ce mot de passe permet de bien avoir un second facteur d'authentification (quelque chose que l'on sait) en plus du premier facteur (quelque chose que l'on possède, ici le fichier de clé privée).

2. Droits d'accès aux fichiers : nos clés publique (`id_ed25519.pub`) et privée (`id_ed25519`) sont stockées sur notre disque dur, sous l'arborescence `~/.ssh` :

```
$ ls -al ~/.ssh/id_ed*
-rw------- 1 cb cb 464 août  10 11:04 /home/cb/.ssh/id_ed25519
-rw-r--r-- 1 cb cb 103 août  10 11:04 /home/cb/.ssh/id_ed25519.pub
```

Vérifiez bien que les droits d'accès sont positionnés comme ci-dessus : accès limité strictement à votre utilisateur pour le fichier de clé privée et en lecture seule pour la clé publique pour les autres utilisateurs.

Copions notre clé sur nos serveurs

La prochaine étape est de déployer notre clé sur nos serveurs afin qu'ils nous proposent la connexion par clé lors de la prochaine connexion :

```
$ ssh-copy-id -i ~/.ssh/id_ed25519 username@host1.mondomaine.fr
```

Il vous sera demandé le mot de passe de l'utilisateur `username` et la clé sera ajoutée au fichier `~/.ssh/authorized_keys2` de l'utilisateur `username` sur le serveur **host1.mondomaine.fr**. La prochaine connexion à ce serveur devrait se faire via la clé SSH dont vous venez de déposer la partie publique sur le serveur (et plus précisément dans le fichier `AuthorizedKeys` de cet utilisateur sur le serveur).

À noter : le prochain mot de passe qui vous sera demandé sera le mot de passe de la clé privée et non pas le mot de passe du compte utilisateur sur le serveur comme précédemment.

POINT POUR LES AUDITEURS SÉCURITÉ : DÉTECTION DES CLÉS SSH SANS PASSPHRASE (OU MOT DE PASSE)

Nous avons vu qu'il est du ressort de l'utilisateur de créer sa propre clé SSH. Libre à lui de la créer avec ou sans un mot de passe.

Or, en tant que responsable de la sécurité de votre système d'information, vous pouvez exiger que les clés privées de vos administrateurs soient protégées par une passphrase (ou mot de passe).

Pour détecter les clés privées non protégées par mot de passe, vous pouvez utiliser la commande `ssh-keygen` avec les paramètres y et f. Voici le résultat avec votre clé protégée par mot de passe (nom de la clé : `id_ed25519`) :

```
$ ~/.ssh$ ssh-keygen -y -f id_ed25519
Enter passphrase:
```

Elle vous demande de saisir le mot de passe.

Dans le cas d'une clé non protégée par mot de passe (nom de la clé : `id_test`), le résultat est le suivant :

```
~/.ssh$ ssh-keygen -y -f id_test
ssh-rsa AAAAB3NzaC1yc2EAAAADA[...]s3R
```

Vous obtenez le contenu de la clé privée (ici le résultat est volontairement tronqué ;-)).

3. CÔTÉ SERVEUR : SÉCURISATION D'OPENSSH

Le fichier de configuration du serveur OpenSSH est localisé sous `/etc/ssh/sshd_config`.

Nous allons mettre en place quelques paramètres afin notamment d'être homogènes avec nos choix de connexion :

- `HostKey /etc/ssh/ssh_host_ed25519_key` : nous demandons ici au serveur d'utiliser la clé de type Ed25519. Les clés du serveur ont été générées à l'installation du package d'OpenSSH.

  Vous pouvez souhaiter les régénérer. Par exemple, lors de la réutilisation de la même ISO pour lancer plusieurs machines virtuelles. Vous pouvez le faire via la commande suivante puis en redémarrant votre service OpenSSH :

```
# dpkg-reconfigure openssh-server
```

- `PasswordAuthentication no` et `PubkeyAuthentication yes` : vous n'autorisez ici que les connexions par clés et vous interdisez les connexions par mot de passe.
- `UsePrivilegeSeparation sandbox` : vous forcez votre service OpenSSH à s'exécuter selon le principe du moindre privilège, ce qui limite les impacts en cas de vulnérabilité sur le service OpenSSH.
- `PermitRootLogin no` : l'utilisateur de la connexion ne peut être root. Cela permet de renforcer la bonne pratique de la connexion avec son utilisateur personnel et de l'usage du `sudo` pour effectuer les actions à privilèges. La combinaison de ces deux actions assure un niveau de traçabilité satisfaisant en cas d'incident de sécurité.

Dans le cadre d'un audit ponctuel ou régulier de votre parc, il est possible d'automatiser ce contrôle via le script suivant :

```
#!/bin/bash

for l in $(ls $HOME/.ssh/*.pub)
do
   key=${l%.*}
   if SSH_ASKPASS=/bin/false ssh-keygen -y -f $key < /dev/null > /dev/null 2>&1
   then
       echo -n "NO password: "
   else
       echo -n "Protected by a password: "
   fi
   echo ${key}
done
```

Nous nommerons ce script `check_ssh_keys_password.sh` et nous l'exécutons :

```
~/.ssh$ ./check_ssh_keys_password.sh
Protected by a password: /home/cb/.ssh/id_assurance-maladie
Protected by a password: /home/cb/.ssh/id_ed25519
NO password: /home/cb/.ssh/id_sync
~/.ssh$
```

Simple, mais efficace :-)

Note : si vous devez absolument autoriser la connexion sous l'utilisateur root, utilisez la valeur `prohibit-password` de la directive `PermitRootLogin`. Cela ne permettra une telle connexion qu'avec une connexion par clé SSH, et pas en simple utilisateur+mot de passe.

- `ListenAddress <address>` et `Ports <port number>` : ces deux paramètres réseaux permettent de fixer respectivement l'adresse et le port d'écoute d'OpenSSH. En effet, OpenSSH écoute par défaut sur le port 22 de toutes les interfaces réseaux de la machine.

  → L'option `ListenAddress` vous permet par exemple de ne faire écouter OpenSSH que sur l'adresse de votre serveur sur le sous-réseau Wireguard. Ainsi, vous n'aurez pas de démon SSH accessible depuis Internet. Je ne peux pas vous faire de recommandation stricte de mon côté, c'est un choix que vous devez faire en fonction de votre modèle de menace.

  → L'option `Port` quant à elle vous permet de fixer le port d'écoute. Si vous ne souhaitez pas avoir le port 22 utilisé, pensez à fixer un autre port privilégié, c'est-à-dire inférieur à 1024. Vous éviterez ainsi les possibles usurpations par des services s'exécutant sur des ports non administrateur.

Stockage de clés serveurs dans le DNS et vérification en ligne

Nous allons positionner un dernier réglage de sécurité réseau qui ne se trouve pas cette fois dans le fichier de configuration.

Nous avons compris que la confiance mise dans une connexion SSH vient en partie du fait que l'on accepte de manière éclairée la clé présentée par le serveur lors de la première connexion (principe *Trust On First Use*). Or, il faut le reconnaître, beaucoup d'entre nous ne vérifient pas cette valeur. Ils font une confiance aveugle à la valeur présentée par ce serveur qui, pour le coup, peut être un serveur malveillant.

OpenSSH nous permet de stocker dans le DNS au travers de l'enregistrement SSHFP les valeurs des clés publiques d'un serveur OpenSSH. Il s'agit d'un enregistrement DNS standard permettant de faire correspondre le hash de la clé publique d'un serveur SSH avec son nom DNS. Chaque enregistrement SSHFP précise aussi le type de hash et le type de la clé. Lors de la connexion au serveur, l'utilitaire SSH peut ainsi opérer une vérification automatique de la clé présentée par le serveur via une requête DNS.

Mise en œuvre de SSHFP

```
# ssh-keygen -r host5.brocas.org.
host5.brocas.org. IN SSHFP 1 1 92af45f2539056f9bd000534a555c038df3ea8d0
host5.brocas.org. IN SSHFP 1 2
b533adab8ea6477a4f3f0c42f65f653547ce8eaa24d0b9811fd322d3f72db8dd
host5.brocas.org. IN SSHFP 2 1 148c96df300c4ff75ddf23998ceb67b5f52f7dfe
host5.brocas.org. IN SSHFP 2 2
8998a67502b8f027322b2a43587b48696f45f77dc967bd1d82278a0df499d33a
host5.brocas.org. IN SSHFP 3 1 c0b49a39d47460460db8c5d3695e103b8f745506
host5.brocas.org. IN SSHFP 3 2
001e4511643763eb9a8cbfe9544218a5b2af015cbb31cab7c3e8da91ff0294f9
host5.brocas.org. IN SSHFP 4 1 ef00fe8de411a463c2bcc56303e7e7fac8a6d3f6
host5.brocas.org. IN SSHFP 4 2
d11c845ef421af07c5a35785b94371fd36f31b29239056ebf6b7cee44a8b3703
```

La commande `ssh-keygen` nous donne la configuration DNS à insérer dans notre zone DNS. Le champ `SSHFP` a trois paramètres :

- le premier argument indique le type de clé : 1 pour RSA, 2 pour DSA, 3 pour ECDSA et 4 pour Ed25519 ;
- le second argument indique le type de hash présent dans le troisième argument. La valeur 1 est pour SHA1 et 2 indique SHA256 ;
- le troisième est la valeur du hash de la clé.

Voici ce que donne la première connexion à un serveur disposant de ce type d'enregistrement DNS :

```
$ ssh -o "VerifyHostKeyDNS  ask" lx.weberlab.de
The authenticity of host 'lx.weberlab.de (193.24.227.230)' can't be
established.
ECDSA key fingerprint is SHA256:Y51GrS3K8r38vdloB8MV8gO7Yk8PViFATY5xYQzc6oU.
Matching host key fingerprint found in DNS.
Are you sure you want to continue connecting (yes/no/[fingerprint])?
```

La vérification d'enregistrement ne peut pas être forcée par le serveur. Il faut donc que ce soit déclenché par la personne qui se connecte via le paramètre `-o "VerifyHostKeyDNS ask"`. La valeur de la clé présentée par le serveur est alors automatiquement vérifiée. Dans notre cas, la valeur est présente dans le DNS comme l'indique la mention « Matching host key fingerprint found in DNS. » . Si tel n'est pas le cas, vous obtenez le message : « No matching host key fingerprint found in DNS. ».

Point d'attention : cette vérification se fait via une requête DNS qui est un protocole en clair et non signé par défaut. Du coup, à moins d'utiliser DNSSEC, les retours du DNS peuvent eux aussi être corrompus par un attaquant, certes puissant et compétent, positionné entre vous et le DNS. Cette vérification DNS fournit donc un élément supplémentaire de sécurité, mais non probant.

4. CONFIGURATION CÔTÉ CLIENT

Automatisation de connexion via le fichier $HOME/.ssh/config

Ce fichier permet d'automatiser certaines actions lors de la connexion à certaines machines (ou pour toutes les connexions) que vous allez émettre depuis votre machine.

Ce fichier est organisé en sections commençant toutes par la directive **Host** :

```
Host hostname1
    SSH_OPTION value
    SSH_OPTION value

Host hostname2
    SSH_OPTION value

Host *
    SSH_OPTION value
```

Par exemple, une section peut ressembler à ceci :

```
Host host1
    HostName host1.mondomaine.fr
    User christophe
    Port 24
    IdentityFile ~/.ssh/id_ed25519
```

Cela vous permet de simplement saisir la commande **ssh host1** et cela se traduit par la connexion au serveur **host1.mondomaine.fr** sur le port **24** avec l'utilisateur **christophe** en utilisant la clé **id_ed25519**.

Vous pouvez utiliser des caractères joker comme *****, **?**, mais aussi le caractère **!** permettant d'indiquer tout sauf le **hostname**.

```
Host host1
    HostName 192.168.10.51

Host host2
    HostName 192.168.10.52

Host host*
    user christophe

Host * !host1
    LogLevel INFO

Host *
    IdentityFile ~/.ssh/id_ed25519
    Compression yes
    VerifyHostKeyDNS yes
```

Quand vous saisirez **ssh host2**, vous activerez toutes les sections ci-dessus sauf la première :

Vous vous connecterez :

- au serveur d'adresse IP **192.168.10.52** ;
- avec l'utilisateur **christophe** ;
- avec un niveau de log **INFO** ;
- avec la clé **id_ed25519** ;
- la compression est activée ;
- et la vérification de l'empreinte de la clé via le DNS l'est aussi.

Chez votre marchand de journaux

et sur **www.ed-diamond.com**

PAPIER

en kiosque

FLIPBOOK HTML5

sur www.ed-diamond.com

CONNECT

LA DOCUMENTATION TECHNIQUE DES PROS DE L'IT

sur connect.ed-diamond.com

Chez votre marchand de journaux

et sur **www.ed-diamond.com**

» Points à retenir

Il est important de retenir trois points importants :

- Tous les paramètres saisis dans les sections sont autant de choses que vous n'avez pas à saisir lors de vos commandes de connexion.
- La section `Host *` est particulièrement utile pour généraliser un paramètre à toutes vos connexions. Par exemple, nous pouvons ajouter systématiquement la vérification DNS dont on a parlé au paragraphe précédent ou l'utilisation d'une clé donnée.
- Tous les paramètres présents dans le fichier s'appliquent à tous les utilitaires reposant sur OpenSSH comme SCP ou SFTP pour la copie de fichiers.

Utilisons l'agent SSH

L'agent SSH est un logiciel présent sur votre distribution qui est lancé à chaque démarrage de session utilisateur. Il va stocker pour vous le mot de passe de votre ou de vos clés et vous n'aurez plus à le saisir pendant votre session.

Pour lancer l'agent :

```
$ eval `ssh-agent`
```

Voici les commandes pour interagir avec l'agent :

- pour lister les clés déjà présentes :

```
$ ssh-add -l
4096 SHA256:o+WtUVSr8S9JyQT0IAazQzsP8EpeITv7f5xuJyW3m4s christophe.brocas@assurance-maladie.fr (RSA)
256 SHA256:U6pTLGHC0EZt6w11NGeZeAJtESG5ncsS0kC+RJ037tA christophe@brocas.org (ED25519)
```

- pour ajouter les clés ayant des noms standards sous `~/.ssh/`, saisissez simplement :

```
$ ssh-add
```

Mais on peut aussi donner le chemin de la clé à ajouter :

```
$ ssh-add /home/user/keys/id_rsa
```

Il vous sera demandé le mot de passe de votre clé et si vous le saisissez correctement, l'agent stockera votre clé privée. Tous les utilitaires SSH (ssh, scp, sftp...) interagiront directement avec votre agent et vous n'aurez plus à saisir votre mot de passe.

» Comment utiliser sa clé SSH une fois connecté sur les serveurs distants ?

Nous avons une clé privée sur notre machine. La clé publique correspondante est déposée sur les machines sur lesquelles vous pouvez vous connecter.

Mais une fois connecté sur une de ses machines, comment faites-vous pour vous connecter sur d'autres serveurs en utilisant la même clé ? La première solution évidente est de dupliquer la clé privée sur ce premier cercle de machines. Or, l'impact sécurité est évident : votre clé privée serait alors exposée à un risque multiplié de vol ou de perte.

» Que faire ? Utiliser l'agent forwarding (avec précaution !)

L'idée va être de propager l'agent sur la machine intermédiaire via la création d'un socket dédié sur la machine distante. Cet agent intermédiaire peut communiquer au travers dudit socket avec l'agent sur votre machine.

Le souci de sécurité se découvre rapidement. Toute personne pouvant accéder au socket sur la machine intermédiaire pourra se faire passer pour vous. Il faut donc avoir une confiance forte dans la machine sur laquelle on réalise le transfert d'agent.

Règles que nous nous fixons :

1. Pas de transfert d'agent systématique. On ne l'active que vers une machine de confiance.
   La règle à prohiber dans le `~/.ssh/config` est donc :

```
Host *
   ForwardAgent yes
```

2. Forwarder l'agent uniquement de manière opportuniste via un argument de la ligne de commandes lorsque l'on se connecte vers une machine relais où nous aurons réellement besoin de rebondir.

Nous ferons cela via l'usage de l'argument `-A` :

```
$ ssh -A user@hos1.mondomaine
```

5. AU-DELÀ DES CLÉS : OPENSSH ET LES CERTIFICATS

Nous avons compris qu'utiliser les clés augmente sensiblement la sécurité de connexion à notre serveur. Mais cela ajoute aussi de la complexité de gestion, car nous devons :

1. Ajouter les clés des utilisateurs autorisés au fichier `AuthorizedKeys` de tous les serveurs auxquels ils ont le droit d'accéder.
2. Ajouter les clés des serveurs accédés au fichier `known_hosts` de notre machine locale.

Ce type d'environnement avec un nombre notable de serveurs et de personnes y accédant peut donc être un frein à l'usage de clés SSH.

OpenSSH vous propose une solution avec la capacité de créer et utiliser une autorité de certification qui vous permettra de générer des certificats tant pour les administrateurs système que les serveurs.

Les deux inconvénients cités plus haut sont alors remplacés par un simple import de la clé publique de l'autorité de certification dans le fichier de configuration du client et du serveur.

Mise en œuvre

Création de l'autorité de certification :

```
$ cd /etc/ssh && sudo ssh-keygen -f CA
```

Cela va créer 2 fichiers, la clé privée **CA** et la clé publique **CA.pub** de l'autorité de certification sous le chemin **/etc/ssh**. C'est la clé privée de l'autorité qui sera utilisée pour générer des certificats. Il faut donc la protéger et la rendre uniquement accessible à l'utilisateur root.

```
$ ls -al /etc/ssh
-rw-------  1 root root   1766 Aug 16 13:19 CA
-rw-r--r--  1 root root    392 Aug 16 13:19 CA.pub
```

Configuration certificat côté serveur

Nous allons décliner les différentes étapes pour le serveur **host1.mondomaine.fr**.

Génération du certificat pour le serveur **host1.mondomaine.fr** :

```
$ sudo ssh-keygen -h -s /etc/ssh/CA -n host1,host1.mondomaine.fr -I
host1 -V +52w KEYFILE.pub
```

où :

- **-s** : clé privée de l'autorité de certification ;
- **-n** : on indique ici tous les noms d'hôtes du serveur ;
- **-I** : nom court permettant de décrire le certificat ;
- **-V** : durée de validité du certificat (ici 52 semaines) ;
- **KEYFILE.pub** : clé publique du serveur que le certificat va certifier. Ici, on choisira la clé publique **/etc/ssh/id_ed25519.pub**.

Cela génère le fichier certificat suivant :

```
-rw-------  1 root root    411 Jul 28 10:25 ssh_host_ed25519_key
-rw-r--r--  1 root root   1054 Aug 16 14:06 ssh_host_ed25519_key-cert.pub
-rw-r--r--  1 root root    101 Jul 28 10:25 ssh_host_ed25519_key.pub
```

Indiquer au serveur **host1.mondomaine.fr** d'utiliser le certificat.

On ajoute la directive suivante dans le fichier **sshd_config** :

```
HostCertificate /etc/ssh/ssh_host_ed25519_key-cert.pub
```

Le serveur va désormais présenter le certificat aux clients qui se connectent.

Indiquer au serveur la clé publique de l'autorité de certification pour valider les certificats utilisateurs.

Pour finir avec la configuration côté serveur, nous ajoutons aussi la clé publique de l'autorité qui validera les certificats des utilisateurs :

```
TrustedUserCAKeys /etc/ssh/CA.pub
```

Il faudra exécuter toutes ces étapes pour les autres serveurs de notre parc :-)

Configuration certificat côté client

» Création du certificat utilisateur

On exécute la commande suivante pour générer le certificat de chaque utilisateur. Elle ressemble beaucoup à celle pour générer le certificat serveur avec seulement l'option **-h** en moins :

```
$ sudo ssh-keygen -s /etc/ssh/CA -I user_cbrocas -n cbrocas -V +52w id_
ed25519.pub
```

» Indiquer au ssh côté client la clé publique de l'autorité de certification pour valider les certificats serveurs

Pour finir avec la configuration côté client, nous ajoutons aussi la clé publique de l'autorité qui validera les certificats serveurs dans le fichier **.ssh/known_hosts** via la directive **@cert-authority** :

```
@cert-authority *.mondomaine.fr ssh-rsa AAAAB[...]QCmt CA@host5
```

LE WORKFLOW : TOUJOURS LE POINT CRITIQUE DU PROCESSUS DE GESTION D'IDENTITÉ

Il est important de noter que pour générer le certificat client il faut avoir accès à la clé privée de l'autorité de certification. Cela implique que l'opération doit s'exécuter sur le serveur hébergeant cette autorité. Pour créer le certificat de chaque utilisateur, il faut donc :

1. transmettre la clé publique de chaque utilisateur sur ce serveur ;

2. renvoyer à chaque utilisateur son certificat.

Il s'agit là du principal écueil que rencontre tout projet de gestion d'identité : la mise en place d'un workflow à la fois sécurisé pour l'entreprise l'utilisant et en même temps suffisamment fluide pour ne pas ruiner l'expérience utilisateur.

À noter que l'on précise aussi le domaine DNS (`*.mondomaine.fr`) des serveurs dont il faudra vérifier le certificat.

Voilà les configurations liées aux certificats sont effectives !

CONCLUSION

OpenSSH est un merveilleux couteau suisse de la connexion sécurisée. Les quelques fonctionnalités que nous avons parcourues ensemble ne couvrent qu'une petite portion de ses possibilités.

Nous avons pu :

- nous connecter de manière sécurisée via l'usage de clés ou de certificats ;
- automatiser beaucoup d'éléments de connexion via le fichier config ;
- sécuriser la configuration côté serveur ;
- utiliser le DNS pour rajouter des éléments de sécurisation ;
- mettre en place un usage efficace et sécurisé des agents SSH.

Mais il reste beaucoup d'autres fonctionnalités à approfondir comme l'usage des tunnels, des systèmes de fichiers distants ou d'un VPN au-dessus d'OpenSSH. Je vous souhaite plein de belles découvertes et expérimentations avec cet incroyable outil qu'est OpenSSH. ▌

RÉFÉRENCE

[1] **https://www.schneier.com/blog/archives/2013/09/the_nsa_is_brea.html#c1675929**

SAUVEGARDEZ VOS DONNÉES, CENTRALISEZ VOS LOGS ET SUPERVISEZ VOTRE SÉCURITÉ

Christophe BROCAS

Nos serveurs présentent désormais une surface d'attaque réseau maîtrisée et une sécurisation système d'un niveau cohérent avec notre modèle de menaces. De même, le service SSH tournant sur ces serveurs est configuré de manière optimisée. Nous pouvons donc être relativement sereins si nos adversaires sont d'un niveau intermédiaire. Et si malgré toutes ces protections, une attaque comme un rançongiciel réussissait ? Et bien dans ce cas-là, pour l'instant, notre infrastructure serait particulièrement vulnérable. Aucune sauvegarde externalisée. Pas de centralisation des traces. Une supervision sécurité inexistante. Remédions à cette situation afin d'élever le niveau de maturité de la sécurité de notre infrastructure.

1. ET SI NOUS REGARDIONS LE PAYSAGE AVANT DE FONCER ?

Avant d'engager un travail conséquent sur nos serveurs, vérifions que nous n'allons pas faire cela pour rien. Regardons par exemple si des menaces pèsent réellement en 2020 sur les serveurs Linux. En cet été 2020, une telle menace a bien été documentée par des agences gouvernementales défensives américaines.

Description d'une menace sophistiquée pour les serveurs Linux

Une fois n'est pas coutume, la NSA et le FBI ont publié un document extrêmement détaillé de 45 pages sur un malware spécifiquement développé pour les plateformes Linux **[1]**.

Le document en question décrit là la fois les mécanismes d'intrusion, de persistance et d'actions qu'offre le malware Dovrorub à ses opérateurs.

Ce document est produit à destination des administrateurs systèmes et sécurité des organismes gouvernementaux américains afin qu'ils l'utilisent pour protéger au mieux leurs systèmes.

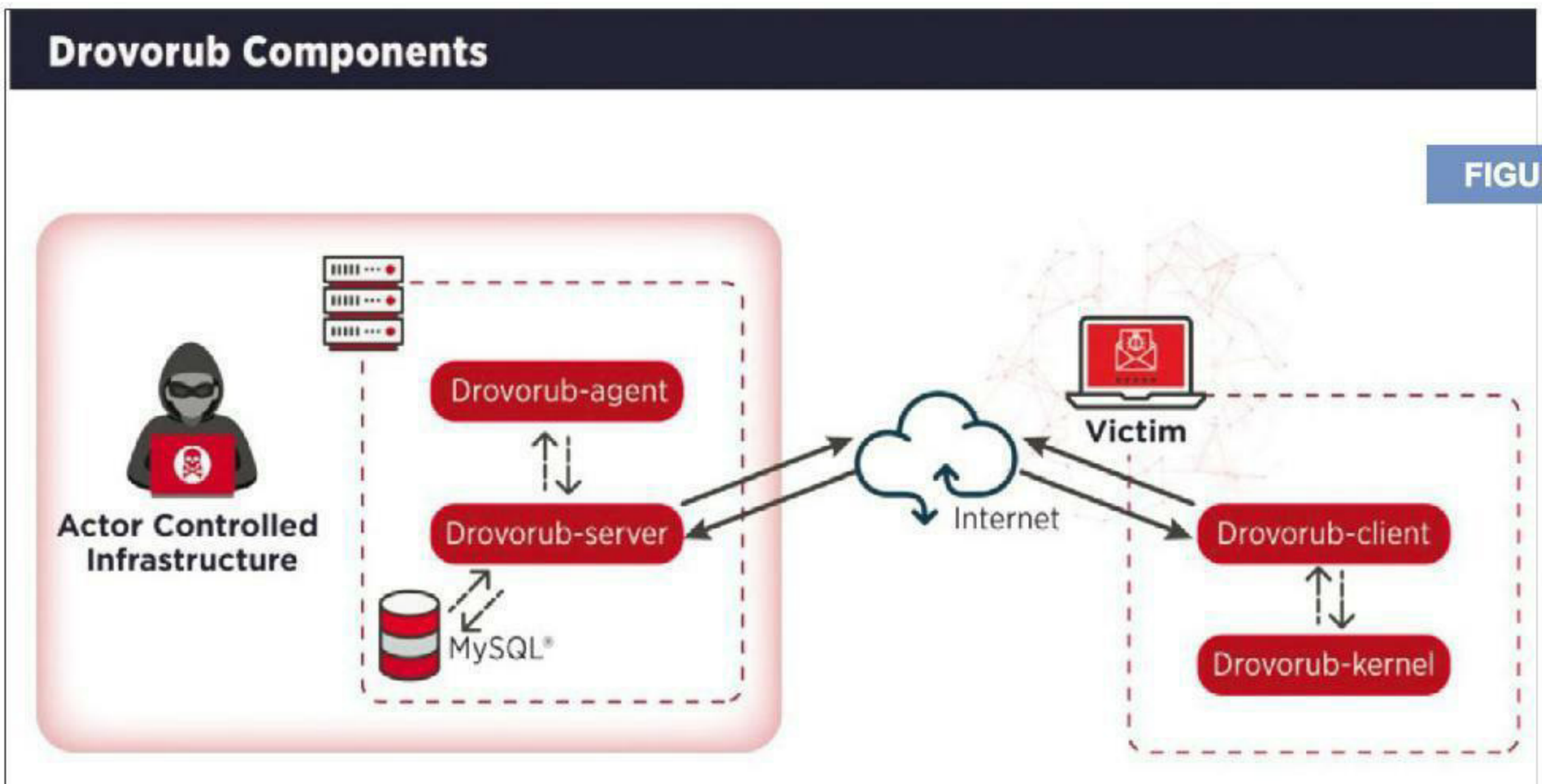


FIGURE 1

Architecture du malware Linux Drovorub [1].

Les composants du malware nous intéressant plus particulièrement sont le client et le module kernel. Comme vous le découvrirez au cours de la lecture des 45 pages du document, ce module kernel empêche toute détection par des commandes exécutées en espace utilisateur. Le module kernel dissimule la présence du malware en interceptant tous les appels systèmes : le module noyau modifie par exemple les retours des appels systèmes `iterate_dir()` ou `vfs_readdir()`. Cela permet de cacher l'existence de tous les fichiers créés et utilisés par le malware sur le serveur aux commandes et processus exécutés en espace utilisateur. Il en sera de même pour les processus lancés par le malware et les connexions réseaux (websockets comprises) qu'il utilise.

Une fois installé, le malware permet donc à son opérateur les actions suivantes sur nos serveurs :

- exécution de commandes distantes arbitraires en mode root ;
- exfiltration et téléchargement de fichiers vers et depuis les serveurs de commandes et de contrôle (C2) de l'infrastructure opérée par l'attaquant.

Une protection coûteuse

Ce module noyau persistant au reboot de nos machines est un vrai souci. Le seul moyen de s'en prémunir est d'avoir des plateformes interdisant l'installation de modules noyau non signés. Cela nécessite d'avoir des machines démarrées en mode UEFI et imposant le SecureBoot.

Ce type de configuration a des impacts sur l'agilité et l'administration des serveurs.

Pour un serveur personnel ou ceux d'une PME, le coût en termes d'administration pourrait s'avérer trop élevé.

Que nous reste-t-il donc comme alternative ?

Il faut que nous disposions de serveurs sur lesquels nous puissions avoir :

- des sauvegardes distantes et intègres de nos données avec une grande profondeur de rétention afin de pouvoir revenir à un état sain précédent l'infection. Cela nous permet aussi de pouvoir réinstaller totalement nos systèmes d'exploitation sans trop de crainte ;
- une externalisation des traces sur un serveur sain ;
- une capacité relative de détection.

Mise en place d'un plan d'action dans nos cordes

Cette démonstration nous indique que nous devons agir.

Cependant, notre plan d'action est limité dans son ambition par un manque de ressources, mais aussi de compétences DFIR (réponse à incident / investigation numérique) au sein d'une structure aussi petite que la nôtre. Pour rappel, notre cas d'usage est la sécurisation d'un serveur personnel ou d'une petite infrastructure informatique de PME.

Le plan, qui nous l'espérons se déroulera sans accroc, se veut donc raisonnable et sera composé des trois pans suivants :

- un plan de sauvegardes externalisées ;
- un serveur de centralisation des traces ;
- un outil de contrôle d'intégrité de nos serveurs.

2. SAUVEGARDE DE NOS SERVEURS

Qu'allons-nous sauvegarder ?

Partons du parc dont nous disposons.

Il s'agit principalement de serveurs hébergeant des services web ou réseaux. Ces services impliquent de devoir sauvegarder :

- les configurations localisées sous `/etc` ;

- des fichiers stockés en général sous `/var/www` ;
- et le contenu de bases de données MariaDB.

Pour les deux premiers points, on peut imaginer une copie directe des fichiers. Pour le dernier point, une étape intermédiaire sera nécessaire : avant de lancer la sauvegarde du serveur, nous exporterons de manière automatisée tous les données contenues dans les bases du serveur de bases de données MariaDB. Le répertoire de destination de ces exports sera sous `/var/backups/db`.

Principes suivis par notre processus de sauvegarde

Les grands principes de nos sauvegardes vont être :

- Les données à sauvegarder sur le serveur principal (**host1.mondomaine.fr**) :
  - → `/var/www` ;
  - → `/var/backups/db` ;
  - → `/etc`.
- La profondeur de rétention des sauvegardes sur le serveur de sauvegarde (**host2.mondomaine.fr**) :
  - → 7 sauvegardes quotidiennes roulantes ;
  - → 12 sauvegardes mensuelles roulantes.
- La localisation des sauvegardes : sur le serveur de sauvegarde **host2.mondomaine.fr** et non sur le serveur principal **host1.mondomaine.fr**, car c'est ce dernier qui possède la plus grande surface d'exposition.

  Note : cependant, si vous disposez de suffisamment d'espace disque sur le serveur principal, vous pouvez tout à fait garder une copie de ces sauvegardes sur disque. En cas de restauration, vous vous économiserez la durée du transfert réseau.

Concernant les détails d'implémentation :

- Sauvegardes de bases de données : nous sauvegarderons les bases de données dans des fichiers séparés (1 fichier par base). La raison est que nous souhaitons pouvoir restaurer le plus simplement possible. Une compression sera mise en œuvre afin d'optimiser l'espace disque consommé.
- Le sens des sauvegardes : nous récupérerons les données depuis le serveur de sauvegarde (sens **PULL**). Cela nous permet de dire que si le serveur principal est compromis, l'attaquant présent sur ce serveur principal ne pourra pas se connecter sur le serveur de sauvegarde pour altérer aussi les sauvegardes. Ce qui serait malheureusement le cas si nous adoptions un sens **PUSH** impliquant une connexion du serveur principal sur le serveur de sauvegarde pour l'export des sauvegardes.

Point ransomwares : l'impossibilité d'accéder aux sauvegardes est un point de sécurité critique dans les dernières attaques de type ransomwares ciblés. En effet, ces dernières, contrairement aux attaques automatisées, sont opérées par des attaquants déterminés, qualifiés et motivés par l'appât du gain. Leur laisser une porte d'entrée sur un serveur de sauvegarde distant relèverait alors d'une faute qui aurait de graves conséquences (perte irrémédiable des données ou coût financier très conséquent).

Étape 1 : export des bases de données sur le serveur principal

1. Création d'un utilisateur MariaDB dédié au backup

Nous allons utiliser un utilisateur dédié aux backups sous MariaDB. Pour cela, nous allons le créer puis lui donner les droits nécessaires à l'exécution de commandes d'export.

```
$ mysql

> CREATE USER 'backupuser'@'localhost' IDENTIFIED BY 'backuupserpwd';

> GRANT SELECT, SHOW VIEW, LOCK TABLES, RELOAD, REPLICATION CLIENT ON
*.* TO 'backupuser'@'localhost';

> FLUSH PRIVILEGES;
```

2. Utilisons ce user MariaDB sous notre user Linux (grâce au fichier $HOME/.my.cnf)

Nous allons utiliser ici une fonctionnalité de MariaDB intéressante tant en termes d'exploitation que de sécurité : le fichier **`$HOME/.my.cf`**.

Localisé à la racine, votre répertoire de travail est protégé de tout accès en dehors de votre utilisateur Linux (**`chmod 600`**), ce fichier vous permet de fixer le nom d'utilisateur MariaDB (et son mot de passe !) que votre utilisateur Linux va utiliser par défaut (surchargeable par l'option **`-u`** de la commande **`mysql`**).

Avantages :

- Intérêt d'administration : plus besoin de préciser l'utilisateur MariaDB et son mot de passe dans un script de backup ou dans une ligne de commandes (ex. : une ligne de la Crontab) ;
- Intérêt sécurité : le mot de passe de cet utilisateur MariaDB n'apparaîtra donc jamais dans une sortie d'un script, dans son contenu ou dans un historique de commandes.

Contenu du fichier **`$HOME/.my.cnf`** :

```
$ more ~/.my.cnf
[client]
user=backupuser
password=p455w0rd
```

3. Export quotidien des bases de données MariaDB et de leurs données

Nous allons mettre en place cet export en ajoutant une entrée de la crontab de votre utilisateur Linux exécutant les backups.

Il existe deux manières de faire à notre disposition :

- Une commande unique **`mysqldump`** : elle va nous permettre d'exporter toutes les bases dans un seul fichier. L'avantage est que vous n'avez aucun script à maintenir, tout est dans la crontab. La commande d'export est la suivante :

```
mysqldump --all-databases > all_databases.sql
```

Et bien entendu, nous allons l'améliorer pour compresser le fichier de sortie :

```
mysqldump --all-databases | gzip > all_databases.sql.gz
```

En effet, les données exportées sont écrites dans un fichier texte. Du coup, les résultats de compression (gzip ici en paramétrage standard) sont très bons. Faisons les 2 exports en ligne de commandes et comparons les résultats :

```
$ mysqldump --all-databases > all_databases.sql
$ mysqldump --all-databases | gzip > all_databases.sql.gz
$ ll all*
-rw-r--r-- 1 root root 101692838 Sep  1 14:11 all_databases.sql
-rw-r--r-- 1 root root  39462472 Sep  1 14:12 all_databases.sql.gz
```

Résultat : la taille du fichier export a diminué d'environ 60 %.

Ligne de commande finale à exécuter quotidiennement via la crontab (ici avec la date dans le nom du fichier si vous le souhaitez) :

```
mysqldump --all-databases | gzip > all_databases-$(date +%Y%m%d).sql.gz
```

- On peut aussi créer un script exportant chaque base dans un fichier différent. C'est la solution que nous allons implémenter. Voici le script **savedb.sh** en question :

```
#!/bin/bash

for DB in $(mysql -e 'show databases' -s --skip-column-names); do
    if [[ "$DB" == "information_schema" || "$DB" == "performance_
schema" ]];
    then
       continue
    fi
    mysqldump $DB | gzip > "/var/backups/db/$DB.sql.gz";
    esac
done
```

4. Paramètrage du crontab de votre user de backup Linux qui exporte tous les jours toutes les bases de données :

```
00  01 * * * /home/backupuser/savedb.sh > /home/backupuser/logfiles/
savedb.log 2>&1
```

Étape 2 : rsync depuis le serveur de backup

» Installation de la clé SSH de l'utilisateur de backup sur le serveur principal

Nous allons nous connecter sur le serveur principal depuis le serveur de sauvegarde en utilisant l'utilisateur Linux *backupuser*. Pour plus de sécurité, nous utiliserons une clé SSH spécialement créée pour cet usage.

Important : cette clé SSH va être utilisée pour des connexions automatisées depuis des scripts et des crontab sans présence d'aucun opérateur pour saisir le mot de passe de la clé. Cette clé doit donc être créée sans mot de passe. La sécurité de la clé repose alors sur les droits d'accès au fichier contenant la clé privée (`chmod 600` ici).

Comment créer une clé sans mot de passe ? Il suffit de taper sur la touche [Entrée] quand `ssh-keygen` vous demande le mot de passe lors de la création de la clé.

Nous copions cette clé publique nouvellement sur le serveur principal via la commande suivante :

```
$ ssh-copy-id -i /home/backupuser/.ssh/id_rsync_nopwd.pub backupuser@
host1.mondomaine.fr
```

Les prérequis nécessaires à la sauvegarde sont désormais en place :

1. export quotidien des bases de données sur le serveur principal ;
2. mise en place de la connexion SSH sur le serveur principal depuis le serveur de backup avec le user *backupuser*.

» Mise en place de la crontab sur le serveur de sauvegarde

Au final, nous aurons :

- 7 sauvegardes quotidiennes roulantes des répertoires `/etc`, `/var/www` et des bases de données avec le nom du jour (option `+%a` de la commande `date` donnant `mon`, `tue`, `wed`...) de la sauvegarde dans le nom du dossier. Simple et efficace lors d'un éventuel besoin de restauration.

```
00  01 * * * rsync -avx --delete --rsync-path="/usr/bin/rsync" -e "ssh
-i /home/backupuser/.ssh/id_rsync_key"  backupuser@host1.mondomaine.
fr:/var/www/ /var/backups/daily/$(date +\%a)/www/

00  01 * * * rsync -avx --delete --rsync-path="/usr/bin/rsync" -e "ssh
-i /home/backupuser/.ssh/id_rsync_key"  backupuser@host1.mondomaine.
fr:/etc/ /var/backups/daily/$(date +\%a)/etc/

00  01 * * * rsync -avx --delete --rsync-path="/usr/bin/rsync" -e "ssh
-i /home/backupuser/.ssh/id_rsync_key"  backupuser@host1.mondomaine.
fr:/var/backups/db/ /var/backups/daily/$(date +\%a)/db/
```

- Pour les 12 sauvegardes mensuelles roulantes, les paramètres de la crontab seront identiques aux valeurs de la sauvegarde quotidienne à part ces 2 points :

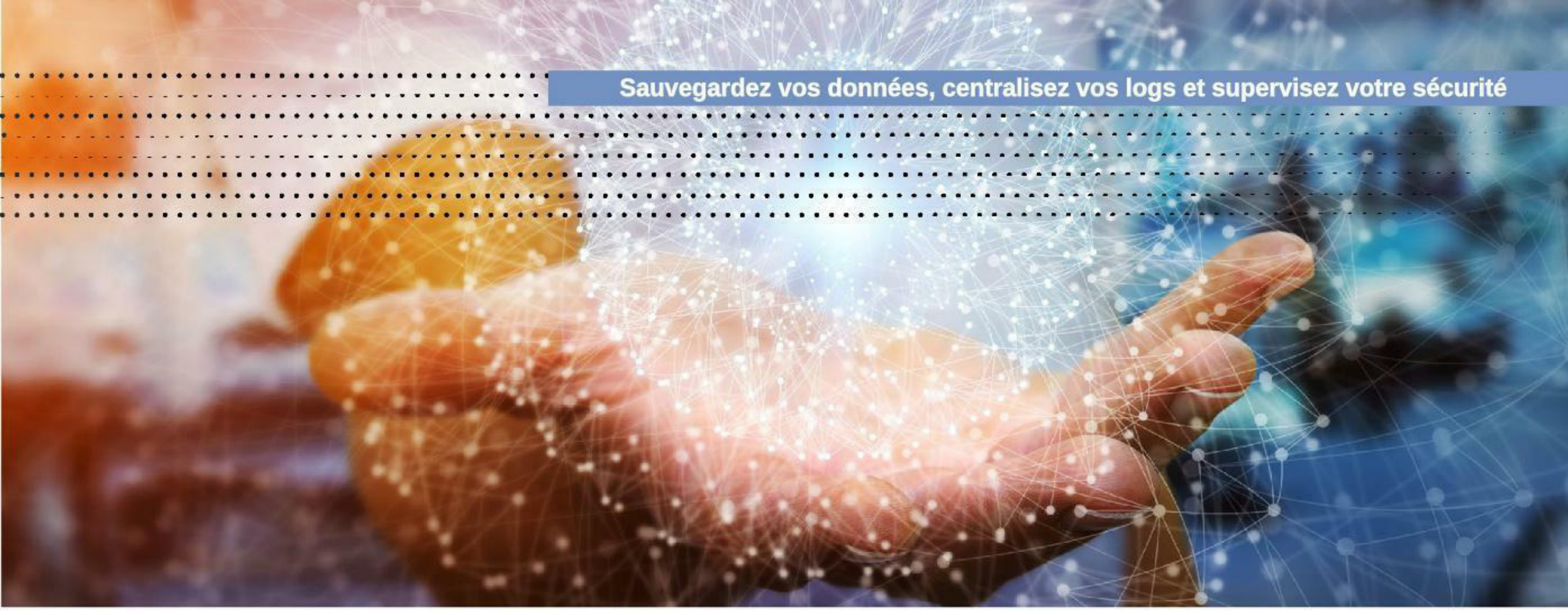
  → fréquence :

```
0 0 1 * *
```

  → localisation des sauvegardes :

```
/var/backups/monthly/$(date +\%b)/db ou www ou etc/
```

Voilà, les sauvegardes sont en production et nous pouvons gagner un peu en sérénité.



3. CENTRALISATION DES LOGS

Les logs générés par tous les processus s'exécutant sur nos machines étaient auparavant collectés dans des fichiers qui sont désormais gérés sous un format binaire par le démon de systemd, **journald**.

Si nous n'avions qu'une machine, **journald** serait suffisant pour gérer et consulter ses logs.

Mais nous voulons mettre en place un envoi des logs de nos machines sur un serveur centralisé. Cette fonctionnalité n'est pas fournie par **journald**. Pour mener à bien cette centralisation, nous allons donc installer Syslog-NG, la référence en matière de centralisation, filtrage et routage des messages de logs.

Installation de Syslog-NG sur tous nos serveurs

Nous allons utiliser la version de Syslog-NG présente dans les dépôts de nos distributions Debian 10 :

```
$ sudo apt install syslog-ng
```

Cela va installer **syslog-ng** (ici en version 3.19.1) en tant que produit et un jeu minimal de packages additionnels amenant par exemple la capacité d'envoyer les logs en bases de données (SQL ou MongoDB).

Il est à noter que Syslog-NG fournit de nombreuses autres possibilités (tant en termes de sources, de filtres ou de destinations) sous la forme de packages additionnels comme les suivants que l'on peut installer unitairement à la demande :

```
Recommended packages:
syslog-ng-mod-smtp syslog-ng-mod-amqp syslog-ng-mod-geoip syslog-ng-mod-geoip2
syslog-ng-mod-pacctformat syslog-ng-mod-redis syslog-ng-mod-stomp syslog-ng-
mod-riemann syslog-ng-mod-graphite syslog-ng-mod-python syslog-ng-mod-add-
contextual-data syslog-ng-mod-getent syslog-ng-mod-stardate syslog-ng-mod-map-
value-pairs syslog-ng-mod-snmptrapd-parser syslog-ng-mod-xml-parser syslog-ng-
mod-extra syslog-ng-mod-tag-parser syslog-ng-mod-examples
```

Principes généraux de configuration

Syslog-NG comprend 4 grands concepts dans ses éléments de configurations :

- Les sources : elles permettent de définir où Syslog-NG fera l'acquisition des nouveaux logs à traiter. Ces sources peuvent être des producteurs de logs systèmes (comme **journald**), des sources réseaux, des fichiers de logs sur disque, etc.

- Les filtres, les parsers et les règles de réécriture : ils permettent respectivement de trier, extraire des informations bien formatées et modifier les lignes de logs que Syslog-NG a lu dans les sources. Ce tri et ces modifications sont totalement optionnelles. Syslog-NG peut très bien transférer à des destinations la totalité des logs lus dans les sources sans aucun tri ni modification préalable.
- Les destinations : une fois les lignes de logs lues, triées et modifiées (si besoin et si on le souhaite), Syslog-NG les envoie à des destinations qui peuvent être de nature très diverses : fichiers, bases de données de tous types (relationnelles, NoSQL, etc.), machines présentes sur le réseau...
- Les directives logs : les 3 types d'éléments précédents sont regroupés au sein des directives logs qui les utilisent pour permettre à l'administrateur système de faire exactement ce qu'il souhaite avec ses lignes de logs.

À ces éléments de configuration, s'ajoutent aussi des options générales qui s'appliquent de manière globale à Syslog-NG.

Architecture de centralisation des logs à base de Syslog-NG

Notre architecture, du fait du peu de serveurs qui la constituent, est très simple.

Le Syslog-NG du serveur principal **host1.mondomaine.fr** est configuré en mode client et envoie tous ses logs au serveur **host2.mondomaine.fr** qui lui est configuré en mode serveur Syslog-NG.

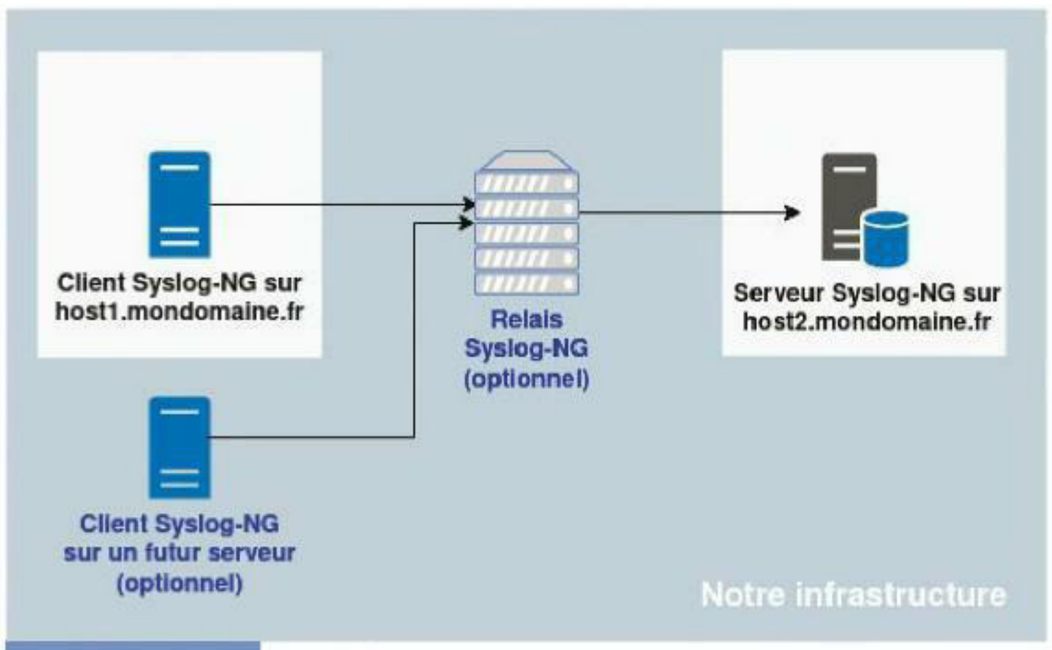


FIGURE 2 *Notre architecture de gestion des logs.*

Le schéma montre aussi un autre rôle pouvant exister : le rôle de serveur relais. Nous ne l'utiliserons pas du fait de la grande simplicité de notre infrastructure. Cependant, détaillons un peu son utilité.

CHOIX DU RÉSEAU DE COMMUNICATION ENTRE VOS SERVEURS & CHIFFREMENT

- **VPN : selon votre analyse de risques et vos échanges avec vos administrateurs systèmes, vous pouvez tout à fait faire le choix de faire circuler les logs entre vos deux serveurs au travers du sous-réseau défini par votre VPN. Cela ne modifie en rien votre infrastructure.**
- **Internet : si vous choisissez de faire communiquer vos clients et serveurs Syslog-NG au-dessus d'Internet, sachez que cela implique deux choses :**

1. **Vous augmentez votre surface d'attaque, mais si vous ajustez votre configuration de pare-feu (autoriser que vos serveurs à dialoguer sur le port de Syslog-NG), les risques sont atténués.**
2. **Chiffrement : vous disposez de la possibilité de faire communiquer vos serveurs de manière chiffrée (TLS) et authentifiée (authentification mutuelle). Il faut cependant disposer d'une autorité de certification interne afin de disposer de certificats pour chaque client et serveur Syslog-NG.**

Il permet de concentrer les logs d'un sous-ensemble de Syslog-NG clients avant de les envoyer à un Syslog-NG serveur ou à d'autres destinations comme un SIEM, une instance ELK ou un serveur Splunk.

Le rôle de relais est notamment très utile dans des zones réseaux sensibles comme les DMZ où ils permettent de restreindre les ouvertures de ports et d'adresses nécessaires dans le pare-feu protégeant cette DMZ pour permettre l'envoi des logs au serveur Syslog-NG. En effet, on ne doit autoriser que l'adresse IP du relais (et non de toutes les adresses des clients Syslog-NG) à communiquer avec le serveur Syslog-NG.

Configuration du client Syslog-NG

Nous avons deux catégories de logs à gérer sur notre serveur principal **host1.mondomaine.fr** :

- les logs générés par le système d'exploitation ;
- les logs générés par les applicatifs web s'exécutant sur notre serveur **host1.mondomaine.fr**.

La configuration vise donc à récupérer ces deux types de logs et à les envoyer à notre serveur Syslog-NG (**host2.mondomaine.fr**).

Important : le fichier `/etc/syslog-ng/syslog-ng.conf` contient déjà toute la configuration requise pour gérer les logs en local. À la fin du fichier se trouve la directive suivante :

```
@include "/etc/syslog-ng/conf.d/*.conf"
```

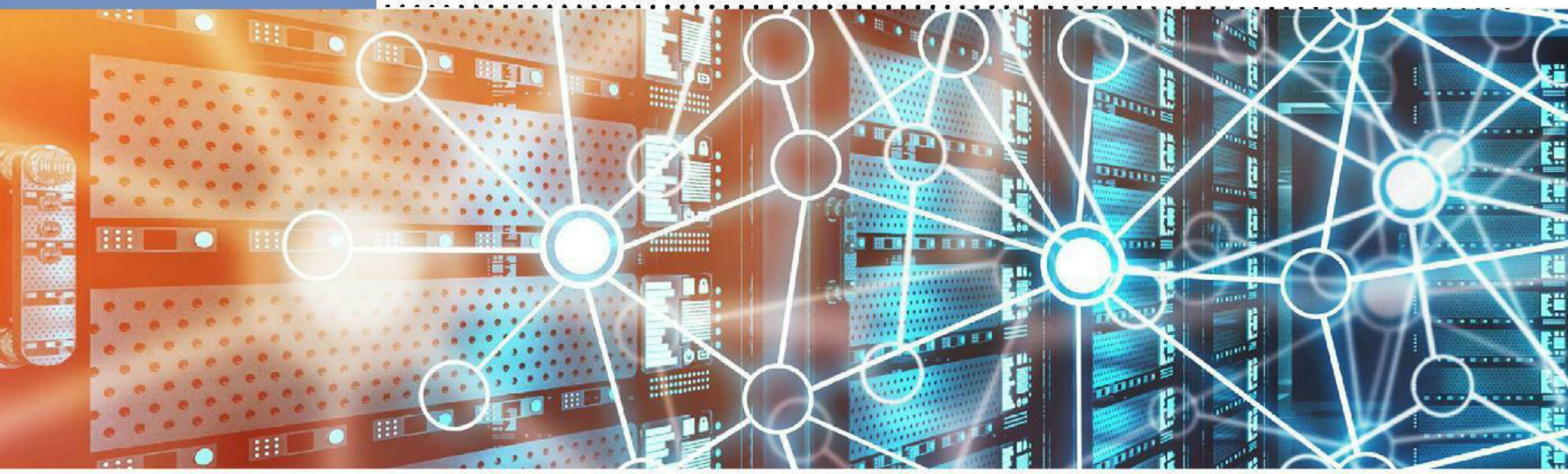
Elle va nous permettre d'ajouter nos éléments supplémentaires de configuration sous forme de fichiers dans ce répertoire. Ainsi nous ne toucherons pas au fichier de configuration initial et nous ne souffrirons d'aucun risque de modification de configuration lors d'une montée de version de Syslog-NG.

- Ajout du fichier `/etc/syslog-ng/conf.d/s_apache_logs.conf` qui permet de lire les logs des vhosts d'Apache (*Access* et *Error logs*) :

```
source s_apache_log_files {
    wildcard-file(
        base-dir("/var/log/apache")
        filename-pattern("*.log")
        recursive(yes)
        follow-freq(1)
    );
};
```

- Ajout du fichier `/etc/syslog-ng/conf.d/d_central_log_server.conf` qui définit la destination du serveur de logs centralisé.

```
destination d_central_log_server {
    syslog(
        "host2.mondomaine.fr"
        port(601)
           disk-buffer(
            mem-buf-length(10000)
            disk-buf-size(2000000)
            reliable(no)
            dir("/tmp/disk-buffer")
        )
    );
};
```



Nous utilisons ici le nom DNS du serveur de centralisation de logs. Vous pouvez préciser à la place l'adresse IP et même l'adresse IP privée au sein du VPN de ce serveur **host2**, à savoir : **192.168.100.1**.

De plus, nous précisons que nous souhaitons sécuriser le transfert de logs par l'usage d'un buffer qui stockera sur disque les logs éventuellement non transmis au serveur central. Ce qui peut par exemple arriver en cas de coupure réseau entre client et serveur.

- Ajout du fichier **/etc/syslog-ng/conf.d/log_logs_to_central-server.conf** qui permet d'envoyer les logs locaux et Apache vers le serveur de logs centralisé.

```
log { source(s_local); source(s_apache_log_files); destination(d_
central_log_server); }
```

Les logs en question sont donc désormais envoyés de manière fiable au serveur central.

» Quelques améliorations et optimisations possibles

Je vous laisse approfondir les possibilités suivantes pour avoir encore plus d'informations pour investiguer sur une suspicion d'attaque (réussie notamment) :

- Qualité des informations présentes dans les logs Apache : par défaut, les serveurs web ne proposent aucune information sur le contenu des requêtes **POST**.

  Piste de solution : regardez les possibilités du module Apache ModSecurity. Il permet d'alimenter son fichier **AuditLog** avec notamment le contenu de la requête **POST**. Il ne restera plus qu'au client Syslog-NG de monitorer ce fichier d'audit de ModSecurity et d'en envoyer le contenu au serveur Syslog-NG.

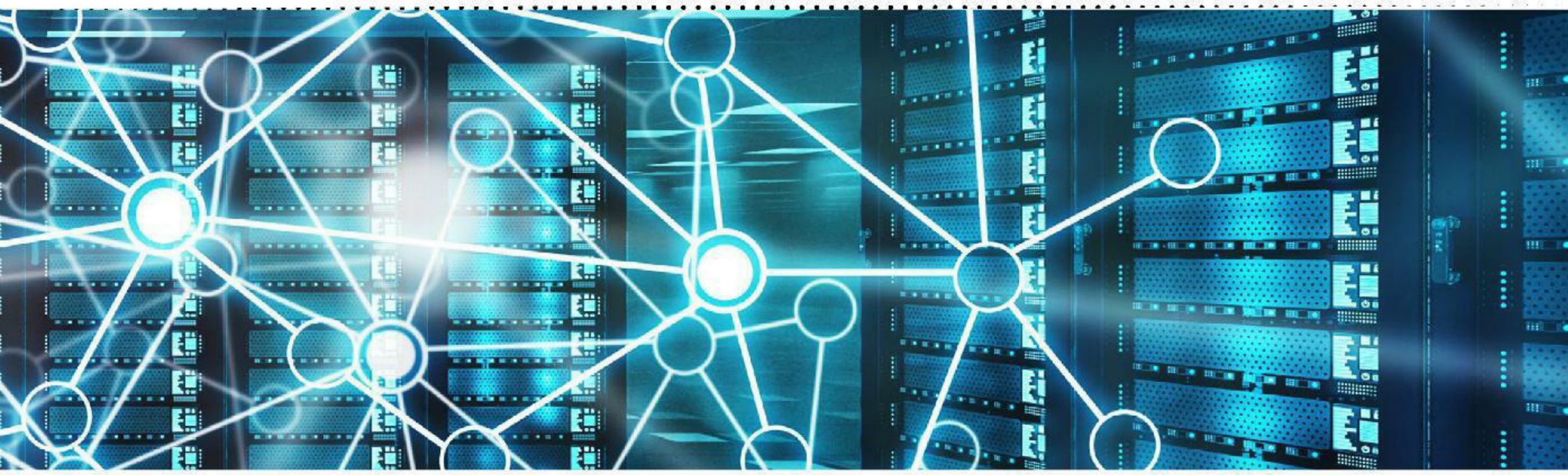
- Envoi de logs en environnement hostile : l'usage d'un réseau privé nous a permis de nous passer de mettre en place une couche de transport chiffrée et authentifiée entre le client et le serveur Syslog-NG.

  Piste de solution : sachez simplement qu'une telle sécurité de transport est tout à fait possible à condition de posséder une autorité de certification interne.

Configuration Syslog-NG serveur sur host2.mondomaine.fr

Il s'agit ici d'assurer la réception et le stockage sur le serveur Syslog-NG (**host2.mondomaine.fr**) des logs émis par le serveur **host1.mondomaine.fr**.

Prérequis réseau : nous allons avoir Syslog-NG qui écoute sur le port 601 en TCP. Le pare-feu doit l'autoriser.



Pour cela, nous lançons la commande suivante :

```
$ sudo ufw allow 601/tcp
```

De la même manière que sur le premier serveur, Syslog-NG doit être installé et sa configuration minimale est opérationnelle dès la sortie d'installation.

Nous allons juste créer une configuration pour la réception des logs et une configuration pour leur écriture sur disque de manière simple (un fichier par serveur source).

Liste des configurations :

- Ajout du fichier `/etc/syslog-ng/conf.d/s_syslog.conf` qui permet de lire les logs en provenance du réseau :

```
source s_syslog {source s_network {default-network-drivers(); }; };
```

- La destination suivante permet de stocker les logs de chaque serveur dans un fichier séparé :

```
destination d_logfile_per_host {
    file("/var/log/syslogng/${HOST}/messages");
};
```

À noter : pour que les répertoires de destination soient créés automatiquement par Syslog-NG, il faut que l'option générale `create-dirs(yes)` soit présente dans la configuration.

- Enfin la ligne `log` suivante permet de rendre actives ces configurations :

```
log {
    source(s_syslog); destination(d_logfile_per_host);
};
```

Quelques améliorations et optimisations possibles :

1. Le rangement des logs dans un fichier unique par serveur sur le serveur Syslog-NG central ne peut fonctionner que si les logs à la source contiennent bien le champ `HOST`. Il faut de même que ce champ soit toujours rempli avec la même information (`hostname` court, `hostname` avec le domaine, adresse IP, etc.). Si ce n'est pas toujours le cas, il faudra travailler sur les logs sources pour rajouter cette information. Utilisez pour cela les règles de réécriture de logs de Syslog-NG :-)

2. On peut souhaiter disposer de plus de logs applicatifs. Pour cela, il faut penser à indiquer à l'application d'envoyer ses logs dans un fichier bien identifié ou mieux encore, de les envoyer au démon Syslog-NG présent sur le serveur. C'est une possibilité qui existe dans beaucoup d'applications, mais aussi dans beaucoup d'équipements matériels ou virtuels (routeurs, équilibreurs de charges, WAF, etc.).
3. De la même manière, vous pouvez souhaiter recevoir des logs dans un format différent. Par exemple, les règles de réécriture syslog-NG permettent d'écrire les logs dans des formats clé=valeur comme le JSON.
4. Vous pouvez vouloir exploiter de manière graphique vos logs en plus de les sauver au format texte pour une éventuelle future exploitation en mode réponse à incident. Sachez que Syslog-NG peut nativement envoyer les logs dans un format compatible à des destinations de type Elastic, Graphite, AMQP, etc.

Exemple (issu de la documentation Syslog-NG 3.19) de destination de type Elastic utilisant l'API HTTP sur le réseau :

```
destination d_elastic {
    elasticsearch2(
        client-mode("http")
        cluster("es-syslog-ng")
        index("x201")
        cluster-url("http://192.168.33.10:9200")
        type("slng_test_type")
        flush-limit("0")
    );
};
```

Conclusion de la partie gestion de logs

Nous sommes désormais assurés de disposer d'une copie intègre des logs système et Apache de notre serveur principal (**host1.mondomaine.fr**) sur notre serveur d'administration (**host2.mondomaine.fr**).

Nous pouvons donc effectuer des recherches dans ses logs dans une situation de réponse à un incident sécurité en ayant la certitude qu'ils n'ont pas été altérés par un attaquant à la suite d'une attaque réussie (la trace de la connexion initiale, éventuelle désactivation du service de logs, etc.).

À l'échelle de notre petite architecture de PME/infrastructure personnelle, nous sommes donc dans une situation bien améliorée en cas d'attaque.

4. OSSEC : FAIRE DE LA VEILLE SÉCURITÉ SUR SON PARC

OSSEC est un logiciel client/serveur qui permet de faire de la supervision sécurité sur un ensemble de machines sur lesquelles tourne l'agent OSSEC (ici, notre serveur principal **host1.mondomaine.fr**). Ces contrôles de sécurité sont lancés de manière automatique depuis le serveur OSSEC (ici le serveur d'administration **host2.mondomaine.fr**).

Installation du serveur et de l'agent

L'installation est quasiment identique que l'on soit sur le serveur ou le client OSSEC.

```
# Ajout de la clé de signature des packages
$ wget -q -O - https://www.atomicorp.com/RPM-GPG-KEY.atomicorp.txt  | sudo apt-key add -
```

```
# Ajout du dépôt compatible avec notre distribution
$ echo "deb https://updates.atomicorp.com/channels/atomic/debian Buster
main" >>  /etc/apt/sources.list.d/atomic.list

$ sudo apt-get update

# Installation du serveur
sudo apt-get install ossec-hids-server

# Installation de l'agent
sudo apt-get install ossec-hids-agent
```

De même pour que la communication entre l'agent et le serveur soit possible, nous allons autoriser la communication sur le port **1514** en UDP :

```
$ sudo ufw allow 1514/udp
```

Création de l'agent sur le serveur et son déploiement

Nous lançons la commande interactive **manage_agents** et créons l'agent en le nommant (**host1**) et en donnant son adresse IP (**a.b.c.d**) :

```
# /var/ossec/bin/manage_agents

****************************************
* OSSEC HIDS v3.6.0 Agent manager.     *
* The following options are available: *
****************************************
   (A)dd an agent (A).
   (E)xtract key for an agent (E).
   (L)ist already added agents (L).
   (R)emove an agent (R).
   (Q)uit.
Choose your action: A,E,L,R or Q: A

- Adding a new agent (use '\q' to return to the main menu).
  Please provide the following:
   * A name for the new agent: host1
   * The IP Address of the new agent: a.b.c.d
   * An ID for the new agent[001]:
Agent information:
   ID:001
   Name:host1
   IP Address:a.b.c.d

Confirm adding it?(y/n): y
Agent added with ID 001.
```

Afin de pouvoir importer cet agent sur le serveur on le déploiera, il faut sa clé cryptographique.

```
****************************************
****************************************
* OSSEC HIDS v3.6.0 Agent manager.     *
* The following options are available: *
****************************************
   (A)dd an agent (A).
   (E)xtract key for an agent (E).
   (L)ist already added agents (L).
   (R)emove an agent (R).
   (Q)uit.
Choose your action: A,E,L,R or Q: E

Available agents:
   ID: 001, Name: host1, IP:  a.b.c.d
Provide the ID of the agent to extract the key (or '\q' to quit): 001

Agent key information for '001' is:
MDAxIGhvc3Q0IDM3LjE4Ny45Ny4[...]
YzdjYTJhODJhMjI0MjZkY2QxMmNkNzBmMDZiYzQ3NzQwZTlmMjIwOTYzMmQ=

** Press ENTER to return to the main menu.

****************************************
* OSSEC HIDS v3.6.0 Agent manager.     *
* The following options are available: *
****************************************
   (A)dd an agent (A).
   (E)xtract key for an agent (E).
   (L)ist already added agents (L).
   (R)emove an agent (R).
   (Q)uit.
Choose your action: A,E,L,R or Q: q
```

Nous nous rendons ensuite sur le serveur hébergeant l'agent et nous importons la configuration de l'agent.

```
****************************************
* OSSEC HIDS v3.6.0 Agent manager.     *
* The following options are available: *
****************************************
   (I)mport key from the server (I).
   (Q)uit.
Choose your action: I or Q: I

* Provide the Key generated by the server.
* The best approach is to cut and paste it.
*** OBS: Do not include spaces or new lines.

Paste it here (or '\q' to quit): MDAxIGhvc3Q0IDM3LjE4Ny45Ny4[...]
YzdjYTJhODJhMjI0MjZkY2QxMmNkNzBmMDZiYzQ3NzQwZTlmMjIwOTYzMmQ=
```

```
Agent information:
   ID:001
   Name:host1
   IP Address:a.b.c.d

Confirm adding it?(y/n): y
Added.
```

Enfin, sur l'agent, nous injectons l'adresse IP du serveur dans la configuration de l'agent sous `/var/ossec/etc/ossec.conf` :

```
<ossec_config>
  <client>
    <server-ip>x.y.z.a</server-ip>
  </client>
```

Et nous relançons le client :

```
$ /var/ossec/bin/ossec-control restart
Killing ossec-logcollector ..
Killing ossec-syscheckd ..
Killing ossec-agentd ..
Killing ossec-execd ..
OSSEC HIDS v3.6.0 Stopped
Starting OSSEC HIDS v3.6.0...
Started ossec-execd...
[...]
Started ossec-agentd...
Started ossec-logcollector...
Started ossec-syscheckd...
Completed.
```

Mise en place des notifications mail sur le serveur

Le fichier de configuration général d'OSSEC sur le serveur est aussi localisé sous `/var/ossec/etc/ossec.conf`.

Une section `global` permet d'activer les notifications mail :

```
<ossec_config>
    <global>
        <email_notification>yes</email_notification>
        <email_to>me@example.com</email_to>
        <smtp_server>mx.example.com</smtp_server>
        <email_from>ossec@example.com</email_from>
```

Vous personnalisez vos paramètres d'e-mail et vous allez pouvoir commencer à recevoir des alertes.

Possibilités d'OSSEC

OSSEC permet d'effectuer les actions suivantes :

1. supervision des logs ;

2. contrôle d'intégrité des fichiers et des répertoires (ajout, modification, suppression de fichiers) ;

3. réponse active à une détection de menace.

Vous voyez que le champ d'exploration est vaste. Par défaut, supervision des logs et contrôle d'intégrité sont actifs.

Les alertes, mais aussi les dysfonctionnements d'OSSEC sont localisées sous `/var/ossec/logs/alerts`.

Une bonne quantité de travail est requise en termes de personnalisation afin de tirer tout le potentiel d'OSSEC. En effet, comme toujours avec les produits générant des contrôles et des alertes, le plus difficile est d'atteindre le bon équilibre entre trop d'alertes d'un côté et rater une intrusion parce que l'on a instauré des contrôles trop permissifs. Cet équilibre dépendra de manière très forte de votre environnement technique, de vos attentes, mais aussi de votre capacité à faire en termes d'analyse d'alertes.

La documentation d'OSSEC est disponible sur **https://www.ossec.net/docs/docs/manual/** pour vous aider dans cet approfondissement.

Toutes les modifications de configuration seront à faire dans le fichier `ossec.conf`.

CONCLUSION

Cet article conclut ce dossier sur la sécurisation de notre infrastructure technique.

J'espère qu'il vous aura apporté un certain nombre de nouvelles connaissances tant en termes de sécurité opérationnelle qu'en termes d'analyse de risques sécurité ou de veille sur des menaces.

Tout au long de ces 4 articles, j'aurais essayé de développer deux axes majeurs :

- accroître notre capacité à protéger notre architecture en nous amenant à utiliser des technologies comme le VPN Wireguard entre nos serveurs ou un service OpenSSH optimisé ;
- parfaire la résilience de nos services en cas d'attaque qui réussit. Nos armes pour mener cela à bien ont été la mise en place de sauvegardes sécurisées et la préservation d'une capacité d'investigation grâce à des logs qui auront été protégés ou la mise en œuvre d'un outil de supervision sécurité.

À l'opposé, j'ai aussi essayé de ne pas nous noyer dans une architecture inutilement trop complexe. Ce surplus de complexité n'aurait, je pense, finalement débouché que sur une diminution de la maîtrise technique de notre architecture. Or, cette maîtrise est un des éléments majeurs de notre capacité à répondre correctement aux incidents de sécurité quand ils adviennent.

Keep It Stupidly Simple est, vous l'aurez compris, un principe que je chéris tant en administration système qu'en sécurité.

Je vous souhaite donc désormais une bonne administration système sécurisée à toutes et à tous ! ▌

RÉFÉRENCE

[1] **https://media.defense.gov/2020/Aug/13/2002476465/-1/-1/0/CSA_DROVORUB_RUSSIAN_GRU_MALWARE_AUG_2020.PDF**

Chez votre marchand de journaux

et sur **www.ed-diamond.com**

PAPIER

en kiosque

FLIPBOOK HTML5

sur www.ed-diamond.com

CONNECT

LA DOCUMENTATION TECHNIQUE DES PROS DE L'IT

sur connect.ed-diamond.com

DEVOPS

AUTOMATISER INTÉGRALEMENT LA MISE EN PLACE DE WILDFLY AVEC ANSIBLE

Romain PELISSE

Si les outils comme Ansible permettent d'aller très loin dans l'automatisation d'un déploiement logiciel, ils sont souvent limités dans leurs capacités de réglage fin d'un outil aussi complexe et avancé qu'un serveur Java EE tel que Wildfly (ou son pendant commercial, JBoss EAP). Afin de résoudre cette problématique, l'outil JCliff a été développé pour permettre à Puppet (un concurrent d'Ansible) de s'intégrer sans difficulté avec ce serveur applicatif. Cet outil est maintenant aussi intégré avec Ansible sous la forme d'une collection et cet article propose un tour exhaustif des capacités d'automatisation du déploiement et de la configuration des sous-systèmes de Wildfly à l'aide de cette nouvelle extension.

Avant de démarrer la partie pratique, nous allons commencer par bien définir les objectifs de notre démonstration. Notre cas d'utilisation est l'automatisation de l'installation, la configuration et le réglage fin d'une instance du serveur Wildfly. Par réglage, on entend ici la modification du comportement par défaut des composantes internes du serveur. Dans le cas de Wildfly, il s'agit en fait de la modification des paramètres de fonctionnement de ses sous-systèmes.

Ceci est important, car l'installation et la configuration de Wildfly sont en effet déjà largement automatisables avec les primitives fournies par Ansible. L'outil permet en effet de facilement installer un paquet logiciel, créer les arborescences de répertoire nécessaires et même déployer les principaux fichiers de configuration du serveur.

C'est une fois ce travail préliminaire effectué que les difficultés, en quelque sorte, commencent. En effet, tous ces réglages fins du serveur vivent dans un fichier de configuration XML (le `standalone.xml` ou le `standalone-full.xml` par exemple). Ce fichier n'est pas statique, il peut être modifié à l'exécution, après le démarrage du serveur. Ce qui en fait donc un très mauvais candidat pour l'utilisation de fichiers patrons (« *template* »). Et c'est donc là que JCliff et sa collection Ansible entrent en jeu.

Il faut donc retenir que la capacité de réglage fin du serveur est la plus-value apportée par JCliff et la collection qui l'accompagne.

1. JCLIFF

JCliff est un outil en ligne de commandes qui permet de dynamiquement modifier l'état d'une instance Wildfly. Il s'appuie lui-même sur une fonctionnalité d'administration du serveur : le célèbre JBoss CLI. Celui-ci est un outil de requêtage qui permet non seulement de récupérer des informations sur l'état et la configuration du serveur, mais aussi de modifier son paramétrage. On peut ainsi utiliser JBoss CLI à des fins de supervision (ex. : surveiller le nombre de connexions à la base de données) ou pour changer (souvent sans le redémarrer) la configuration du serveur (augmenter le nombre de connexions à la base de données).

C'est lors de ces opérations que le serveur est parfois amené à mettre le jour le fichier de configuration XML. Ce qui interdit à Ansible de suivre ce fichier à l'aide de patrons (« template ») comme l'outil peut le faire avec d'autres fichiers de configurations.

C'est pour cette raison que JCliff a été développé. En effet, l'outil de requêtes JBoss CLI est au bout du compte assez peu adapté au besoin d'Ansible ou de Puppet, car ces outils manipulent des états. De leur point de vue, soit le serveur est dans l'état désiré, soit son état doit être modifié pour correspondre aux attentes. La sémantique de JBoss CLI permet certes d'obtenir ces informations, mais ne fournit pas naturellement la réponse à la question (est-ce que l'état est conforme aux attentes ?).

JCliff vient donc faire cette liaison. À partir des informations communiquées par Ansible sur l'état désiré du serveur, l'outil va générer une série de requêtes JBoss CLI destinées à vérifier que l'état est conforme. Si ce n'est pas le cas, JCliff va aussi générer les requêtes pour aligner la configuration du serveur avec les attentes. En quelques mots, JCliff fournit à Ansible la sémantique qui lui est nécessaire pour faire de Wildfly une ressource comme une autre, dont il peut contrôler et gérer l'état.

Cette approche a aussi une conséquence appréciable : le « *playbook* » utilisé par Ansible résume, en quelques lignes seulement, l'ensemble des modifications effectuées sur la configuration une fois celle de base déployée. On peut donc voir, en un coup d'œil, quels sont les paramétrages du serveur qui lui sont spécifiques à ce déploiement.

Voilà pour la théorie et le contexte, passons (enfin !) à la pratique avec la mise en place de notre environnement de travail.

2. PRÉREQUIS ET MISE EN PLACE

Pour que notre démonstration fonctionne, il nous faut d'abord nous assurer que ces prérequis ont été mis en place :

- il est recommandé que le système d'exploitation de la machine cible soit un système RHEL, CentOS, Scientific Linux ou Fedora (bref, une distribution de Linux utilisant RPM ainsi que YUM ou DNF) ;
- une machine virtuelle Java dans sa version 8 (ou plus récente) a été installée et est fonctionnelle ;
- Ansible, dans sa version 2.9 ou plus récente, est aussi installé et fonctionnel ;
- enfin, le serveur Wildfly (dans sa version 19) ou son pendant commercial (JBoss EAP 7.3) est installé sur le système a été démarré.

Un premier « playbook »

Pour faciliter la compréhension de cette démonstration, nous allons utiliser Ansible de manière locale. Ce qui signifie qu'il s'exécutera directement sur la machine cible. Il n'y a donc pas de contrôleur Ansible et de système (distant) cible, tout se déroule sur la même machine. Ceci permet au lecteur de reproduire tout ce que nous allons décrire dans cet article au sein d'une seule machine virtuelle ou même simplement d'un conteneur Docker.

Commençons par mettre un « *playbook* » minimum qui s'exécute donc sur le système local :

```
---
- hosts: localhost
  gather_facts: true
  vars:
  tasks:
```

Avant d'aller plus loin, vérifions que ce « *playbook* » soit fonctionnel :

```
$ ansible-playbook playbook.yml
[WARNING]: provided hosts list is empty, only localhost is available.
Note that the implicit localhost does not match 'all'

PLAY [localhost] ***************************************************
*******************************************************************

TASK [Gathering Facts] *********************************************
*******************************************************************
ok: [localhost]

PLAY RECAP *********************************************************
*******************************************************************
localhost                  : ok=1    changed=0    unreachable=0
failed=0    skipped=0    rescued=0    ignored=0
```

Parfait ! Tout ceci fonctionne comme prévu.

Le lecteur attentif prendra soin de noter que nous avons configuré Ansible pour collecter, à son exécution, les informations sur le système (« *gather_facts* »). C'est un point important, car cette opération est quelque peu coûteuse en termes de performances, mais les informations collectées vont permettre de faciliter grandement l'installation de JCliff comme nous allons le voir un peu plus loin.

En effet, la collection Ansible fournissant l'intégration avec JCliff, elle vient aussi avec un **role** qui automatise intégralement son installation, quel que soit le système utilisé. En effet, JCliff peut être installé à l'aide de RPM sur une distribution qui le permet ou à l'aide de **homebrew** sur macOS. Seul Windows n'est pas encore supporté, mais le travail est en cours de développement et cette intégration sera peut-être disponible lorsque vous lirez ces lignes.

Dernière remarque : comme la collecte d'informations sur le système peut prendre quelques secondes, il souvent pratique de la désactiver afin d'accélérer l'exécution d'Ansible. Cette collecte n'est pas réellement une contrainte forte de la collection. En fait, elle n'utilise que quelques-unes des informations collectées, il est donc aisé, si besoin est, d'ajouter ces données au « *playbook* » et de désactiver la collection de données.

Installer la collection JCliff pour Ansible

Ansible étant déjà installé, nous allons maintenant lui ajouter la collection qui fournit l'intégration avec JCliff. Elle est nommée : **willdfy.jcliff**. Cette opération est très simple, il suffit pour la réaliser d'utiliser l'outil **ansible-galaxy** fourni avec Ansible (et de disposer d'une connexion internet, bien sûr) :

```
# ansible-galaxy collection install wildfly.jcliff
Process install dependency map
Starting collection install process
Installing 'wildfly.jcliff:0.0.2' to '/root/.ansible/collections/
ansible_collections/wildfly/jcliff'
Installing JCliff using the provided role
```

Installer JCliff à l'aide de sa collection pour Ansible

Une fois cette collection installée, Ansible dispose non seulement de l'intégration avec JCliff (que nous allons étudier en détail tout au long de cet article), mais aussi d'une procédure d'installation de l'outil Java lui-même. Cette procédure ne requiert qu'un seul paramètre : l'emplacement du serveur Wildfly sur le système, soit la valeur systématiquement référencée sous le nom de **JBOSS_HOME**.

Pour rappel, avant qu'il ne soit renommé Wildfly, le serveur se nommait JBoss AS. Seul le pendant commercial de ce projet, soit JBoss EAP, a conservé ce nom.

Cette valeur est souvent définie et disponible sous forme d'une variable d'environnement. Nous allons faire simple et configurer Ansible pour récupérer cette information à partir de la variable d'environnement :

```
---
- hosts: localhost
  gather_facts: true
  vars:
    jboss_home: "{{ lookup('env','JBOSS_HOME') }}"
  collections:
    - wildfly.jcliff
  roles:
    - jcliff
  tasks:
```

Quelques remarques sur le « *playbook* » ci-dessus :

- la variable **jboss_home** est définie à partir du contenu de la variable d'environnement **JBOSS_HOME** récupérée à l'aide de la fonction **lookup** ;
- ce « *playbook* » requiert la présence de la collection Ansible **wildfly.jcliff** sur le système exécutant Ansible (nous venons de l'installer à l'aide de **ansible-galaxy**) ;
- le « *playbook* » importe le rôle nommé **jcliff** (fourni avec la collection) et l'exécute avant d'exécuter les tâches qu'il contient. Ce qui signifie que, même en l'absence de tâches définies sous le mot-clé **tasks:** Ansible installera tout de même JCliff lors de son exécution.

Voyons si tout ceci fonctionne comme prévu :

```
# ansible-playbook playbook.yml
[WARNING]: provided hosts list is empty, only localhost is available.
Note that the implicit localhost does not match 'all'

PLAY [localhost] ***************************************************
*******************************************************************

TASK [Gathering Facts] *********************************************
*******************************************************************
ok: [localhost]

TASK [wildfly.jcliff.jcliff : Collect Supported Operating Systems] ****
*******************************************************************
ok: [localhost] => (item={u'key': u'homebrew', u'value': [u'MacOSX']})
ok: [localhost] => (item={u'key': u'rpm', u'value': [u'Fedora',
u'CentOS', u'RedHat']})

TASK [wildfly.jcliff.jcliff : Verify supported Operating Systems] *****
*******************************************************************
ok: [localhost]

TASK [wildfly.jcliff.jcliff : Install JCliff using HomeBrew] **********
*******************************************************************
skipping: [localhost]
```

```
TASK [wildfly.jcliff.jcliff : Install JCliff using RPM] ***************
*******************************************************************
included: /root/.ansible/collections/ansible_collections/wildfly/
jcliff/roles/jcliff/tasks/install_rpm.yml for localhost

TASK [wildfly.jcliff.jcliff : Add JCliff Yum Repository (RedHat)] *****
*******************************************************************
ok: [localhost]

TASK [wildfly.jcliff.jcliff : Test if package jcliff is already
installed] ********************************************************
*
fatal: [localhost]: FAILED! => {"changed": false, "cmd": ["rpm",
"-q", "jcliff"], "delta": "0:00:00.489702", "end": "2020-08-17
08:54:34.190992", "msg": "non-zero return code", "rc": 1, "start":
"2020-08-17 08:54:33.701290", "stderr": "", "stderr_lines": [],
"stdout": "package jcliff is not installed", "stdout_lines": ["package
jcliff is not installed"]}

TASK [wildfly.jcliff.jcliff : Ensure JCliff is installed] *************
*******************************************************************
changed: [localhost]

TASK [wildfly.jcliff.jcliff : Install Jcliff using standalone binary]
*******************************************************************
skipping: [localhost]

PLAY RECAP ********************************************************
*******************************************************************
localhost                  : ok=6    changed=1    unreachable=0
failed=0    skipped=2    rescued=1    ignored=0
```

Comme indiqué plus haut, l'exécution de ce « *playbook* » a permis d'installer JCliff à l'aide de la procédure fournie par la collection Ansible **`wildfly.jcliff`**. En effet, l'outil a noté l'absence de JCliff sur le système et s'est donc occupé de l'installer.

Vérifions que ceci ait en effet bien fonctionné :

```
# jcliff --version
No JBOSS_HOME provided, aborting...
```

Définir la variable d'environnement JBOSS_HOME

JCliff est bien installé, mais à l'image du « *playbook* », il requiert que la variable **`JBOSS_HOME`** soit définie :

```
$ export JBOSS_HOME=/path/to/wildfly/home
```

Une fois la variable d'environnement requise définie, nous pouvons exécuter à nouveau JCliff :

```
# jcliff
Jcliff version 2.12.5
Usage:
    jcliff [options] file(s)
where options are:
  --cli=Path : jboss-cli.sh. Defaults to
               /usr/share/jbossas/bin/jboss-cli.sh
  --controller=host         : EAP6 host. Defaults to localhost.
  --user=username           : EAP6 admin user name
  --password=pwd            : EAP6 admin password
  --ruledir=Path            : Location of jcliff rules.
  --noop                    : Read-only mode
  --json                    : Use json to parse input files
  -v                        : Verbose output
  --timeout=timeout         : Command timeout in milliseconds
  --output=Path             : Log output file
  --reload                  : Reload after each subsystem configuration
if required
  --waitport=waitport       : Wait this many seconds for the port to be
opened
  --nobatch                 : Don't use batch mode of jboss-cli
  --redeploy                : Redeploy all apps
  --reconnect-delay=delay : Wait this many milliseconds after a :reload
for the server to restart
  --leavetmp                : Don't erase temp files
  --pre=str                 : Prepend str to all commands (can be used
for domain mode support)
```

Vérifier le respect de l'idempotence

Avec un outil comme Ansible, il est essentiel de s'assurer de l'idempotence de son exécution. Ce qui signifie, en langage vernaculaire, que les tâches décrites dans le « *playbook* » ne doivent être exécutées que si l'état du système n'est pas en cohérence avec les attentes. Dans notre cas, l'outil JCliff étant désormais installé, une nouvelle exécution du « *playbook* » ne doit, en aucun cas, aboutir à de nouvelles opérations de modifications sur le système.

Vérifions que la procédure d'installation de JCliff fournie avec la collection `wildfly.jcliff` respecte bien cette politique :

```
# ansible-playbook  playbook.yml

PLAY [localhost] ***************************************************
********************************************************************
*********************************************

TASK [Gathering Facts] *********************************************
********************************************************************
*********************************************
ok: [localhost]

TASK [wildfly.jcliff.jcliff : Collect Supported Operating Systems] ****
********************************************************************
*********************************************
ok: [localhost] => (item={u'key': u'homebrew', u'value': [u'MacOSX']})
ok: [localhost] => (item={u'key': u'rpm', u'value': [u'Fedora',
u'CentOS', u'RedHat']})

TASK [wildfly.jcliff.jcliff : Verify supported Operating Systems] *****
********************************************************************
*********************************************
ok: [localhost]

TASK [wildfly.jcliff.jcliff : Install JCliff using HomeBrew] **********
********************************************************************
*********************************************
skipping: [localhost]

TASK [wildfly.jcliff.jcliff : Install JCliff using RPM] ***************
********************************************************************
*********************************************
included: /root/.ansible/collections/ansible_collections/wildfly/
jcliff/roles/jcliff/tasks/install_rpm.yml for localhost
```

```
TASK [wildfly.jcliff.jcliff : Add JCliff Yum Repository (RedHat)] *****
*******************************************************************
*******************************************
ok: [localhost]

TASK [wildfly.jcliff.jcliff : Test if package jcliff is already
installed] ********************************************************
*******************************************
ok: [localhost]

TASK [wildfly.jcliff.jcliff : Install Jcliff using standalone binary]
*******************************************************************
*******************************************
skipping: [localhost]

PLAY RECAP *********************************************************
*******************************************************************
*******************************************
localhost                  : ok=6    changed=0    unreachable=0
failed=0    skipped=2    rescued=0    ignored=0
```

Avec cette dernière vérification, nous en avons fini avec les prérequis, et nous voilà enfin prêts à aborder l'utilisation de JCliff au sein d'un « *playbook* » Ansible.

3. UTILISER JCLIFF AVEC ANSIBLE

Nous arrivons maintenant au cœur de notre démonstration. Tout ce que nous avons mis en place jusqu'à maintenant n'avait qu'un objectif : nous permettre d'utiliser le nouveau module Ansible `jcliff:`, mis à disposition par la collection `wildfly.jcliff`. Ce nouveau module accepte une série de sous-éléments, chacun identifiant un sous-système spécifique de Wildfly.

Configurer des variables système au sein d'un serveur Wildfly

Commençons par utiliser une des fonctionnalités les plus simples de JCliff. Ceci nous permettra de valider que tout fonctionne bien avant d'aller plus loin.

Cette fonctionnalité permet de configurer une propriété système, soit une valeur globale, partagée entre toutes les applications hébergées par Wildfly. Nous allons donc définir une propriété nommée `jcliff.enabled` donc le contenu indiquera si le serveur a été configuré (ou non) par Ansible. Il ne s'agit bien sûr que d'une convention, mais son utilisation est fortement recommandée. En effet, sa présence permet aux applications déployées sur le serveur JEE de vérifier qu'Ansible a été utilisé ou non pour le configurer.

```
---
- hosts: localhost
  gather_facts: true
  vars:
    jboss_home: "{{ lookup('env','JBOSS_HOME') }}"
```

```
collections:
  - wildfly.jcliff
roles:
  - jcliff
tasks:
  - jcliff:
      wfly_home: "{{ jboss_home }}"
      subsystems:
        - system_props:
            - name: jcliff.enabled
              value: 'enabled'
```

La configuration ci-dessus est relativement explicite. La seule information que requiert le module `jcliff:` est, sans surprise, l'emplacement du serveur Wildfly à configurer (soit le contenu de la variable `jboss_home`). Et c'est d'ailleurs à cette fin que nous avons pris soin de nous assurer de sa définition et de sa présence lors de la mise en place des prérequis.

Le lecteur prendra soin de noter que JCliff n'a besoin de cette valeur qu'afin de localiser et utiliser le script `${JBOSS_HOME}/bin/jboss-cli.sh`. Fourni avec Wildfly, ce script permet d'utiliser JBoss CLI en ligne de commandes.

Exécutons notre « *playbook* » modifié :

```
# ansible-playbook playbook.yml

PLAY [localhost] ***************************************************
********************************************************************
*********************************************

TASK [Gathering Facts] *********************************************
********************************************************************
*********************************************
ok: [localhost]

TASK [wildfly.jcliff.jcliff : Collect Supported Operating Systems] ****
********************************************************************
*********************************************
ok: [localhost] => (item={u'key': u'homebrew', u'value': [u'MacOSX']})
ok: [localhost] => (item={u'key': u'rpm', u'value': [u'Fedora',
u'CentOS', u'RedHat']})

TASK [wildfly.jcliff.jcliff : Verify supported Operating Systems] *****
********************************************************************
*********************************************
ok: [localhost]

TASK [wildfly.jcliff.jcliff : Install JCliff using HomeBrew] **********
********************************************************************
*********************************************
skipping: [localhost]
```

```
TASK [wildfly.jcliff.jcliff : Install JCliff using RPM] ****************
*******************************************************************
*********************************************
included: /root/.ansible/collections/ansible_collections/wildfly/
jcliff/roles/jcliff/tasks/install_rpm.yml for localhost

TASK [wildfly.jcliff.jcliff : Add JCliff Yum Repository (RedHat)] *****
*******************************************************************
*********************************************
ok: [localhost]

TASK [wildfly.jcliff.jcliff : Test if package jcliff is already
installed] ********************************************************
*********************************************
ok: [localhost]

TASK [wildfly.jcliff.jcliff : Install Jcliff using standalone binary]
*******************************************************************
*********************************************
skipping: [localhost]

TASK [jcliff] ******************************************************
*******************************************************************
*********************************************
changed: [localhost]

PLAY RECAP *********************************************************
*******************************************************************
*********************************************
localhost                  : ok=7    changed=1    unreachable=0
failed=0    skipped=2    rescued=0    ignored=0
```

Tout s'est bien déroulé et Ansible nous signale que la tâche `jcliff:` a bien abouti à une modification d'état sur le système cible. En effet, la propriété que nous avons choisi de déclarer au sein de la configuration de Wildfly n'était pas définie, donc JCliff a remarqué l'état erroné et a modifié en conséquence la configuration en y déclarant cette nouvelle propriété.

Tout ceci a lieu à l'aide d'une communication entre JCliff et le serveur rendue possible par l'utilisation de JBoss CLI.

Néanmoins, soyons sceptiques et vérifions que l'état obtenu est bien celui que l'on attend. On peut effectuer cette vérification en exécutant simplement manuellement la requête JBoss CLI suivante :

```
# ${JBOSS_HOME}/bin/jboss-cli.sh --connect --command='/system-
property=jcliff.enabled:read-resource'
{
    "outcome" => "success",
    "result" => {"value" => "enabled.plus"}
}
```

Comme il s'agit de notre toute première utilisation du module `jcliff:` soyons exhaustifs dans notre vérification de son bon comportement et allons vérifier que le fichier de configuration XML du serveur a aussi été mis à jour :

```
...
 </extensions>
    <system-properties>
        <property name="jcliff.enabled" value="enabled.plus"/>
    </system-properties>
    <management>
...
```

À l'aide de cette première fonctionnalité, très simple, du nouveau module, nous avons pu valider la bonne configuration de notre « *playbook* » ainsi que le fonctionnement de cette intégration. Il est donc bien temps de passer à la vitesse supérieure et de nous attaquer à la configuration d'un sous-système bien plus complexe de Wildlfy.

Configurer une source de données

De nombreuses applications Java consomment et manipulent des informations issues de différentes bases de données relationnelles. Ces systèmes sont abstraits au sein du serveur d'applications JEE sous le terme de source de données. Le serveur permet donc d'isoler les applicatifs qu'il héberge de ces systèmes et leur permet d'interagir avec eux à partir d'un pilote logiciel (un « *driver JDBC* »). Wildlfy prend donc en charge le nombre de connexions à la base de données afin d'assurer qu'elle ne soit pas débordée par le nombre de requêtes si la charge du serveur applicatif venait s'accroître subitement.

Nous allons donc maintenant illustrer comment on peut utiliser le module `jcliff:` afin de configurer une telle source de données au sein du serveur d'application JEE.

Il faut noter que le serveur supporte de nombreux pilotes JDBC, mais qu'un seul est installé par défaut : le pilote pour la base de données embarquée Hypersonic (ou h2). Pour ses propres besoins, le serveur peut en effet instancier une telle source de données en mémoire.

Pour commencer, nous allons donc instancier une nouvelle source source de données de ce type, en étudiant celle déjà définie au sein de la configuration de Wildfly par défaut :

```
        <subsystem xmlns="urn:jboss:domain:datasources:6.0">
            <datasources>
                <datasource jndi-name="java:jboss/datasources/
ExampleDS" pool-name="ExampleDS" enabled="true" use-java-context="true"
statistics-enabled="${wildfly.datasources.statistics-enabled:${wildfly.
statistics-enabled:false}}">
                    <connection-url>jdbc:h2:mem:test;DB_CLOSE_DELAY=-
1;DB_CLOSE_ON_EXIT=FALSE</connection-url>
                    <driver>h2</driver>
                    <security>
                        <user-name>sa</user-name>
                        <password>sa</password>
                    </security>
                </datasource>
...
```

Nous allons donc chercher à reproduire cette configuration afin de créer une seconde instance de **h2** que nous allons nommer **H2DS4Test** :

```
---
- hosts: localhost
  gather_facts: true
  vars:
    jboss_home: "{{ lookup('env','JBOSS_HOME') }}"
  collections:
    - wildfly.jcliff
  roles:
    - jcliff
  tasks:

    - jcliff:
        wfly_home: "{{ jboss_home }}"
        subsystems:
          - system_props:
              - name: jcliff.enabled
                value: 'enabled.plus'
          - datasources:
              - name: H2DS4Test
                jndi_name: java:jboss/datasources/H2DS4Test
                connection_url: "jdbc:h2:mem:test2;DB_CLOSE_DELAY=-
1;DB_CLOSE_ON_EXIT=FALSE"
                driver_name: h2
```

Là encore, la configuration au sein du module **jcliff:** est relativement explicite. Nous avons juste besoin d'ajouter les informations de connexion à la base de données sous forme d'URL, le type du pilote (**h2**) et le nom JNDI attribué (« *Java Naming and Directory Interface* », soit un mécanisme de localisation de ressources applicatives spécifique à Java).

Une fois le « *playbook* » exécuté sans erreur, le serveur dispose d'une nouvelle source de données nommée **H2DS4Test**. On peut vérifier ceci simplement en consultant le fichier de journalisation du serveur :

```
...
09:53:03,542 INFO  [org.jboss.as.connector.subsystems.datasources]
(MSC service thread 1-6) WFLYJCA0001: Bound data source [java:jboss/
datasources/H2DS4Test]
...
```

Déployer un nouveau driver JDBC

Comme indiqué dans la précédente section, Wildfly ne dispose d'aucun pilote JDBC à son installation. Déployer un nouveau pilote au sein du serveur est donc une tâche très commune et probablement la première opération que l'on souhaitera automatiser avec Ansible. C'est donc à la mise en place d'une telle automatisation que nous allons nous intéresser dans cette section.

La toute première étape va consister à récupérer le pilote logiciel en lui-même, généralement fourni sous forme d'archive JAR. Le déploiement de cette archive est très dépendant du système d'information et de son infrastructure. Ici, nous allons donc faire simple et récupérer l'archive jar directement depuis un dépôt Maven public. Ceci n'est évidemment pas recommandé pour un système en production (pour des raisons évidentes de sécurité), mais suffit amplement à notre démonstration.

Une fois l'archive récupérée, il faut la déployer au sein d'un module JBoss qui est constitué essentiellement d'une arborescence de fichiers, située sous `${JBOSS_HOME}/modules` et d'un descripteur XML.

Mettons tout ceci en place en utilisant les primitives habituelles d'Ansible :

```
...
  vars:
    jboss_home: "{{ lookup('env','JBOSS_HOME') }}"
    psql_module_home: "{{ jboss_home }}//modules/org/postgresql/main/"
    jdbc_driver_version: 9.2-1002-jdbc4
    jdbc_driver_jar_filename: "postgresql-{{ jdbc_driver_version
}}.jar"

  collections:
    - wildfly.jcliff
  roles:
    - jcliff
  tasks:

   - name: "Set up module dir for Postgres JDBC Driver"
     file:
       path: "{{ psql_module_home }}"
       state: directory
       recurse: yes

   - name: "Ensures Wildfly Postgres Driver is present"
     uri:
       url: "https://repo.maven.apache.org/maven2/org/postgresql/
postgresql/{{ jdbc_driver_version }}/{{ jdbc_driver_jar_filename }}"
       dest: "{{ psql_module_home }}"
```

```
    creates: "{{ psql_module_home }}"

- name: "Deploy module.xml for Postgres JDBC Driver"
  template:
    src: templates/pgsql_jdbc_driver_module.xml.j2
    dest: "{{ psql_module_home }}/module.xml"
```

En seulement trois tâches, nous avons intégralement automatisé l'installation d'un nouveau pilote JDBC pour Wildfly. En l'occurrence, il s'agit d'un pilote pour la base de données PostgreSQL.

À titre informatif, voici le contenu du fichier patron (« *template* ») utilisé pour générer le descripteur du module JBoss :

```
<module xmlns="urn:jboss:module:1.0" name="org.postgresql">
  <resources>
    <resource-root path="{{ jdbc_driver_jar_filename }}"/>
  </resources>
  <dependencies>
    <module name="javax.api"/>
    <module name="javax.transaction.api"/>
  </dependencies>
</module>
```

À la différence du fichier de configuration de Wildfly mentionné plus haut, ce fichier est entièrement statique. Son contenu ne change pas après le démarrage du serveur, il forme donc un parfait candidat pour être généré à l'aide de la fonctionnalité de « *templating* » d'Ansible.

Si les fichiers nécessaires au déploiement du nouveau pilote ont été mis en place, il reste encore à en activer l'utilisation au sein du serveur Wildfly. Il s'agit en fait de la dernière étape de l'installation du module. Sans cette dernière, les fichiers déployés n'ont aucune incidence sur l'état du serveur.

Cependant, avant d'étudier ceci, vérifions déjà que l'installation que nous avons automatisée fonctionne bien comme prévu :

```
$ ansible-playbook playbook.yml
...
TASK [wildfly.jcliff.jcliff : Test if package jcliff is already
installed] ***********************************************************
*************************************************
ok: [localhost]

TASK [wildfly.jcliff.jcliff : Install Jcliff using standalone binary]
***********************************************************************
*************************************************
skipping: [localhost]

TASK [Set up module dir for Postgres JDBC Driver] ********************
***********************************************************************
*************************************************
changed: [localhost]
```

```
TASK [Ensures Wildfly Postgres Driver is present] ***********************
**********************************************************************
**********************************************
ok: [localhost]

TASK [Deploy module.xml for Postgres JDBC Driver] ***********************
**********************************************************************
**********************************************
changed: [localhost]

TASK [jcliff] ***********************************************************
**********************************************************************
**********************************************
ok: [localhost]

PLAY RECAP **************************************************************
**********************************************************************
**********************************************
localhost                  : ok=10   changed=2      unreachable=0
failed=0 skipped=2 rescued=0 ignored=0
```

Parfait, tout est en place, il ne nous reste plus qu'à activer ce module JBoss. Pour ce faire, nous allons donc utiliser la fonctionnalité associée fournie par le module `jcliff:` de la collection éponyme :

```
- jcliff:
 wfly_home: "{{ jboss_home }}"
 subsystems:
   - system_props:
         - name: jcliff.enabled
       value: 'enabled.plus'
   - drivers:
         - driver_name: postgresql
       driver_module_name: org.postgresql
       driver_class_name: org.postgresql.Driver
       driver_xa_datasource_class_name: org.postgresql.xa.PGXADataSource
  - datasources:
         - name: H2DS4Test
       use_java_context: 'true'
       jndi_name: java:jboss/datasources/H2DS4Test
       connection_url: "jdbc:h2:mem:test2;DB_CLOSE_DELAY=-1;DB_CLOSE_ON_
EXIT=FALSE"
       driver_name: h2
```

Exécutons à nouveau notre « *playbook* » et observons le fichier de journalisation du serveur Wildfly afin de confirmer que le nouveau pilote JDBC est bien disponible désormais :

```
...
13:24:25,474 INFO  [org.jboss.as.connector.subsystems.datasources] (management-
handler-thread - 1) WFLYJCA0005: Deploying non-JDBC-compliant driver class org.
postgresql.Driver (version 9.2)
13:24:25,474 INFO  [org.jboss.as.connector.deployers.jdbc] (MSC service thread
1-1) WFLYJCA0018: Started Driver service with driver-name = postgresql
13:24:27,464 INFO  [org.jboss.as.connector.deployers.jdbc] (MSC service thread
1-2) WFLYJCA0019: Stopped Driver service with driver-name = postgresql
...
```

Déployer des applications avec JCliff

Jusqu'à maintenant, nous avons seulement utilisé JCliff pour sa capacité à configurer les sous-systèmes du serveur Java JEE Wildfly. Cependant, l'outil peut aller plus loin et même nous permettre de déployer des applications au sein du serveur. Nous allons ici en faire la démonstration en déployant le serveur d'intégration continue Jenkins.

Tout d'abord, nous allons mettre en place un dépôt Yum sur le système cible afin de pouvoir installer facilement l'archive WAR associée à Jenkins. Ceci est bien évidemment entièrement pris en charge par Ansible :

```
...
   - name: Jenkins Yum Repository
  yum_repository:
      name: jenkins
      description: Jenkins
      baseurl: http://pkg.jenkins.io/redhat
      gpgcheck: 1

   - name: Install Jenkins
  yum:
      name: jenkins
      state: present
...
```

Attention !

L'ajout de ce dépôt requiert l'import d'une clé GPG, n'oubliez pas d'ajouter celle-ci avant d'exécuter le « *playbook* » ci-dessus !

```
rpm --import https://pkg.jenkins.io/redhat/jenkins.io.key
```

Une fois que le « *playbook* » est exécuté sans erreur, le paquet RPM associé à Jenkins est installé sur le système. Ceci signifie que le fichier WAR dont nous avons besoin pour déployer le service est désormais disponible sur le système cible :

```
$ yum whatprovides */jenkins*.war
...
jenkins-2.253-1.1.noarch : Jenkins Automation Server
Repo     : @jenkins
Matched from:
Filename : /usr/lib/jenkins/jenkins.war
```

Nous n'avons plus qu'à déployer l'application sur le serveur Wildlfy. Pour ce faire, il nous suffit d'ajouter cette application à la configuration du module **jcliff:**, comme indiqué ci-dessous :

```
- jcliff:
 wfly_home: "{{ jboss_home }}"
 subsystems:
   - system_props:
       - name: jcliff.enabled
         value: 'enabled.plus'
   - drivers:
       - driver_name: postgresql
         driver_module_name: org.postgresql
         driver_class_name: org.postgresql.Driver
         driver_xa_datasource_class_name: org.postgresql.
xa.PGXADataSource
   - datasources:
       - name: H2DS4Test
         use_java_context: 'true'
         jndi_name: java:jboss/datasources/H2DS4Test
         connection_url: "jdbc:h2:mem:test2;DB_CLOSE_DELAY=-1;DB_
CLOSE_ON_EXIT=FALSE"
         driver_name: h2
   - deployments:
       - name: jenkins
         path: /usr/lib/jenkins/jenkins.war
```

Seuls le nom et le chemin vers l'archive WAR sont requis ! À partir de ces deux seules informations, Ansible et JCliff se chargent de tout ! On notera aussi que le module `jcliff:` permet d'aller encore plus loin, car il peut gérer le déploiement de mises à jour de cet applicatif.

Pour le moment, vérifions déjà que l'application soit correctement déployée au sein du serveur JEE :

```
...
14:06:09,148 INFO  [org.jboss.as.repository] (management-handler-thread
- 2) WFLYDR0001: Content added at location /opt/wildfly-20.0.1.Final/
standalone/data/content/6b/30899e9a28a361241d19561c965977aea70d7f/
content
14:06:09,171 INFO  [org.jboss.as.server.deployment] (MSC service thread
1-5) WFLYSRV0027: Starting deployment of "jenkins" (runtime-name:
"jenkins")
14:06:09,840 WARN  [org.jboss.as.server.deployment] (MSC service thread
1-5) WFLYSRV0274: Excluded dependency com.fasterxml.jackson.core.
jackson-core via jboss-deployment-structure.xml does not exist.
14:06:09,841 WARN  [org.jboss.as.server.deployment] (MSC service thread
1-5) WFLYSRV0274: Excluded dependency org.jboss.resteasy.resteasy-
jackson-provider via jboss-deployment-structure.xml does not exist.
14:06:09,841 WARN  [org.jboss.as.server.deployment] (MSC service thread
1-5) WFLYSRV0274: Excluded dependency com.fasterxml.jackson.core.
jackson-databind via jboss-deployment-structure.xml does not exist.
14:06:09,841 WARN  [org.jboss.as.server.deployment] (MSC service thread
1-5) WFLYSRV0274: Excluded dependency com.fasterxml.jackson.core.
jackson-annotations via jboss-deployment-structure.xml does not exist.
14:06:10,189 INFO  [org.infinispan.PERSISTENCE] (MSC service thread
1-7) ISPN000556: Starting user marshaller 'org.wildfly.clustering.
infinispan.marshalling.jboss.JBossMarshaller'
14:06:10,209 INFO  [org.infinispan.CONTAINER] (MSC service thread 1-7)
ISPN000128: Infinispan version: Infinispan 'Turia' 10.1.8.Final
14:06:10,501 INFO  [org.jboss.as.clustering.infinispan] (ServerService
Thread Pool -- 78) WFLYCLINF0002: Started client-mappings cache from
ejb container
14:06:10,557 INFO  [org.jboss.as.server] (management-handler-thread -
2) WFLYSRV0010: Deployed "jenkins" (runtime-name : "jenkins")
...
```

4. UTILISER UNE RÈGLE JCLIFF PERSONNALISÉE

Le module `jcliff:` permet déjà de configurer les sous-systèmes les plus souvent utilisés de Wildfly, mais l'outil JCliff en propose un éventail beaucoup plus large qui n'est pas encore intégré à ceux supportés par la collection pour Ansible. En outre, JCliff permet de concevoir sa propre règle et ouvre ainsi la porte à l'automatisation de configurations relativement avancées et complexes de Wildfly.

Fort heureusement, l'absence d'intégration directe au sein du module `jcliff:` n'interdit aucunement à son utilisateur d'avoir recours aux nombreuses fonctionnalités supplémentaires de l'outil. En effet, le module permet d'exécuter une règle JCliff personnalisée qui lui est fournie sous la forme d'un simple fichier texte.

Nous allons donc en faire ici la démonstration à l'aide d'une règle toute prête, fournie par le projet JCliff. Cette règle permet la configuration du sous-système mail de Wildfly. On notera que les dernières versions de la collection intègrent désormais cette fonctionnalité au module `jcliff:`.

Toujours disponible sur

www.ed-diamond.com

PAPIER

FLIPBOOK HTML5

sur www.ed-diamond.com

CONNECT

LA DOCUMENTATION TECHNIQUE DES PROS DE L'IT

sur connect.ed-diamond.com

Voici le contenu de la règle JCliff issue du site du projet :

```
{
  "mail" => {
     "mail-session" => {
        "testSession" => {
           "from" => "a@b.com",
           "jndi-name" => "java:jboss/mail/testSession",
           "server" => {
              "smtp" => {
                 "outbound-socket-binding-ref" => "mail-smtp",
                 "ssl" => false
              }
           }
        }
     }
  }
}
```

Commençons par automatiser le téléchargement de cette règle :

```
vars:
...
  jcliff_config_files_dir: /opt/jcliff
...
  - name: Create directory for jcliff rules
    file:
      name: "{{ jcliff_config_files_dir }}"
      state: directory

  - name: Download mail.test configuration file for JCLiff
    uri:
      url: https://raw.githubusercontent.com/bserdar/jcliff/master/
testscripts/mail.test
      dest: "{{ jcliff_config_files_dir }}/mail.test"
      creates: "{{ jcliff_config_files_dir }}/mail.test"
```

Ceci effectué, mettons en place l'exécution de cette règle par le module **jcliff:** au sein de notre « *playbook* ». Cette mise en place ne requiert que la définition d'un répertoire contenant l'ensemble des règles JCliff qu'Ansible devra exécuter :

```
...
- jcliff:
      wfly_home: "{{ jboss_home }}"
      rule_file: "{{ jcliff_config_files_dir }}"
      subsystems:
...
```

Une fois le « *playbook* » exécuté sans incident, on peut vérifier que la configuration du sous-système **mail** a bien été effectué en étudiant le fichier de journalisation de Wildlfy :

```
...
14:38:36,372 INFO  [org.jboss.as.mail.extension] (MSC service thread
1-6) WFLYMAIL0001: Bound mail session [java:jboss/mail/testSession]
...
```

Si cette entrée confirme la présence de la session, elle ne nous donne que peu d'informations. Afin de vérifier que notre configuration est conforme aux instructions de notre règle JCliff dans ses moindres détails, nous pouvons utiliser la requête JBoss CLI suivante :

```
/subsystem=mail/mail-session=testSession:read-resource
{
 "outcome" => "success",
 "result" => {
     "debug" => false,
     "from" => "a@b.com",
     "jndi-name" => "java:jboss/mail/testSession",
     "custom" => undefined,
     "server" => {"smtp" => undefined}
 },
 "response-headers" => {"process-state" => "reload-required"}
}
```

Le lecteur attentif aura remarqué que, dans sa réponse, le serveur Wildfly signale qu'un redémarrage est nécessaire (« *reload-required* »). Si le serveur a été déployé comme un service système, il est aisé d'utiliser les primitives d'Ansible afin d'automatiser ceci :

```
- name: "Restart server"
  service:
    name: 'jbossas'
    state: restarted
  changed_when: false
```

Si ce n'est pas le cas, on peut aussi simplement assurer ce redémarrage à l'aide de l'exécution d'une requête JBoss CLI :

```
- name: "Restart Wildfly"
   command: "{{ jboss_home }}/bin/jboss-cli.sh --connect --command=':reload'"
```

5. COMMENT ANALYSER ET RÉSOUDRE UN PROBLÈME AVEC JCLIFF ?

Si la démonstration de cet article s'est écoulée sans la moindre difficulté, il serait malhonnête de suggérer qu'il est impossible de rencontrer des problèmes dans la mise en place et l'utilisation de JCliff et de son module pour Ansible. Nous allons donc évoquer ici quelques techniques et pratiques afin de faciliter l'analyse et la résolution de celles-ci.

Le point le plus important à retenir est que l'implémentation du module `jcliff:` ne contient pas réellement de logique de traitement. Il se contente de générer une règle JCliff à partir des informations fournies et de s'assurer que JCliff l'exécute. À l'exception d'une erreur de configuration sur les chemins d'un fichier, il y a donc assez peu de possibilités d'erreur. Le meilleur outil pour analyser ce genre de difficulté reste l'affichage complet des messages d'informations d'Ansible à l'aide de l'option `-vvvv`.

Si une erreur apparaît lors de l'exécution de la règle JCliff sur le serveur Wildfly en train d'être configuré, le plus simple est de sortir entièrement du contexte d'utilisation d'Ansible et de vérifier que la règle en question (qu'elle soit générée par le module ou fournie à celui-ci) soit fonctionnelle. Dans le cas d'une règle générée, il suffit de récupérer le fichier produit par le module à partir des fichiers temporaires associés à l'exécution du « *playbook* ».

Bien évidemment, ces fichiers sont supprimés après l'exécution, il est donc nécessaire d'indiquer à Ansible de les conserver :

```
export ANSIBLE_KEEP_REMOTE_FILES=1
```

Une fois cette configuration mise en place, exécutez à nouveau le « *playbook* » à l'aide de la commande `ansible-playbook` et avec l'option `-vvvv` afin de pouvoir retrouver, au sein de la sortie générée, l'emplacement et le nom du fichier temporaire généré par le module. À titre d'exemple, le fichier généré pour implémenter la configuration de pilote JDBC se nommera : `drivers-0.jcliff.yml`.

Le fichier localisé, il suffit de l'exécuter directement avec JCliff, avec l'option `-v` si nécessaire, pour en apprendre plus sur la source de l'erreur :

```
$ jcliff -v ~/.ansible/tmp/ansible-
tmp-1597756555.46-5332-269490407528192//drivers-0.jcliff.yml
```

Si le problème n'est pas apparent immédiatement, il est possible de récupérer, à partir de la sortie complète de JCliff, les requêtes JBoss CLI exécutées sur le serveur. En exécutant celles-ci, une par une, à l'aide de JBoss CLI, il devrait alors être facile d'identifier laquelle pose problème.

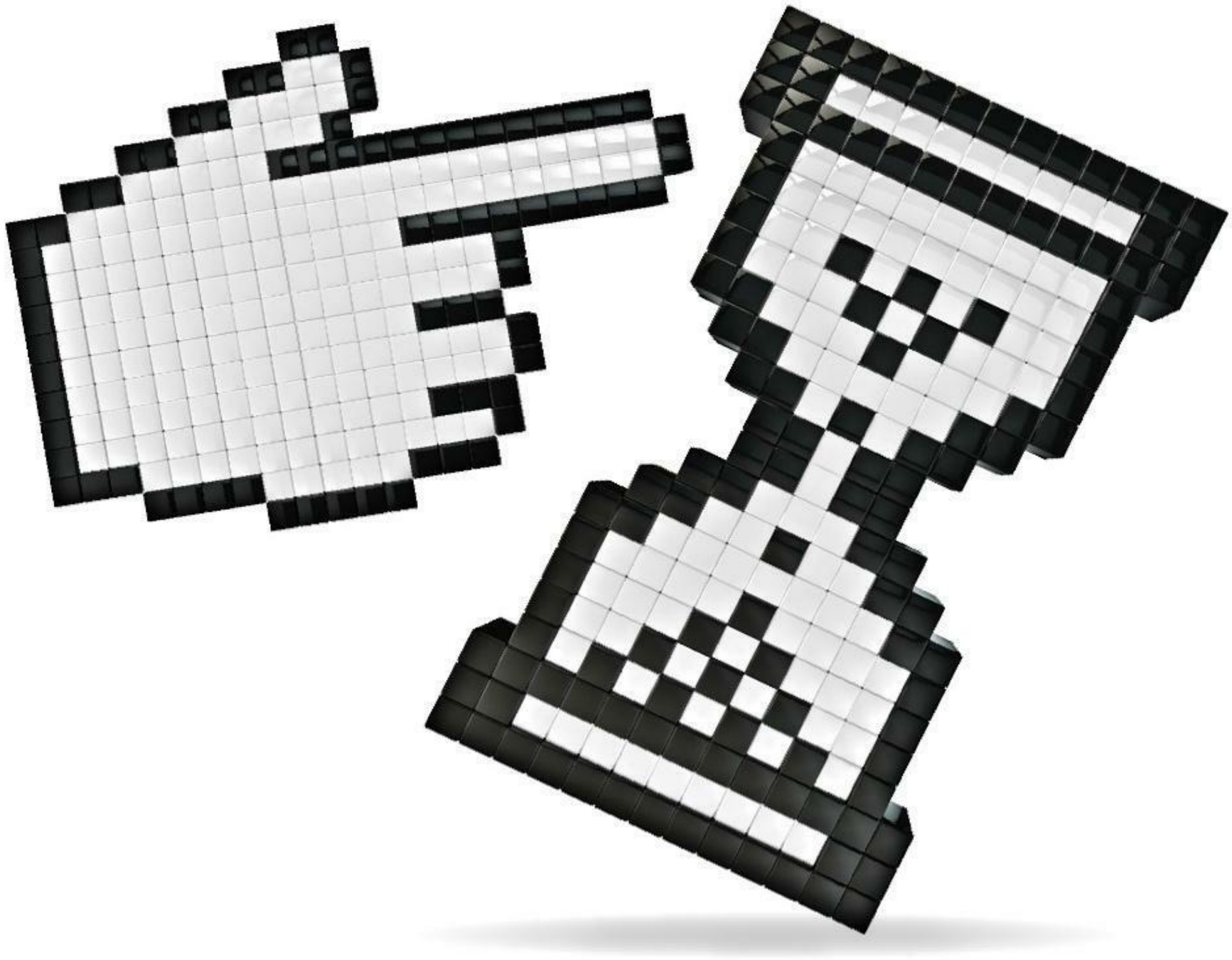
CONCLUSION

Avant la publication de cette collection JCliff pour Ansible, l'automatisation du déploiement et de la configuration d'un serveur Wildlfy était fortement limitée et exigeait de concevoir de nombreuses requêtes JBoss CLI, afin de contourner la limitation induite par la mise à jour, à l'exécution, du fichier de configuration du serveur. Même si l'on avait déjà identifié JCliff comme une solution à cette problématique, le manque d'intégration au sein d'Ansible rendait son utilisation compliquée.

Grâce à cette intégration, la manipulation d'une instance Wildfly ressemble exactement à celle de n'importe quelles autres ressources placées sous le contrôle d'Ansible. L'outil maîtrise réellement et complètement l'état du service et est donc en mesure de détecter les éventuelles incohérences avec la configuration souhaitée, et aussi résoudre le problème.

Bénéfice supplémentaire : la déviation du service, par rapport à sa configuration de base, est documentée, en quelques lignes au sein du « *playbook* » et non répartie sur un ensemble plus ou moins complexe de requêtes JBoss CLI (ou, pire, un autre mécanisme « maison »)... ▌

PLUS ENVIE D'ATTENDRE ?
PRENEZ UN TEMPS D'AVANCE !



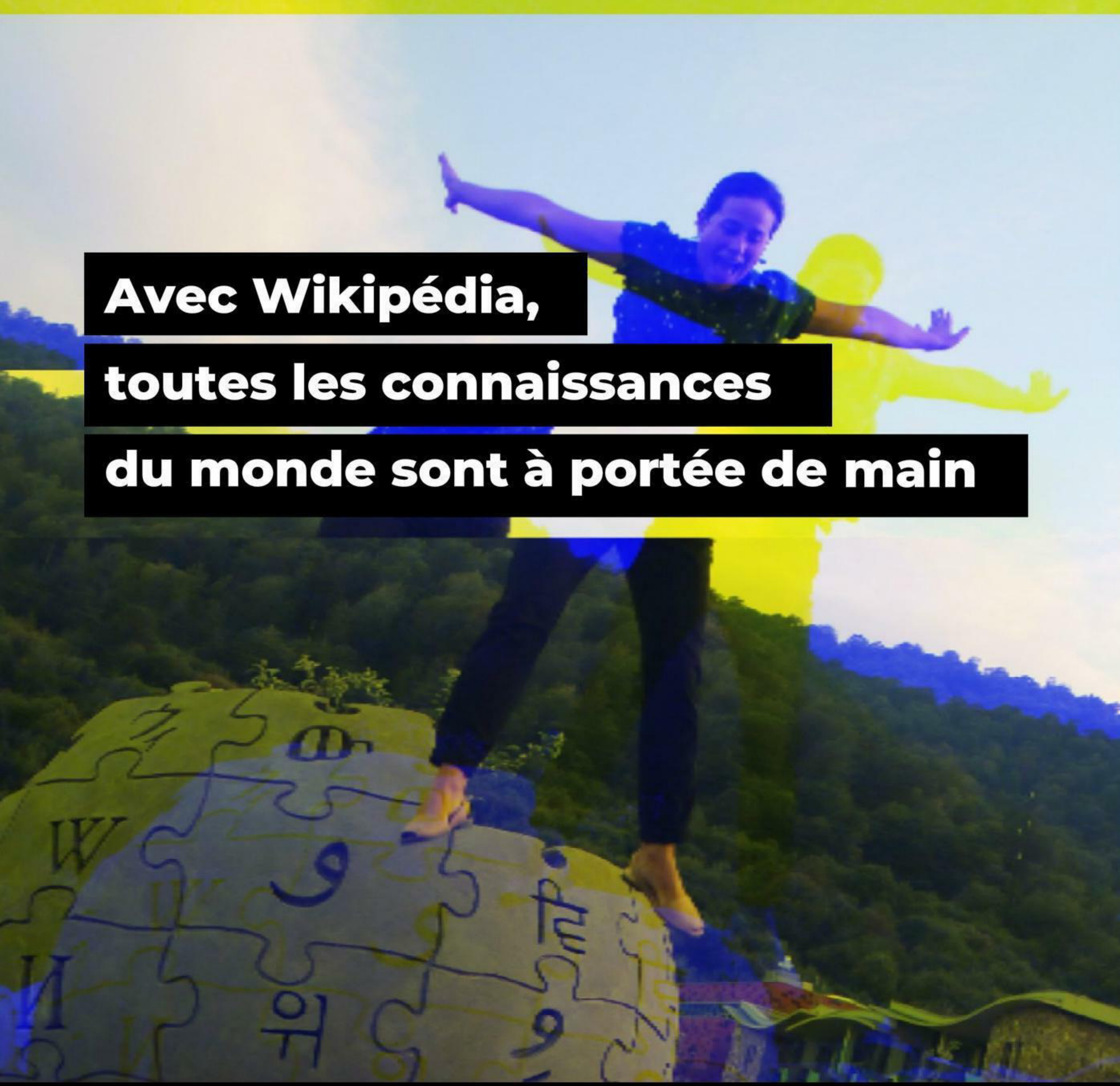
Lisez nos articles en ligne en avant-première !

EXCLUSIVEMENT SUR

CONNECT

LA DOCUMENTATION TECHNIQUE DES PROS DE L'IT

connect.ed-diamond.com



Faites un don
pour la connaissance libre
→ dons.wikimedia.fr

WIKIMEDIA
FRANCE