RÉPUBLIQUE
FRANÇAISE
Liberté
Égalité
Fraternité

RECOMMANDATIONS DE SÉCURITÉ RELATIVES AU DÉPLOIEMENT D'UNE INFRASTRUCTURE OPENSTACK POUR UN SERVICE IAAS SECNUMCLOUD

GUIDE ANSSI

ANSSI-BP-104
16/04/2024

PUBLIC VISÉ :

Développeur **Administrateur** RSSI DSI Utilisateur

Informations

Attention

Ce document rédigé par l'ANSSI s'intitule **« Recommandations de sécurité relatives au déploiement d'une infrastructure OpenStack pour un service IaaS SecNumCloud »**. Il est téléchargeable sur le site `cyber.gouv.fr`.

Sauf disposition réglementaire contraire, les recommandations n'ont pas de caractère normatif ; elles sont livrées en l'état et adaptées aux menaces au jour de leur publication. Au regard de la diversité des systèmes d'information, l'ANSSI ne peut garantir que ces informations puissent être reprises sans adaptation sur les systèmes d'information cibles. Dans tous les cas, la pertinence de l'implémentation des éléments proposés par l'ANSSI doit être soumise, au préalable, à la validation de l'administrateur du système et/ou des personnes en charge de la sécurité des systèmes d'information.

Évolutions du document :

VERSION	DATE	NATURE DES MODIFICATIONS
1.0	16/04/2024	Version initiale

Table des matières

1
Introduction

1.1 Objectif du guide

Ce guide est destiné à des personnes chargées de la mise en œuvre d'une infrastructure OpenStack sécurisée. Il présente des exemples de solutions permettant d'être conforme aux exigences du référentiel SecNumCloud qui ne sont pas directement couvertes par une fonctionnalité implémentée dans la suite logicielle.

Il s'adresse à des personnes ayant déjà une bonne connaissance des fonctions implémentées dans les services OpenStack, des échanges effectués avec ces services et entre ces services, ainsi que des connaissances système Linux.

Les recommandations présentes dans ce document s'appuient sur les fonctionnalités standard des services OpenStack. Les éléments de configuration nécessaires à la mise en œuvre de ces fonctionnalités pouvant varier selon la solution de déploiement retenue, le détail de la configuration des services ne sera pas présenté. Ce document est le fruit de travaux réalisés sur des solutions de déploiement communautaires.

1.2 Organisation du guide

Après une brève présentation de la suite logicielle OpenStack ainsi que du référentiel SecNumCloud (chapitre 1), ce guide présente au chapitre 2 différentes techniques permettant de séparer les interfaces d'administration à destination du prestataire de celles offertes aux commanditaires.

Le chapitre 3 présente la mise en place d'un mécanisme d'authentification multi-facteurs lors de l'accès aux interfaces d'administration.

L'utilisation de chiffrement pour la protection des flux et des données des commanditaires est étudiée au chapitre 4.

Le chapitre 5 aborde le cloisonnement des ressources et l'effacement sécurisé des données.

Enfin, le chapitre 6 présente la mise en place de la journalisation.

1.3 Liste des sigles et acronymes

administrateur Personnes en charge des opérations d'administration liées à l'infrastructure virtualisée.

API Les *Application Programming Interface* sont les interfaces permettant d'interagir avec l'infrastructure nuagique. Pour OpenStack, ces APIs gèrent des transactions formatées en JSON avec une entête HTTP et un éventuel chiffrement TLS.

biens essentiels Dans le cadre d'une infrastructure mutualisée, les biens essentiels sont les secrets manipulés par les machines virtuelles, les images importées par le commanditaire, les volumes associés aux machines virtuelles, ainsi que les données présentes dans la mémoire de ces dernières.

catalogue Liste des services de l'infrastructure OpenStack accessibles à un projet ou un domaine défini. Ce catalogue des services est maintenu à jour par Keystone. Son contenu est envoyé au client dans la réponse à une requête d'authentification.

client OpenStack Les clients OpenStack sont des composants logiciels chargés des échanges avec les APIs OpenStack. Ils peuvent prendre la forme de programmes exécutables utilisables par un opérateur ou un script déclenché de manière automatique. Ils peuvent aussi prendre la forme de librairies utilisables par des programmes tiers. Ils sont en charge du formatage des requêtes, du déformatage des réponses et de la communication réseau avec les APIs. Par défaut OpenStack dispose d'un client python en ligne de commande.

commanditaire Entité juridique utilisant le service d'informatique nuagique.

domaine Regroupement logique d'utilisateurs en charge de l'administration d'un ensemble de ressources virtulisées. Les utilisateur d'un domaine ne sont pas visibles en dehors de ce domaine.

HSM *Hardware Security Module* - Périphérique cryptographique.

informatique nuagique Un service d'informatique nuagique est une infrastructure virtualisée dont les ressources sont accessibles en libre service au moyen d'interfaces standardisées. Ces ressources sont partagées entre plusieurs projets et potentiellement plusieurs commanditaires. Elles sont attribuées à la demande et sans intervention humaine. La facturation du service est effectuée à l'usage.

KMIP *Key Management Interoperability Protocol* - Protocole d'interface standard destiné aux échanges avec les gestionnaires de clés.

MCO Maintien en conditions opérationnelles.

MCS Maintien en conditions de sécurité.

nœud de stockage Un nœud de stockage est un serveur ou un ensemble de serveurs supportant des protocoles de stockage réseau tels que iSCSI, NFS ou Fibre Channel. Ce ou ces serveurs ont de plus accès à de l'espace disque sur lequel seront effectivement stockés les volumes virtuels ou les images utilisés par les machines virtuelles.

nœud de calcul Un nœud de calcul est un serveur sur lequel s'exécute un hyperviseur et qui héberge des machines virtuelles en leur mettant à disposition du temps processeur et de la mémoire.

NTP Network Time Protocol est un protocole réseau qui permet d'utiliser un serveur externe comme référence pour l'horloge interne d'un système.

OIDC *OpenID Connect* est un protocole d'authentification basé sur OAuth 2.0.

PKCS#11 Protocole d'interface standard destiné aux échanges avec les périphériques cryptographiques.

prestataire Organisation mettant en œuvre le service d'informatique nuagique.

projet Regroupement logique d'utilisateurs en charge de l'administration d'un ensemble de ressources virtualisées. Le terme *project* a été introduit lors du passage de l'API *Keystone Identity* de V2.0 à V3.x (version *Grizzly* de 2013) et remplace le terme *tenant* [15].

ressources virtualisées Ensemble des ressources virtuelles utilisables par les commanditaires pour construire leurs infrastructures virtualisées (vCPU, mémoire, espace disque, réseau). On parle de ressources virtualisées car elles sont indépendantes des composants matériels sous-jacents ; une machine virtuelle conservera ses ressources virtualisées lorsqu'elle est migrée vers une nouvelle machine hôte.

réseau défini par logiciel Ensemble de technologies appliquées à l'architecture réseau, qui permettent de programmer le réseau de manière indépendante du réseau sous-jacent et centralisée, à l'aide de briques logicielles.

SAML 2.0 *Security Assertion Markup Language* est un protocole d'authentification basé sur XML.

secret Les *secrets* sont des éléments cryptographiques utilisés par les infrastructures virtualisées, accessibles aux utilisateurs des APIs OpenStack et gérés par le service barbican. Les secrets peuvent être des mots de passe, des clés symétriques ou asymétriques, des certificats X509 ou des blocs de données brutes.

1.4 Présentation rapide d'OpenStack

OpenStack est une suite logicielle libre destinée à faciliter la mise en place et la gestion d'infrastructures virtualisées et de services d'informatique nuagique.

Elle n'intègre pas son propre hyperviseur ou sa propre solution de réseau défini par logiciel, mais s'appuie sur les principales solutions du marché, qu'elles soient propriétaires ou libres.

Elle permet la gestion de l'infrastructure virtualisée au moyen d'interfaces accessibles en réseau. Elle intègre les notions de projets et de domaines qui permettent à un groupe d'utilisateurs d'assurer eux-mêmes la gestion d'une partie des ressources virtualisées dans le cadre de service d'informatique nuagique.

La suite logicielle OpenStack est maintenue par un ensemble d'acteurs majeurs de l'industrie

du développement logiciel [16]. Elle peut être déployée au moyen de distributions propriétaires proposées par une partie de ces acteurs ou au moyen de distributions communautaires.

La communauté met à disposition une version majeure de la suite logicielle tous les 6 mois (Austin, Bexar, Cactus, Diablo, Essex, Folsom, Grizzly, Havana, Icehouse, Juno, Kilo, Liberty, Mikata, Newton, Ocata, Pike, Queens, Rocky, Stein, Train). Chaque version est maintenue pendant 18 mois.

La suite logicielle est constituée de services. Les administrateurs interagissent avec ces services au travers d'API réseau. Dans le cas des administrateurs des commanditaires, ces échanges peuvent transiter par un réseau public. Le nombre et le périmètre de ces services évoluent entre chaque version. Certains services ont néanmoins atteint le niveau de maturité et de fiabilité requis pour être considérés comme les briques essentielles d'une infrastructure OpenStack [33]. Ces briques essentielles permettent la réalisation d'un service d'informatique nuagique de type *Infrastructure as a Service* (IaaS) :

- **Nova** : Ce service pilote les hyperviseurs qui exécutent les machines virtuelles. Il s'appuie sur les principaux hyperviseurs du marché (KVM, VMWare ESX, Xen, Hyper-V, ...) ;
- **Neutron** : Ce service pilote les réseaux définis par logiciel associés à l'infrastructure virtualisé. Il s'appuie sur les principales solutions du marché (OpenVSwitch, VMWare, LinuxBridge, CISCO, ...) ;
- **Swift** : Ce service est en charge du stockage objet. Il assure un stockage redondant et évolutif via une grappe de serveurs ;
- **Cinder** : Ce service est en charge du stockage par blocs. Il permet aux machines virtuelles de disposer de volumes disques. Il s'appuie sur un ensemble de solutions de stockage (Ceph, NetApp, LVM,...) ;
- **Glance** : Ce service est en charge du stockage et de la mise à disposition des images utilisées par les machines virtuelles. Il s'appuie sur un ensemble de solutions de stockage (Swift, Cinder, NFS, Vsphere, ...) ;
- **Keystone** : Ce service est en charge de la gestion des identités et des rôles. Il s'appuie sur un ensemble de modules (base de données, annuaire LDAP, ...) ;
- **Horizon** : Ce service offre une interface utilisateur accessible via un client léger. Elle permet d'interagir avec les différents services au travers d'un environnement graphique ;
- **Barbican** : Ce service est en charge de la gestion sécurisée des secrets et du contrôle d'accès à ceux-ci. Les secrets sont stockés chiffrés. Leur chiffrement peut être assuré via un module de cryptographie matétielle ou de façon logicielle.

Les liens de dépendance entre ces différents services sont représentés sur la figure 1 :

1.5 Risques pris en compte dans ce document

Ce document ne traite que certains risques spécifiques à l'utilisation de l'informatique nuagique. Ces risques sont ceux liés aux interfaces d'administration de l'infrastructure nuagique, l'hébergement et l'exploitation de données potentiellement sensibles via une infrastructure mutualisée.

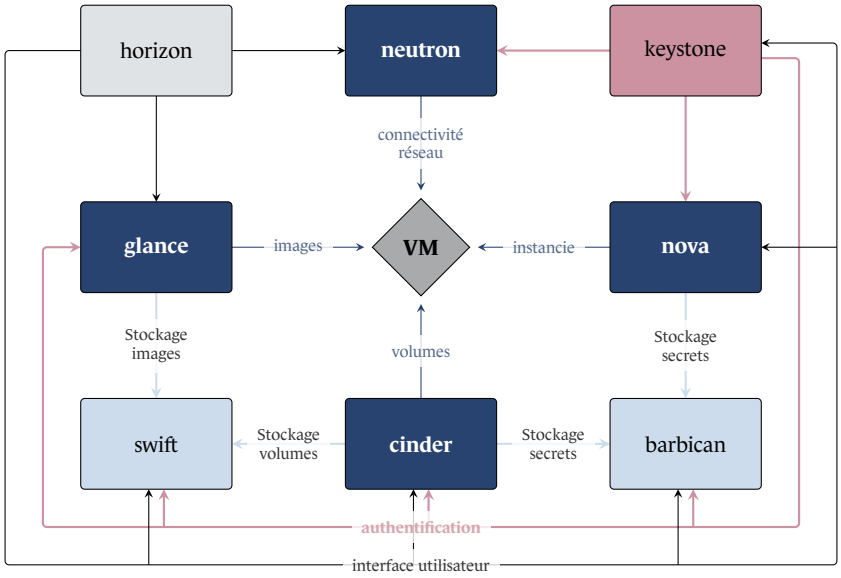

FIGURE 1 – Architecture conceptuelle des principaux services OPENSTACK

1.5.1 Risques liés aux APIs d'administration

1.5.1.1 Les types d'administrateurs

Les administrateurs d'une infrastructure peuvent être classés selon leur domaine de responsabilité :

- Les administrateurs du prestataire sont en charge de la mise en place des ressources physiques associées à l'infrastructure virtualisée, du MCO/MCS des ressources physiques et logiciels utilisées par l'infrastructure virtualisée, de la création et de la suppression des projets et domaines utilisés par les commanditaires. Ils interagissent avec OPENSTACK au moyen des APIs **admin**.
- Les administrateurs d'un commanditaire sont en charge de la gestion des ressources virtualisées associées aux projets et domaines affecté à ce commanditaire. Pour un domaine, ils sont aussi chargés d'affecter les utilisateurs du commanditaire aux différents groupes d'utilisateurs disponibles. Ils interagissent avec OPENSTACK au moyen des APIs **public**.
- Les administrateurs en charge du MCO/MCS des machines virtuelles :
  - Pour les machines virtuelles permanentes, les opérations d'administration sont effectuées via un shell (console, ssh, ...). Les droits des administrateurs sont gérés par le système d'exploitation des machines virtuelles et non par OPENSTACK. C'est pourquoi ce type d'opérations d'administration n'est pas traité dans le présent document.
  - Pour les machines virtuelles éphémères, les opérations d'administration sont effectuées via la mise à jour des images. La chaîne de création des images n'étant pas prise en charge par OPENSTACK, ce type d'opérations d'administration n'est pas traité dans le présent document.

Pour les utilisateurs des machines virtuelles ou des services applicatifs qu'elles hébergent, la gestion des droits d'accès est à la charge du système d'exploitation de la machine virtuelle ou d'un contrôle d'accès applicatif, mais n'est pas de la responsabilité d'OPENSTACK. C'est pourquoi la gestion des utilisateurs des machines virtuelles ou des services applicatifs qu'elles hébergent n'est pas traité dans le présent document.

Les différents modes d'interaction entre les utilisateurs et une infrastructure OpenStack sont illustrés dans la figure 2.

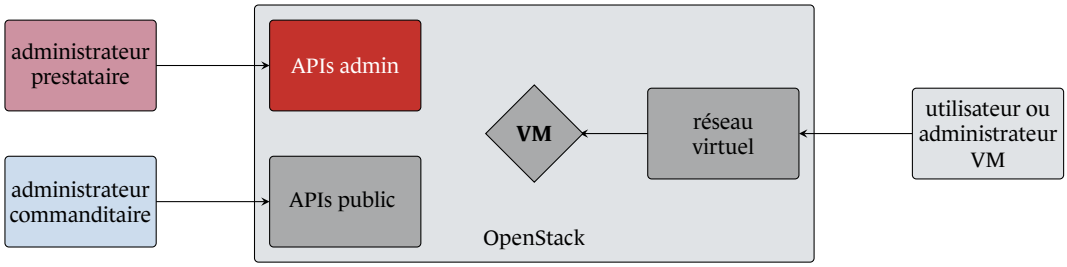

Figure 2 – Interactions entre les différents types d'utilisateurs et l'infrastructure OpenStack

1.5.1.2 Risques pesants sur les interfaces d'administration

Une utilisation incontrôlée des interfaces dédiées aux administrateur du prestataire est susceptible de fortement impacter les ressources physiques (noeuds de calcul, espace disque, ...) utilisées par les infrastructures virtualisées des commanditaires, et peut entraîner des interruptions de services ou des pertes de données. Ces APIs étant particulièrement sensibles, elles ne doivent pas être exposées sur des réseaux publics mais être accessibles uniquement depuis le système d'information du prestataire.

Une utilisation incontrôlée des interfaces dédiées aux administrateur d'un commanditaire est susceptible de fortement impacter l'infrastructure virtuelle (machines virtuelles, volumes, images, secrets, ...) de ce dernier, et peut entraîner des interruptions de services, ainsi que des pertes, altérations ou des vols de données.

A l'exception de certains services d'informatique nuagique privés mis en place directement par le commanditaire, l'accès aux interfaces d'administration du commanditaire se fait au travers d'un réseau public. Ce type d'accès expose au risque d'un acteur malveillant qui s'introduit dans un projet ou un domaine via ces interfaces.

1.5.2 Risques liés à l'utilisation d'une infrastructure mutualisée

L'utilisation d'une infrastructure mutualisée présente un certain nombre de risques vis-à-vis des biens essentiels manipulés par cette infrastructure.

Les biens essentiels externalisés sont susceptibles d'être atteints par un utilisateur illégitime, en cas de défaut d'isolation entre les données de différents commanditaires. Le vol ou la modification de ces biens essentiels est susceptible d'impacter l'activité ou la réputation du commanditaire. Dans le cas d'un service d'informatique nuagique, la mise en place du chiffrement au niveau des machines virtuelles, pour protéger ces biens essentiels, n'est que partiellement efficace. En effet, pour interagir avec ces biens essentiels, l'infrastructure virtuelle doit être à même d'accéder aux clés cryptographiques utilisées pour les chiffrer. Un accès illégitime à ces clés cryptographiques permet à un utilisateur malveillant d'accéder aux biens essentiels. Afin de réduire ce risque, l'infrastructure nuagique doit implémenter des mesures de sécurité telles que le stockage chiffré des

clés, un contrôle d'accès approprié et une traçabilité des accès, ainsi qu'un cloisonnement approprié.

Ces garanties doivent inclure la capacité par le commanditaire de transférer ses biens essentiels et de pouvoir les effacer de manière sécurisée lorsqu'il ne souhaite plus que ces derniers soient externalisés chez le prestataire ou pour les transférer à un autre prestataire.

Dans le cas d'une infrastructure nuagique publique, les ressources physiques sous-jacentes sont mutualisées pour exécuter les infrastructures virtuelles de commanditaires juridiquement distinctes.

Une faille de sécurité impactant l'infrastructure virtuelle d'un commanditaire ne doit pas être en mesure de remettre en cause la sécurité de l'infrastructure virtuelle d'un autre commanditaire.

1.6 Le référentiel SecNumCloud

Le référentiel **SecNumCloud** [9] s'adresse aux prestataires proposant des services d'informatique nuagique. Il leur permet de proposer des prestations pour lesquelles la problématique de sécurité n'est pas négociée individuellement contrat par contrat.

La qualification **SecNumCloud**, supervisée par l'ANSSI permet de garantir la prise en compte de l'ensemble des exigences du référentiel par le prestataire. Elle permet donc aux commanditaires d'avoir des garanties vis-à-vis de certains points relatifs à la sécurité, lors de l'établissement du contrat de prestation.

Le référentiel porte sur les ensembles d'exigences suivants :

- Politiques de sécurité de l'information et gestion du risque ;
- Organisation de la sécurité de l'information ;
- Sécurité des ressources humaines ;
- Gestion des actifs ;
- Contrôle d'accès et gestion des identités ;
- Cryptologie ;
- Sécurité physique et environnementale ;
- Sécurité liée à l'exploitation ;
- Sécurité des communications ;
- Acquisition, développement et maintenance des systèmes d'information ;
- Relations avec les tiers ;
- Gestion des incidents liés à la sécurité de l'information ;
- Continuité d'activité ;

- Conformité ;
- Exigences supplémentaires.

Certaines de ces exigences concernent des fonctionnalités assurées par la suite logicielle OpenStack et la configuration de cette dernière devra en garantir le respect. Ces exigences sont détaillées dans la table 1.

Table 1 – Exigences SecNumCloud devant être mises en œuvre par OpenStack

Réf.	**Intitulé de l'exigence**
9.3b	Le prestataire doit mettre à la disposition de ses commanditaires les outils et les moyens qui permettent une différenciation des rôles des utilisateurs du service, par exemple suivant leur rôle fonctionnel.
9.3d	Le prestataire doit être en mesure de fournir, pour une ressource donnée mettant en œuvre le service, la liste de tous les utilisateurs y ayant accès, qu'ils soient sous la responsabilité du prestataire ou du commanditaire ainsi que les droits d'accès qui leurs ont été attribués.
9.3e	Le prestataire doit être en mesure de fournir, pour un utilisateur donné, qu'ils soient sous la responsabilité du prestataire ou du commanditaire, la liste de tous ses droits d'accès sur les différents éléments du système d'information du service.
9.3g	Le prestataire doit inclure dans la procédure de gestion des droits d'accès les actions de révocation ou de suspension des droits de tout utilisateur.
9.4b	Le prestataire doit mettre à disposition du commanditaire un outil facilitant la revue des droits d'accès des utilisateurs placés sous la responsabilité de ce dernier.
9.5a	Le prestataire doit formaliser et mettre en oeuvre des procédures de gestion de l'authentification des utilisateurs. En accord avec les exigences du chapitre 10, celles-ci doivent notamment porter sur : - la gestion des moyens d'authentification (émission et réinitialisation de mot de passe, mise à jour des CRL et import des certificats racines en cas d'utilisation de certificats, etc.). - la mise en place des moyens permettant une authentification à multiples facteurs afin de répondre aux différents cas d'usage du référentiel. - les systèmes qui génèrent des mots de passe ou vérifient leur robustesse, lorsqu'une authentification par mot de passe est utilisée. Ils doivent suivre les recommandations de [G_AUTH] [8].
9.5b	Tout mécanisme d'authentification doit prévoir le blocage d'un compte après un nombre limité de tentatives infructueuses.
9.6a	Les comptes d'administration sous la responsabilité du prestataire doivent être gérés à l'aide d'outils et d'annuaires distincts de ceux utilisés pour la gestion des comptes utilisateurs placés sous la responsabilité du commanditaire.
9.6b	Les interfaces d'administration mises à disposition des commanditaires doivent être distinctes des interfaces d'administration utilisées par le prestataire.
9.6c	Les interfaces d'administration mises à disposition des commanditaires ne doivent permettre aucune connexion avec des comptes d'administrateurs sous la responsabilité du prestataire.
9.6d	Les interfaces d'administration utilisées par le prestataire ne doivent pas être accessibles à partir d'un réseau public et ainsi ne doivent permettre aucune connexion des utilisateurs sous la responsabilité du commanditaire.

Réf.	Intitulé de l'exigence
9.6e	Si des interfaces d'administration sont mises à disposition des commanditaires avec un accès via un réseau public, les flux d'administration doivent être authentifiés et chiffrés avec des moyens en accord avec les exigences du chapitre 10.2.
9.6f	Le prestataire doit mettre en place un système d'authentification multifacteur fort pour l'accès : - aux interfaces d'administration utilisées par le prestataire ; - aux interfaces d'administration dédiées aux commanditaires.
9.6h	Dès lors qu'une interface d'administration est accessible depuis un réseau public, le processus d'authentification doit avoir lieu avant toute interaction entre l'utilisateur et l'interface en question.
9.7a	Le prestataire doit mettre en œuvre des mesures de cloisonnement appropriées entre ses commanditaires.
10.1a	Le prestataire doit définir et mettre en œuvre un mécanisme de chiffrement empêchant la récupération des données des commanditaires en cas de réallocation d'une ressource ou de récupération du support physique. Dans le cas d'un service IaaS, cet objectif pourra par exemple être atteint : - par un chiffrement du disque ou du système de fichier, lorsque le protocole d'accès en mode fichiers garantit que seuls des blocs vides peuvent être alloués (par exemple stockage de type NAS dans lequel un bloc physique n'est effectivement affecté qu'au moment de l'écriture) ; - par un chiffrement par volume dans le cas d'un accès en mode bloc (par exemple stockage de type SAN ou stockage local), avec au moins une clé par commanditaire ; - dans le cas d'un service PaaS ou SaaS, cet objectif pourra être atteint en utilisant un chiffrement applicatif dans le périmètre du prestataire, avec au moins une clé par commanditaire.
10.1b	Le prestataire doit utiliser une méthode de chiffrement des données respectant les règles de [CRYPTO_B1] [7].
10.1c	Il est recommandé d'utiliser une méthode de chiffrement des données respectant les recommandations de [CRYPTO_B1] [7].
10.2a	Lorsque le prestataire met en œuvre un mécanisme de chiffrement des flux réseau, celui-ci doit respecter les règles de [CRYPTO_B1] [7].
10.2b	Lorsque le prestataire met en œuvre un mécanisme de chiffrement des flux réseau, il est recommandé que celui-ci respecte les recommandations de [CRYPTO_B1] [7].
10.2c	Si le protocole TLS est mis en œuvre, le prestataire doit appliquer les recommandations de [NT_TLS] [4].
10.2d	Si le protocole IPsec est mis en œuvre, le prestataire doit appliquer les recommandations de [NT_IPSEC] [3].
10.2e	Si le protocole SSH est mis en œuvre, le prestataire doit appliquer les recommandations de [NT_SSH] [1].
10.3a	Le prestataire ne doit stocker que l'empreinte des mots de passe des utilisateurs et des comptes techniques.
10.3b	Le prestataire doit mettre en œuvre une fonction de hachage respectant les règles de [CRYPTO_B1] [7].
10.3c	Il est recommandé que le prestataire mette en œuvre une fonction de hachage respectant les recommandations de [CRYPTO_B1] [7].
10.3d	Le prestataire doit générer les empreintes des mots de passe avec une fonction de hachage associée à l'utilisation d'un sel cryptographique respectant les règles de [CRYPTO_B1] [7].

Réf.	Intitulé de l'exigence
10.5a	Le prestataire doit mettre en œuvre des clés cryptographiques respectant les règles de [CRYPTO_B2] [6].
10.5c	Le prestataire doit protéger l'accès aux clés cryptographiques et autres secrets utilisés pour le chiffrement des données par un moyen adapté : conteneur de sécurité (logiciel ou matériel) ou support disjoint.
11.9a	Le prestataire doit documenter et mettre en œuvre des moyens permettant d'effacer de manière sécurisée par réécriture de motifs aléatoires tout support de données mis à disposition d'un commanditaire. Si l'espace de stockage est chiffré dans le cadre de l'exigence 10.1 a), l'effacement peut être réalisé par un effacement sécurisé de la clé de chiffrement.
12.6	Journalisation des évènements.
13.2a	Le prestataire doit documenter et mettre en œuvre, pour le système d'information du service, les mesures de cloisonnement (logique, physique ou par chiffrement) pour séparer les flux réseau selon : - la sensibilité des informations transmises ; - la nature des flux (production, administration, supervision, etc.) ; - le domaine d'appartenance des flux (des commanditaires – avec distinction par commanditaire ou ensemble de commanditaires, du prestataire, des tiers, etc.) ; - le domaine technique (traitement, stockage, etc.).
13.2d	Le prestataire doit mettre en place et configurer un pare-feu applicatif pour protéger les interfaces d'administration destinées à ses commanditaires et exposées sur un réseau public.
13.3	Le prestataire doit disposer une ou plusieurs sondes de détection d'incidents de sécurité sur le système d'information du service. Ces sondes doivent notamment permettre la supervision de chacune des interconnexions du système d'information du service avec des systèmes d'information tiers et des réseaux publics. Ces sondes doivent être des sources de collecte pour l'infrastructure d'analyse et de corrélation des événements (voir chapitre 12.9).
19.4a	À la fin du contrat liant le prestataire et le commanditaire, que le contrat soit arrivé à son terme ou pour toute autre cause, le prestataire doit assurer un effacement sécurisé de l'intégralité des données du commanditaire. Cet effacement peut être réalisé suivant l'une des méthodes suivantes, et ce dans un délai précisé dans la convention de service : - effacement par réécriture complète de tout support ayant hébergé ces données ; - effacement des clés utilisées pour le chiffrement des espaces de stockage du commanditaire décrit au chapitre 10.1 ; - recyclage sécurisé, dans les conditions énoncées au chapitre 11.9.
19.4b	À la fin du contrat, le prestataire doit supprimer les données techniques relatives au commanditaire (annuaire, certificats, configuration des accès, etc.).

1.7 Périmètre du document

Parmis les exigences détaillées dans la table 1, ce guide porte sur celles relatives aux APIs d'administration, à l'authentification des administrateurs, au chiffrement des données et des flux, ainsi qu'au cloisonnement entre les commanditaires.

Les modules de base d'OpenStack ne couvrant que la mise en œuvre d'un service d'informatique nuagique de type IaaS, ce document se bornera aux exigences relatives à ce type de service.

Afin de permettre aux prestataires de mettre en œuvre l'architecture la plus adaptée à leurs besoins, les seules contraintes d'architecture présentes dans ce document sont celles qui sont strictement nécessaires à la réalisation des fonctions de sécurité présentées.

Enfin, les implémentations suggérées dans le présent document s'appuient sur les fonctionnalités offertes par les versions d'OpenStack supportées à la date de rédaction de ce document. Les implémentations suggérées sont susceptibles de devenir obsolètes, que ce soit par l'ajout de nouvelles fonctionnalités plus à même de répondre aux exigences ou par le retrait des fonctionnalités utilisées dans le présent document.

2
Cloisonnement des accès et des données d'authentification entre le prestataire et ses commanditaires

2.1 Séparation des données d'authentification entre le prestataire et ses commanditaires

Le but est ici de se protéger contre le vol des données d'authentification d'un administrateur du prestataire faisant suite à une compromission des APIs utilisées par les commanditaires. Un tel vol pouvant conduire à une utilisation illégitime des interfaces dédiées aux administrateurs du prestataire dont les impacts sont décrits dans la section 1.5.1. Ce risque est couvert par l'exigence 9.6a du référentiel SecNumCloud :

SecNumCloud, Version 3.2, exigence 9.6a

Les comptes d'administration sous la responsabilité du prestataire doivent être gérés à l'aide d'outils et d'annuaires distincts de ceux utilisés pour la gestion des comptes utilisateurs sous la responsabilité du commanditaire.

Le service permettant de satisfaire cette exigence est le service **keystone**. Le schéma de déploiement le plus simple pour le service **keystone** repose sur l'utilisation d'une base de données dans laquelle les informations relatives à l'ensemble des utilisateurs de l'infrastructure sont stockées (éléments d'authentification, rôles assignés aux différents projets, ...). Même avec des instances du services **keystone** séparées entre prestataire et commanditaires, la base de données reste commune. La compromission du service par un agent malveillant ayant compromis un projet (ou du domaine) du commanditaire peut entraîner un accès illégitime à cette base et ainsi permettre le vol des éléments d'authentification des administrateurs du prestataire.

Il est possible de configurer le service **keystone** pour qu'il délègue l'authentification des utilisateurs à un annuaire LDAP [22]. La délégation de l'authentification permet de s'affranchir du stockage des éléments sensibles dans la base **keystone**. Il permet le déport de la gestion de la complexité des mots de passe et du mécanismes de verrouillage des comptes côté LDAP. L'utilisation d'un serveur LDAP pour lequel les mécanismes de sécurité sont à l'état de l'art permettra donc de renforcer les mécanismes de protection des comptes et des authentifiants qui leurs sont associés.

R1

Configurer keystone pour utiliser un annuaire LDAP

Il est recommandé de configurer le service **keystone** pour qu'il délègue l'authentification des utilisateurs à un annuaire LDAP. Cet annuaire doit être géré par un serveur LDAP pour lequel les mécanismes de sécurité sont à l'état de l'art.

En étendant le recours aux annuaires LDAP, il est possible de faire appel à un annuaire destiné à la gestion des administrateurs du prestataire distinct de celui ou ceux utilisé(s) par le ou les commanditaires. Une telle séparation des annuaires présente les avantages suivants :

- La compromission d'un annuaire commanditaire ne permet pas d'avoir accès aux authentifiants des administrateurs du prestataire ;
- Une part importante de la gestion des utilisateurs du ou des commanditaire(s) est effectuée via les outils LDAP, par les commanditaires eux-même. Cela permet de limiter les droits accordés aux rôles dédiés aux administrateurs des commanditaires et les failles de sécurité potentiellement associées à ces droits étendus.

Depuis la version *Grizzly* (publiée en 2013), il est possible de décomposer un déploiement OpenStack en domaines. Les domaines permettent de regrouper une hiérarchie de projets, ainsi qu'un ensemble d'utilisateurs. C'est cette fonctionnalité qui va être exploitée pour la séparation des annuaires.

Les domaines sont gérés par le service **keystone**. Il est en particulier possible d'avoir des paramètres de configuration spécifiques pour chaque domaine. Cela permet d'utiliser des annuaires LDAP distincts pour le domaine racine (*Default*) utilisé par les administrateurs du prestataire, et les domaines dédiés aux différents commanditaires [20]. Ce modèle de déploiement est représenté par la figure 3.

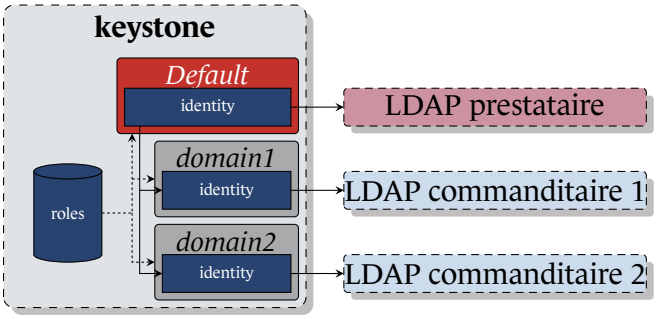

Figure 3 – Séparation des annuaires

R2

Utiliser plusieurs domaines et des annuaires LDAP dédiés par domaine

Il est recommandé de décomposer le déploiement OpenStack en domaines. Le domaine racine *Default* utilisé par les administrateurs du prestataire sera alors configuré pour utiliser un annuaire LDAP dédié. Le ou les autres domaines seront configurés pour utiliser un ou plusieurs annuaires dédiés aux commanditaires.

Le mécanisme de contrôle des droits d'accès d'OpenStack est détaillé dans l'annexe A.1. Il repose sur l'utilisation de rôles [32]. Le module **oslo.policy** permet d'affecter les droits d'accès aux rôles service par service (fichiers policy). Le service **keystone** permet d'associer des rôles à un utilisateur. L'utilisateur dispose alors de droits d'accès pour utiliser les différents services. Les rôles étant gérés de façon transverse aux domaines, un domaine ajouté à l'infrastructure OpenStack hérite donc des rôles définis.

En complément de la gestion des utilisateurs, les annuaires LDAP permettent la gestion de groupes d'utilisateurs. Ces groupes d'utilisateurs sont eux aussi gérés par le service **keystone**. Des rôles peuvent être associés à un groupe d'utilisateurs. L'ensemble des membres du groupe héritent alors des droits d'accès affectés aux rôles associés au groupe [31]. La gestion des rôles et des groupes d'utilisateurs est illustrée par la figure 4.

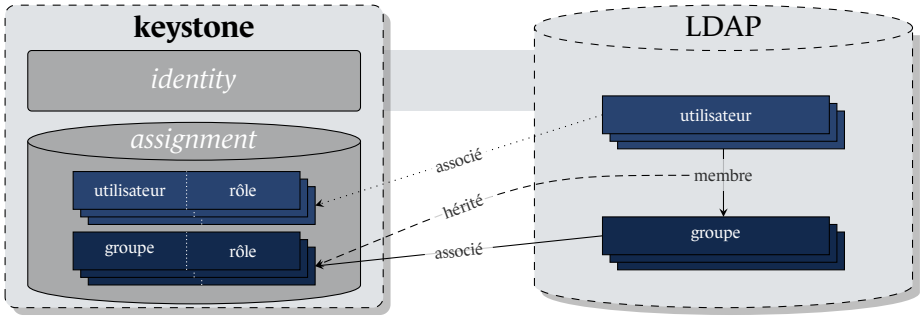

Figure 4 – Gestion des rôles et des groupes d'utilisateurs

Les exigences SecNumCloud relatives à la robustesse des mots de passe (9.5a), au bloquage des comptes (9.5b) et au stockage des mots de passe (10.3a, 10.3b, 10.3c et 10.3d) devront être mises en œuvre par les annuaires LDAP utilisés par **keystone** :

SecNumCloud, Version 3.2, exigence 9.5a

Le prestataire doit formaliser et mettre en œuvre des procédures de gestion de l'authentification des utilisateurs. En accord avec les exigences du chapitre 10, celles-ci doivent notamment porter sur :

- *la gestion des moyens d'authentification (émission et réinitialisation de mot de passe, mise à jour des CRL et import des certificats racines en cas d'utilisation de certificats, etc.).*
- *la mise en place des moyens permettant une authentification à multiples facteurs afin de répondre aux différents cas d'usage du référentiel.*
- *les systèmes qui génèrent des mots de passe ou vérifient leur robustesse, lorsqu'une authentification par mot de passe est utilisée. Ils doivent suivre les recommandations de [G_AUTH] [8].*

SecNumCloud, Version 3.2, exigence 9.5b

Tout mécanisme d'authentification doit prévoir le blocage d'un compte après un nombre limité de tentatives infructueuses.

SecNumCloud, Version 3.2, exigence 10.3a

Le prestataire ne doit stocker que l'empreinte des mots de passe des utilisateurs et des comptes techniques.

SecNumCloud, Version 3.2, exigence 10.3b

Le prestataire doit mettre en œuvre une fonction de hachage respectant les règles de [CRYPTO_B1] [7].

SecNumCloud, Version 3.2, exigence 10.3c

Il est recommandé que le prestataire mette en œuvre une fonction de hachage respectant les recommandations de [CRYPTO_B1] [7].

SecNumCloud, Version 3.2, exigence 10.3d

Le prestataire doit générer les empreintes des mots de passe avec une fonction de hachage associée à l'utilisation d'un sel cryptographique respectant les règles de [CRYPTO_B1] [7].

L'utilisation d'annuaires LDAP spécifiques pour chaque domaine permet d'augmenter l'isolation des données d'authentification entre les domaines définis pour l'infrastructure. Cela augmente fortement le travail devant être réalisé par un attaquant désirant avoir accès aux données d'authentification d'un domaine cible après avoir compromis un domaine initial.

Toutefois, si cet attaquant est en mesure d'écrire dans la base **keystone**, il lui sera toujours possible d'affecter des utilisateurs compromis au rôle **admin**, et ce faisant, d'effectuer certaines opérations d'administration de l'infrastructure. Ce risque résiduel devra être pris en compte lors de l'analyse de risque portant sur l'infrastructure.

2.2 Séparation des APIs d'administration entre le prestataire et ses commanditaires

Le but est ici d'éviter qu'une attaque par déni de service visant les APIs utilisées par les commanditaires n'affecte celles utilisées par les administrateurs du prestataire. Une telle situation pourrait empêcher ces derniers de répondre à un incident, entraînant ainsi des dysfonctionnements durables de l'infrastructure OpenStack. Ce risque est couvert par l'exigence 9.6b du référentiel SecNumCloud :

SecNumCloud, Version 3.2, exigence 9.6b

Les interfaces d'administration mises à disposition du commanditaire doivent être distinctes des interfaces d'administration utilisées par le prestataire.

Pour chaque service de son catalogue, OpenStack permet de définir trois points d'entrée distincts :

- **Internal** : Point d'entrée utilisé par les autres services, lorsqu'ils souhaitent utiliser les fonctionnalités du service ;
- **Admin** : Point d'entrée utilisé par les administrateurs du prestataire afin d'administrer le service ;
- **Public** : Point d'entrée utilisé par les administrateurs des commanditaires lorsqu'ils souhaitent interagir avec le service.

A l'issue d'une installation OpenStack par défaut, ces trois point d'entrée pointent vers une URL unique. Cette configuration ne permet pas de répondre à l'exigence présentée ci-dessus. Nous cherchons donc à mettre en place des URL distinctes pour éviter que les accès commanditaires et les accès prestataire ne soient effectués sur les mêmes URLs.

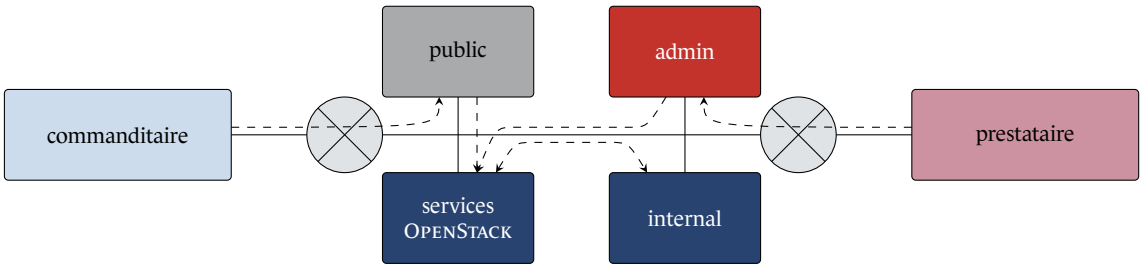

Figure 5 – Séparation des APIs d'administration

Un premier niveau de séparation des APIs consiste à mettre en place des ports TCP distincts pour les points d'entrée **admin** et **public** des services. Idéalement, il est possible de renforcer la séparation des APIs en utilisant trois ports TCP distincts. Néanmoins certaines surcouches de déploiement ne permettent pas l'utilisation de ports TCP distincts pour les points d'entrée **internal** et **admin** (kolla-ansible, **internal_vip_address** et **external_vip_address**) [45]. Ce premier niveau de séparation des APIs est illustré par la figure 5.

R3

Utiliser des ports TCP distincts pour les points d'entrée admin et public des services

Il est recommandé de mettre en place des ports TCP distincts pour les points d'entrée **admin** et **public** des services OpenStack.

L'utilisation de ports TCP distincts en écoute sur un réseau unique est susceptible de faciliter la tâche d'un attaquant cherchant à atteindre l'API **admin** à l'usage du prestataire après compromission de l'API **public** à l'usage du commanditaire. Dans le cadre de ce guide, des travaux ont été menés afin de placer les APIs **internal**, **admin** et **public** sur des réseaux distincts. La séparation des APIs **public** et **admin** est illustrée par la figure 6.

Attention

Si certaines surcouches de déploiement permettent facilement de placer les APIs **public** sur une URL distincte (kolla-ansible, **kolla_external_vip_interface**) [18], l'utilisation d'URLs distinctes pour les points d'entrée **admin** et **internal** est complexe

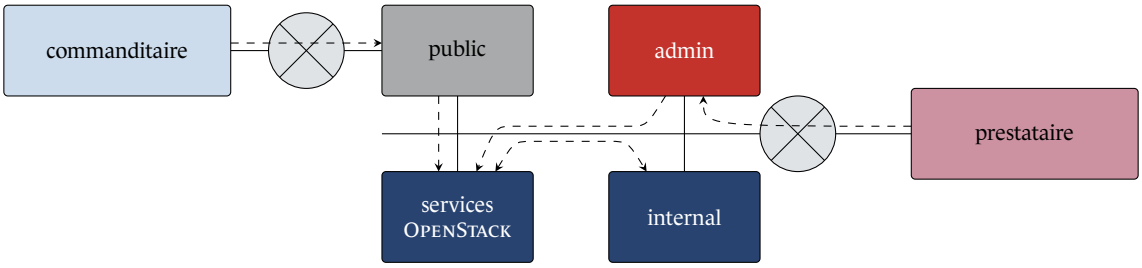

FIGURE 6 – Séparation des réseaux

à réaliser de manière expérimentale, n'est pas automatisable avec les surcouches de déploiement standard et donc irréaliste pour l'industrialisation.

R4

Placer les APIs public et admin sur des réseaux distincts

Il est recommandé de placer les APIs **public** sur un réseau distinct de celui utilisé pour les APIs **admin** et **internal**.

A ce stade, les APIs affectées à l'usage du prestataire et celles à destination du ou des commanditaires sont bien séparées. L'utilisation de réseaux séparés permet en outre d'avoir un premier niveau de séparation entre ces APIs. Néanmoins, les points d'entrée des services étant accessibles via leur adresse DNS, cette architecture oblige à rendre les APIs **public** directement accessibles aux commanditaires. Dans le cas où ils accèdent à l'infrastructure nuagique au travers d'un réseau ouvert, cela oblige donc à exposer les points d'entrée qui leur sont destinés directement sur le réseau ouvert.

La mise en place d'un serveur mandataire inverse permet de résoudre ce problème en créant une séparation entre les points d'entée à destination du ou des commanditaire(s) et le réseau ouvert. Seul le serveur mandataire inverse est exposé sur le réseau ouvert.

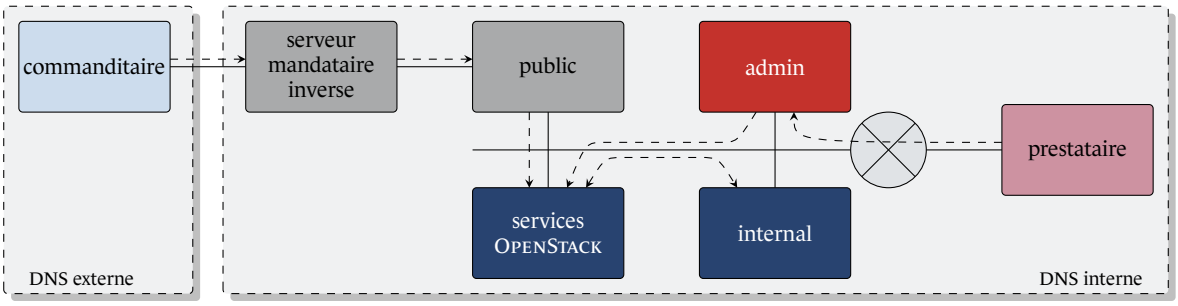

FIGURE 7 – Mise en œuvre d'un serveur mandataire inverse

R5

Mettre en place un serveur mandataire inverse en amont des APIs à destination des commanditaires

Il est recommandé de mettre en place un serveur mandataire inverse entre les points d'entrée des services OpenStack à destination des commanditaires et le réseau ouvert au travers duquel ils accèdent aux services.

C'est le service identité qui transmet la liste des services OpenStack lors de la délivrance du jeton d'authentification. Le client OpenStack utilisé par le commanditaire essaie donc de se connecter en utilisant les adresses DNS des points d'entrée placés derrière le serveur mandataire. Pour que cela fonctionne, il est impératif que du côté du commanditaire, les adresses DNS des points d'entrée **public** soient résolues en pointant vers le serveur mandataire. Alors que pour les machines de l'infrastructure, il faut qu'elles soient résolues en pointant vers les serveurs hébergeant effectivement les points d'entrée **public**. L'utilisation d'un serveur mandataire inverse pour la séparation des APIs est illustrée par la figure 7.

Bien que les points d'entrée **public** et **admin** soient placés en écoute sur des réseaux distincts, certaines surcouches de déploiement associent les différents points d'entrée de chaque service à un processus unique. Dans un tel cas de figure, une attaque par déni de service affectant le processus est toujours possible. Il sera alors impératif de mettre en place un filtrage applicatif en amont des APIs, conformément à l'exigence du 13.2d du référentiel SecNumCloud, ainsi qu'une supervision de l'activité des serveurs hébergeant ces processus conformément à l'exigence 12.6 :

SecNumCloud, Version 3.2, exigence 13.2d

Le prestataire doit mettre en place et configurer un pare-feu applicatif pour protéger les interfaces d'administration destinées à ses commanditaires et exposées sur un réseau public.

SecNumCloud, Version 3.2, exigence 12.6

a) Le prestataire doit documenter et mettre en œuvre une politique de journalisation incluant au minimum les éléments suivants :

- *la liste des sources de collecte ;*
- *la liste des événements à journaliser par source ;*
- *l'objet de la journalisation par événement ;*
- *la fréquence de la collecte et base de temps utilisée ;*
- *la durée de rétention locale et centralisée ;*
- *les mesures de protection des journaux (dont chiffrement et duplication) ;*
- *la localisation des journaux.*

b) Le prestataire doit générer et collecter les événements suivants :

- *les activités des utilisateurs liées à la sécurité de l'information ;*
- *la modification des droits d'accès dans le périmètre de sa responsabilité ;*
- *les événements issus des mécanismes de lutte contre les codes malveillants (voir 12.4) ;*
- *les exceptions ;*
- *les défaillances ;*
- *tout autre événement lié à la sécurité de l'information.*

c) *Le prestataire doit conserver les événements issus de la journalisation pendant une durée minimale de six mois sous réserve du respect des exigences légales et réglementaires.*

d) *Le prestataire doit fournir, sur demande d'un commanditaire, l'ensemble des événements le concernant.*

e) *Il est recommandé que le système de journalisation mis en place par le prestataire respecte les recommandations de [NT_JOURNAL] [5].*

R6

Mettre en place un filtrage applicatif en amont des APIs

Il est recommandé de mettre en place un filtrage applicatif en amont des APIs, conformément à l'exigence 13.2d du référentiel SecNumCloud.

2.3 Isolation des API d'administration

Le but est ici de se protéger d'un acteur malveillant disposant des authentifiants d'un administrateur du prestataire et cherchant à se connecter à l'infrastructure OpenStack en utilisant les APIs dédiées aux commanditaires accessibles depuis un réseau public. Les impacts d'une telle utilisation des APIs **public** sont décrits à la section 1.5.1. Ce risque est couvert par l'exigence 9.6c du référentiel SecNumCloud :

SecNumCloud, Version 3.2, exigence 9.6c

Les interfaces d'administration mises à disposition du commanditaire ne doivent permettre aucune connexion avec des comptes d'administrateurs sous la responsabilité du prestataire.

Les APIs OpenStack mettent en œuvre un contrôle d'accès basé sur des rôles. Pour chaque service, un fichier **policy** défini les actions autorisées pour un ensemble de rôles. Chaque utilisateur souhaitant interagir avec un service doit, au préalable, s'être authentifié pour un périmètre donné (projet ou domaine) et un rôle donné. Pour chaque interaction, le service vérifie alors que l'action est autorisée en fonction des règles présentes dans le fichier **policy** pour le rôle associé à l'utilisateur.

Le rôle **admin** est un rôle attribué à certains administrateurs du prestataire. Ce rôle dispose de privilèges étendus. Parmi ces privilèges étendus, il faut noter que pour la majorité des interfaces OpenStack, le rôle **admin** permet d'effectuer sans restrictions l'ensemble des actions possibles pour le service.

Pour simplifier les interactions entre les APIs OpenStack et l'utilisateur, ce dernier se voit attribuer un jeton d'authentification (*token*) valide quelques heures.

Les jetons sont produits et signés par le service **identité**. La signature est effectuée au moyen

d'une clé qui est partagée entre les instances du service **identité** [30]. Pour interagir avec un service OpenStack, l'utilisateur fourni ce jeton au service en question qui va alors vérifier sa validité auprès du service **identité**.

Si un administrateur du prestataire dispose d'un jeton valide avec le rôle **admin**, il sera en mesure de se connecter via l'API à destination du ou des commanditaire(s) et d'interagir avec le système. Ceci constitue une non conformité à l'exigence SecNumCloud 9.6c.

La première approche envisagée pour empêcher ce type d'interactions est la mise en place de clés distinctes pour les points d'entrée **public** et **admin** de l'API identité. Le fonctionnement de ce mécanisme de protection a été validé expérimentalement, mais il s'avère très compliqué à industrialiser, surtout dans les cas où les APIs **internal** et **admin** sont communes.

Une autre approche possible est de faire évoluer le serveur mandataire inverse pour qu'il bloque les éventuels jetons **admin**, utilisés par des acteurs malveillants, sur les APIs à destination du ou des commanditaire(s). Pour cela, il faut lui ajouter les fonctions suivantes :

1. Sur réception d'une requête sans jeton (requête d'authentification), la faire suivre à l'API identité **public**.
2. Sur réception de la réponse de l'API identité, si un jeton est présent, stocker ce jeton dans un cache.
3. Sur réception d'une requête avec jeton, vérifier que le jeton est déjà présent dans le cache avant de faire suivre la requête à l'API destinataire.

Ce mécanisme permet de n'accepter que les jetons produits par l'API identité **public**. L'utilisation du cache de jetons est illustrée par la figure 8.

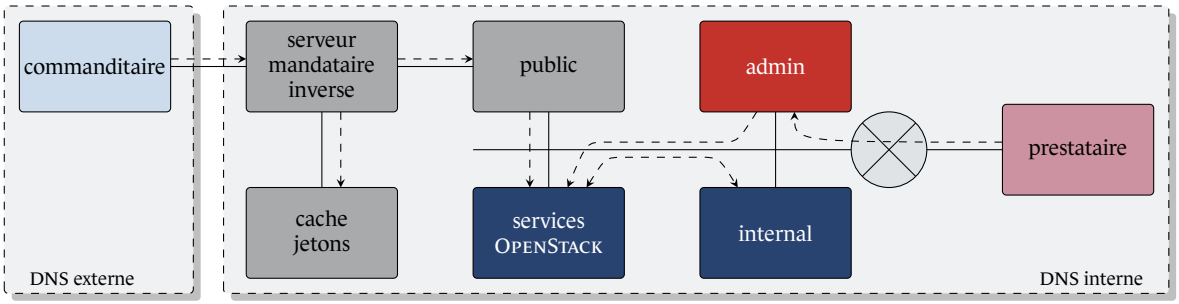

Figure 8 – Mise en œuvre d'un cache de jetons

Afin de réduire les conséquences d'un vol de jetons faisant suite à la compromission du serveur mandataire inverse et du cache. Ce ne sont pas les jetons qui sont conservés, mais leur empreinte.

R7

Mettre en place un mécanisme de de bloquage des jetons admin pour les APIs à destination des commanditaires

Il est recommandé d'implémenter un mécansime de bloquage des jetons **admin** au niveau du serveur mandataire inverse à destination des commanditaires.

Si le mécanisme décrit ci-dessus permet bien d'empêcher les tentatives de connexion des administrateurs du prestataire via les APIs destinées aux commanditaires, il est aussi impératif d'empêcher les tentatives de connexion des administrateurs des commanditaires via les APIs destinées au prestataire. Ce dernier cas de figure conduit à l'usurpation de l'identité d'un administrateur d'un commanditaire, usurpation d'identité dont les conséquences sont décrites à la section 1.5.1 et qui est couvert par l'exigence 9.6d du référentiel SecNumCloud :

SecNumCloud, Version 3.2, exigence 9.6d

Les interfaces d'administration utilisées par le prestataire ne doivent pas être accessibles à partir d'un réseau public et ainsi ne doivent permettre aucune connexion des utilisateurs sous la responsabilité du commanditaire.

R8

Mettre en place un mécanisme de de bloquage des jetons public pour les APIs à destination du prestataire

Il est recommandé d'implémenter un mécanisme de blocage des jetons **public** en amont des APIs à destination des administrateurs du prestataire.

L'isolation des APIs d'administration n'est toutefois que partiellement atteinte à ce point. En effet, aucune corrélation n'est faite entre l'identité de l'administrateur effectuant une requête et le jeton associé à cet administrateur. Si un acteur malveillant compromet le serveur mandataire, il lui est alors possible d'introduire artificiellemet des jetons arbitraires dans le cache et donc de contourner ce mécanisme de protection. La mise en place d'une corrélation entre les identités OpenStack et les jetons sera étudiée à la section 3.2.

3
Possibilité d'authentification multi-facteurs

Le but est ici de se protéger contre l'utilisation frauduleuse des données d'authentification d'un administrateur qui lui auraient été préalablement dérobées. Les impacts d'une telle utilisation de données d'authentification sont décrits dans la section 1.5.1. Ce risque est couvert par les exigences 9.5a et 9.6f du référentiel SecNumCloud :

SecNumCloud, Version 3.2, exigence 9.5a

Le prestataire doit formaliser et mettre en œuvre des procédures de gestion de l'authentification des utilisateurs. En accord avec les exigences du chapitre 10, celles-ci doivent notamment porter sur :

- *la gestion des moyens d'authentification (émission et réinitialisation de mot de passe, mise à jour des CRL et import des certificats racines en cas d'utilisation de certificats, etc.).*
- *la mise en place des moyens permettant une authentification à multiples facteurs afin de répondre aux différents cas d'usage du référentiel.*
- *les systèmes qui génèrent des mots de passe ou vérifient leur robustesse, lorsqu'une authentification par mot de passe est utilisée. Ils doivent suivre les recommandations de [G_AUTH] [8].*

SecNumCloud, Version 3.2, exigence 9.6f

Le prestataire doit mettre en place un système d'authentification à double facteur pour l'accès :

- *aux interfaces d'administration utilisées par le prestataire ;*
- *aux interfaces d'administration dédiées aux commanditaires.*

Les administrateurs peuvent interagir avec une infrastructure OpenStack de deux manières : soit au moyen des API d'administration, soit via le tableau de bord **horizon**. Le mécanisme d'authentification multi-facteurs retenu doit supporter ces deux types d'interfaces.

Enfin, de nombreux administrateurs font appel à des automates pour la gestion de leur infrastructure virtualisée, ce mécanisme devra aussi permettre à des automates de se connecter aux APIs.

3.1 Authentification multi-facteurs par fédération d'identité

Keystone en tant que fournisseur d'identité ne supporte pas l'authentification multi-facteurs. Il supporte néanmoins la fédération d'identité et peut ainsi déléguer l'authentification des utilisateurs à un fournisseur d'identité externe supportant l'authentification multi-facteurs [29]. Pour la fédération d'identité, **keystone** supporte les protocoles SAML 2.0 et OIDC [21].

3.1.1 Fédération d'identité reposant sur SAML 2.0

Avec une fédération d'identité reposant sur SAML 2.0, le synoptique illustré par la figure 9 est exécuté lors de l'authentification du client :

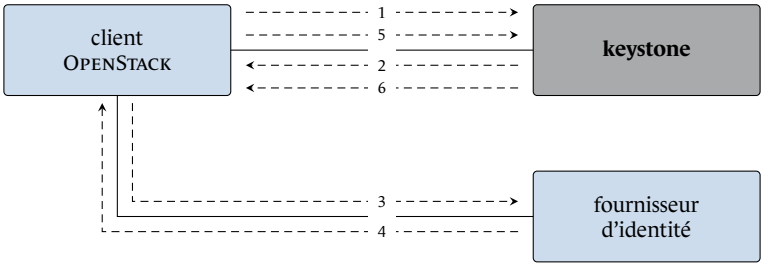

FIGURE 9 – Authentification multi-facteurs via SAML 2.0

1. Le client émet une requête d'authentification auprès du service **keystone**.
2. Le service **keystone** répond avec une demande de redirection incluant l'adresse du fournisseur d'identité externe.
3. Le client émet une requête d'authentification auprès du fournisseur d'identité externe.
4. Après authentification, le fournisseur d'identité externe répond avec une assertion SAML 2.0.
5. Le client renvoie cette assertion au service **keystone**.
6. Le service **keystone** valide l'assertion afin d'authentifier le client.

Keystone est compatible avec les profiles SAML 2.0 **WebSSO** et **ESP**. Le profil **WebSSO** est supporté par les navigateurs Web mais n'est pas supporté par le client OpenStack en ligne de commande. En l'absence de développements spécifiques côté client OpenStack, ce profil n'est utilisable qu'avec l'interface graphique **horizon**. Le profil **ESP** est quant à lui supporté par le client OpenStack en ligne de commande, mais n'est compatible qu'avec un nombre réduit de fournisseurs d'identité.

3.1.2 Fédération d'identité reposant sur OIDC

Avec une fédération d'identité reposant sur OIDC, le synoptique illustré par la figure 10 est exécuté lors de l'authentification du client :

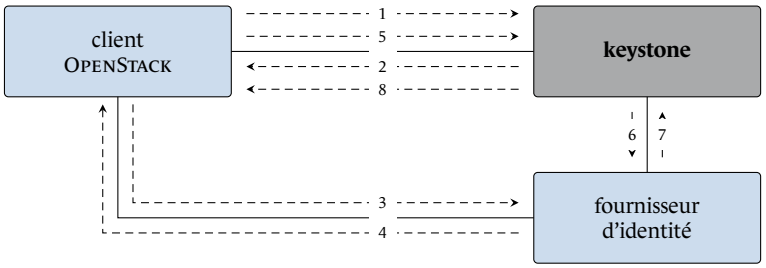

FIGURE 10 – Authentification multi-facteurs via OpenIDC

1. Le client émet une requête d'authentification auprès du service **keystone**.
2. Le service **keystone** répond avec une demande de redirection incluant l'adresse du fournisseur d'identité externe.
3. Le client émet une requête d'authentification auprès du fournisseur d'identité externe.
4. Après authentification, le fournisseur d'identité externe répond avec un code OIDC.
5. Le client renvoie ce code au service **keystone**.
6. Le service **keystone** utilise se code pour interroger le fournisseur d'identité externe.
7. Le fournisseur d'identité externe valide le code et retourne des informations relatives à l'utilisateur.
8. Le service **keystone** valide la connexion.

Comme pour la fédération d'identité reposant sur SAML 2.0, la fédération d'identité reposant sur OIDC est directement utilisable avec l'interface graphique **horizon**. Mais elle n'est pas directement utilisable avec le client OpenStack en ligne de commande.

3.2 Utilisation de l'authentification par certificat X.509 comme second facteur

En l'absence d'un fournisseur d'identité répondant aux besoins, nous souhaitons évaluer la possibilité de mettre en place un mécanisme d'authentification multi-facteurs en amont des interfaces OpenStack. Le serveur mandataire inverse étant placé en amont des interfaces OpenStack, nous allons maintenant étudier comment adapter son fonctionnement pour réaliser une authentification multi-facteurs.

En l'état, le serveur mandataire inverse reçoit deux types de requêtes avec des éléments d'authentification distincts :

- Les requêtes d'authentification à destination de l'interface de **keystone**. Ces requêtes comprennent l'identifiant de l'utilisateur, ainsi que son mot de passe ;
- Les requêtes authentifiées à destination des interfaces OpenStack. Ces requêtes sont authentifiées par la présence d'un jeton valide dans leur en-tête HTTP.

Si l'établissement de la connexion entre le client OpenStack et le serveur mandataire inverse se fait avec une authentification mutuelle par certificats X.509, il est possible de se baser sur les données du certificat client pour disposer d'un nouveau facteur d'authentification :

- Le serveur mandataire utilise les mécanismes IGC classiques pour vérifier que le certificat reçu est en cours de validité et qu'il a été émis par une chaîne de certification valide ;
- Il extrait le champ DN du certificat et l'identifiant de l'utilisateur présent dans la requête puis utilise ces deux éléments pour interroger le serveur LDAP ;
- Le serveur LDAP stockant le champ DN du certificat associé à chaque utilisateur, le serveur mandataire est alors en mesure de s'assurer que le certificat reçu est bien celui associé à l'utilisateur OpenStack à l'origine de la requête.

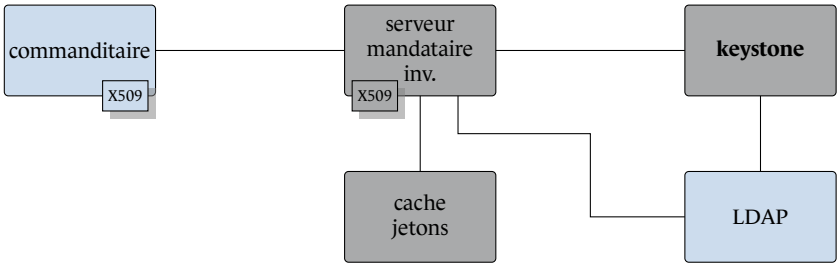

Figure 11 – Authentification multi-facteurs via le serveur mandataire inverse

R9

Mettre en place une authentification mutuelle entre le client OpenStack et le serveur mandataire

Il est recommandé de mettre en place une authentification mutuelle basée sur des certificats X.509 entre le client OpenStack et le serveur mandataire inverse placé en amont des APIs **public**.

R10

Mettre en place un contrôle de cohérence entre le certificat X.509 et l'identité utilisateur

Il est recommandé de mettre en place un contrôle de cohérence entre le champ **Distinguished Name** des certificats X.509 reçus et l'identité utilisateur fournie à l'annuaire LDAP.

L'identifiant utilisateur est présent chiffré dans le jeton et le serveur mandataire n'a pas accès à la clé de chiffrement. En modifiant le mécanisme de mémorisation des jetons pour ne plus stocker simplement l'empreinte des jetons mais stocker des paires constituée des empreintes de jetons et des identifiants utilisateurs, il devient possible de faire la vérification pour les deux types de requêtes. Ce fonctionnement est illustré par la figure 12.

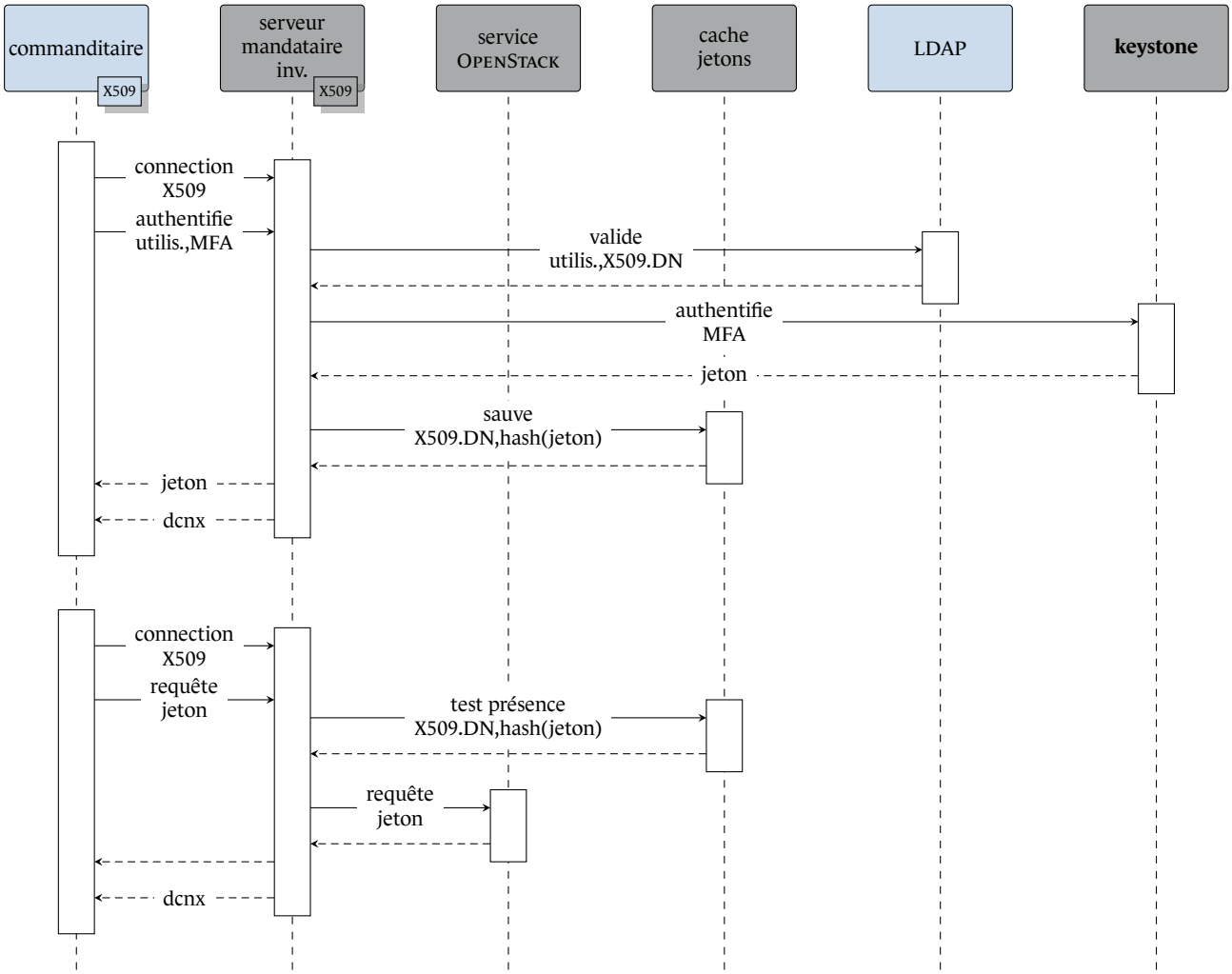

FIGURE 12 – Diagramme de séquence de l'authentification multi-facteurs

R11

Mémoriser les paires constituées du champ DN des certificats X.509 et des jetons émis par le service keystone

Il est recommandé de mémoriser des paires constituées du champ **Distinguished Name** des certificats X.509 reçus et de l'empreinte des jetons retournés par l'API **keystone**.

R12

Vérifier les associations de jetons envoyés par les clients OpenStack et des certificats X.509 utilisés

Il est recommandé, pour tout jeton reçu, de vérifier la cohérence entre son empreinte et le champ **Distinguished Name** du certificat X.509 reçu en se basant sur les paires mémorisées précédemment.

Le serveur mandataire inverse ayant besoin d'accéder aux données applicatives pour effectuer l'authentification des utilisateurs, il doit impérativement déchiffrer le flux TLS reçu du client OpenStack, puis le chiffrer à nouveau pour dialoguer avec l'API destinataire.

Bien que le mécanisme présenté ci-dessus permette de mettre en œuvre une authentification multi-facteurs, il n'offre pas une protection efficace contre le vol des données d'authentification côté client lorsque celles-ci sont utilisées par un automate qui se connecte aux APIs OpenStack. En effet, afin de se connecter à l'infrastructure OpenStack de manière autonome, l'automate doit disposer de l'ensemble des facteurs d'authentification.

En complément des mesures de contrôle d'accès aux données d'authentification qui doivent être mises en place côté client, la mise en œuvre des *applications credentials* [35] proposées par OpenStack permet de restreindre les droits accordés par OpenStack à l'automate au strict nécessaire pour son bon fonctionnement. De plus, l'utilisation des *applications credentials* permet de ne pas avoir à rendre le mot de passe de l'administrateur accessible à l'automate.

4
Protection des flux et des données

Le but est ici de nous protéger contre le vol ou la modification du logiciel ou des données mises en œuvre par les commanditaires dans leurs infrastructures virtualisées. Les impacts d'un tel vol ou d'une telle modification sont présentés dans la section 1.5.2. Ce risque est couvert par le référentiel SecNumCloud par la capacité offerte aux commanditaires de chiffrer les données stockées chez le prestataire, ainsi que des données échangées avec les machines virtuelles utilisées par le commanditaire.

4.1 Chiffrement des données stockées

Le chiffrement des données stockées chez le prestataire permet de protéger celles-ci d'un accès illégitime. Ce risque est couvert par les exigences 10.1b et 10.1c du référentiel SecNumCloud :

SecNumCloud, Version 3.2, exigence 10.1b

Le prestataire doit utiliser une méthode de chiffrement des données respectant les règles de [CRYPTO_B1] [7].

SecNumCloud, Version 3.2, exigence 10.1c

Il est recommandé d'utiliser une méthode de chiffrement des données respectant les recommandations de [CRYPTO_B1] [7].

OpenStack propose deux entités dédiées au stockage ; les **volumes** et les **images**.

Les **images** hébergent le système de fichiers qui sera utilisé par une ou plusieurs machines virtuelles de l'infrastructure. Si les machines virtuelles d'un commanditaire doivent exécuter des traitements particuliers, ce dernier a la possibilité d'importer dans l'infrastructure ces propres images incluant le code associé à ces traitements.

Lorsqu'une machine virtuelle utilise une image, les éventuelles modifications apportées au système de fichiers resteront locales à la machine et ne seront pas sauvegardées dans l'image associée. Afin de pouvoir sauvegarder des données liées à son fonctionnement, une machine virtuelle doit monter un **volume** dans lequel elle peut écrire des données qui sont alors conservées de manière persistante.

OpenStack intègre des capacités de chiffrement des volumes depuis 2015 avec la version *kilo* [38]. Le chiffrement des volumes peut être géré :

- Par volume, chaque volume est chiffré avec une clé unique ;
- Par machine virtuelle, l'ensemble des volumes de cette machine virtuelle sont chiffrés avec une clé unique ;
- Par projet ou domaine, l'ensemble des volumes du projet ou du domaine sont chiffrés avec une clé unique ;
- Par commanditaire, l'ensemble des volumes du commanditaire sont chiffrés avec une clé unique.

Le chiffrement des volumes est pris en charge par le service **nova** en s'appuyant sur les capacités de chiffrement de l'hyperviseur sous-jacent. Ce type de chiffrement est transparent pour le service de gestion des volumes virtuels **cinder** et pour le service en charge du stockage **swift**. Les données qui transitent sur le réseau ou qui sont stockées sur disques sont nativement chiffrées par le ou les nœuds de calcul qui mettent en œuvre le volume.

R13

Activer le chiffrement des volumes du service nova

Il est recommandé d'utiliser le mécanisme de chiffrement des volumes offert par le service **nova**.

Même si un mécanisme de chiffrement des images est actuellement à l'étude au niveau du service **glance** [27], les versions courantes de ce service ne permettent pas de chiffrer les images. La seule possibilité offerte pour protéger les images stockées est d'utiliser la capacité de chiffrement du service **swift** [49], ainsi que le mécanisme de contrôle d'accès **oslo.policy** commun à l'ensemble des services OpenStack.

R14

Activer le chiffrement du stockage du service swift

Il est recommandé d'utiliser le mécanisme de chiffrement du stockage offert par le service **swift**.

Dans ce cas, le chiffrement étant interne au service **swift**, les données associées aux images circulent donc en clair entre **swift** et **glance** et entre **glance** et **nova**. Afin de garantir leur confidentialité lors des échanges entre services, il est impératif de mettre en œuvre un mécanisme de chiffrement des échanges.

R15

Activer le chiffrement des échanges entre les services swift et glance

Il est recommandé de chiffrer les échanges entre les services **swift** et **glance**.

R16

Activer le chiffrement des échanges entre les services glance et nova

Il est recommandé de chiffrer les échanges entre les services **glance** et **nova**.

Bien que des mécanismes de chiffrement des données soient mis en place, les données doivent toujours pouvoir être accédées en clair par le nœud de calcul qui exécute la machine virtuelle. Un attaquant ayant accès à une autre machine virtuelle exécutée sur le même nœud de calcul et réussissant à s'évader de cette machine virtuelle aura alors potentiellement accès aux données en clair. Des mesures de cloisonnement (section 5.1) doivent donc aussi être mises en œuvre pour assurer une protection efficace des données.

Enfin, il est impératif de mettre en place des mécanismes de protection des clés de chiffrement (section 4.3) afin d'empêcher qu'un administrateur malveillant du prestataire n'accède aux données en clair en volant les données chiffrées et les clés associées.

R17

Protéger les clés de chiffrement

Il est recommandé de mettre en place des mécanismes de protection des clés de chiffrement.

4.2 Chiffrement des flux

Le chiffrement des flux échangés entre les différents composants de l'infrastructure permet de protéger leur contenu contre le vol ou la modification. Ce risque est couvert par les exigences 9.6e, 10.2a, 10.2b, 10.2c, 10.2d et 10.2e du référentiel SecNumCloud :

SecNumCloud, Version 3.2, exigence 9.6e

Si des interfaces d'administration sont mises à disposition du commanditaire avec un accès via un réseau public, les flux d'administration doivent être authentifiés et chiffrés avec des moyens en accord avec les exigences du chapitre 10.2.

SecNumCloud, Version 3.2, exigence 10.2a

Lorsque le prestataire met en œuvre un mécanisme de chiffrement des flux réseau, celui-ci doit respecter les règles de [CRYPTO_B1] [7].

SecNumCloud, Version 3.2, exigence 10.2b

Lorsque le prestataire met en œuvre un mécanisme de chiffrement des flux réseau, il est recommandé que celui-ci respecte les recommandations de [CRYPTO_B1] [7].

SecNumCloud, Version 3.2, exigence 10.2c

Si le protocole TLS est mis en œuvre, le prestataire doit appliquer les recommandations de [NT_TLS] [4].

SecNumCloud, Version 3.2, exigence 10.2d

Si le protocole IPsec est mis en œuvre, le prestataire doit appliquer les recommandations de [NT_IPSEC] [3].

SecNumCloud, Version 3.2, exigence 10.2e

Si le protocole SSH est mis en œuvre, le prestataire doit appliquer les recommandations de [NT_SSH] [1].

Les échanges entre les divers éléments constitutifs de l'infrastructure OpenStack comprennent :

- Les échanges entre les APIs OpenStack et le ou les serveurs mandataires inverses ;
- Les échanges entre les serveurs mandataires inverses et les clients ;
- Les échanges entre les services OpenStack et la file d'attente de messages ;
- Les échanges entre les services OpenStack et la base de données ;
- Les échanges entre les services OpenStack et le cache de jetons.

Les flux, autres que ceux des APIs, nécessaires à l'authentification multi-facteurs étudiée dans la section 3.2 n'étant pas natifs de la suite logicielle OpenStack, ils ne seront pas abordés ici. Ils doivent néanmoins être chiffrés conformément aux recommandations du référentiel SecNumCloud.

Les APIs OpenStack s'appuient sur des briques logicielles standard qui ne posent pas de problème pour la mise en place d'un chiffrement TLS [46]. Afin que les chiffrements TLS utilisés par l'infrastructure OpenStack soient à l'état de l'art, leur paramètrage doit s'appuyer sur les recommandations publiées par l'ANSSI [4].

R18

Configurer le chiffrement TLS à l'état de l'art pour les APIs des services OpenStack

Il est recommandé de configurer l'ensemble des APIs OpenStack pour qu'elles mettent en œuvre du chiffrement TLS à l'état de l'art.

Historiquement, certaines surcouches de déploiement ne permettaient pas la mise en œuvre de ce chiffrement TLS sur l'ensemble des APIs, mais ce n'est plus le cas avec les versions les plus récentes (kolla-ansible, **kolla_enable_tls_internal**, **kolla_internal_fqdn_cert**) [18].

Comme indiqué précédemment, le client openstack interagit avec les APIs en s'appuyant sur les adresses fournies par le catalogue des services retourné par le service identité lors de la phase d'authentification. Ces adresses comprenant une indication de présence du chiffrement TLS, le niveau de chiffrement entre le client et le serveur mandataire inverse doit être cohérent avec celui entre le serveur mandataire inverse et les points d'entrée des services OpenStack.

Dans le cas où le chiffrement TLS est présent sur l'intégralité du chemin du flux, le serveur mandataire inverse devra déchiffrer les données reçues d'un côté avant de les chiffrer à nouveau pour les transmettre de l'autre côté. En effet, celui-ci exploitant activement le contenu des messages échangés, il ne pourra pas remplir les fonctions qui lui sont assignées sans avoir accès au contenu des échanges en clair.

Lorsqu'une requête asynchrone est envoyée à un service, elle est placée dans la file d'attente des

messages (**oslo.messaging.rpc**, **oslo.messaging.notify**) par l'agent en charge de l'interface du service. Elle est ensuite récupérée par l'agent en charge des traitements lorsqu'il est en mesure de l'exécuter.

Un accès illégitime à cette file d'attente étant de nature à fortement compromettre le fonctionnement de l'infrastructure. Il est impératif de mettre en place le mécanisme de chiffrement TLS avec authentification mutuelle nativement offert par le service de messagerie afin de réduire ce risque [43].

R19

Configurer le chiffrement TLS avec authentification mutuelle pour le service de file d'attente de messages

Il est recommandé de configurer le service de file d'attente de messages pour qu'il mette en œuvre du chiffrement TLS avec authentification mutuelle.

La base de données est utilisée par les services pour stocker des données de travail. Certaines de ces données sont de nature à fortement compromettre l'intégrité de l'infrastructure (tables du module assignment du service **keystone**). Il est impératif de mettre en place le mécanisme de chiffrement TLS nativement offert par le serveur de base de données afin d'atténuer ce risque [42]. Ce chiffrement TLS doit être accompagné d'une authentification mutuelle [23].

R20

Configurer le chiffrement TLS avec authentification mutuelle pour le service base de données

Il est recommandé de configurer le service base de données pour qu'il mette en œuvre du chiffrement TLS avec authentification mutuelle.

Le cache de jetons est utilisé par le module identité du service **keystone** pour résoudre des problèmes de performance [19]. Le vol de jetons pouvant conduire à l'usurpation de l'identité d'un utilisateur ou d'un administrateur, il est impératif de chiffrer les échanges entre le cache de jetons et les services OpenStack qui l'utilisent. Le chiffrement TLS n'est supporté par le service **memcached** que depuis la version 1.5.13. Le support de TLS par la couche **oslo.cache** n'est pas finalisé à la date de rédaction de ce document. En attendant la disponibilité de cette fonctionnalité, il est recommandé de mettre en place du chiffrement par tunnel IPsec entre le service **memcached** et les services OpenStack clients.

R21

Mettre en place des tunnels IPsec entre le cache de jetons et les services OpenStack

Il est recommandé de mettre en place du chiffrement par tunnel IPsec entre le service **memcached** et les services OpenStack clients.

Comme évoqué dans la section 4.1, l'absence de mécanisme de chiffrement des images via **nova** impose la mise en place d'un chiffrement des flux entre les services par lesquels transite le contenu des images. Pour que cela fonctionne, il faut configurer **glance** pour qu'il s'appuie sur **cinder** pour

le stockage des images [37], et **cinder** pour forcer les échanges via des flux TLS (**nova_api_insecure** et **glance_api_insecure**) [36].

R22

Configurer le service glance pour qu'il s'appuis sur le service cinder

Il est recommandé de configurer le service **glance** pour qu'il s'appuie sur **cinder** pour le stockage des images.

R23

Configurer le chiffrement TLS pour le service glance

Il est recommandé de configurer le service **glance** pour mette en œuvre du chiffrement TLS.

Enfin, et de manière identique aux clés de chiffrement des données, il est impératif de mettre en place des mécanismes de protection des clés utilisées lors des phases d'authentification mutuelle associées au chiffrement TLS.

4.3 Gestion des secrets

Le but est ici de se protéger contre le vol des clés cryptographiques utilisées dans le cadre du chiffrement des données ou du chiffrement des flux. Les impacts d'un tel vol sont présentés dans la section 1.5.2. Ce risque est couvert par les exigences 10.5a et 10.5b du référentiel SecNumCloud :

SecNumCloud, Version 3.2, exigence 10.5a

Le prestataire doit mettre en œuvre des clés cryptographiques respectant les règles de [CRYPTO_B2] [6].

SecNumCloud, Version 3.2, exigence 10.5b

Il est recommandé que le prestataire mette en œuvre des clés cryptographiques respectant les recommandations de [CRYPTO_B2] [6].

Dans une infrastructure OpenStack, les différents services doivent pouvoir accéder aux secrets nécessaires à leur bon fonctionnement, sans intervention humaine [44]. L'accès à ces secrets doit de plus être restreint à leurs seuls utilisateurs légitimes, ainsi qu'aux projets auxquels ils sont associés. C'est le service de gestion des secrets **barbican** qui est chargé de répondre à ces contraintes.

Le service barbican permet de gérer les secrets suivants :

- Mots de passe ;
- Certificats X.509 ;
- Clés symétriques ;
- Clés asymétriques ;

- Données binaires.

Barbican peut être utilisé comme générateur de secrets. Néanmoins, seules les clés symétriques et asymétriques peuvent être générées.

Pour protéger les secrets, **barbican** s'appuie sur deux modules :

- Un module cryptographique en charge du chiffrement/déchiffrement des secrets. Il est possible, par simple configuration, de sélectionner un module mettant en œuvre de la cryptographie logicielle (*simple_crypto*), un module d'interface avec un serveur KMIP (*kmip_plugin*) ou un module d'interface avec un HSM via une librairie PKCS#11 (*p11_crypto_plugin*). Parmi les matériels compatibles, seul le HSM Proteccio est certifié [48], il sera donc utilisé ;
- Un module en charge du stockage des chiffrés manipulés par le service. Il est possible, par simple configuration, de sélectionner un module s'interfaçant avec un gestionnaire de base de données (*store_crypto*) ou un module d'interface avec un serveur KMIP (*kmip_plugin*). Seul le stockage en base de données étant compatible avec le module cryptographique PKCS#11, c'est ce dernier qui sera utilisé.

R24

Configurer le service barbican pour qu'il s'appuie sur un HSM)

Il est recommandé de configurer le service **barbican** pour qu'il utilise le module *p11_crypto_plugin* afin de s'interfacer avec un HSM via une librairie **PKCS#11**.

R25

Utiliser un HSM ayant fait l'objet d'une certification de sécurité

Il est recommandé d'utiliser un HSM ayant fait l'objet d'une certification de sécurité.

Lorsque le service est configuré pour s'interfacer avec un HSM, les secrets sont chiffrés au moyen de clés projet qui sont elles-mêmes protégées avec la clé maître du HSM. Les secrets chiffrés , ainsi que les clés projet protégées sont stockés dans la base de données. L'intégrité des clés projet est garantie par une signature via un algorithme de HMAC. La signature est assurée par le HSM.

L'accès aux différents secrets est protégé par le mécanisme de contrôle d'accès commun à l'ensemble des services OpenStack (**oslo.policy**). Par défaut, un secret est accessible en lecture/écriture à son propriétaire et en lecture seule aux membres du projet. Il est possible de restreindre l'accès au seul propriétaire du secret en rendant ce dernier privé. Lorsqu'un secret est privé, il est possible de définir une liste blanche d'utilisateurs qui sont autorisés à le lire.

R26

Utiliser le contrôle d'accès de barbican pour restreindre l'accès aux secrets

Il est recommandé d'utiliser le mécanisme de contrôle d'accès **oslo.policy** afin de restreindre l'accès aux secrets, conformément au principe de moindre privilège.

Le service **barbican** offre une fonction de suppression des secrets présents dans sa base. Si cette fonction permet de garantir que le secret ne sera plus accessible au travers d'OpenStack, elle

n'inclut pas un effacement sécurisé du secret. Le secret étant néanmoins chiffré au moyen de sa clé projet, il ne pourra être déchiffré qu'au moyen d'un appel HSM incluant cette dernière clé.

R27

Ne permettre l'accès au HSM que depuis les machines hébergeant le service barbican

Il est recommandé de ne permettre l'accès aux interfaces de l'HSM qu'aux machines hébergeant le service **barbican**.

Le service barbican est en mesure d'associer des métadonnées aux secrets (date d'expiration, ...). Si le contrôle d'accès à ces métadonnées est bien assuré par OpenStack, ce dernier n'implémente pas de mécanismes de gestion du cycle de vie des secrets basé sur ces métadonnées. Si une automatisation de ce cycle de vie doit être mise en œuvre, elle devra l'être au moyen d'outils externes faisant appel aux APIs OpenStack.

R28

Mettre en place des procédures pour automatiser le cycle de vie des secrets

Il est recommandé de mettre en place des procédures basées des appels aux APIs OpenStack afin d'automatiser le cycle de vie des secrets.

Les secrets devant rester disponibles en clair pour les différents services qui les utilisent, leur protection est fortement liée au mécanisme qui en contrôle l'accès. Des mécanismes d'audit et de gestion des droits d'accès aux secrets doivent être mis à disposition des commanditaires, conformément aux exigences 9.3d, 9.3e, 9.3g et 9.4b du référentiel SecNumCloud :

SecNumCloud, Version 3.2, exigence 9.3d

Le prestataire doit être en mesure de fournir, pour une ressource donnée mettant en œuvre le service, la liste de tous les utilisateurs y ayant accès, qu'ils soient sous la responsabilité du prestataire ou du commanditaire ainsi que les droits d'accès qui leurs ont été attribués.

SecNumCloud, Version 3.2, exigence 9.3e

Le prestataire doit être en mesure de fournir, pour un utilisateur donné, qu'ils soient sous la responsabilité du prestataire ou du commanditaire, la liste de tous ses droits d'accès sur les différents éléments du système d'information du service.

SecNumCloud, Version 3.2, exigence 9.3g

Le prestataire doit inclure dans la procédure de gestion des droits d'accès les actions de révocation ou de suspension des droits de tout utilisateur.

SecNumCloud, Version 3.2, exigence 9.4b

Le prestataire doit mettre à disposition du commanditaire un outil facilitant la revue des droits d'accès des utilisateurs placés sous la responsabilité de ce dernier.

R29

Mettre à disposition des commanditaires des procédures d'audit et de gestion des droits d'accès aux secrets

Il est recommandé de mettre à disposition des commanditaires, des procédures d'audit et de gestion des droits d'accès aux secrets.

Les accès aux secrets doivent être journalisés (voir annexe A.2), conformément à l'exigence 12.6. Des outils doivent être mis à disposition des commanditaires pour l'effacement des secrets lorsqu'ils ne sont plus utilisés (section 5.2) ou en fin de contrat (section 5.4). Le risque de défaillance du contrôle d'accès aux secrets doit enfin être pris en compte lors de l'analyse de risque portant sur l'externalisation des biens sensibles.

R30

Journaliser les accès aux secrets

Il est recommandé de journaliser les accès aux secrets.

R31

Mettre à disposition des commanditaires des procédures d'effacement des secrets

Il est recommandé de mettre à disposition des commanditaires des procédures d'effacement des secrets lorsqu'ils ne sont plus utilisés ou en fin de contrat.

4.4 Gestion des supports amovibles et des supports de sauvegarde

Le but est ici de se protéger contre la perte ou le vol des logiciels ou des données placées sur des supports amovibles ou des supports de sauvegarde et amenés à quitter le périmètre de sécurité physique du site. Les impacts d'une telle perte ou d'un tel vol sont présentés dans la section 1.5.2. Ce risque est couvert par l'exigence 11.8 du référentiel SecNumCloud :

SecNumCloud, Version 3.2, exigence 11.8

Le prestataire doit documenter et mettre en œuvre une procédure de transfert hors site de données du commanditaire, équipements et logiciels. Cette procédure doit nécessiter que la direction du prestataire donne son autorisation écrite. Dans tous les cas, le prestataire doit mettre en œuvre les moyens permettant de garantir que le niveau de protection en confidentialité et en intégrité des actifs durant leur transport est équivalent à celui sur site.

OpenStack n'intégre pas de service assurant le chiffrement des données placées sur les supports amovibles et les supports de sauvegarde amenés à quitter le périmètre de sécurité physique du site. Néanmoins, l'utilisation du chiffrement des volumes assuré par **nova** et le chiffrement des images assuré par **swift** permet de garantir le chiffrement des données stockées, y compris lorsqu'elles sont placées sur des supports amovibles ou des supports de sauvegarde.

Le risque résiduel est ici identique à celui lié au chiffrement des données stockées (section 4.1).

4.5 Cloisonnement des dépendances de composants

Les composants OpenStack nécessitent pour la plupart la mise en œuvre de services tiers pour fonctionner. Le fait que les composants OpenStack soient développés à partir d'un unique framework réduit la variété des services tiers à la liste suivante :

- **SGBDR** : Souvent **MariaDB** ou **MySQL** mais il est possible de mettre en œuvre **PostgreSQL** ou **MSSQL** ;
- **Cache** : Il est possible de fonctionner sans service de cache mais ce mode de fonctionnement impacte fortement les performances et n'est donc pas viable en production. Le service de cache nécessaire doit être conforme au protocole **MemCache**. Il n'est donc pas obligatoire de mettre en œuvre **memcached**. Un autre produit peut être utilisé en substitution : **Redis, Infinispam**. Le choix doit prendre en compte les fonctionnalités spécifiques souhaitées, les performances attendues et le type de support utilisé ;
- **Message Queue** : Souvent **RabbitMQ**, OpenStack utilise le standard de message **AMQP**. Il est donc possible de mettre en œuvre des alternatives : **ActiveMQ, Apollo, HornetQ, Qpid ou ZeroMQ**. Le choix est à effectuer en fonction du niveau de compatibilité **AMQP** et du support ;
- **Serveur Web** : Souvent **Apache HTTPD**, il est possible de mettre en œuvre **Nginx** comme alternative.

4.5.1 Nom mutualisation des dépendances

Le fait de trouver, en dépendance, les mêmes services tiers pour beaucoup de composants OpenStack n'implique pas la mutualisation de ceux-ci. Bien que cette mutualisation soit fréquemment décrite dans les documentations officielles ou réalisée « de facto » par les installeurs automatiques, cette pratique tend à mettre en œuvre de possibles goulots d'étranglement et expose des données à des services sans considération de sécurité.

R32

Ne pas mutualiser les services de bases de données

Il est recommandé de ne pas mutualiser les services de bases de données pour des composants différents au sein d'un même serveur ou au sein d'un même cluster de serveurs.

En effet, le cloisonnement simple par l'utilisation de bases de données distinctes mais mutualisées au sein d'un même serveur ou ensemble de serveurs n'offre pas un niveau de cloisonnement suffisant.

Exemple

La mutualisation des bases de données au sein d'un même serveur ou cluster de serveurs :

- Expose des données de la vérification de signature des images (**Glance**) aux composants en charge de la gestion du réseau (**Neutron**) et des instances et projets (**Nova**) ;
- Expose des données d'authentification utilisateur (**Keystone**) aux composants en

charge de la gestion des images (**Glance**), du réseau (**Neutron**) et des instances et projets (**Nova**).

R33

Ne pas mutualiser les services de caches

Il est recommandé de ne pas mutualiser les services de caches pour des composants différents au sein d'un même serveur ou au sein d'un même cluster de serveurs.

Les services de cache sont généralement des services privilégiant la haute-disponibilité et la performance à la sécurité. Il n'est pas toujours possible de cloisonner les données entre elles, de plus l'authentification ou le chiffrement des flux de mise à jour ne sont souvent pas proposés par les implémentations.

4.5.2 Séparation réseau des services tiers

Les logiciels tiers utilisés par OpenStack en tant que stockage persistant (**MariaDB,Redis**) ou permettant la haute-disponibilité (**Pacemaker,Corosync**) utilisent des comptes applicatifs avec de forts privilèges pour effectuer des synchronisations ou des réplications de données. L'exposition de ces données sur un réseau similaire aux composants de l'infrastructure présente une vulnérabilité. Il est souhaitable de voir ces logiciels disposés sur un ou des réseaux dédiés.

R34

Mettre en œuvre un réseau spécifique pour les services tiers

Il est recommandé de mettre en place un ou plusieurs réseaux dédiés à l'hébergement des données, caches et messaging.

R35

Mettre en œuvre plusieurs réseaux spécifiques pour certains services tiers

Il est recommandé de mettre en œuvre un réseau dédié pour les services tiers faisant usage de protocole **multicast** (exemple : **Pacemaker et Corosync**).

5
Protection des données des commanditaires

5.1 Cloisonnement des ressources

Le but est ici de se protéger contre un défaut d'isolation résultant d'une défaillance des mécanismes de virtualisation, que cette défaillance soit malveillante ou accidentelle. Un tel défaut d'isolation est susceptible de permettre à un utilisateur illégitime d'accéder à la mémoire des machines virtuelles, aux flux réseau ou aux données stockées sur disques. Les impacts d'un tel accès illégitime sont présentés dans la section 1.5.2. Ce risque est couvert par l'exigence 9.7a du référentiel SecNumCloud :

SecNumCloud, Version 3.2, exigence 9.7a

Le prestataire doit mettre en œuvre des mesures de cloisonnement appropriées entre ses commanditaires.

Les mesures de cloisonnement des ressources entre les commanditaires comprennent :

- Celles relatives aux nœuds de calcul ;
- Celles relatives au stockage ;
- Celles relatives aux échanges réseau.

5.1.1 Cloisonnement des nœuds de calcul

L'approche retenue ici pour obtenir un cloisonnement des nœuds de calcul est de dédier un ou plusieurs nœuds de calcul à un commanditaire. Selon la sensibilité des environnements virtualisés mis en œuvre par chaque commanditaire, un cloisonnement plus fin devra être envisagé (plusieurs ensembles de nœuds de calcul pour un commanditaire unique).

5.1.1.1 Affectation de nœuds de calcul à un commanditaire

Nova intégre un mécanisme de sélection des nœuds de calcul basé sur des filtres [25]. Un des filtres disponible force l'exécution des machines virtuelles d'un projet donné sur un nœud de calcul spécifique (**AggregateMultiTenancyIsolation**, **filter_tenant_id**).

R36

Permettre la création d'aggrégats de nœuds de calculs dédiés par commanditaire

Il est recommandé de permettre l'utilisation des filtres pour le service **nova**, afin de permettre la création d'aggregats de nœuds de calculs dédiés pour les commanditaires.

Ce mécanisme se généralise à des groupes de machines en utilisant des aggrégats de nœuds de calcul plutôt que des nœuds unitaires.

Attention

Si ce mécanisme interdit bien l'exécution de machines virtuelles d'autres projets sur les nœuds filtrés, il n'empêche pas l'exécution de machines virtuelles associées au projet sur d'autres nœuds n'ayant pas de mécanisme de filtrage par projet.

Par défaut, le mécanisme de placement des machines virtuelles sélectionne le nœud de calcul le moins chargé. Pour forcer la création des machines virtuelles sur les nœuds dédiés au projet, il faut créer des modèles de machines virtuelles avec des métadonnées utilisés par les filtres **nova** pour filtrer les nœuds de calcul utilisables (**aggregate_instance_extra_specs :filter_tenant_id**).

R37

Proposer aux commanditaires des modèles de Machines Virtuelles dédiés aux aggrégats de nœuds de calcul

Il est recommandé d'intégrer aux modèles de machines virtuelles proposées aux commanditaires, des métadonnées utilisables par les filters **nova** pour sélectionner les aggrégats de nœuds de calcul sur lesquels instancier les machines virtuelles.

5.1.1.2 Prise en compte des mécanismes de migration à chaud

La migration à chaud des machines virtuelles est une des fonctionnalité d'intérêt dans les infrastructures nuagiques. Elle permet de déplacer une machine virtuelle d'un nœud de calcul vers un autre sans qu'elle soit arrêtée (cas d'usage typique : libération d'un nœud de calcul en vue d'une opération de MCO/MCS).

Ces opérations devant être effectuées le plus rapidement possible afin de réduire au maximum le risque d'attaque par déni de service [12]. Nous ne considérons ici que le cas des machines virtuelles dont les disques virtuels sont physiquement présents sur des nœuds de stockage distants, et pour lesquels les transferts réseau sont assurés par la couche d'abstraction au moyen de protocoles tel que iSCSI ou NFS. La migration à chaud de la machine virtuelle consiste alors à recopier le contenu de la mémoire qu'elle utilise, du nœud de calcul initial vers le nœud de calcul de destination.

Lors de cette opération, le contenu de la mémoire transite entre les deux machines via le réseau. Par défaut **nova** utilise un flux **ssh** pour la recopie mémoire [40]. Ces échanges devant être réalisés de manière automatique et nécessitant l'utilisation du compte *root*, les différents nœuds de calcul de l'infrastructure doivent être configurés pour permettre à ces comptes hautement privilégiés de se connecter depuis n'importe quel nœud de calcul vers n'importe quel autre nœud de calcul sans

avoir à saisir d'élément d'authentification.

Avec une telle configuration, un utilisateur malveillant ayant compromis le compte *root* d'un des nœuds est alors en mesure de prendre le contrôle de l'ensemble des nœuds. À partir des versions 4.4.0 de *libvirt* et 2.11 de *QEMU*, il est possible de sécuriser cette configuration en faisant transiter le contenu de la mémoire des machines virtuelles en cours de migration par la couche applicative *QEMU* en lieu et place de *ssh* [41]. Cette nouvelle approche évite d'avoir à configurer le compte *root* pour qu'il puisse qu'il puisse se connecter d'un nœud de calcul vers un autre sans avoir à fournir d'élément d'authentification.

R38

Configurer la migration à chaud des Machines Virtuelles pour qu'elle s'appuie sur QEMU

Il est recommandé de configurer le service **nova** et la couche **libvirt** pour qu'ils utilisent *QEMU* et non *ssh* pour les migrations à chaud de machines virtuelles entre nœuds de calcul.

La création d'aggrégats de nœuds de calcul et l'utilisation de *QEMU* pour la migration à chaud des machines virtuelles permettent bien d'obtenir un cloisonnement des nœuds de calcul. Il faut néanmoins noter que la création de nouveaux aggrégats ou l'ajout de nœuds à un aggrégat nécessitent des opérations d'administration côté prestataire et induisent des délais qui doivent être pris en compte lors de la planification des montées en charge des environnements virtualisés. Enfin, *QEMU* doit être configuré avec un chiffrement TLS à l'état de l'art comme pour les autres flux présentés à la section 4.2.

R39

Configurer QEMU pour mettre en œuvre du chiffrement TLS à l'état de l'art lors de la migration des Machines Virtuelles

Il est recommandé de configurer la couche **QEMU** pour que les flux utilisés pour la migration à chaud des machines virtuelles mettent en œuvre un chiffrement TLS à l'état de l'art.

5.1.2 Cloisonnement du stockage

L'approche retenue ici pour obtenir un cloisonnement du stockage est de dédier un espace de stockages à un commanditaire. Selon la sensibilité des environnements virtualisés mis en œuvre par chaque commanditaire, un cloisonnement plus fin devra être envisagé (plusieurs espaces de stockage pour un commanditaire unique).

Swift permet l'utilisation de plusieurs ensembles de stockages (appelés *rings*) auxquels sont rattachés les périphériques de stockage. Pour chaque ensemble de stockages, il est possible de définir une politique de stockage [28]. Ces politiques de stockage ne comprennent pas de paramètres permettant de dédier un ensemble de stockages à un projet ou à un domaine.

R40

Configurer le service swift pour qu'il dispose de plusieurs ensembles de stockage

Il est recommandé de configurer le service **swift** pour disposer de plusieurs ensembles de stockage (*rings*).

Cinder permet l'utilisation de plusieurs ensembles de stockages (appelés *backends*). Si on dispose de plusieurs ensembles de stockage, il est possible de détourner le mécanisme de quotas pour dédier un ensemble de stockage à un projet donné (on force le quota à 0 pour l'ensemble des projets sauf celui que l'on souhaite autoriser). Pour forcer un projet à utiliser un ensemble de stockage donné, il faut que tous les ensembles de stockages dont on souhaite interdire l'utilisation aient un quota à 0 [53].

R41

Configurer le service cinder pour qu'il associe un backend à chaque ensemble de stockage

Il est recommandé de configurer le service **cinder** pour qu'il associe un *backend* à chaque *ring* configuré avec **swift**.

R42

Configurer les quotas cinder pour restreindre l'accès des projets aux backends

Il est recommandé de configurer le service **cinder** pour restreindre l'accès des projets à des *backends* spécifiques via les *quotas*.

Aucun mécanisme de cloisonnement physique du stockage des images n'a été identifié sur **glance**.

Le cloisonnement du stockage n'est ici que partiellement atteint car les images ne peuvent pas être stockées sur un espace dédié au commanditaire. Le risque résiduel qui en résulte doit être pris en compte lors des analyses de risques.

Il faut enfin noter que la mise en place d'espaces de stockage dédiés nécessitent des opérations d'administration côté prestataire et induisent des délais qui doivent être pris en compte lors de la planification des montées en charge de la capacité de stockage.

5.1.3 Cloisonnement du réseau

La mise en place d'un réseau physique dédié pour chaque commanditaire n'étant pas réaliste, l'approche retenue ici est d'obtenir un cloisonnement du réseau en reforcant la ségmentation logique nativement mise en œuvre par le service **neutron** avec le chiffrement des flux échangés sur ces segments logiques. Selon la sensibilité des environnements virtualisés mis en œuvre par chaque commanditaire, un cloisonnement plus fin devra être envisagé (plusieurs segments réseau pour un commanditaire unique).

Neutron permet d'interfacer directement les machines virtuelles avec le réseau sous-jacent (réseau à plat ou réseau ségmenté au moyen de VLANs). Il permet aussi de construire des réseaux

virtualisés totalement indépendants de la structure du réseau sous-jacent, via un mécanisme d'encapsulation (VXLAN, GENEVE).

Le nombre de réseaux nécessaires pour une infrastructure virtualisée multi-commanditaires étant trop important pour une segmentation à base de VLANs, cette approche ne sera pas étudiée ici.

Avec un mécanisme d'encapsulation, les trames de niveau 2 produites par les réseaux virtuels sont acheminées par le réseau sous jacent après avoir été encapsulées dans des paquets de niveau 3. Cette opération d'encapsulation est assurée par les commutateurs virtuels distribués entre les machines hôtes hébergeant des machines virtuelles connectées au réseau.

Même si du point de vue du réseau virtualisé, les segments niveau 2 semblent isolés les uns des autres, les paquets échangés entre les machines hôtes circulent sur le même réseau sous-jacent. Si un attaquant réussi à s'évader de sa machine virtuelle et à prendre le contrôle du nœud de calcul hébergeant celle-ci, il aura alors accès à l'ensemble des flux reçus par le nœud et sera potentiellement en mesure de détourner des flux vers ce nœud en forgeant des trames à destination du plan de contrôle du réseau défini par logiciel.

Afin de cloisonner les flux échangés sur ces réseaux, il faut configurer leurs commutateurs virtuels distribués pour que les échanges entre les machines hôtes soient chiffrés [34]. La mise en œuvre de cette fonctionnalité n'est pas assurée automatiquement par le service **neutron** de OpenStack, ou par Kolla-Ansible.

R43 +

Configurer le service neutron pour encapsuler les échanges entre les nœuds de calcul dans des tunnels IPSec

Il est recommandé de configurer le service **neutron** pour que les échanges entre les nœuds de calcul soint encapsulés dans des tunnels IPSec.

La création de segments réseau virtuels et le chiffrement automatique des flux échangés permettent bien d'obtenir un cloisonnement réseau. Une attention particulière doit être portée à la protection de l'accès aux clés utilisées pour le chiffrement des flux échangés sur ces réseaux virtuels, tel qu'indiqué dans l'exigence 10.5c du référentiel SecNumCloud :

SecNumCloud, Version 3.2, exigence 10.5c

Le prestataire doit protéger l'accès aux clés cryptographiques et autres secrets utilisés pour le chiffrement des données par un moyen adapté : conteneur de sécurité (logiciel ou matériel) ou support disjoint.

Il faut enfin noter que la création de nouveaux segments réseau ou l'ajout de nœuds à ces segments réseau nécessitent des opérations d'administration côté prestataire et induisent des délais qui doivent être pris en compte lors de la planification des montées en charge des environnements virtualisés.

5.2 Effacement sécurisé des supports de données

Nous cherchons ici à nous protéger contre un accès aux données précédemment stockées par un commanditaire. Ces données restent présentes dans un espace de stockage rendu disponible lorsque le commanditaire libère le stockage virtualisé associé.

Les données restant accessibles sur les supports de stockage, elles sont potentiellement consultables par un utilisateur illégitime, que ce soit de façon malveillante ou accidentelle. Les impacts d'un tel accès illégitime sont présentés dans la section 1.5.2. Ce risque est couvert par l'exigence 11.9a du référentiel SecNumCloud :

SecNumCloud, Version 3.2, exigence 11.9a

Le prestataire doit documenter et mettre en œuvre des moyens permettant d'effacer de manière sécurisée par réécriture de motifs aléatoires tout support de données mis à disposition d'un commanditaire. Si l'espace de stockage est chiffré dans le cadre de l'exigence 10.1 a), l'effacement peut être réalisé par un effacement sécurisé de la clé de chiffrement.

Les services OpenStack n'incluent pas de mécanisme d'effacement sécurisé par réécriture de motifs aléatoires sur les supports physiques utilisés pour le stockage. Lorsque les données sont chiffrées avant d'être placées sur les supports, un effacement cryptographique peut être réalisé.

Selon le conteneur logique utilisé, les possibilités d'effacement cryptographique ou non sont présentées ci-dessous :

- Volumes : Lorsque le service **nova** prend en charge le chiffrement des volumes en s'appuyant sur l'hyperviseur sous jacent pour les opérations cryptographiques et sur le service **barbican** pour le stockage sécurisé des clés de chiffrement, il est possible de procéder à un effacement du contenu des volumes par suppression de leur clé de chiffrement (effacement cryptographique) ;
- Images : En absence de mécanisme de chiffrement des images, et sachant qu'elles sont stockées sur des supports partagés entre les commanditaires (section 4.1), il n'est pas possible de procéder à l'effacement effectif de leur données ;
- Snapshot d'image : Les Snapshot d'images permettent la création « à la volée » de nouvelles images basées sur le contenu d'instances en cours d'utilisation. Ces snapshot étant gérés par **glance** au même titre que les images elles-meme, l'absence de capacité d'effacement de ces dernières s'applique aussi au Snapshot d'images ;
- Snapshot de Volume : Les Snapshot de volumes permettent la création de volume à partir d'instances suspendues ou arrêtées. Les capacités d'effacement cryptographique applicables aux volumes s'appliquent aussi aux snapshots de volumes ;
- Métadonnées utilisateur : L'utilisation d'un serveur LDAP permet d'éviter la présence de métadonnées utilisateur dans la base **keystone** (section 2.1). Un mécanisme d'effacement sécurisé est nécessaire au niveau du serveur LDAP si celui-ci n'est pas géré directement par le commanditaire ;

- Autres secrets : L'ensemble des éléments secrets associés à un projet étant protégés par une clé spécifiquement dédiée à la protection des secrets du projet (la clé projet), il est possible de procéder à l'effacement cryptographique des secrets par suppression de cette clé projet ;
- Données présentes sur des supports amovibles : Si celles-ci sont chiffrées en utilisant un secret associé au projet et stocké par **barbican**, la suppression de la clé projet permet de procéder à l'effacement cryptographique des données présentes sur les supports amovibles.

Les clés projet sont effacées à la suppression de ce dernier (module **barbican-keystone-listener**). Ce mécanisme permet en théorie de garantir, à la suppression du projet, l'effacement cryptographique de l'ensemble des éléments secret du projet gérés par **barbican**. En pratique, l'effacement cryptographique n'est effectif que si la clé projet n'est présente sur aucun support de sauvegarde.

Les clés projet étant protégées par la clé maître du HSM, il est néanmoins possible de garantir l'effacement cryptographique de ces clés en changeant la clé maître du HSM :

- Pour les projets actifs, la clé projet est transchiffrée avec la nouvelle clé maître ;
- Pour les projets supprimés, la clé projet éventuellement encore présente dans un support de sauvegarde est effacée de manière cryptographique par destruction de l'ancienne clé maître.

En mettant en œuvre une politique de renouvellement périodique de la clé maître du HSM, on est donc en mesure de garantir une durée maximale de rétention des données projet égale à la durée de vie de la clé maître (durée d'utilisation comme clé maître, à laquelle il faut ajouter la durée de conservation permettant de garantir le transchiffrement des clés projet pour les projets actifs).

R44

Mettre en place une politique de renouvellement périodique des clés maitres du HSM

Il est recommandé de mettre en œuvre une politique de renouvellement périodique des clés maîtres du HSM.

Barbican ne disposant pas d'une fonction de renouvellement de la clé projet avec transchiffrement de l'ensemble des secrets actifs pour le projet, l'effacement cryptographique de ses secrets actifs ne peut être réalisé que par suppression du projet auquel est associé le secret. Le risque résiduel qui en résulte, ainsi que celui portant sur l'impossibilité d'effacement sécurisé des images doivent être prise en compte dans les analyses de risques.

5.3 Suppression des données à la désinscription d'un utilisateur

Le but est ici de se protéger contre un accès illégitime aux APIs par un utilisateur associé à un commanditaire et désinscrit par ce dernier. Les impacts d'un tel accès illégitime sont présentés dans la section 1.5.1. Ce risque est couvert par l'exigence 9.2a du référentiel SecNumCloud :

SecNumCloud, Version 3.2, exigence 9.2a

Le prestataire doit documenter et mettre en œuvre une procédure d'enregistrement et de désinscription des utilisateurs s'appuyant sur une interface de gestion des comptes et des droits d'accès. Cette procédure doit indiquer quelles données doivent être supprimées au départ d'un utilisateur.

Nous chechons aussi à nous protéger contre un accès illégitime ou une utilisation illégitime des ressources virtualisées sous la responsabilité de cet utilisateur. Les impacts d'un tel accès illégitime sont présentés dans la section 1.5.2. Ce risque est lui aussi couvert par l'exigence 9.2a du référentiel SecNumCloud.

Dans le cadre d'une infrastructure nuagique, les données associées à un utilisateur sont :

- Les données liées à son identité et ses données d'authentification. Pour une infrastructure OpenStack, ces données sont gérées par le service **keystone** (avec une éventuelle délégation à un serveur LDAP) ;
- Les éléments de l'infrastructure virtuelle associés à l'utilisateur (machines virtuelles, images, volumes, ...). Ces éléments peuvent être divisées en deux ensembles :
  - Ceux qui sont de la responsabilité de l'utilisateur, mais dont la portée est un projet ou une arborescence de projets ;
  - Ceux qui sont utilisées exclusivement par l'utilisateur.
- Les secrets qui sont de la responsabilité de l'utilisateur. Comme pour les éléments de l'infrastructure virtuelle, les secrets peuvent être partagés au niveau projet, arborescence de projets, ou uniquement utilisées par leur propriétaire.

Lorsque la gestion des utilisateurs est faite via un serveur LDAP, la majeure partie des données relatives à l'utilisateur restent stockées sur ce serveur. La base de données utilisée par le service **keystone** ne contient que le minimum d'informations nécessaires pour associer les ressources OpenStack à l'utilisateur [26]. On parle alors d'un "profil fantôme". Ce profil fantôme seul ne permet pas à l'utilisateur de continuer à interagir avec les APIs et ne doit être supprimé qu'une fois l'ensemble des ressources précédemment sous la responsabilité de l'utilisateur ont été supprimées ou affectées à d'autres utilisateurs.

OpenStack ne dispose pas de fonction permettant l'effacement automatique d'un ensemble prédéfini de ressources virtualisées associées à un utilisateur lors de la suppression du profil de ce dernier. Le risque résiduel qui en résulte doit être pris en compte dans les analyses de risques.

5.4 Effacement sécurisé de l'intégralité des données du commanditaire en fin de contrat

Le but est ici de se protéger contre un accès aux données précédemment stockées par le commanditaire. Ces données restent présentes dans un espace de stockage rendu disponible lorsque le

domaine associé au commanditaire est supprimé par le prestataire.

Les données restant accessibles sur les supports de stockage, elles sont potentiellement consultables par un utilisateur illégitime, que ce soit de façon malveillante ou accidentelle. Les impacts d'un tel accès illégitime sont présentés dans la section 1.5.2. Ce risque est couvert par les exigences 19.4a et 19.4b du référentiel SecNumCloud :

SecNumCloud, Version 3.2, exigence 19.4a

À la fin du contrat liant le prestataire et le commanditaire, que le contrat soit arrivé à son terme ou pour toute autre cause, le prestataire doit assurer un effacement sécurisé de l'intégralité des données du commanditaire. Cet effacement peut être réalisé suivant l'une des méthodes suivantes, et ce dans un délai précisé dans la convention de service :

- *effacement par réécriture complète de tout support ayant hébergé ces données ;*
- *effacement des clés utilisées pour le chiffrement des espaces de stockage du commanditaire décrit au chapitre 10.1 ;*
- *recyclage sécurisé, dans les conditions énoncées au chapitre 11.9.*

SecNumCloud, Version 3.2, exigence 19.4b

À la fin du contrat, le prestataire doit supprimer les données techniques relatives au commanditaire (annuaire, certificats, configuration des accès, etc.).

En fin de contrat, la suppression des projets et domaines associés au commanditaire entraîne la libération des espaces de stockage associés. Ces suppressions s'accompagnent aussi de la suppression des clés projet associées. Conformément à ce qui a été présenté dans les sections 4.1 et 5.2, cela induit un effacement cryptographique du contenu des espaces de stockage utilisés par le commanditaire, à l'exception des images qui ne sont pas chiffrées avec les clés projet.

L'utilisation d'un serveur LDAP permet d'éviter la présence de métadonnées utilisateur dans la base **keystone** (voir section 2.1). Un mécanisme d'effacement sécurisé est nécessaire au niveau du serveur LDAP si celui-ci a été mis à disposition par le prestataire.

R45

Mettre en place une procédure d'effacement sécurisé des données du commanditaire présentes sur le serveur LDAP

] Il est recommandé de mettre en place une procédure d'effacement sécurisé des données du commanditaire présentes sur le serveur LDAP si celui-ci est mis à disposition par le prestataire.

Le risque résiduel associé à l'absence d'effacement sécurisé des images doit être pris en compte dans les analyses de risques.

5.5 Durcissement des composants

5.5.1 Sécurisation des hyperviseurs

Il a été remarqué que dans le cas d'utilisation de **qemu/kvm** le compte Unix utilisé pour l'instanciation des processus était toujours le même. Cette pratique présente un risque car elle ne permet pas une isolation suffisante entre les processus d'exécution des machines virtuelles.

Information

Une « configuration » spécifique de **SELinux** permet de contenir les risques d'interactions sur un même nœud de calcul, d'une machine virtuelle sur une autre machine virtuelle, d'une machine virtuelle sur le nœud de calcul.

R46

Utiliser un module de sécurité Linux (LSM)

Il est recommandé de mettre en œuvre un **Linux Security Module** sur les hyperviseurs ou un système de cloisonnement similaire.

5.5.2 Durcissement des services tiers

5.5.2.1 Durcissement du service de messaging

Le service de messaging étant installé sur un réseau spécifique, les connexions entrantes vers celui-ci sont à filtrer en considérant que seuls les composants en ayant la nécessité sont autorisés à utiliser ce service. Cette remarque est à appliquer pour les composants *EndPoints* de l'infrastructure comme pour les composants présentés sur le frontal utilisateur.

En exemple, les composants suivants n'ont pas accès au messaging :

- **Horizon** ;
- **Keystone** ;
- **Glance** ;
- ...

R47

Sécuriser la configuration du messaging (RabbitMQ uniquement)

Il est recommandé de supprimer le compte administrateur par défaut (**guest**). Pour le cas où un compte d'administrateur est nécessaire, le nom de ce compte doit être différent.

Enfin une ACL d'accès restrictive doit être réalisée afin de contrôler les accès des comptes aux « topics ».

6
Mise en œuvre de la journalisation sur les services OpenStack

6.1 Objectifs

La mise en place d'un niveau de journalisation correcte est une étape importante lors de la mise en place d'un environnement OpenStack. Autant pour la détection des comportements malveillants que pour suivre le bon fonctionnement des différents services, la mise en place d'un système de journalisation est nécessaire. Sa configuration peut de prime à bord sembler fastidieuse, ainsi l'objectif de ce chapitre est de présenter les différents modules de journalisation proposés par OpenStack, leur intérêt et les configurations nécessaires de manière à faciliter la détection et l'analyse des incidents de sécurité.

Ce chapitre reprendra les principaux éléments du guide recommandations de sécurité pour l'architecture d'un système de journalisation de l'ANSSI [5] et apportera des précisions sur les points spécifiques à OpenStack.

6.2 Synchronisation des horloges

Les différents serveurs hébergeant les services OpenStack vont générer des journaux en utilisant l'horloge du système. Les horloges de tous les équipements dérivent naturellement dans le temps. Ces écarts peuvent se mesurer en secondes voire en minutes après quelques semaines de dérive. Or, la compréhension de l'enchaînement précis d'évènements issus de plusieurs journaux est plus difficile lorsque les équipements qui produisent ces journaux ne disposent pas du même temps de référence.

Il est donc important de synchroniser les horloges des différents serveurs de l'infrastructure. Une des méthodes les plus répandues est l'utilisation du protocole NTP.

NTP

Le protocole NTP est largement utilisé pour synchroniser les équipements sur une ou plusieurs sources de temps. Il est disponible pour la plupart des systèmes d'exploitation, quel que soit le type d'équipement. Une architecture NTP type comprend généralement un ou plusieurs serveurs NTP internes sur lesquels se synchronisent les machines du SI. Ces serveurs référents peuvent, quant à eux, calibrer leurs horloges à l'aide de matériel spécifique (intégrant une horloge atomique ou utilisant les signaux

radio ou satellitaires par exemple) ou bien se synchroniser avec des serveurs NTP publics. Une source de temps peut avoir une précision plus ou moins importante par rapport au temps universel.

R48

Veiller à la synchronisation des horloges

Les horloges des serveurs doivent être synchronisées sur plusieurs sources de temps internes cohérentes entre elles. Ces dernières peuvent elles-mêmes être synchronisées sur plusieurs sources de temps externes fiables, sauf dans le cas particulier de réseaux physiquement isolés.

6.3 Les sources de journalisation applicative d'OpenStack

L'environnement OpenStack met à disposition des administrateurs plusieurs sources de journalisation. Chacune d'entre elles possède ses propres caractéristiques et fonctionalités.

La principale source de journalisation fournie par OpenStack est la génération de "notifications". On peut distinguer 2 types de notifications : les notifications internes des différents services et les notifications d'audit des requêtes API.

Les notifications internes permettent d'avoir une vision des différentes actions qui ont été réalisées au sens applicatif. On obtiendra par exemple des notifications lors de la création d'une instance, son démarrage, sa sauvegarde ou encore sa suppression. Ces notifications contiennent différentes informations, les principales étant : la ressource ciblée, l'action effectuée, la date, l'utilisateur associé à l'action et différentes données spécifiques à chaque type de notification.

Les APIs exposées par les différents services OpenStack peuvent aussi générer des notifications. Ces APIs suivent la norme WSGI (Web Server Gateway Interface). Cette norme permet l'utilisation de "middleware", des composants qui peuvent interagir avec les différentes requêtes faites à l'API, les "middleware" s'exécutant sur chaque requête les uns après les autres. Un "middleware" permettant la mise en place d'audit des APIs est fourni par OpenStack : *keystonemiddleware*. L'utilisation de ce composant va permettre la journalisation d'informations décrivant les requêtes et réponses des APIs. On y retrouve notamment les endpoints requêtés, les utilisateurs associés aux requêtes et le contexte de la requête (ip, date, code de retour, etc...).

Ces deux premières sources de données utilisent les modules **oslo.log** et **oslo.messaging** qui font partie des modules de base des différents services. Le premier est une surcouche à la bibliothèque de journalisation standard de Pythonet permet de rediriger les évènements vers un espace de traitement ou de stockage, par exemple un fichier ou service syslog. Le second permet de rediriger les notifications vers différentes sources comme RabbitMQ, ZMQ ou encore Kafka.

Les différents services OpenStack produisent également des journaux d'information sur le fonctionnement des différents modules qui les composent. Leur génération repose uniquement sur le module **oslo.log** et sa configuration.

Les services qui exposent des APIs le font par le biais de serveurs **apache**. Cette solution dispose de

son propre système de journalisation. Ces journaux contiennent des informations sur l'ensemble des requêtes traitées par le serveur **apache**.

Certains services proposent une journalisation spécifique, c'est le cas du service **swift**. La journalisation disponible avec le service **swift** permet de suivre les différentes actions réalisées par les utilisateurs (lecture, écriture, suppression, etc...) sur les données stockées dans les différents conteneurs. Les journaux produits contiennent le même type d'information que les journaux HTTP (**keystonemiddleware** et **apache**) en plus de précisions sur la ressource ciblée. Ces journaux sont générés par la partie serveur proxy su serveur **swift**.

Le service **horizon**, permettant d'interagir avec les services OpenStack depuis une interface Web, dispose d'un système de journalisation propre. Le service utilise **django** comme serveur Web. Il possède donc la journalisation de ce dernier. Un second système de journalisation est également disponible avec les journaux *Operation* d'**horizon**. Ils permettent de suivre les différentes actions réalisées sur l'environnement OpenStack depuis l'interface Web d'**horizon**. Ces journaux contiennent le même type d'information que les notifications des différents services OpenStack, mais permettent de suivre plus simplement les actions qu'un utilisateur d'OpenStack passant par l'interface Web.

On peut résumer la répartition des sources de journalisation des principaux services d'OpenStack avec la table 2.

Table 2 – Répartition des sources de journalisation des principaux services OpenStack

Service	**Notifications**	**Audit middleware**	**Journaux HTTP**	**Journaux spécifiques**
Keystone	X		X	
Nova	X	X	X	
Cinder	X	X	X	
Glance	X	X	X	
Swift				Journaux d'accès aux ressources Swift
Neutron	X	X	X	
Horizon			X	Journaux Opération

6.4 Collecte, stockage et rotation des journaux

Afin d'assurer la disponibilité des informations essentielles aux activités de détection et d'analyse, il est nécessaire de mettre en place une politique de rétention des journaux d'évènements. Celle-ci doit notamment tenir compte des contraintes opérationnelles impliquées par le stockage d'un grand volume d'informations. Pour ces raisons, il est d'usage de mettre en place un système de collecte et de centralisation des journaux. Cette pratique consiste à acheminer l'ensemble des journaux générés vers un espace de stockage dédié, pour en faciliter la conservation et la manipulation à des fins d'analyse. Cette pratique permet de dissocier le stockage des journaux de celui des données d'OpenStack, simplifiant par la même occasion la gestion des différents espaces de stockage. Concernant les solutions de collecte des journaux, il n'existe pas de solution générique dans l'en-

vironnement OpenStack tant la diversité des journaux est importante. Il convient donc d'établir une solution propre à l'environnement et aux types de journaux activés. **Syslog** étant supporté par l'ensemble des sources de journaux, **syslog** se présente comme une solution à privilégier.

Il est également d'usage de mettre en place une rotation des journaux. Cela permet entre autre d'éviter de manipuler des fichiers de journaux trop volumineux, de les compresser et de garder le contrôle sur le système de stockage en archivant les journaux trop anciens au profit de nouveaux. Cette pratique porte le nom de *rotation des journaux* et s'avère indispensable à tout système de journalisation pérenne.

Comme évoqué précédemment, la mise en place d'un système de journalisation dépend des journaux qu'on souhaite conserver. L'utilisation de **syslog** est à privilégier pour la mise en place d'un tel système dans l'environnement OpenStack. De plus, il est important de suivre les différentes recommandation présentées dans le guide de journalisation de l'ANSSI dédié à la mise en place d'un système de journalisation (Chapitre 3 : Architecture et conception d'un système de journalisation) [5].

6.5 Configuration de la journalisation système via AuditD

En plus de la journalisation applicative, il peut être intéressant de mettre en place de la journalisation système pour identifier des compromissions de l'infrastructure. Un des moyens les plus répandus dans l'univers Linux est l'utilisation du framework **auditD**. **Auditd** est un service de journalisation avancé souvent présent sur les distributions GNU/Linux. Il permet de journaliser différentes actions effectuées sur un système Linux.

AuditD

Le framework **auditD** est un ensemble de composants permettant la génération d'évènements d'audits dans un environnement Linux. Il est disponible sur la majorité des systèmes Linux et peut être configuré par l'administrateur. Son composant principal est le démon **auditd**, il est responsable de la réception des évènements d'audits et de leur stockagesur la machine. Ce framework permet, entre autre, d'auditer les accès à des fichiers, des appels système ou encore des commandes effectuées par les utilisateurs.

Un exemple de mise en place d'**auditd** est disponible dans le guide "Recommandations de configuration d'un système GNU/Linux" (Chapitre 6 : Configuration et services système) [2].

R49

Auditer les accès aux fichiers de configuration des services OpenStack

Les fichiers de configuration OpenStack étant susceptibles de contenir des éléments sensibles, une bonne pratique consiste à metre en place un démon **auditd** et à configurer ce démon pour qu'il trace l'ensemble des accès à ces fichiers de configuration, que ces accès soient des accès en lecture ou en écriture.

6.6 Configuration recommandée pour la journalisation

Dans un objectif de supervision de l'environnement OpenStack, il est crucial d'être en capacité d'identifier toutes les tentatives de connexion. Les environnements cloud étant principalement basés sur l'authentification des utilisateurs, les tentatives de connexion font partie du chemin critique et elles permettent généralement de détecter des compromissions. Ainsi, la surveillance des journaux issus du service **keystone** est la brique de base nécessaire à toute stratégie de supervision. Elle rend notamment possible la détection de scénario génériques comme les attaques par force brute sur des identifiants de connexion.

Pour améliorer la capacité de détection, il convient ensuite d'utiliser d'autres sources de journaux. Il peut en effet être intéressant d'avoir la capacité de pivoter entre les différentes requêtes émises par un utilisateur pour mieux comprendre les actions effectuées par l'utilisateur et établir un potentiel schéma d'attaque. Pour cela, on peut s'appuyer sur les journaux générés par les services principaux d'OpenStack autre que **keystone**. Les services critiques comme la virtualisation, le stockage des images ou encore l'interface d'administration, sont des cibles de choix pour un attaquant et traitent à eux seuls la majorité des requêtes qui transitent dans l'environnement OpenStack. La supervision de ces journaux répond à un enjeu double : superviser la majorité de l'environnement et protéger les services critiques. En pratique on peut donc s'appuyer sur les notifications de ces services et les journaux spécifiques pour les services dépourvus de notifications.

Enfin, il est encore possible d'enrichir les informations déjà à disposition en utilisant les journaux d'Audit middleware et HTTP (apache et django) pour tous les services déjà surveillés qui les supportent. Ces derniers peuvent apporter un complément d'information notamment dans le cas d'une requête post-compromission. Le surplus d'informations qu'ils fournissent peut s'avérer volumineux et redondant avec les informations générées par les autres sources de données, mais apporte également des précisions sur les actions de l'utilisateur. Ces nouvelles informations ne sont pas aisément utilisables pour établir des règles de détection (peu de détails sur l'action effectuée), mais peuvent fournir des informations exploitables à posteriori, notamment pour retracer le parcours d'un attaquant.

In fine, les recommandations concernant les choix de journalisation à effectuer sont les suivantes :

R50 -

Activer la journalisation essentielle à la détection de compromission

A minima, les notifications du service **keystone** et les journaux opération du service **horizon** doivent être activés et configurés (notamment le format des journaux opération) pour détecter une intrusion et son origine.

R50

Activer la journalisation importante pour la détection de compromission

Il est recommandé d'activer et de configurer les notifications des services **keystone**, **glance**, **cinder**, **nova** et **neutron** ainsi que les journaux opération du service **horizon** et les journaux de **swift** pour détecter un comportement malveillant au sein de l'environnement OpenStack. L'ajout des champs **account**, **container** et **object** dans le format des journaux **swift** étant également conseillé.

R50 +

Activer l'ensemble des systèmes de journalisation

En plus des recommandations précédentes, on peut ajouter au moins une source des journaux HTTP : les journaux générés par django pour **horizon**, et pour tous les services avec API, les journaux apache ou **keystonemiddleware** sont à favoriser (ceci permet d'uniformiser le traitement des notifications des services OpenStack).

Ces journaux pouvant être volumineux, il faudra porter attention aux différentes problématiques de traitement et de stockage.

Annexe A
Annexe

A.1 Gestion des droits d'accès aux ressources

OpenStack met en œuvre une politique de gestion des droits d'accès basée sur des rôles. Cette politique est implémenté via la librairie **oslo.policy** qui est utilisée par l'ensemble des services OpenStack pour gérer leurs politiques d'accès respectives.

Chaque service défini ses propres points de contrôles et les droits par défaut. De ces éléments sont dérivés les fichiers de configuration des services (**policy.json** ou **policy.yaml** selon la version d'OpenStack). Ces fichiers définissent les droits d'accès par défaut aux différentes fonctionnalités de l'API de chaque service, ainsi que les droits par défaut pour les rôles autorisés à appeler ces fonctionnalités.

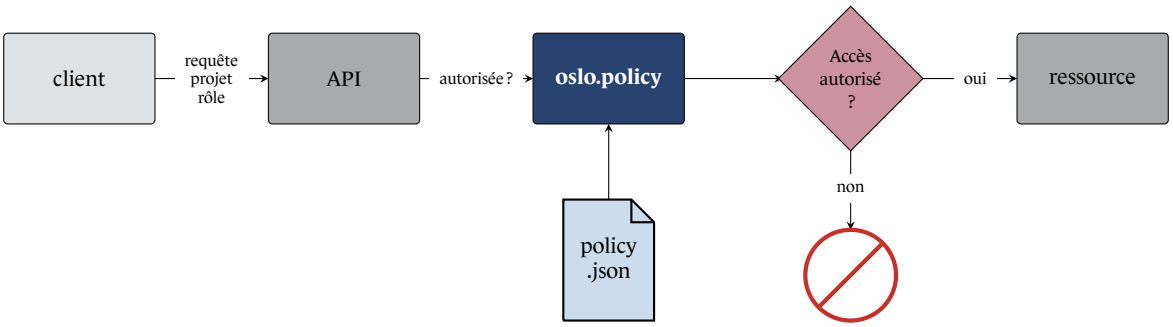

Figure 13 – Gestion des droits d'accès

Il est possible de personnaliser ce fichier de configuration afin d'ajouter de nouveaux rôles pour le service ou de modifier les droits attribués aux rôles standards (une telle modification est néanmoins fortement déconseillé par la documentation OpenStack).

Les rôles sont transverses aux domaines. Il est néanmoins possible de restreindre les droits accordés par domaine en configurant le fichier **policy.json** pour que la librairie **oslo.policy** de contrôle que le domaine auquel appartient l'utilisateur est aussi le domaine auquel appartient le projet avec lequel il essaie d'interagir.

Afin de faciliter la gestion des droits des utilisateurs, il est possible d'associer des rôles à des groupes d'utilisateurs. Chaque utilisateur membre du groupe hérite alors du ou des rôles associés au groupe, et des autorisations d'accès qui en découlent.

En utilisant conjointement les rôles affectés aux groupes et la restriction des droits au domaine

d'appartenance du projet cible, il est possible de définir de manière globale des droits restreints au domaine d'appartenance pour des groupes d'utilisateur donnés. Ces groupes et les droits associés peuvent être gérés de manière globale à l'infrastructure OpenStack. Ils sont alors disponibles pour chaque nouveau domaine créé, sans modifier la configuration des services autres que **keystone**.

A.2 Mise en place de la journalisation

A.2.1 Configuration des services de journalisation

Les parties qui suivent ont pour ambition de présenter les différents types de journalisation évoqués ci-dessus, et d'évoquer les changements de configuration nécessaires à leur bon fonctionnement. En effet, la configuration par défaut n'active pas forcément les types de journaux souhaités. Une modification préalable est donc nécessaire.

A.2.1.1 Notifications

Ces logs, appelés **notifications**, sont générés par défaut par la plupart des services d'OpenStack. Ils sont également par défaut au format CADF [14] (*Cloud Auditing Data Federation*) que nous utiliserons par la suite. Pour chacun des services concernés, il faut modifier le fichier de configuration correspondant, afin d'y définir le driver souhaité pour la génération des logs.

Information

Pour connaître les différentes valeurs que peut pendre le paramètre **driver**, dans la section **[oslo_messaging_notifications]**, il est possible de se référer à la documentation d'OpenStack [39].

Information spécifique keystone

Il est possible pour le service **keystone**, qui génère beaucoup de logs par défault, de ne pas générer une partie de ceux-ci. Cette sélection s'effectue en ajoutant, dans le paramètre **notification_opt_out**, le nom du type d'évènement que l'on souhaite omettre de la génération de logs.

Il est recommandé de stopper la génération des évènements de type **identity.authenticate.pending**. En fonction des ressources disponibles, il peut également être opportun de stopper la génération des évènements **identity.authenticate.success**.

A.2.1.2 Journaux Audit *Middleware*

Comme énoncé plus haut, il existe un WSGI middleware permettant de générer des logs d'audit pour chacune des requêtes reçues par les services qui l'activent. Ce filtre fonctionne à la manière d'un pipeline à l'entrée duquel arrivent les différentes requêtes à destination des services OpenStack, et duquel sortent in fine les requêtes qui sont réellement transmises aux services correspondants. A l'intérieur de ce pipeline, les requêtes vont être filtrées si elles ne contiennent pas

le token d'authentification, car comme présenté sur la figure 1.1, chaque requête envoyée à un service OpenStack nécessite au préalable d'être authentifié par le service **keystone**. Une fois la vérification effectuée, la requête continue son cheminement au sein du pipeline et est enrichie avec le contexte connu par **keystone**. Enfin, à la sortie de cet enrichissement se trouve le filtre d'audit middleware qui nous intéresse ici. Ce dernier reçoit donc la requête en question, génère le log adéquat, et la transmet au service destinataire.

Pour profiter de ce middleware, il faut également effectuer des modifications dans le fichier de configuration du service choisi.

De la même manière que pour les notifications, il est possible de définir le driver qui recevra les logs générés via le paramètre **driver**, dans la section **[audit_middleware_notifications]**. Il peut d'ailleurs s'agir d'un driver différent de celui utilisé pour les notifications si l'on souhaite séparer les logs. Il est également possible d'avoir plusieurs drivers de sortie pour un même log, il suffit de définir plusieurs fois le champ **driver**.

Il faut également effectuer des modifications dans le fichier **api-paste.ini**, de manière à y intégrer le filtre d'audit au sein du pipeline de traitement des requêtes.

Information

Pour plus d'informations concernant la configuration du module *Audit middleware*, il est possible de se référer à la documentation OpenStack [13].

A.2.1.3 Journaux HTTP

La plupart des services utilisent une API basée sur un serveur web Apache. Pour plus d'informations concernant le fonctionnement de ce serveur HTTP open source, se référer à la documentation officielle du projet.

Il n'y a pas de modification de configuration à effectuer pour utiliser ces journaux.

A.2.1.4 Journaux spécifiques à swift

Swift est un service à part en ce qui concerne le type de journalisation. Il ne met à disposition que des logs spécifiques issus du proxy avec lequel il s'interface.

/etc/swift/proxy-server.conf
[pipeline :main] pipeline = catch_errors gatekeeper healthcheck cache container_sync bulk tempurl ratelimit authtoken keystoneauth container_quotas account_quotas slo dlo proxy-logging proxy-server

Information

Pour plus d'informations concernant la configuration du module *proxy server* du service **swift**, il est possible de se référer à la documentation OpenStack [52].

Pour plus d'informations concernant la configuration du module *middleware*

> du service **swift**, il est possible de se référer à la documentation OpenStack [51].

A.2.1.5 Journaux spécifiques à horizon

L'interface web du service **horizon** possède un module WSGI Django, nommé **OperationLogMiddleware**, qui permet de générer des journaux. Pour activer ces journaux, il faut modifier les configurations du service **horizon**.

/etc/horizon/local_settings

```
OPERATION_LOG_ENABLE = True
OPERATION_LOG_OTIONS = {
  'mask_fields' : ['password', 'secret']
  'target_methods' : ['POST', 'GET', 'PUT', 'DELETE']
  'format' : ("[%(domain_name)s] [%(domain_id)s] [%(project_name)s]"
    " [%(project_id)s] [%(user_name)s] [%(user_id)s] [%(request_scheme)s]"
    " [%(referer_url)s] [%(request_url)s] [%(message)s] [%(method)s]"
    " [%(http_status)s] [%(param)s]")
}
```

A.2.2 Présentation du contenu des différents journaux

A.2.2.1 Notifications

Comme évoqué plus haut, les notifications sont au format CADF (*Cloud Audit Data Federation*). De plus, elles contiennent toujours un champ *event_type* spécifiant à quel type d'évènement correspond la log en question. Plus précisément, le champ *event_type* est composé de deux ou trois mots séparés par un point : une ressource suivie d'une action, elle-même parfois suivie des mots "*start*" ou "*end*" dans le cas d'opérations ayant un délai d'exécution.

Exemple

Voici un exemple de log de notification que l'on peut obtenir avec **Keystone** :

Keystone notification


```
{
  ”message_id” : ”d71546b5-263a-4d98-b82a-c6046a831b92”,
  ”publisher_id” : ”identity.openstack”,
  ”event_type” : ”identity.authenticate”,
  ”priority” : ”INFO”,
  ”payload” : {
    ”typeURI” : ”http ://schemas.dmtf.org/cloud/audit/1.0/event”,
    ”eventType” : ”activity”,
    ”id” : ”c6036c58-f7ee-5908-9f13-4fd8d6c4912c”,
    ”eventTime” : ”2022-02-02T07 :49 :53.854687+0000”,
    ”action” : ”authenticate”,
    ”outcome” : ”failure”,
    ”observer” : {
      ”id” : ”b4294d6a821f47f0a274c7e68c50b1e1”,
      ”typeURI” : ”service/security”
    },
    ”initiator” : {
      ”id” : ”8b06a55d-80b3-51ec-b158-2ffbf9c05b0d”,
      ”typeURI” : ”service/security/account/user”,
      ”host” : {
        ”address” : ”192.168.56.200”,
        ”agent” : ”openstack_auth keystoneauth1/4.4.0 python-requests/2.26.0 CPython/3.8.10”
      },
      ”request_id” : ”req-f9433773-c778-416d-84cd-c277440a74d7”,
      ”user_name” : ”adm”,
      ”domain_id” : ”default”,
      ”domain_name” : ”Default”
    },
    ”target” : {
      ”id” : ”a1086b28-aa95-55cb-9bab-37b35a10ca4f”,
      ”typeURI” : ”service/security/account/user”
    },
      ”reason” : {
      ”reasonType” : ”Could not find user : adm.”,
      ”reasonCode” : ”404”
    }
  },
  ”timestamp” : ”2022-02-02 07 :49 :53.916773”
}
```


Ici, la valeur ”*identity.authenticate*” du champ *event_type* correspond bien au champ attendu : ”*identity*” en est la ressource et ”*authenticate*” l'action. Pour plus d'explications concernant le contenu de ce log d'exemple, on peut se référer au tableau ci-dessous décrivant les champs importants :

Champ	Valeur	Explication
event_type	identity.authenticate	Signifie qu'il s'agit d'une requête d'authentification auprès de Keystone
outcome	failure	Indique que l'authentification a échoué
address	192.168.56.200	Adresse IP de l'émetteur de la requête
agent	openstack_auth keystoneauth1/4.4.0 python-requests/2.26.0 CPython/3.8.10	Nom de l'agent à l'origine de la requête
user_name	adm	Nom d'utilisateur fourni pour la demande d'authentification
reasonType	Could not find user : adm.	Raison expliquant le rejet de l'authentification
reasonCode	404	Code d'erreur du refus d'authentification

Principaux évènements à surveiller

Il existe un sous-ensemble des notifications qu'il est particulièrement important d'observer dans le cadre de la détection de compromission. Voici des évènements qu'il est conseillé de surveiller :

Service	Ressource	Action
Keystone	identity.authenticate	
	identity.application_credential	create
		delete
	identity.domain	create
		update
		delete
	identity.group	create
		update
		delete
	identity.access_token	create
		delete
	identity.consumer	create
		update
		delete
	identity.request_token	create
	identity.trust	create
		delete
	identity.endpoint	create
		update
		delete
	identity.policy	create
		update
		delete

Service	Ressource	Action
Keystone	identity.project	create
		update
		delete
	identity.region	create
		update
		delete
	identity.role	create
		update
		delete
	identity.service	create
		update
		delete
	identity.user	create
		update
		delete
	identity.role_assignment	add
		remove
Nova	flavor	create
		update
		delete
	compute.instance	create
		create_ip
		delete
		delete_ip
		pause
		poweroff
		power_on
		reboot
		rebuild
		rescue
		resize
		resize.revert
		restore
		resume
		shutdown
		snapshot
		soft_delete
		suspend
		unpause
		unrescue
		update
		volume_attach
		volume_detach
		volume_swap

Service	Ressource	Action
Nova	keypair	create
		delete
		import
	servergroup	addmember
		create
Neutron	floatingip	create
		update
		delete
	network	create
		update
		delete
	port	create
		update
		delete
	port_association	create
		delete
	port_forwarding	creztr
		delete
	rbac_policy	create
		delete
	router	create
		update
		delete
	router.interface	create
		delete
	security_group	create
		update
		delete
	security_group_rule	create
		delete
	subnet	create
		update
		delete
	subnetpool	create
		update
		delete
	group	create
		update
		delete
		disable_replication
		enable_replication
		failover_replication
	group_snapshot	create
		delete
		reset_status

Service	Ressource	Action
Neutron	scheduler	create_volume
		migrate_volume_to_host
	snapshot	create
		update
		delete
		reset_status
		revert
	volume	create
		update
		delete
		resize
		attach
		detach
		retype
		reset_status
		revert
		manage_existing
	volume.transfer	create
		delete
		accept
	volume_type_encryption	create
		update
Glance	image	activate
		create
		delete
		prepare
		send
		update
		upload
	image.member	create
		update
		delete

A.2.2.2 Audit *Middleware*

Ces logs ont un format similaire aux notifications présentées précédemment étant donné qu'ils sont également au format CADF (*Cloud Audit Data Federation*) et possèdent aussi un champ "*event_type*" pour spécifier le type d'évènement. En revanche, puisque ces logs correspondent aux requêtes envoyées aux différents services, le champ "*event_type*" ne prend que deux valeurs : "*audit.http.request*" dans le cas d'une requête initiée vers un service, et "*audit.http.response*" dans le cas d'une réponse de celui-ci.

Il existe également un champ *action* dont les valeurs indiquent si la requête entraîne une modification de données du côté du service :

Valeur	Signification
read	Requête associée à une demande d'information pour une ressource
read/list	Requête associée à une demande d'information concernant un ensemble de ressources (par exemple la liste des serveurs)
update	Requête qui met à jour des informations concernant une ressource
write	Requête qui crée une ressource

Exemple

Voici un exemple de logs obtenus lors d'une requête à destination du service **Nova** :

Audit middleware log

```
2022-01-24 09 : 33 : 45.873 19 INFO oslo.messaging.notification.audit.http.request
[req-db9d7dfb-71d5-492e-818a-889f4ddb0396 17228c0a0be5423a850bd49b9356b5df
622053d249724e539ddabedb8714c0d9 - default default
] {
  "message_id" : "33aa497e-2579-4b3d-a34d-1b660f163f4d",
  "publisher_id" : "mod_wsgi",
  "event_type" : "audit.http.request",
  "priority" : "INFO",
  "payload" : {
    "typeURI" : "http ://schemas.dmtf.org/cloud/audit/1.0/event",
    "eventType" : "activity",
    "id" : "ee2d4cc2-0665-56e8-bea1-418edec17051",
    "eventTime" : "2022-01-24T09 :33 :45.872824+0000",
    "action" : "read/list",
    "outcome" : "pending",
    "observer" : {
      "id" : "target"
    },
    "initiator" : {
      "id" : "17228c0a0be5423a850bd49b9356b5df",
      "typeURI" : "service/security/account/user",
      "name" : "admin",
      "credential" : {
        "token" : "***",
        "identity_status" : "Confirmed"
      },
      "host" : {
        "address" : "192.168.56.1",
        "agent" : "python-novaclient"
      },
      "project_id" : "622053d249724e539ddabedb8714c0d9",
      "request_id" : "req-db9d7dfb-71d5-492e-818a-889f4ddb0396"
    },
```

Audit middleware log


```
    "target" : {
      "id" : "nova_legacy",
      "typeURI" : "unknown",
      "name" : "nova_legacy",
      "addresses" : [
        {
          "url" : "http ://192.168.56.200 :8774/v2/622053d249724e539ddabedb8714c0d9",
          "name" : "admin"
        },
        {
          "url" : "http ://192.168.56.200 :8774/v2/622053d249724e539ddabedb8714c0d9",
          "name" : "private"
        },
        {
          "url" : "http ://192.168.56.200 :8774/v2/622053d249724e539ddabedb8714c0d9",
          "name" : "public"
        }
      ]
    },
    "requestPath" : "/v2.1/servers/detail",
    "tags" : [
      "correlation_id ?value=86357bc7-ee0e-52d1-a4a8-49de79d9b943"
    ]
  },
  "timestamp" : "2022-01-24 09 :33 :45.873437"
}

2022-01-24 09 : 33 : 46.047 19 INFO oslo.messaging.notification.audit.http.response
[req-db9d7dfb-71d5-492e-818a-889f4ddb0396 17228c0a0be5423a850bd49b9356b5df
622053d249724e539ddabedb8714c0d9 - default default
] {
  "message_id" : "3dcb8033-29d8-426c-a827-f61755d1d6c7",
  "publisher_id" : "mod_wsgi",
  "event_type" : "audit.http.response",
  "priority" : "INFO",
  "payload" : {
    "typeURI" : "http ://schemas.dmtf.org/cloud/audit/1.0/event",
    "eventType" : "activity",
    "id" : "ee2d4cc2-0665-56e8-bea1-418edec17051",
    "eventTime" : "2022-01-24T09 :33 :45.872824+0000",
    "action" : "read/list",
    "outcome" : "success",
    "observer" : {
      "id" : "target"
    },
    "initiator" : {
```

Audit middleware log


```
    "id" : "17228c0a0be5423a850bd49b9356b5df",
    "typeURI" : "service/security/account/user",
    "name" : "admin",
    "credential" : {
      "token" : "***",
      "identity_status" : "Confirmed"
    },
    "host" : {
      "address" : "192.168.56.1",
      "agent" : "python-novaclient"
    },
    "project_id" : "622053d249724e539ddabedb8714c0d9",
    "request_id" : "req-db9d7dfb-71d5-492e-818a-889f4ddb0396"
  },
  "target" : {
    "id" : "nova_legacy",
    "typeURI" : "unknown",
    "name" : "nova_legacy",
    "addresses" : [
      {
        "url" : "http ://192.168.56.200 :8774/v2/622053d249724e539ddabedb8714c0d9",
        "name" : "admin"
      },
      {
        "url" : "http ://192.168.56.200 :8774/v2/622053d249724e539ddabedb8714c0d9",
        "name" : "private"
      },
      {
        "url" : "http ://192.168.56.200 :8774/v2/622053d249724e539ddabedb8714c0d9",
        "name" : "public"
      }
    ]
  },
  "requestPath" : "/v2.1/servers/detail",
  "tags" : [
    "correlation_id ?value=86357bc7-ee0e-52d1-a4a8-49de79d9b943"
  ],
  "reason" : {
    "reasonType" : "HTTP",
    "reasonCode" : "200"
  },
  "reporterchain" : [
    {
      "role" : "modifier",
      "reporterTime" : "2022-01-24T09 :33 :46.046849+0000",
      "reporter" : {
```

Audit middleware log

```
            "id" : "target"
          }
        }
      ]
    },
    "timestamp" : "2022-01-24 09 :33 :46.047173"
}
```

On peut constater qu'il y a eu génération des deux types d'évènements décrits plus haut : *audit.http.request* et *audit.http.response*. On peut analyser les champs les plus importants pour extraire de ces logs de l'information :

Champ	**Valeur**	**Explication**
action	read/list	Il s'agit d'une requête qui demande une information, mais n'effectue pas de modifications
outcome	pending puis success	La requête était au status pending le temps d'être traitée, puis a été acceptée
initiator. typeURI	service/security/ account/user	Signifie que l'expéditeur de la requête est un utilisateur
initiator.name	admin	Signifie que l'expéditeur de la requête a pour nom "admin"
initiator.id	17228c0a0be5423a 850bd49b9356b5df	Identifiant de l'expéditeur de la requête, donc ici de l'utilisateur "admin"
initiator. project_id	622053d249724e53 9ddabedb8714c0d9	Identifiant du projet (au sens Openstack) utilisé par l'utilisateur
initiator. request_id	req-db9d7dfb- 71d5 -492e-818a- 889f4ddb0396	Identifiant de la requête
initiator. credential. identity_status	Confirmed	Signifie que le token d'authentification a bien été validé par Keystone
initiator. host. address	192.168.56.1	IP de l'expéditeur
initiator. host. agent	python-novaclient	Agent utilisé par l'expéditeur
initiator. request_id	req-db9d7dfb- 71d5 -492e-818a- 889f4ddb0396	Identifiant unique par requête permettant d'associer le log d'émission de requête avec celui de la réponse
target.name	nova_legacy	Nom du destinataire de la requête. Ici il s'agit bien du service Nova.

Champ	Valeur	Explication
target.addresses	Plusieurs URLs avec des noms associés différents	Chacune de ces URLs correspond à l'endpoint de destination de la requête en fonction du niveau d'accès requis (admin, private, public)
requestPath	/v2.1/servers /detail	Endpoint de l'API destinataire. Ici, cela signifie que la requête demande la liste des détails pour chacun des serveurs.
reasonCode	200	Corresponds au code HTTP de retour de l'API. Ici, la valeur 200 nous confirme que la requête a bien été traitée.

Information

Pour des informations complémentaires concernant les logs *Audit middleware*, il est possible de se référer à la documentation OpenStack [47].

Il est également possible d'obtenir plus d'informations sur les différents *endpoints* des API OpenStack en consultant la documentation de l'API correspondante [10]. Par exemple, pour l'API Novaelle se trouve ici [11]. Ceci permettant de comprendre, à l'aide du champ **requestPath**, quelle est l'action demandée par la requête.

A.2.2.3 Journaux HTTP

Comme expliqué plus haut, les services proposant des API utilisent un serveur Apache. Il est donc également possible d'accéder à ces logs plus classiques. Le format par défaut est le suivant :

Format par défaut des journaux Apache
"%{X-Forwarded-For}i %l %u %t \"%r\" %>s %b %D \"%{Referer}i\" \"%{User-Agent}i\"" logformat

Information

Il est possible de se référer à la documentation officielle pour connaître les différents types de format string supportés [24].

Ces logs contiennent des informations sur les requêtes effectuées sur les API en question, et peuvent également permettre de suivre les actions des utilisateurs.

Exemple

Les logs Apache suivants ont été générés par le service **Horizon** et permettent d'observer les différentes requêtes émises aux autres services :

Journaux HTTP du service Horizon
- - - [01/Feb/2022 :13 :42 :51 +0000] ”GET /api/glance/images/?sort_dir=asc&sort_key=name&status=active HTTP/1.1” 200 449 38797 ”http ://192.168.56.102/project/instances/” ”Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv :96.0) Gecko/20100101 Firefox/96.0”
- - - [01/Feb/2022 :13 :42 :51 +0000] ”GET /api/cinder/volumes/?bootable=1&status=available HTTP/1.1” 200 70 79199 ”http ://192.168.56.102/project/instances/” ”Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv :96.0) Gecko/20100101 Firefox/96.0”
- - - [01/Feb/2022 :13 :42 :51 +0000] ”GET /api/neutron/networks/ HTTP/1.1” 200 755 188837 ”http ://192.168.56.102/project/instances/” ”Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv :96.0) Gecko/20100101 Firefox/96.0”

On peut constater ici que le service **Horizon** a d'abord demandé au service **Glance** la liste des images actives. Puis, la liste des volumes disponibles au service **Cinder**, avant de demander la liste des réseaux au service **Neutron**

A.2.2.4 Journaux spécifiques à swift

Swift permet de générer des logs d'une manière qui lui est propre par le biais de son serveur proxy. Le format par défaut des logs est le suivant :

Format par défaut des journaux Swift
{client_ip} {remote_addr} {end_time.datetime} {method} {path} {protocol} {status_int} {referer} {user_agent} {auth_token} {bytes_recvd} {bytes_sent} {client_etag} {transaction_id} {headers} {request_time} {source} {log_info} {start_time} {end_time} {policy_index}

Il est recommandé d'ajouter les champs *account*, *container* et *object* dans le format des journaux générés, pour avec les informations disponibles sans traitement supplémentaire.

Information

Il est possible d'ajuster le format des logs grâce aux différentes variables existantes et documentées par OpenStack [50].

Exemple

Journaux du proxy du service Swift
192.168.56.200 192.168.56.102 09/Mar/2022/13/26/16 PUT /v1/AUTH_622053d249724e539ddabedb8714c0d9/Swift_ Container HTTP/1.0 201 - python-swiftclient-3.12.0 gAAAAABiKKrs9GsG... -tx72cd4c556362491ea9334-006228aaf8 - - - 0 AUTH_622053d249724e539ddabedb8714c0d9 Swift_Container - 201

Journaux du proxy du service Swift
192.168.56.200 192.168.56.102 09/Mar/2022/13/26/18 GET /v1/AUTH_622053d249724e539ddabedb8714c0d9/Swift_Container%3Fformat%3Djson%26limit%3D1001%26delimiter%3D%252F%26states%3Dlisting HTTP/1.0 200 - pythonswiftclient-3.12.0 gAAAAABiKKrs9GsG... - txe0e3d684a3c54d4d89389-006228aafa - - - 0 AUTH_622053d249724e539ddabedb8714c0d9 Swift_Container - 200 192.168.56.200 192.168.56.102 09/Mar/2022/13/26/43 PUT /v1/AUTH_622053d249724e539ddabedb8714c0d9/Swift_Container/Testfileupload.png HTTP/1.0 201 - python-swiftclient-3.12.0 gAAAAABiKKrs9GsG... - txbf78886a819e4d91bd271-006228ab13 - - - 0 AUTH_622053d249724e539ddabedb8714c0d9 Swift_Container Testfileupload.png 201 192.168.56.200 192.168.56.102 09/Mar/2022/13/27/14 GET /v1/AUTH_622053d249724e539ddabedb8714c0d9/Swift_Container/Testfileupload.png HTTP/1.0 200 - python-swiftclient-3.12.0 gAAAAABiKKrs9GsG... - tx26b627e1b9f94e99bdd18-006228ab32 - - - 0 AUTH_622053d249724e539ddabedb8714c0d9 Swift_Container Testfileupload.png 200

A.2.2.5 Journaux spécifiques à horizon

Comme expliqué plus haut, l'interface web du service **Horizon** met à disposition des journaux spécifiques par le biais d'un middleware. Les journaux générés contiennent un ensemble de champs de contexte, notamment l'heure de génération, et des informations sur l'action effectuée. Cette seconde partie suit un format dont la valeur par défaut est la suivante :

Format par défaut des champs de contexte Horizon
[%(domain_name)s] [%(domain_id)s] [%(project_name)s] [%(project_id)s] [%(user_name)s] [%(user_id)s] [%(request_scheme)s] [%(referer_url)s] [%(request_url)s] [%(message)s] [%(method)s] [%(http_status)s] [%(param)s]

Information

Les variables utilisées ici sont les seules disponibles. Pour en savoir plus, il est possible de se référer au code source [17].

Exemple

Voici un exemple de logs que l'on peut obtenir :

Journaux du service Horizon

```
[Tue Feb 01 14:03:59.741325 2022] [wsgi:error] [pid 25:tid 140133369485056]
 [remote 192.168.56.102:50286] [None] [None] [admin]
 [622053d249724e539ddabedb8714c0d9] [admin] [17228c0a0be5423a850bd49b9356b5df]
 [http] [/auth/login/] [/auth/login/] [None] [POST] [302]
 [{"csrfmiddlewaretoken":
 "T5KUdTGHvbm3Q7r3v2xTMECzN3Ifjf3TdyGqwdkzslQOLtL8KueJLmrXBfjyD1wu",
 "fake_email": "admin", "fake_password":
 "dfpLUD6jQY0iZIGbsgsUUoIcBcdi3JseEyHqeR6o",
 "region": "default", "username": "admin", "password": "********"}]
[Tue Feb 01 14:03:59.748525 2022] [wsgi:error] [pid 26:tid 140133369485056]
 [remote 192.168.56.102:50286] [None] [None] [admin]
 [622053d249724e539ddabedb8714c0d9] [admin] [17228c0a0be5423a850bd49b9356b5df]
 [http] [/auth/login/] [/] [None] [GET] [302] [{}]
[Tue Feb 01 14:04:00.104671 2022] [wsgi:error] [pid 27:tid 140133369485056]
 [remote 192.168.56.102:50286] [None] [None] [admin]
 [622053d249724e539ddabedb8714c0d9] [admin] [17228c0a0be5423a850bd49b9356b5df]
 [http] [/auth/login/] [/project/] [None] [GET] [200] [{}]
[Tue Feb 01 14:04:00.492432 2022] [wsgi:error] [pid 24:tid 140133369485056]
 [remote 192.168.56.102:50500] [None] [None] [admin]
 [622053d249724e539ddabedb8714c0d9] [admin] [17228c0a0be5423a850bd49b9356b5df]
 [http] [/project/] [/header/] [None] [GET] [200] [{}]
```

Ici il est possible d'observer ici que l'utilisateur "admin" s'est connecté, puis a navigué ves les *end-points* "project" et "header".

Liste des recommandations

Bibliographie

[1] *Recommandations pour un usage sécurisé d'(Open)SSH*.
Note technique DAT-NT-007/ANSSI/SDE/NP v1.2, ANSSI, août 2015.
https://cyber.gouv.fr/guide-ssh.

[2] *Recommandations de sécurité relatives à un système GNU/Linux*.
Guide ANSSI-BP-028 v2.0, ANSSI, octobre 2022.
https://cyber.gouv.fr/guide-linux.

[3] *Recommandations de sécurité relatives à IPsec pour la protection des flux réseau*.
Note technique DAT-NT-003/ANSSI/SDE/NP v1.1, ANSSI, août 2015.
https://cyber.gouv.fr/guide-ipsec.

[4] *Recommandations de sécurité relatives à TLS*.
Guide ANSSI-PA-035 v1.2, ANSSI, mars 2020.
https://cyber.gouv.fr/guide-tls.

[5] *Recommandations de sécurité pour l'architecture d'un système de journalisation*.
Guide DAT-PA-012 v2.0, ANSSI, janvier 2022.
https://cyber.gouv.fr/guide-journalisation.

[6] *RGS Annexe B2 : Règles et recommandations concernant la gestion des clés utilisées dans les mécanismes cryptographiques*.
Référentiel Version 2.0, ANSSI, juin 2012.
https://cyber.gouv.fr/rgs.

[7] *RGS Annexe B1 : Règles et recommandations concernant le choix et le dimensionnement des mécanismes cryptographiques*.
Référentiel Version 2.03, ANSSI, février 2014.
https://cyber.gouv.fr/rgs.

[8] *Authentification multifacteurs et mots de passe*.
Guide ANSSI-PG-078 v1.0, ANSSI, octobre 2021.
https://cyber.gouv.fr/guide-authentification.

[9] *Prestataires de services d'informatique en nuage (SecNumCloud). Référentiel d'exigences*.
Référentiel Version 3.2, ANSSI, mars 2022.
https://cyber.gouv.fr/secnumcloud.

[10] *Openstack - API reference*, 2022.
https://docs.openstack.org/api-ref.

[11] *Openstack - Compute API*, 2022.
https://docs.openstack.org/api-ref/compute/.

[12] Ahmed Atya, Azeem Aqil, Karim Khalil, Zhiyun Qian, Srikanth V. Krishnamurthy, and Thomas F. La Porta.
Stalling Live Migrations on the Cloud.
In *11th USENIX Workshop on Offensive Technologies (WOOT 17)*, Vancouver, BC, août 2017.

USENIX Association.
https://www.usenix.org/conference/woot17/workshop-program/presentation/atya.

[13] *Swift - middleware*, 2019.
https://docs.openstack.org/swift/latest/middleware.html.

[14] *Cloud Auditing Data Federation*, 2022.
https://www.dmtf.org/standards/cadf.

[15] *OpenStack - Keystone - Identity API V2.0 and V3 history*.
OpenStack Foundation.
Technical report, OpenStack Foundation, 2020.
https://docs.openstack.org/keystone/pike/contributor/http-api.html.

[16] *OpenStack Foundation - Reviews by company*.
OpenStack Foundation.
Technical report, OpenStack Foundation, 2020.
https://www.stackalytics.com/.

[17] *horizon.middleware.operation_log - horizon*, 2022.
https://docs.openstack.org/horizon/latest/_modules/horizon/middleware/operation_log.html.

[18] *Kolla-Ansible - Advanced Configuration - Endpoint Network Configuration*, 2020.
https://docs.openstack.org/kolla-ansible/latest/admin/advanced-configuration.html.

[19] *Keystone administration - Identity caching layer*, 2020.
https://docs.openstack.org/keystone/pike/admin/identity-caching-layer.html.

[20] *Keystone - Domain-specific configuration*, 2019.
https://docs.openstack.org/keystone/latest/admin/configuration.html.

[21] *OpenStack - Configuring Keystone for Federation*, 2020.
https://docs.openstack.org/keystone/latest/admin/federation/configure_federation.html.

[22] *Keystone - Integrate Identity with LDAP*, 2020.
https://docs.openstack.org/keystone/pike/admin/identity-integrate-with-ldap.html.

[23] *Mariadb - Securing Connections for Client and Server*, 2020.
https://mariadb.com/kb/en/securing-connections-for-client-and-server/#requiring-tls.

[24] *mod_log_config - Serveur HTTP Apache Version 2.4*, 2022.
https://httpd.apache.org/docs/2.4/mod/mod_log_config.html#formats.

[25] *OpenStack configuration - Compute schedulers*, 2019.
https://docs.openstack.org/newton/config-reference/compute/schedulers.html.

[26] *Keystone - identity specs - shadow users*.
Openstack.
Technical report, Openstack, 2016.

https://specs.openstack.org/openstack/keystone-specs/specs/keystone/mitaka/shadow-users.html.

[27] *Glance - Blueprints - Encrypt image data for image data security*.
OpenStack.
Technical report, OpenStack, 2018.
https://blueprints.launchpad.net/glance/+spec/encrypt-image-data.

[28] *Swift - Storage policies*.
Openstack.
Technical report, Openstack, 2019.
https://docs.openstack.org/swift/latest/overview_policies.html.

[29] *OpenStack Introduction to Keystone Federation*.
OpenStack.
Technical report, OpenStack, 2019.
https://docs.openstack.org/keystone/latest/admin/federation/introduction.html.

[30] *OpenStack Fernet - Frequently Asked Questions*.
OpenStack.
Technical report, OpenStack, 2020.
https://docs.openstack.org/keystone/pike/admin/identity-fernet-token-faq.html.

[31] *OpenStack Identity concepts*.
OpenStack.
Technical report, OpenStack, 2020.
https://docs.openstack.org/keystone/pike/admin/identity-concepts.html.

[32] *OpenStack Policies*.
OpenStack.
Technical report, OpenStack, 2020.
https://docs.openstack.org/security-guide/identity/policies.html.

[33] *OpenStack Services*.
OpenStack.
Technical report, OpenStack, 2020.
https://www.openstack.org/software/project-navigator/openstack-components#openstack-services.

[34] *Encrypt Open vSwitch Tunnels with IPsec*.
OpenvSwitch.
Technical report, OpenvSwitch, 2019.
https://docs.openvswitch.org/en/latest/howto/ipsec/.

[35] *OpenStack - Python OpenStack client - Application Credentials*, 2020.
https://docs.openstack.org/python-openstackclient/latest/cli/command-objects/application-credentials.html.

[36] *Cinder - Configuration - Additional options*, 2020.
https://docs.openstack.org/cinder/rocky/configuration/block-storage/config-options.html.

[37] *Cinder - Administration - Volume-backed image*, 2017.
https://docs.openstack.org/cinder/latest/admin/blockstorage-volume-backed-image.html.

[38] *OpenStack - Security Guide - Data encryption*, 2020.
https://docs.openstack.org/security-guide/tenant-data/data-encryption.html.

[39] *Oslo Messaging - Configuration options*, 2017.
https://docs.openstack.org/oslo.messaging/latest/configuration/opts.html.

[40] *Nova - Administration - Configure live migrations*, 2019.
https://docs.openstack.org/nova/latest/admin/configuring-migrations.html.

[41] *Nova - Administration - Secure live migration with QEMU-native TLS*, 2019.
https://docs.openstack.org/nova/latest/admin/secure-live-migration-with-qemu-native-tls.html.

[42] *OpenStack - Security Guide - Database transport security*, 2020.
https://docs.openstack.org/security-guide/databases/database-transport-security.html.

[43] *OpenStack - Security Guide - Messaging security*, 2020.
https://docs.openstack.org/security-guide/messaging/security.html.

[44] *OpenStack - Security Guide - Secrets management use cases*, 2020.
https://docs.openstack.org/security-guide/secrets-management/secrets-management-use-cases.html.

[45] *OpenStack - Security Guide - API endpoint configuration recommandations*, 2020.
https://docs.openstack.org/security-guide/api-endpoints/api-endpoint-configuration-recommendations.html.

[46] *OpenStack - Security Guide - Introduction to TLS and SSL*, 2020.
https://docs.openstack.org/security-guide/secure-communication/introduction-to-ssl-and-tls.html.

[47] *Openstack - pyCADF events*, 2017.
https://docs.openstack.org/pycadf/latest/event_concept.html.

[48] *Produits certifiés CC - HSM TrustWay Proteccio – Version V128/X130*.
Amossys Serma Technologies.
Technical report, ANSSI, 2016.
https://cyber.gouv.fr/produits-certifies/hsm-trustway-proteccio-version-v128x130.

[49] *Swift - Data encryption*, 2020.
https://docs.openstack.org/swift/latest/overview_encryption.html.

[50] *Logs - Swift documentation*, 2022.
https://docs.openstack.org/swift/latest/logs.html.

[51] *Audit middleware - keystonemiddleware*, 2016.
https://docs.openstack.org/keystonemiddleware/latest/audit.html.

[52] *Swift - Proxy Server Configuration*, 2020.
https://docs.openstack.org/swift/latest/config/proxy_server_config.html#proxy-server.

[53] *Multi-Tenancy Security in Cloud Computing - Edge Computing and Distributed Cloud*.
Ali Shokrollahi Yancheshmeh.
Technical report, KTH Royal Institute Of Technology- Stockholm, Sweden, 2019.
http://kth.diva-portal.org/smash/get/diva2:1412847/FULLTEXT01.pdf.

Version 1.0 - 16/04/2024 - ANSSI-BP-104

ISBN : 978-2-11-167166-9 (papier)
ISBN : 978-2-11-167167-6 (numérique)
Dépôt légal : avril 2024

AGENCE NATIONALE DE LA SÉCURITÉ DES SYSTÈMES D'INFORMATION

ANSSI - 51 boulevard de La Tour-Maubourg, 75700 PARIS 07 SP
cyber.gouv.fr / conseil.technique@ssi.gouv.fr

RÉPUBLIQUE FRANÇAISE
Liberté
Égalité
Fraternité