ESCUELA TÉCNICA SUPERIOR DE INGENIEROS INDUSTRIALES Y DE TELECOMUNICACIÓN

UNIVERSIDAD DE CANTABRIA

Trabajo Fin de Grado

Estudio de las técnicas Forenses en el entorno Cloud: Implementación de funcionalidades sobre una maqueta de pruebas

(Study of Forensic techniques in the Cloud environment: Implementation of functionalities on a model of tests)

Para acceder al Titulo de

Graduado en Ingeniería de Tecnologías de Telecomunicación

Autor: Pablo Peris Barreda

Julio - 2019

Agradecimientos

A Jose Ángel y Alberto Eloy por su paciencia, dedicación y ayuda con la elaboración de este trabajo. A mis otros profesores del grado por su ayuda incondicional que sin la cual no hubiese llegado hasta aquí. A Fede que me ayudó con la expresión y a mi familia y amigos que me han empujado a seguir adelante. En conclusión gracias a todas las personas que me apoyaron en todo momento.

Resumen

Las técnicas forenses en entornos cloud son el estudio analítico de la estructura de un sistema de computación en la nube con el fin de, tras la ocurrencia de un error, encontrar las trazas que llevaron al sistema a ese punto. La infraestructura de computación en la nube es un cambio de paradigma en la computación que conlleva grandes retos para los investigadores debido, principalmente, a las características que hacen atractiva a esta tecnología. Sabiendo que este tipo de entorno es problemático, especialmente para la tecnología tradicional forense, en primer lugar se estudia este tipo de entono, la tecnología forense tradicional y los problemas y soluciones que se presentan en este cambio de paradigma, que es la computación en la nube. Posteriormente se crea un entorno de pruebas en el que se estudian dos programas para la detección de ciberataques a un entorno de DevStack; además, también se estudia la capacidad de generar registros de eventos en este tipo de entornos. Finalmente se analizan las evidencias generadas desde el entono, en el caso de uno de los programas no se llega a probar la configuración inicial que se pretendía, el resto de las pruebas resultan satisfactorias. Se concuye que este tipo de entornos aun necesitan mucho trabajo en el diseño e implementación de muchos tipos herramientas forenses que se utilizan en entornos tradicionales y necesitan su adaptación al cloud; además se enumeran una serie de otros posibles entornos de pruebas para las técnicas forenses en la computación en la nube.

Abstract

Forensic techniques in cloud environments are the analytical study of the structure of a cloud computing system in order to, after the occurrence of an error, find the traces that led the system to that point. The infrastructure of cloud computing is a paradigm shift in computing that entails great challenges for researchers due, mainly, to the characteristics that make this technology attractive. Knowing that this type of environment is problematic, especially for traditional forensic technology, first of all this type of environment is studied, the traditional forensic technology and the problems and solutions that arise in this paradigm shift, which is cloud computing. Subsequently, a test environment is created in which two programs for the detection of cyber attacks to a DevStack environment are studied; In addition, the ability to generate event records in this type of environment is also studied. Finally, the evidence generated from the environment is analyzed. In the case of one of the programs, it is not possible to prove the initial configuration that was intended, the rest of the tests are satisfactory. It is assumed that this type of environments still need a lot of work in the design and implementation of many types of forensic tools that are used in traditional environments and need their adaptation to the cloud; A number of other possible testing environments for forensic techniques in cloud computing are listed.

Índice general

Agradecimientos **I**

Resumen **III**

Abstract **V**

Índice general **VII**

Índice de figuras **IX**

1. Introducción **1**
1.1. Motivación y objetivos del proyecto 2
1.2. Organización del documento 2

2. Conceptos teóricos y Estado del arte **5**
2.1. El entorno cloud 5
2.1.1. Virtualización 6
2.1.2. Modelos de despliegue 7
2.1.3. Modelos de servicio 8
2.1.4. Amenazas a los entornos cloud 9
2.1.5. Soluciones de creación de entornos cloud 11
2.2. Técnicas forenses digitales 20
2.2.1. Modelos de técnicas forenses digitales 20
2.2.2. Tipos de herramientas forenses 21
2.3. Análisis forense en entornos cloud 23
2.3.1. Usos de las técnicas forenses en entornos cloud 23
2.3.2. Retos y soluciones de las técnicas forenses en entornos cloud 24
2.3.3. Herramientas forenses en entornos cloud 29

3. Implemetación práctica **33**
3.1. Descripción del demostrador 33
3.1.1. Hardware e hipervisor 33
3.1.2. Sistema operativo y entorno 35
3.1.3. Programas forenses 38
3.1.4. Actores implicados 42
3.2. Preparación del demostrador 43
3.2.1. Especificaciones de la máquina virtual 43
3.2.2. Instalación del sistema 43
3.2.3. Preparación de los complementos 45
3.3. Generación de evidencias 54
3.3.1. Ataque MitM en una red virtual 54
3.3.2. Generación de alarmas en Monasca 57
3.4. Detección, preservación y análisis de evidencias 57
3.4.1. Evidencias en Congress 58
3.4.2. Evidencias en Monasca 59
3.4.3. Evidencias en los registros 60

4. Conclusiones y líneas futuras **63**

Bibliografía **65**

Apéndice I - local.conf **71**

Apéndice II - Archivo rc (admin-openrc.sh) **73**

Apéndice III - politica.yaml **75**

Apéndice IV - Log de Nova **79**

Apéndice V - Log de Neutron **83**

Apéndice VI - Log de Keystone **85**

Índice de figuras

2.1. Comparación de los hipervisores nativos y hosted. 7
2.2. Responsabilidad según el modelo de servicio. 8
2.3. Definiciones de la computación en la nube. 9
2.4. Ejemplo de espionaje de tráfico. 9
2.5. Ejemplo de un ataque de intermediario malicioso. 10
2.6. Ejemplo de denegación de servicio. 10
2.7. Ejemplo de acceso con autorización insuficiente. 11
2.8. Servicios opcionales OpenStack. 14
2.9. Infraestuctura de CloudStack. 15
2.10. Infraestructura de OpenNebula. 16
2.11. Infraestructura lógica de Eucalyptus. 17
2.12. Diagrama de flujo del proceso forense. 21

3.1. Memoria máxima recomendada para virtualización 34
3.2. Máximo de núcleos recomendados para virtualización 34
3.3. Diagrama lógico de red del entorno 35
3.4. Aplicación de las técnicas forenses 36
3.5. Infraestructura lógica de OpenStack 37
3.6. Infraestructura de monasca 39
3.7. Dashboard de Grafana 40
3.8. Resultado de la instalación de OpenStack mediante DevStack 46
3.9. Red del usuario malintencionado 47
3.10. Red del usuario bienintencionado 48
3.11. Principio de la lista de métricas que, por defecto, se obtienen tras la instalación del sistema 51
3.12. Métricas recogidas por Ceilometer y enviadas a Monasca . . . 52
3.13. Definición de una alarma sobre la métrica *test* 53
3.14. Definición de una notificación en Monasca 53
3.15. La red de usuarios bienintencionados se encuentra en modo compartido 54
3.16. La red compartida se ve desde otros tennats 55

3.17. El usuario malicioso despliega una máquina en una red de otro proyecto 56
3.18. Máquina desplegada por el usuario malicioso 56
3.19. Alarma generada en el dashboard de Congress 58
3.20. Registro de creación en Horizon 59
3.21. Alarma activada 60
3.22. Historial de los estados de la alarma 60

1. Introducción

Las palabras "en la nube"se han convertido en parte de la jerga habitual y cada vez se utilizan más, pero en muchos casos se desconoce realmente a qué se refiere este concepto y aun menos como funciona realmente la computación en la nube.

El término nube (cloud) tiene un comienzo difuso que se sitúan en la década de 1960 [1], siendo esta una idea muy poco madura sobre lo que hoy en día se conoce como la nube. Un concepto más aproximado de los comienzos del cloud, como actualmente se entiende, se puede situar en 2006, en la conferencia sobre el motor de búsqueda de Google: *"What's interesting [now] is that there is an emergent new model, and you all are here because you are part of that new model. I don't think people have really understood how big this opportunity really is. It starts with the premise that the data services and architecture should be on servers. We call it cloud computing – they should be in a "cloud"somewhere."*[1][2] En ese mismo mes, 25 días más tarde, Amazon lanza la beta de Amazon EC2 (Amazon Elastic Compute Cloud) [3]. Con las ideas expuestas por Google y la beta de la nube de Amazon se comienza a atisbar el cambio de paradigma de la computación.

Lo que hoy se conoce como cloud computing es básicamente una cantidad ilimitada de recursos de los que un usuario o una compañía puede disponer de manera flexible pagando únicamente por aquellos que utilice durante el tiempo que desee. Esto se realiza mediante la creación y eliminación de máquinas virtuales en grandes datacenters[2] controlados por hipervisores[4].

[1] "Lo que es interesante [ahora] es que hay un nuevo modelo emergente, y todos vosotros estáis aquí por que sois parte de ese nuevo modelo. No creo que la gente haya entendido realmente cómo de grande es esta oportunidad. Comienza con la premisa de que la arquitectura y los servicios de datos se deberían encontrar en los servidores. Lo llamamos computación en la nube – deberían estar en una "nube", en alguna parte"

[2] También conocidos como Centros de Procesamiento de Datos, o por sus siglas CPD, es un espacio donde las compañías almacenan los recursos informáticos necesarios para el procesamiento de sus datos

Pero, como toda tecnología, está sujeta a errores y dichos errores son un punto de acceso para que los usuarios malintencionados puedan obtener un beneficio ilegítimo[5]. Es en este punto donde la seguridad en la nube entra en juego, evitando que parte de esas acciones malintencionadas se lleven a cabo o, al menos, que el hecho en sí sea registrado y se pueda reconstruir el "escenario del crimen", para descubrir al atacante. Esta última parte es lo que se conoce como técnicas forenses digitales, que es el principal punto de estudio de este documento.

1.1. Motivación y objetivos del proyecto

Las técnicas forenses digitales son el estudio analítico de una infraestructura tecnológica. Las tecnologías de computación en el cloud aún son jóvenes, empresas y usuarios están comenzando a migrar a la nube sus procesos menos sensibles, pero la tendencia hacia la migración completa pasa por crear un entorno más seguro para dichos procesos.

Para conseguir un entorno más seguro es necesaria la implementación de una estructura de análisis forense y, aunque en la actualidad no existe una cantidad demasiado amplia de herramientas para llevar a cabo este tipo de estudio, va apareciendo paulatinamente nuevo software forense.

El objetivo del trabajo es el estudio de estas herramientas, sus capacidades y limitaciones e incluso si existe la manera de adaptar software forense no diseñado específicamente para la computación en la nube al análisis de este entorno de una manera escalable. Para, luego, implementar una simulación en la que se utilizaría este software forense en un entorno controlado sobre el que se realizarían ciertos ataques de red.

1.2. Organización del documento

El trabajo está dividido en dos secciones, una primera, en la que se estudiará el entorno de trabajo, así como las técnicas forenses digitales en sistemas informáticos comunes y como se han adaptado estos procedimientos al estudio forense en los entornos cloud. Y una segunda en la que se relatará el proceso de creación del entorno de estudio que se implementó y las pruebas que se realizaron sobre él.

Al finalizar la lectura de este documento se tendrán conocimientos sobre:

- Entornos de computación en la nube: definición, arquitectura y modelos de servicio.
- Técnicas forenses digitales: proceso y requisitos del análisis.
- Aplicación de las técnicas forenses en entornos cloud.
- Software forense en entornos cloud.
- Diseño de un laboratorio de computación en la nube.
- Resultados y funcionamiento de una serie de herramientas forenses.

2. Conceptos teóricos y Estado del arte

Previo al estudio del análisis forense en entornos cloud es necesario definir algunos conceptos de interés tanto del modelo del entorno, como del proceso de investigación forense informática. Tras estos conocimientos previos se procederá a explicar las actuales técnicas forenses en este tipo de entornos y los problemas a los que se enfrenta.

2.1. El entorno cloud

La computación en la nube es un modelo de servicio en el cual los recursos informáticos, masivamente escalables, son repartidos entre los clientes a través de las tecnologías de internet[6].

Las principales características del modelo son [7]:

- Uso bajo demanda: el consumidor de recursos cloud puede acceder unilateralmente a dichos recursos. Una vez que el recurso es configurado es posible automatizar su uso, siendo así innecesario la intervención humana desde el lado del consumidor o de la del proveedor.
- Acceso ubicuo: la habilidad de un servicio para ser utilizado por tecnologías heterogéneas; diversos dispositivos, protocolos, interfaces y tecnologías de seguridad
- Multiusuario (compartición de recursos): característica del software que permite el acceso de múltiples usuarios (tenants) a los mismos recursos repartiéndolos entre estos bajo demanda (polling). Esta característica es obtenida de la capacidad de virtualización del software utilizado en los servidores de computación en la nube.
- Elasticidad: capacidad de aumentar (o disminuir) los recursos utilizados por una nube dependiendo de los picos de trabajo, siempre bajo

configuración del usuario o del proveedor. La elasticidad es la principal característica que atrae a los consumidores a utilizar esta tecnología.

- Uso medido: capacidad de la plataforma de calcular los recursos que están siendo utilizados por cada usuario. Principalmente es utilizado por el proveedor del servicio para realizar la facturación en función de tiempo de uso y cantidad de recurso ocupado.

A parte Thomas Erl [8] especifica otra característica que, si bien no es básica en el modelo, varias arquitecturas la implementan.

- Resiliencia: Colocación de recursos informáticos redundantes a fin de evitar un colapso del servicio a causa de la caída de uno de los recursos. Los recursos redundantes pueden pertenecer a la misma nube, pero encontrarse en emplazamientos físicos diferentes.

2.1.1. Virtualización

El funcionamiento de un entorno nube está basado en la virtualización, que consiste en la asignación de recursos físicos para la posterior instalación de un sistema operativo [8]. Los sistemas operativos virtualizados funcionan independientemente unos de otros como si se encontrasen en máquina físicas separadas de tal manera que aun encontrándose virtualizados dentro de otro equipo físico o virtual estos, en principio, no pueden acceder a datos del equipo en el que se encuentran implementados [9].
Para realizar la virtualización se necesita de un hipervisor[1], se pueden encontrar 2 tipos de hipervisores [10]:

- Nativos: El software se instala directamente sobre el servidor físico
  - VMware ESXi
  - Xen
  - Citrix XenServer
  - Microsoft Hyper-V Server
- Hosted: El software se encuentra instalado en un sistema operativo
  - KVM
  - VirtualBox
  - QEMU

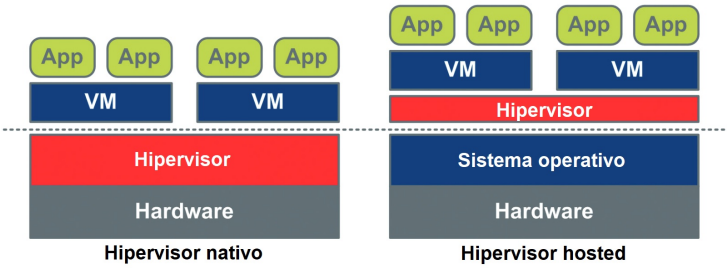

Figura 2.1: Comparación de los hipervisores nativos y hosted.

En la figura 2.1 se puede ver una comparación de arquitectura entre dos máquinas, una con un hipervisor tipo hosted y otro nativo.

2.1.2. Modelos de despliegue

En primera instancia, para poder adjudicar una responsabilidad sobre la seguridad de un entorno cloud debemos conocer las distintas maneras en las que se puede desplegar este tipo de entornos. Los modelos de despliegue coinciden en la mayor parte de la bibliografía [4][7][8]:

- Nube pública: el despliegue público de una nube está disponible para usuarios individuales y grandes empresas que contratan servicios de virtualización a proveedores externos.

- Nube privada: la infraestructura de nube privada es operada y utilizada únicamente por una compañía. El equipo físico se puede encontrar tanto dentro como fuera de sus instalaciones.

- Nube híbrida: combina infraestructuras de distintas nubes gestionadas por distintas entidades, pero actúan como una única infraestructura.

- Nube de comunidad: es una nube diseñada para servir a un propósito común. Puede ser utilizada por una o por varias compañías, estas comparten preocupaciones comunes como objetivo, políticas, seguridad, necesidades de regulación, etc.

[1]Un hipervisor es una plataforma que permite aplicar diversas técnicas de control de virtualización para utilizar, al mismo tiempo, diferentes sistemas operativos en una misma computadora.[10]

Figura 2.2: Responsabilidad según el modelo de servicio.

2.1.3. Modelos de servicio

Dentro de las nubes con infraestructura pública (públicas e híbridas), el nivel de responsabilidad depende de hasta qué punto el usuario tiene nivel de acceso, como se muestra en la figura 2.2. Al igual que en el punto anterior la bibliografía coincide en la clasificación [4][7][8]:

- Infraestructura como servicio: (IaaS [Infraestructure as a Service]) provee de virtualización de sistemas, almacenamiento, virtualización de infraestructura y otros elementos de hardware. El proveedor se encarga de la infraestructura física, mientras que el usuario se encarga del resto de elementos del despliegue.
- Plataforma como servicio: (PaaS [Platform as a Service]) provee de máquinas virtuales, aplicaciones, servicios, transacciones y estructuras de control.
- Software como servicio: (SaaS [Software as a Service]) provee de un sistema operativo con aplicaciones, administración e interfaz de usuario.

Con esto ya se plantan las bases de cómo definir lo que es la computación en la nube y como se puede clasificar los distintos tipos de entornos cloud que existen y quedan resumidos en la figura 2.3

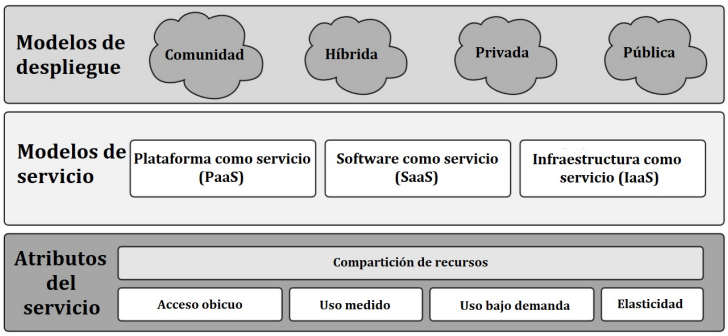

Figura 2.3: Definiciones de la computación en la nube.

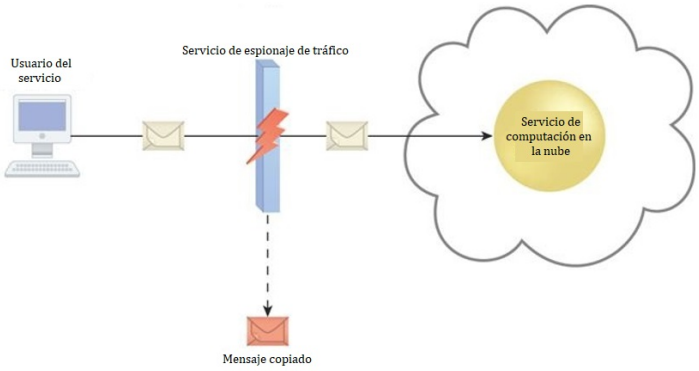

Figura 2.4: Ejemplo de espionaje de tráfico.

2.1.4. Amenazas a los entornos cloud

Dado que la seguridad es el asunto principal de este estudio hay que entender los posibles ataques que este tipo de entonos pueden recibir. Siendo las amenazas más significativas [8]:

Espionaje de tráfico (Traffic eavesdropping) También llamado ataque man in the middle, o por sus siglas MitM. Los paquetes de datos enviados entre usuario y la nube son interceptados por un agente malicioso para reunir datos de manera ilegítima. Este tipo de ataque compromete la validez de los datos y la relación entre el usuario y el proveedor de los servicios, como se ve en la figura 2.4.

Intermediario malicioso Los mensajes son interceptados por un agente malicioso y son modificados amenazando la integridad y la confidencialidad de estos. Además, la modificación puede llegar a inyectar código malicioso

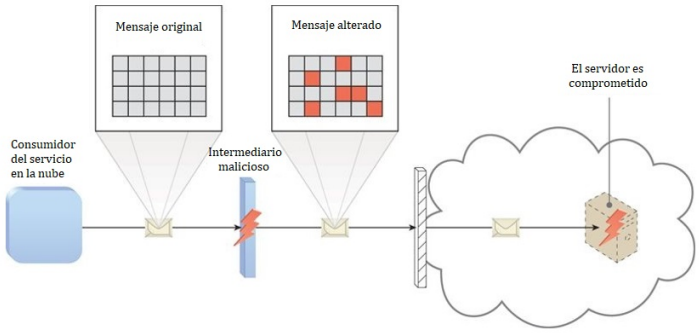

Figura 2.5: Ejemplo de un ataque de intermediario malicioso.

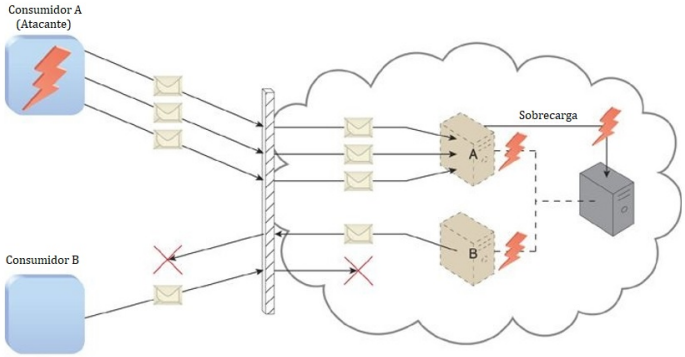

Figura 2.6: Ejemplo de denegación de servicio.

comprometiendo al servidor. Un ejemplo se muestra en la figura 2.5.

Denegación de servicio (DoS) Sobrecarga los recursos informáticos del servidor haciéndolo inaccesible para el resto de los usuarios, como en la figura 2.6. Este tipo de ataques se pueden ejecutar de distintas maneras:

- La sobrecarga de los servidores se efectúa mediante peticiones de comunicaciones repetidas.
- Reducir la capacidad de la red mediante la sobrecarga de esta.
- Se solicitan múltiples servicios a la nube haciendo que la memoria y el procesamiento sean consumidos en exceso.

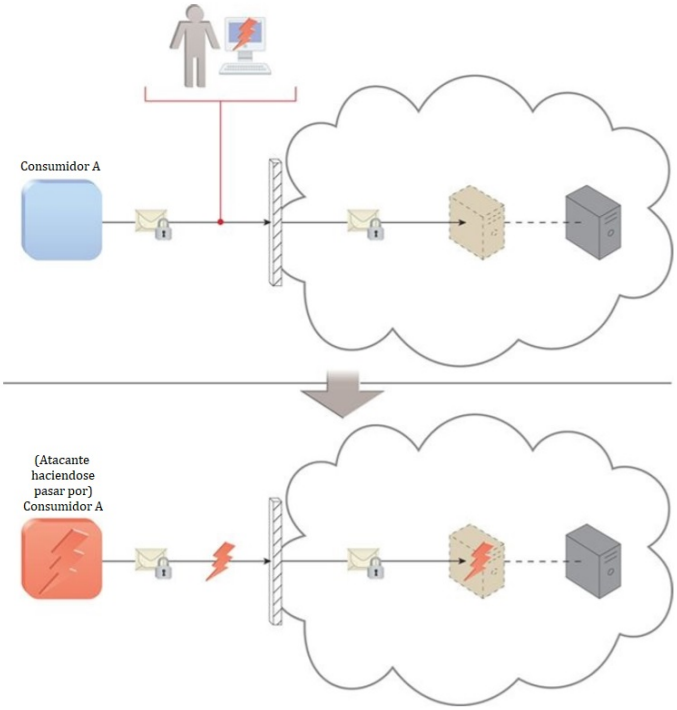

Figura 2.7: Ejemplo de acceso con autorización insuficiente.

Autorización insuficiente El acceso es concedido a un atacante de manera errónea, esto resulta en que el usuario malintencionado recibe capacidad de acceso a recursos normalmente protegidos. Una versión de este ataque se puede ver en la figura 2.7

Ataque de virtualización Como se explicó antes los recursos informáticos virtuales están aislados unos de otros. Este ataque explota las vulnerabilidades del hipervisor poniendo en peligro la confidencialidad, integridad y disponibilidad de los datos almacenados en el servidor.

Superposición de los límites de confianza Un usuario malicioso con acceso al entorno cloud puede tener acceso a los mismos recursos que otros usuarios, en este caso el atacante tiene como objetivo estos recursos compartidos de los que dispone poniendo en compromiso la integridad de estos.

2.1.5. Soluciones de creación de entornos cloud

Como se ha visto, existen multitud de modelos de despliegue y servicio y, por lo tanto, existen multitud de soluciones para cada combinación, tanto

open source[2] como propietarias.

Con los conocimientos que se han adquirido a lo largo de esta sección se dará una descripción de los enfoques más comunes, para posteriormente poder decidir cuál es la más adecuada para el desarrollo del entorno de pruebas.

OpenStack

OpenStack es un sistema operativo cloud, que gestiona grandes cantidades de recursos computacionales, de almacenamiento y de red a través de un datacenter controlado por un "dashboard"que dota a los administradores de capacidades de gestión y da a los usuarios la posibilidad de aprovisionarse de medios a través de una interfaz web[12].

OpenStack es un proyecto de código libre distribuido bajo la licencia Apache 2.0[3]. Existe una gran cantidad de colaboradores en el desarrollo del proyecto, lo que permite a este sistema mantenerse actualizado e ir añadiendo complementos de manera continua.

La funcionalidad del proyecto OpenStack se debe a el trabajo continuo de distintos servicios ejecutándose en servidores distintos de manera cooperativa. OpenStack distingue entre 2 tipos de servicios:

Core Services Son los servicios que otorgan una funcionalidad básica a OpenStack, que vienen descritos en el cuadro 2.1.

Servicios opcionales Son el resto de los servicios que OpenStack puede albergar, dotan al sistema de muchas otras funcionalidades que complementan el servicio con otras utilidades. Algunos de estos servicios se pueden ver en la figura 2.8

[2]Una licencia de código abierto es una licencia de software que permite que tanto el código fuente como los archivos binarios sean modificados y redistribuidos libremente y sin tener que pagar al autor original. Sin embargo, ciertas licencias de código abierto pueden incorporar algunas restricciones, como el requisito de mantener el nombre de los autores y la declaración de derechos de autor en el código, o permitir la modificación del código sólo para usos personales o la redistribución del software para usos no comerciales.[11]

[3]La licencia Apache es una licencia de software libre creada en 2004 por Apache Software Foundation. Es una licencia considerada permisiva[13]. Requiere la conservación del aviso de derechos de autor y el aviso de responsabilidad.

Componente	Funcionalidad	Descripción
Swift	Almacenamiento de objetos	Almacena objetos de datos desestructurados, a través de una API RESTful basada en HTTP. Tiene una gran tolerancia a fallos, mediante la replicación de datos y una arquitectura escalable.
Keystone	Identidad	Proporciona un servicio de autenticación y autorización para otros servicios de OpenStack.
Nova	Computación	Gestiona el ciclo de vida de las instancias en un entorno OpenStack. Entre sus responsabilidades se incluyen la generación, programación y clausura de máquinas bajo demanda.
Neutron	Redes	Habilita la conectividad de red como servicio para otros servicios de OpenStack. Ofrece una API para permitir a los usuarios que definan redes.
Cinder	Almacenamiento de bloques	Proporciona almacenamiento de bloques persistente, para correr las instancias.
Glance	Servicio de imágenes	Almacena las imágenes de las máquinas virtuales. Este servicio es usado durante el aprovisionamiento de instancias.

Cuadro 2.1: Core Services de OpenStack

Figura 2.8: Servicios opcionales OpenStack.

CloudStack

CloudStack de Apache, es un software de código abierto bajo la licencia Apache 2.0 diseñado para desplegar y gestionar grandes redes de máquinas virtuales, como una plataforma de computación en la nube con alta disponibilidad y altamente escalable en modo de infraestructura como servicio (IaaS). CloudStack es utilizado por varios proveedores de servicio para ofrecer infraestructuras públicas de computación en la nube y por varias empresas para proveerse de servicios cloud en la localización de la estas.

CloudStack es una "navaja suiza"que incluye todas las herramientas y características que la mayoría de las organizaciones desean en una nube IaaS: orquestación computacional, red como servicio, administración de cuentas y usuarios, una API nativa completa y abierta, contabilidad de recursos y una interfaz de usuario (UI).[14]

La principal característica de CloudStack es el despliegue de la infraestructura que se describe en la figura 2.9. La infraestructura se compone de:

- Región, es la mayor unidad de organización disponible. Una región se compone de distintas zonas. Cada región está controlada por su propio cluster con servidores de gestión.
- Zona, típicamente se corresponde con un datacenter, aunque es posible tener múltiples zonas dentro de un centro. Las zonas permiten aislar físicamente y añaden redundancia.
- Pod, suele representar un único rack. Todos los hipervisores de un pod pertenecen a la misma subred.

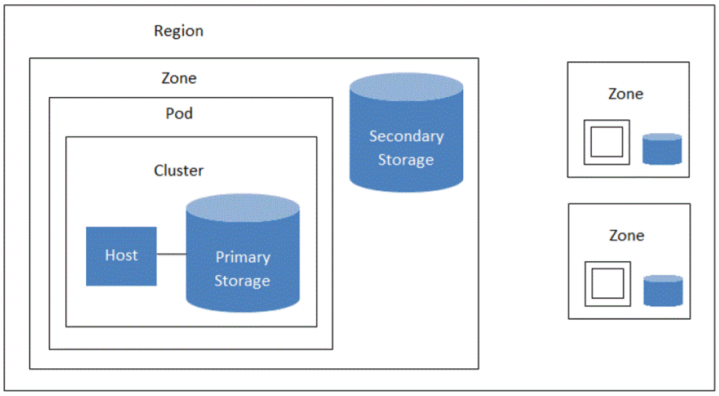

Figura 2.9: Infraestuctura de CloudStack.

- Cluster, es una medida que se toma para agrupar hipervisores, todos los host en el cluster tienen características similares.
- Host, representa a un único ordenador, que tiene un hipervisor instalado.
- Almacenamiento primario, provee al sistema de los discos de almacenamiento virtuales de las máquinas que son ejecutadas por los hosts en su cluster.
- Almacenamiento secundario, provee a las zonas con imágenes de los sistemas operativos, imágenes de los discos duros como backup...

OpenNebula

OpenNebula es otra solución de software libre, bajo la licencia Apache 2.0. Administra centros de datos heterogéneos distribuidos. La plataforma OpenNebula gestiona la infraestructura virtual de un centro de datos para construir nubes privadas, públicas e híbridas en modo infraestructura como servicio (IaaS). Los dos usos principales de OpenNebula son la virtualización de centros de datos y la creación de infraestructuras en la nube.

La infraestructura de CloudStack es análoga a la de un clúster clásico, que se compone de :

- Nodo maestro, es la máquina real donde está instalado OpenNebula. Los servicios de OpenNebula en la máquina de aplicaciones para el usuario incluyen el servicio de administración (oned), el programador (sched), el servidor de interfaz web (servidor Sunstone) y otros componentes.

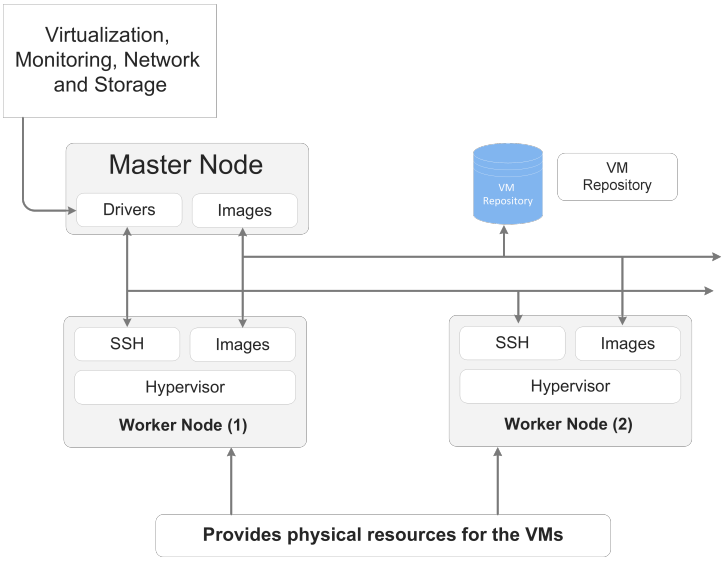

Figura 2.10: Infraestructura de OpenNebula.

- Nodos de trabajo, proporcionan los recursos informáticos reales necesarios para procesar todos los trabajos enviados por el nodo maestro. Los hosts habilitados para el hipervisor OpenNebula utilizan un hipervisor de virtualización como Vmware, Xen o KVM.

- Almacenes de datos, contienen las imágenes base de las máquinas virtuales. Los almacenes de datos deben ser accesibles para el front-end, esto puede lograrse mediante el uso de una de las diversas tecnologías disponibles, como NAS, SAN o almacenamiento de conexión directa.

- Red de comunicaciones física permite la interconexión entre servidores de almacenamiento y máquinas virtuales en ubicaciones remotas. También es esencial que la máquina de front-end pueda conectarse a todos los nodos de trabajo o hosts.

Eucalyptus

HPE Helion Eucalyptus es una solución open source bajo la licencia GPL v3[4] para la implementación de una nube privada. Su nombre hace referencia al acrónimo "Elastic Utility Computing Architecture for Linking Your

[4]La GPL de GNU se refiere a la licencia general pública de GNU (GNU General Public License), la licencia implica que el software y su documentación sean de dominio público además de obligar a todas las obras derivadas de mantener la misma licencia.[15]

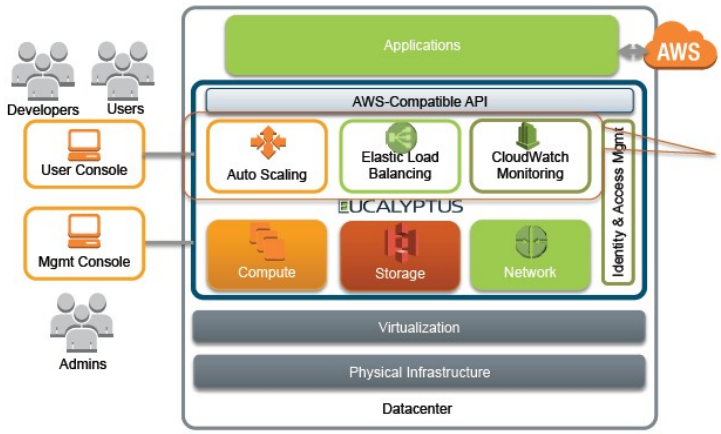

Figura 2.11: Infraestructura lógica de Eucalyptus.

Programs To Useful Systems"que puede traducirse como "Utilidad de arquitectura de computación elástica para vincular sus programas a sistemas útiles".

Eucalyptus es compatible con la api de Amazon Web Services, por lo tanto se puede utilizar para montar una infraestructura de nube híbrida[16]. Tiene la capacidad de balancear la carga de computación entre los recursos privados y públicos. El diagrama de funcionalidades de Eucalyptus se encuentra en la figura 2.11.

Amazon Web Services

Amazon Web Services, AWS por sus siglas, ofrece recursos de cloud computing públicos bajo demanda, los recursos son ofrecidos en un modelo de pago por uso. AWS ofrece varios modelos de servicio, los más importantes se describen a continuación:

Amazon EC2 Amazon Elastic Cloud Compute, se encarga de gestionar los recursos de computación, utilizar este servicio permite ahorrar coste en recursos hardware.

Amazon S3 Amazon Simple Storage Service es un servicio de almacenamiento en la nube. Permite almacenar y recuperar cualquier cantidad de datos en cualquier momento. El objetivo de este servicio es maximizar los beneficios de la escalabilidad. Principalmente se utiliza para almacenar backups, recuperar datos, análisis de Big Data, aplicaciones en la nube, etc.

Amazon EBS Amazon Elastic Block Store, proporciona volúmenes de almacenamiento a nivel de bloques, persistentes y diseñados para ser usados conjuntamente con las instancias de EC2 en la nube de Amazon. Cada volumen de Amazon EBS se replica automáticamente dentro de una zona de disponibilidad, con el fin de proporcionar protección ante fallos de los componentes, y ofreciendo una alta disponibilidad y durabilidad.

Microsoft Azure

Microsoft Azure, también conocido como Windows Azure o Azure Services Plataform es una solución para la gestión de recursos en los servidores de Microsoft, el equipo controlador tiene un sistema operativo homónimo y los equipos controlados llevan instalado Microsoft Server con una versión modificada del software de virtualización Hyper-V[17]. Microsoft Azure proporciona una serie de servicios diferenciados descritos en el cuadro 2.2.

Servicio	Campo	Descripción
Windows Azure Compute	Computación	Hospeda y administra aplicaciones en los centros de datos de Microsoft, pueden ser webs, aplicaciones o máquinas virtuales.
Windows Azure Storage	Almacenamiento	Se encarga del almacenamiento duradero en la nube, almacenando objetos como blobs, tablas, colas, etc.
Microsoft SQL Azure	Bases de datos	Es un servicio de bases de datos basado en SQL Server.
CDN de Windows Azure	Distribución	Acerca datos de Microsoft a los puntos de distribución, como por ejemplo los mapas de Bing o datos de actualización de una app de un desarrollador que utilice Azure
Azure AppFabric	Servicios Complementarios	Principalmente da servicios de autenticación, autorización y mensajería
Azure Market Place	Tienda de aplicaciones	Permite compartir, comprar y vender aplicaciones SaaS y datos.
Azure Virtual Network	Red Virtual	Aporta una serie de funciones de red, entre otras permite a los recursos locales y virtuales tener conectividad IP, como si se encontrasen conectados al mismo enrutador.

Cuadro 2.2: Servicios de Microsoft Azure

2.2. Técnicas forenses digitales

Las técnicas forenses digitales se definen como el uso de métodos científicamente probados para la preservación, colección, validación, identificación, análisis, interpretación, documentación y preservación de evidencias derivadas de fuentes digitales con el propósito de posteriormente reconstruir acciones criminales, o ayudar a anticipar acciones no autorizadas que potencialmente interrumpirían el funcionamiento esperado [18].
Existen diversos modelos del proceso forense digital, en este documento se utilizará el modelo en 4 fases descrito por A. Eleyan y D. Eleyan [19] que se encuentra destinado a su aplicación en entornos cloud. Las fases, que se muestran en la figura 2.12, son:

1. Identificación: es la fase en el que se reporta el abuso o uso maligno del dispositivo informático. El proceso forense comienza con la identificación de la evidencia, que puede ser una imagen del disco duro, un archivo o los logs (también llamados archivos de registro) de alguna aplicación del sistema.
   Por lo tanto, este paso se divide a su vez en 2 partes, identificación del problema e identificación de la prueba.

2. Colección y preservación: este es la frase principal del proceso forense. Un error durante esta fase puede afectar al resto del proceso forense. La evidencia es extraída de del dispositivo afectado mediante un proceso denominado mirroring.

3. Procesado y análisis: Es el empleo de técnicas y herramientas forenses para detectar y extraer los datos de la evidencia, mientras se protege su integridad. Si el análisis no llega a una conclusión valida se deberá volver al paso 1.

4. Diseminación de resultados: reporte sobre el proceso realizado, se deben indicar los datos obtenidos, así como el proceso para obtenerlos; también, las recomendaciones sobre seguridad que necesita el sistema a fin de evitar que vuelva a ser víctima del mismo tipo de ataque.

2.2.1. Modelos de técnicas forenses digitales

Según el momento de aplicación de las técnicas forenses digitales podemos distinguir entre dos modelos [20]:

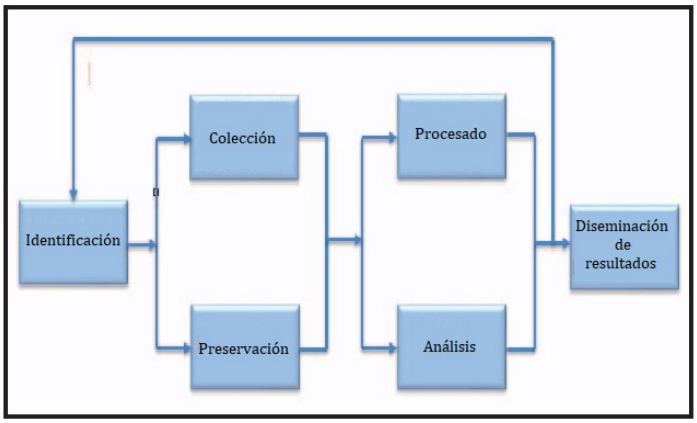

Figura 2.12: Diagrama de flujo del proceso forense.

Técnicas forenses reactivas son el método clásico de análisis forense de sistemas, se realizan después de que el crimen haya sido cometido. En este proceso las evidencias obtenidas son de dos tipos, reactivas, ficheros que hayan quedado almacenados en memoria, y activas, todo tipo de evidencias dinámicas como por ejemplo procesos almacenados en memorias volátiles (RAM).

Técnicas forenses proactivas es un método más moderno. Se analizan partes del sistema de manera continua, cuando el sistema es víctima de un ataque se dispara una alarma y se almacena y preserva la evidencia, generándose un reporte automático para un análisis futuro. A diferencia del modelo anterior la preservación es posterior a la colección, pues el incidente no ha sido identificado.

2.2.2. Tipos de herramientas forenses

Dependiendo del ataque recibido los procedimientos para la extracción de la evidencia digital y por tanto las herramientas utilizadas para su obtención son diferentes.

Imágenes de disco el proceso de copiar bit a bit de un disco, de una manera más general, se puede aplicar a cualquier proceso que se considere un flujo de bits (volúmenes de datos físicos y lógicos, flujos de datos en red...).

Recuperación de datos rescata datos que de manera normal no pueden ser leídos, por encontrarse inaccesibles, perdidos, corrompidos, dañados o formateados.

Análisis de archivos Existen distintos tipos de software para el análisis de archivos, mientras que unos detectan datos copiados entre documentos, otros detectan código malicioso introducido en otros archivos, etc.

Extracción de metadatos de documentos Leen los metadatos de los documentos a fin de conocer más información sobre ciertos archivos que pueden ser utilizados como evidencia informática, como fecha de creación y modificación, tamaño del archivo o tipo de archivo entre otros.

Imágenes de memoria Mediante un principio similar a la creación de imágenes de discos se realiza una copia bit a bit de una memoria.

Análisis de memoria estudia los programas y procesos que se estaban ejecutando en una memoria en el momento que se realizó la imagen del disco.

Herramientas forenses de red engloba distintos tipos de herramientas que leen paquetes de red (sniffers), detectan actividad anómala en la red (intrusion detection sistems) o herramientas de búsquedas de direcciones IP que tienen el fin de conocer al atacante que dejó la evidencia.

Análisis de logs Ciertas aplicaciones cuando realizan actividades van almacenando dichas actividades en archivos de registro o logs, la manera de almacenar los logs es bastante heterogénea. Estas herramientas facilitan la lectura de los registros de aplicaciones.

Herramientas antiforenses es un grupo de herramientas cuyo objetivo es frustrar que la evidencia obtenida por cualquiera de las herramientas anteriores sea veraz o sea imposible de obtener. Las más habituales son: la encriptación de datos o protocolos de red; la modificación de metadatos y aprovecharse de la estructura de archivos del sistema operativo para ocultar datos o crear archivos que llevan a una sección vacía de un disco, dificultando la investigación del forense.

Eliminación segura Eliminan datos de discos por completo, haciendo imposible su recuperación. De tal manera que solo si se tenía un conocimiento previo de la existencia de estos archivos se puede detectar que han sido eliminados.

2.3. Análisis forense en entornos cloud

Conociendo cómo funciona el análisis forense de una manera general y qué tipo de herramientas y técnicas tiene investigador a su disposición, la evolución hacia los entornos en la nube no es trivial, pues los recursos informáticos a los que se intenta acceder no suelen ser accesibles al investigador. Para este fin, se indica que el modelo más adecuado es el análisis forense de redes[5]. En este tipo de análisis la adquisición y la verificación de la evidencia digital es más rápida, debido a procesos automatizados como el hashing, que se produce cuando se almacenan datos en el entorno cloud [5].
Pero no todos los procesos forenses se pueden realizar siguiendo la técnica anterior y debe evaluarse otra serie de opciones como la utilización de imágenes de la máquina virtual, mediante la toma de "snapshots"[6] de las instancias de la máquina [21].

2.3.1. Usos de las técnicas forenses en entornos cloud

Podemos definir varios usos de técnicas forenses que actualmente se pueden realizar en entornos cloud [22]:

Investigación especialemente la investiación de ciberataques, así como su resconstrucción y la constatación de la validez de las pruebas. También la comprobación que las políticas de entornos multijurisdiccionales y multiusuario se están aplicando o la investigación de transacciones, operaciones o sistemas sospechosos.

Solución de problemas como por ejemplo la recuperación de archivos o instancias eliminados, la localización las causas de incidentes, el rastreo eventos o la resolución de problemas funcionales u operacionales; así como el manejo de incidentes de seguridad.

[5]El análisis forense de redes se define como estudio sistemático de los flujos de datos de una red.

[6]La toma de snapshots es un concepto similar al disk mirroring que se realiza sobre los equipos tradicionales.

Monitorización de registros Colección, análisis y correlación de diversas entradas de logs de múltiples sistemas en un entorno cloud.

Recuperación de datos y sistemas tanto datos erronea o malintencionadamente borrados, así como desencripción de datos encriptados cuya clave ha sido perdida, recuperación de sistemas tras un ataque o la adquisición de datos que vayan a ser redesplegados, retirados o que tengan que ser desinfectados.

Cumplimiento normativo ayudar a organizaciones a cumplir con la normativa vigente en materia de protección de la información, mantenimiento de registros para auditorías, notificación a usuarios cuando sus datos han sido expuestos, etc.

2.3.2. Retos y soluciones de las técnicas forenses en entornos cloud

Existen muchas partes interesadas en el desarrollo de las técnicas forenses aplicadas en los entornos cloud, entre estas se encuentran gobiernos, empresas e investigadores de las especialidades de comunicaciones y tecnologías de la información. Estos retos se pueden clasificar en técnicos, legales u organizativos[23]. Cuando se habla de retos de las técnicas forenses aplicadas en entornos cloud, solamente se habla de aquellos retos intrínsecos a la aplicación de dichas técnicas al tipo de entorno en el que se trabaja y no a las propias limitaciones de las técnicas que se apliquen, aunque, los objetivos del análisis forense tanto dentro como fuera de entornos cloud deberían ser los mismos.
En el documento del Instituto Nacional de Estándares y Tecnología de Estados Unidos referenciado anteriormente[23] se establece una lista de los 65 mayores problemas que existen sobre la aplicación de técnicas forenses en entornos cloud, que se sintetizan a continuación:

Arquitectura dados los diferentes tipos de arquitecturas físicas de nubes y las diferencias entre distintos proveedores dar con la localización de las evidencias resulta complejo, pues pueden tener varios puntos finales desde donde dotan de servicio a la infraestructura, además de que pueden no haber previsto la manera de incautar las pruebas sin afectar al resto de usuarios.

Colección teniendo en cuenta la gran cantidad de información que se maneja en un entorno cloud no toda ella puede ser presentada como evidencia

forense y la localización de esta, la extracción de información de máquinas virtuales y el mantenimiento de la integridad de los datos en un entorno multitenant sin violar la privacidad de los otros tenants se convierte en otro problema.

Análisis el mayor reto del análisis es la reconstrucción de los hechos dentro del entorno, tarea que se vuelve mucho más complicada cuando son varios proveedores distintos los que ofrecen conjuntamente el servicio. La tarea de organizar los logs de eventos de manera temporal entre distintos servicios que no están correctamente sincronizados, encontrar la correlación entre los eventos o la reconstrucción de escenarios a partir de imágenes y datos del almacenamiento.

Respuesta a incidentes confianza, competencia e integridad de los datos recabados por los proveedores del servicio, que actúan los primeros ante un incidente y realizan una colección inicial de los datos.

Retos legales Por lo general en un cibercrimen realizado en un entorno cloud se pueden encontrar hasta 3 países implicados: donde se encuentran los proveedores, donde se encuentra el atacante y donde se encuentra el atacado, esto genera una serie de problemas jurisdiccionales que son agravados por la falta de medios para la comunicación y cooperación internacional durante la investigación. La adquisición de datos se basa en la cooperación de los proveedores de servicios cloud además de en su competencia e integridad.

Estandarización los mayores problemas del trabajo forense en este tipo de entornos vienen de la mano de la falta de unos estándares de operación, prácticas y herramientas y la falta de interoperabilidad entre los distintos proveedores.

A parte de estos seis puntos problemáticos se habla del uso de técnicas antiforenses, de la administración de roles y de la falta, y antigüedad, de la documentación sobre el análisis forense.

Las principales dificultades que se pueden llegar a estudiar en este documento son identificación, colección y análisis. Por ser los más directamente relacionados con el campo de la investigación forense digital y de redes. Por lo tanto a continuación se amplía la información referente a estos campos.

Identificación

Es la principal fase del proceso forense y empieza definiendo las fuentes de datos de las que se dispone, que son [24]:

- Desarrollador del entorno: Dado que la mayor parte de los procesos del entorno cloud son realizados de manera online una gran parte de los artefactos forenses se encuentran en las instalaciones del servidor de computación.

- Proveedor del servicio: En algunos casos el desarrollador del entorno no es accesible o no facilitará los artefactos forenses, el proveedor no dispone de tantos datos pero normalmente es suficiente como para avanzar en la investigación.

- Cliente: En otras ocasiones, la información volátil almacenada en la memoria RAM del cliente puede contener trazas de información útil para la investigación.

Como se indicó en apartados anteriores el modelo de servicio indica hasta donde el usuario puede llegar a tener acceso a evidencias, la figura 2.2 es un ejemplo de esto. También es un dato que ayuda a conocer qué tipos de artefactos forenses se pueden llegar a obtener [24]:

- Software como servicio: Monitores asignados a usuarios y logs de nivel de aplicación (acceso, identificación, transacciones, uso, información de la cuenta, problemas de rendimiento, volumen de datos...)

- Plataforma como servicio: registros específicos de las aplicaciones de la plataforma, errores y advertencias del sistema operativo.

- Infraestructura como servicio: logs a nivel del sistema, eventos del hipervisor, archivos de máquinas virtuales (imágenes de disco duro, memoria RAM...), capturas de red, backups...

Dependiendo de la implementación del servidor de computación no siempre es posible llegar a acceder a las evidencias, generalmente el acceso a estos datos es controlado por el desarrollador del entorno y en algunos casos no se llegan a generar por decisión de este. Por ejemplo, la creación de backups en un servidor de OpenStack (IaaS) es realizada mediante un plugin opcional.

Colección

Mientras que la colección desde el lado cliente se puede realizar utilizando las técnicas forenses tradicionales, para obtener los datos procedentes del servidor solo se dispone de la información que el proveedor o desarrollador ofrecen (a través de una API o mediante una solicitud).

En el artículo de Alqahtany et al. se recogen los problemas de la adquisición y análisis forense y sus soluciones, son las siguientes [25]:

- Dependencia de los proveedores: Como se ha visto los usuarios e investigadores son muy dependientes de los proveedores de servicio y de los desarrolladores del entorno. Esta dependencia genera problemas con la integridad de los datos y la cadena de confianza con los proveedores. Además, hay varios motivos técnicos y económicos por los que los proveedores no darán cierta información, por ejemplo:
  - El volumen de datos no permite almacenar backups de todas las instancias.
  - Los proveedores de servicio suelen ocultar la localización de los datos.
  - En caso de incidente el proveedor de servicio intentará primeramente restablecer el servicio antes de reunir las evidencias.
  - La formación de los investigadores puede no ser suficiente para mantener la integridad de los artefactos forenses. Por lo tanto, las evidencias pueden ser cuestionadas en un juicio.
- Aislamiento de una instancia: en cualquier escenario de cibercrimen es necesario aislar el equipo investigado, para evitar que las evidencias desaparezcan, se adulteren o modifiquen. Hacer esto en un entorno de computación en la nube no es trivial, pues muchos de los recursos son compartidos entre las distintas instancias.
  Se proponen dos métodos de aislamiento de instancias, el primero, la recolocación de la instancia en un nodo aparte; y el segundo, el aislamiento de la instancia en una sandbox[7].
- Procedencia de los datos: La procedencia de los datos juega un rol importante en la investigación forense en este tipo de entornos. Conocer la procedencia de los datos permite al investigador conocer quien es

[7]Un sandbox es un mecanismo de seguridad para disponer de un entorno aislado del resto del sistema operativo.[26]

el dueño de los datos y cuando y quien accedió a ciertos recursos en un momento concreto. Li et al. proponen un método para registrar la propiedad y el uso de archivos y recursos[27].

- Integridad de los datos: La preservación de los datos es un paso crítico en el proceso forense y es crucial que las evidencias permanezcan inmutables. En un escenario en el que múltiples actores son presentados no siempre es posible tener un control absoluto de todos los datos.
  Con el objetivo de preservar la integridad se presenta el Módulo de Plataforma de Confianza (Trust Platform Module o, por sus siglas, TPM)[28]. Que permite mantener la integridad y la confidencialidad de los datos. También permite autentificar máquinas, encriptar hardware, realizar firmas digitales, almacenamiento seguro de claves y validar la integridad de la información.

- Sincronización temporal: La diferencia en las zonas horarias entre los servidores de la nube y los clientes de la nube puede afectar la integridad, confiabilidad y admisibilidad de la evidencia.
  Usar la misma zona horaria, por ejemplo GMT, entre las distintas entidades implicadas puede ayudar a realizar el análisis temporal de los hechos investigados[29].

Análisis

Es un gran desafío llevar a cabo un análisis exhaustivo de un escenario en la nube debido al gran volumen de datos que hay que analizar y la falta de procesos y evidencias que se pueden encontrar. Además, no existe una solución unificada para la extracción, dado que el usuario puede acceder a sus recursos desde una gran variedad de dispositivos (Tablets, PCs y teléfonos móviles).

Además, debido a la naturaleza distribuida y compartida de la nube, cada evento del crimen puede ocurrir en diferentes países. Lo que dificultará el establecer el orden lógico en que tuvo lugar. Los investigadores podrían enfrentar una amplia gama de desafíos cuando realicen la etapa de examen y análisis, incluyendo:

- Falta de herramientas forenses: Se sabe que las herramientas forenses disponibles tienen varias limitaciones y no pueden hacer frente a las características distribuidas y elásticas de la computación en la nube. También existe una alta demanda de herramientas forenses para que el proveedor de servicio y los clientes realicen una investigación forense en

un entorno cloud. Por lo tanto, es crucial desarrollar herramientas que puedan utilizarse para identificar, recopilar y analizar evidencias halladas en la nube. Se necesita una combinación de herramientas forenses informáticas y forenses de red para obtener datos forenses y luego poder analizarlos. Las herramientas forenses tradicionales se pueden utilizar para recopilar los datos activos mientras se mantiene su integridad al tiempo que se pueden utilizar herramientas forenses de red para recopilar datos adicionales a través de la red, incluidos los logs.
Dykstra y Sherman indican que las herramientas forenses se deben encontrar desde el plano administrativo[28]. También indican que utilizar las herramientas desde el plano de administración es la solución más equilibrada entre confianza en el proveedor y velocidad de obtención de datos. Posteriormente, los mismos, diseñan una utilidad llamada FROST, Forensics Open-Stack Tools (Herramientas Forenses para Open-Stack) [30] que sienta la base para lo que debieran ser las herramientas posteriormente desarrolladas para ejecutarse desde el plano administrativo.

- Reconstrucción de la escena del crimen: Es crucial reconstruir la escena del crimen para entender cómo se cometieron las actividades ilegales. Por ejemplo, cuando un usuario malintencionado cierra su instancia virtual después de cometer ciertas actividades maliciosas, la reconstrucción de la escena del crimen será imposible.
Geethakumari y Belorkar propusieron un método que permite a los investigadores repetir el evento del ataque y restaurar el sistema al estado anterior al ataque mediante el uso de snapshots.[31]

2.3.3. Herramientas forenses en entornos cloud

El cloud es una virtualización de una red de computadores, por lo tanto, muchas de las herramientas forenses de redes y de equipos informáticos son válidas para este tipo de entornos, pero, como se ha visto, las características del cloud hacen más complicado que un software forense tradicional pueda desenvolverse con normalidad. Por ello se analizarán una serie de herramientas que pueden ser utilizadas en investigaciones forenses en entornos de nube.

Se estudiarán una serie de herramientas dedicadas al cloud, más concretamente a la solución OpenStack, debido a que es la que se utilizará en el siguiente capítulo para el diseño del laboratorio de pruebas. También se estudiarán otro tipo de herramientas que, si bien, no pertenecen a esta tipología de entorno, han sido utilizados en alguna ocasión.

Wireshark

Wireshark es un analizador de protocolos. El programa se conecta a un interfaz de red y muestra los paquetes de datos que estén pasando por dicho interfaz. Puede reconstruir una sesión TCP o UDP. Este software está pensado para trabajar en un equipo real, para su adaptación al cloud es necesario o bien que se instale en el host para capturar el tráfico entrante o saliente del entorno o que se instale en un equipo virtual conectado a todas las redes virtuales que se necesite analizar[32].

Ceilometer

El proyecto Ceilometer es un servicio de recopilación de datos que da la capacidad de normalizar y transformar datos en todos los componentes principales de OpenStack actuales. Ceilometer es un proyecto de telemetría, sus datos se pueden utilizar para proporcionar capacidades de facturación, seguimiento de recursos y alarmas a los clientes en todos los componentes principales de OpenStack. La principal ventaja de este software en cuanto a capacidad forense es la capacidad de marcar alarmas de eventos, como por ejemplo un intento excesivo de accesos a una cuenta (como en caso de intentar entrar mediante un ataque de fuerza bruta).[33]

Monasca

El proyecto Monasca es una solución de monitorización, se autodenomina MONaaS (Monitoring as a Service), para la plataforma OpenStack, está compuesto de una multitud de capacidades entre las que se encuentran [34]:

- Monasca Persister: Obtiene métricas y alarmas y las almacena en una base de datos.
- Predictor de anomalías: es un prototipo aún, pero se encarga de tomar métricas actuales y pasadas y trata de prever situaciones problemáticas en el sistema.
- Mecanismo de detección de métricas: Cuando una métrica sobrepasa un umbral es capaz de marcar una alarma.
- Mecanismo de notificación: cuando una alarma es realizada por el sistema el mecanismo de notificación actúa, por ejemplo, mandando un e-mail.

- Interfaz de usuario: Muchas de las capacidades del proyecto son configurables desde Horizon, la interfaz de usuario de OpenStack, y además es capaz de integrar las métricas en Grafana, un software de visualización de métricas.
- Tiene la posibilidad de integrarse con Ceilometer y que Monasca gestione los datos que ceilometer genera.

Snort

Snort es un sistema de detección de intrusiones de código libre, se compone de tres subprogramas: un analizador de paquetes (que funciona de manera análoga a Wireshark), un registro (que almacena los paquetes que obtiene el analizador) y un NIDS (sistema de detección de intrusiones de red)[35]. Los tres componentes juntos son capaces de analizar el tráfico de una red y encontrar situaciones anómalas en dicha red. Al no tratarse de un software diseñado para la nube se debe realizar una implementación similar a la de Wireshark.

Congress

Congress es un proyecto de OpenStack para proporcionar política como servicio (Policy as a Service) a través de cualquier colección de servicios en la nube, con el fin de ofrecer normas y obligar su cumplimiento[36]. Congress dispone de una interfaz web desde la que se puede gestionar. Las políticas se pueden configurar para mandar alarmas de situaciones no deseadas en la nube, por ejemplo, un equipo de un tenant conectado a una red que no le pertenece, si no tiene permiso para ello; también permite, por ejemplo, pausar activamente el equipo que está violando dicha norma.

Freezer

Freezer es una plataforma para gestionar copias de seguridad distribuidas y recuperación de desastres como servicio. Está diseñado para ser multi sistema operativo (Linux, Windows, OSX, BSD), enfocado en proporcionar eficiencia y flexibilidad para copias de seguridad basadas en bloques, copias de seguridad incrementales basadas en archivos, acciones puntuales, sincronización de trabajos (es decir, sincronización de copias de seguridad en múltiples nodos) y muchas otras características[37]. Está dirigido a ser útil para todos los entornos, incluidas grandes nubes. Dispone de una interfaz de usuario integrada en Horizon de OpenStack.

3. Implemetación práctica

Sentadas las bases teóricas, se puede proceder al diseño del demostrador que se utilizará para comprobar las capacidades forenses que una serie de programas pueden llegar a ofrecer. Primeramente se hablará qué utilidades van a ser usadas y se ampliará la información sobre estas, seguidamente se describirá el proceso realizado para la implementación de la prueba y para finalizar se realizará la simulación de un ciberataque y se estudiarán las evidencias generadas.

3.1. Descripción del demostrador

A la hora de describir el demostrador es importante escoger un hardware suficiente para que el equipo pueda funcionar sin problemas y evitar fallos en el sistema debido a un hardware limitado. También es necesario establecer que máquinas virtuales se van a utilizar, el sistema de despliegue y el software que se utilizará, así como las configuraciones que se apliquen a los anteriores.

Antes de comenzar el desligue es necesario decidir que componentes van a ser utilizados, en este caso se va a virtualizar una nube de OpenStack mediante DevStack, que introduce una serie de cambios que se explican más adelante. Sobre ese entorno se añadirán los complementos Monasca, Ceilometer y Congress, que también se ampliará la información, respecto a lo indicado en el apartado 2.3.3.

3.1.1. Hardware e hipervisor

Para la maqueta de pruebas se ha escogido virtualizar el entono sobre un portatil ASUS ROG GL552JX con Windows 10 instalado como sistema base, que es del que se dispone. El equipo posee 8 Núcleos y 16 GB de memoria RAM. Se podía haber optado por un arranque dual para aprovechar al máximo las especificaciones del equipo, pero son suficientemente altas como

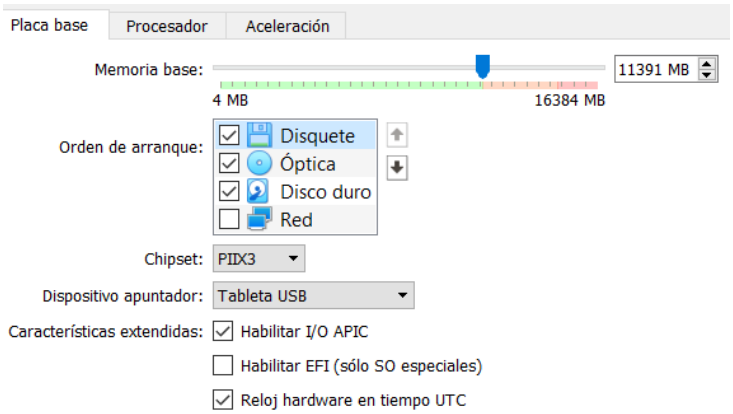

Figura 3.1: Memoria máxima recomendada para virtualización

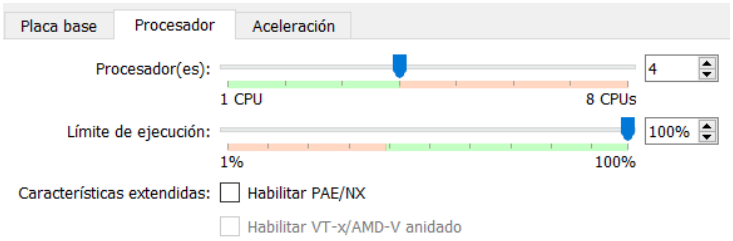

Figura 3.2: Máximo de núcleos recomendados para virtualización

para poder escoger instalar un hipervisor y arrancar una o varias máquinas virtuales para la maqueta.

Para la virtualización del entono se escoge el hipervisor Oracle VM VirtualBox, sin ningún motivo en especial (siendo válido cualquier otro hipervisor de tipo hosted). Antes de decidir cuantos recursos va a utilizar cada máquina en las figuras 3.1 y 3.2 se ven la memoria RAM máxima y procesadores máximos que se recomienda dedicar a la virtualización respectivamente; hay disponibles algo más de 11GB de memoria y exactamente 4 núcleos.

Para la virtualización de las redes se decide utilizar el modo bridged networking, lo que a nivel de red hace que cada máquina virtual se conecte de manera independiente al enrutador, como se indica en la figura 3.3. Aunque en realidad están compartiendo la tarjeta de red del anfitrión como si de un bridge se tratase.

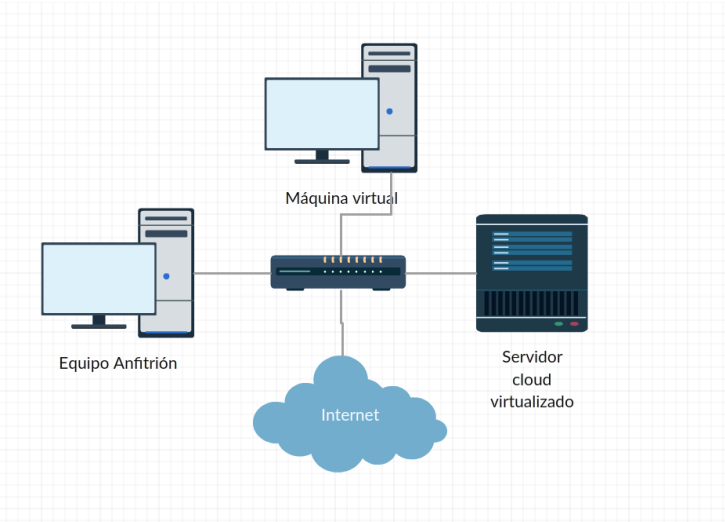

Figura 3.3: Diagrama lógico de red del entorno

3.1.2. Sistema operativo y entorno

Para la implementación se ha decidido utilizar DevStack para poder desplegar una nube de OpenStack. OpenStack es una infraestructura para la gestión de nodos de computación para poder ofrecer recursos informáticos a través de internet. Como ya se vió en la sección 2.1.5 existen una serie de servicios básicos y otros opcionales, aunque realmente la configuración mínima de OpenStack necesita de 2 nodos, uno para Nova (computación) y otro para la gestión con Keystone (identidad), Glance (imágenes) y Neutron (redes), si se está instalando la última versión estable (Stein) en el nodo de gestión se debe instalar además el servicio de colocación [38].

De una manera más general se presenta la figura 3.4, en la que se muestra la arquitectura lógica de un entorno real de producción. Además se ha marcado donde se aplican las técnicas forenses que se detallaron en la sección 2.3.2. En primer lugar la colección se realizará mediante un software de monitorización, cuyo agente recabará datos de los distintos nodos del sistema. La identificación se realizará a nivel de gestión, representada por la conexión superior de los nodos. Finalmente se realizará un análisis de distintos archivos de registro o logs de los servicios de control situados en el nodo homónimo.

Esta solución no se escogió por que se tendrían que virtualizar varios servidores agregando más carga al sistema y aumentando la complejidad de la instalación. Un ejemplo de la estructura lógica de OpenStack aparece en la

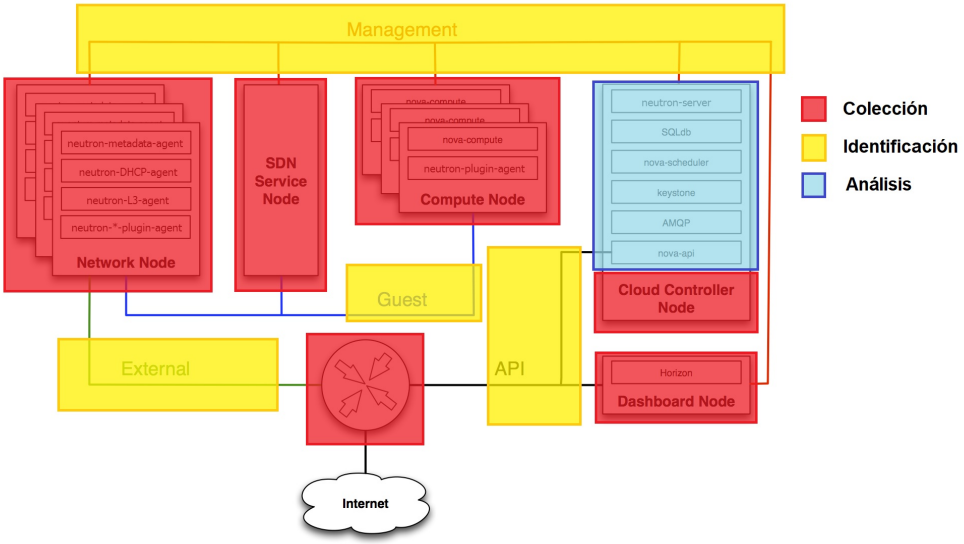

Figura 3.4: Aplicación de las técnicas forenses

figura 3.5 en la que se detalla el funcionamiento de varios de los servicios. Es por ello que se estudian 2 modelos de despliegue de una nube de OpenStack que se de detallan a continuación.

DevStack es una serie de scripts que despliegan una nube de OpenStack en uno o varios servidores, para ello aísla los distintos servicios ejecutándolos en unidades de systemd lo que permite que todos funcionen sobre el mismo nodo, pero también implica que todos los servicios deben de ser especificados antes de la instalación.

Devstack no es una implementación de producción y, como tal, tras un reinicio varios de los componentes dejan de funcionar y los logs no son accesibles desde un archivo, únicamente se accede a ellos desde journald[39]. Lo cual dificulta el trabajo con este sistema pero no lo imposibilita, todas las pruebas se deban realizar sin apagar el equipo (en un principio se probó tomando snapshots de la máquina pero algunos complementos se desincronizaban y daban problemas) y para acceder a los logs se hace mediante el comando `journalctl`.

PackStack se refiere a un conjunto de módulos "puppet", que únicamente funcionan sobre sistemas basados en RedHat[40]. Se trata de un sistema más similar al que existiría en un entorno de OpenStack real, pero no dis-

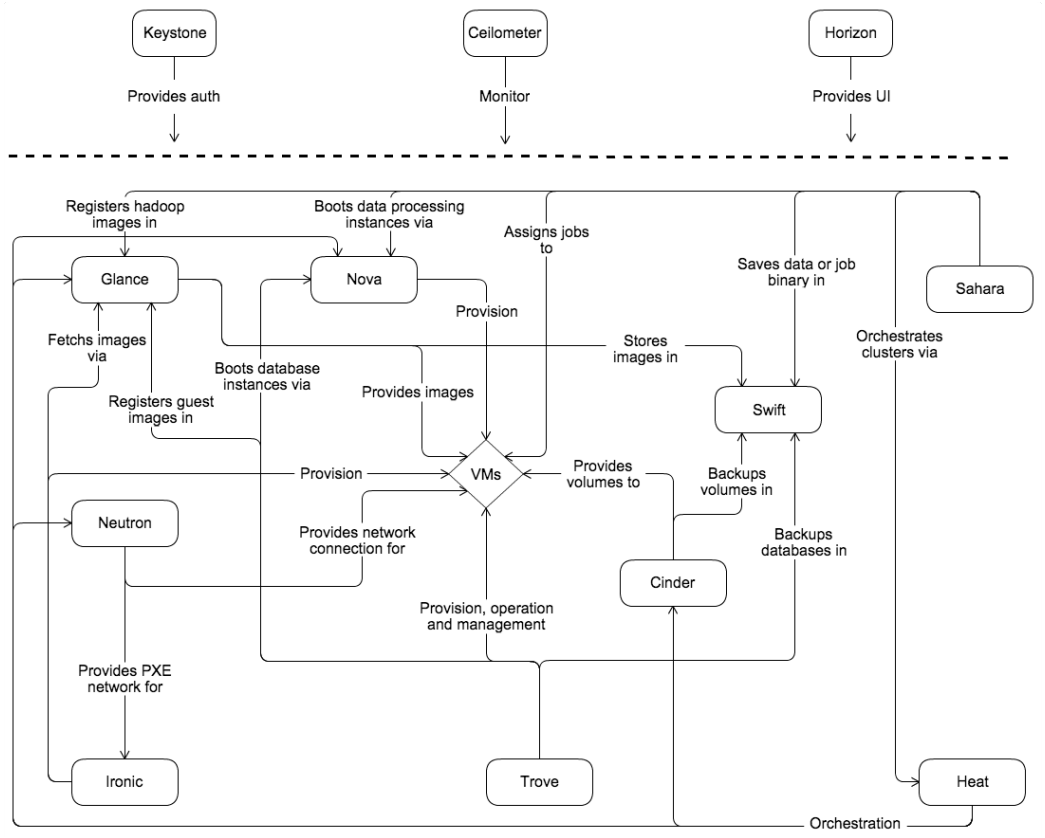

Figura 3.5: Infraestructura lógica de OpenStack

pone de tantos complementos como los que llega a tener DevStack, que, prácticamente, tiene todos; pues se trata de un entorno de pruebas para los desarrolladores.

Es por ello que para el sistema operativo se utiliza Ubuntu 16.04, que es la versión más testeada en DevStack rocky [41], que era la última versión estable cuando se probó por primera vez el demostrador.

3.1.3. Programas forenses

Decidida ya la impletación del entorno cloud se pasa a decidir que programas forenses se van a utilizar, se decide seguir las indicaciones dadas por Dykstra y Sherman [30], que indican que las funcionalidades deben encontrarse desde el dashboard. En 2.3.3 se habla de varios programas accesibles desde el dashboard: Monasca, Freezer y Congress. Freezer por tratarse de un programa para la toma de snapshots se descarta pues el campo de estudio es más típico de ingeniería informática que de telecomunicaciones. Por lo tanto se deciden utilizar los otros dos.

Congress como monitor de políticas se utiliza para estudiar redes compartidas entre tenants en las que un tenant no autorizado se intenta conectar. Por el momento, una red compartida en OpenStack es compartida entre todos los tenant; si se quiere crear una red compartida entre 2 de los tenant se debe añadir otro complemento para controlar el acceso.

Por su parte Monasca se utiliza como monitor de métricas, monitoriza métricas de Ceilometer mediante Celiosca. En concreto utilizar la métrica `indentity.authenticate.failure` con el fin de detectar un ataque de fuerza bruta.

Monitorización

Como se ve en el apartado 2.3.3, Monasca es una solución de monitorización para la plataforma. En un entorno OpenStack real, un nodo computacional se encarga de la gestión y ejecuta los principales programas de gestión, así como la base de datos donde se almacenan los datos registrados por el programa agente (`monasca-agent`) que se encuentra en el resto de nodos de computación. Dado que no es un entorno de OpenStack de producción, sino que es un entorno de desarrollo, DevStack, instalado sobre un solo nodo de computación todos los programas se encuentran ejecutándose sobre el mismo nodo; tanto los básicos de funcionamiento de OpenStack como el gestor y los

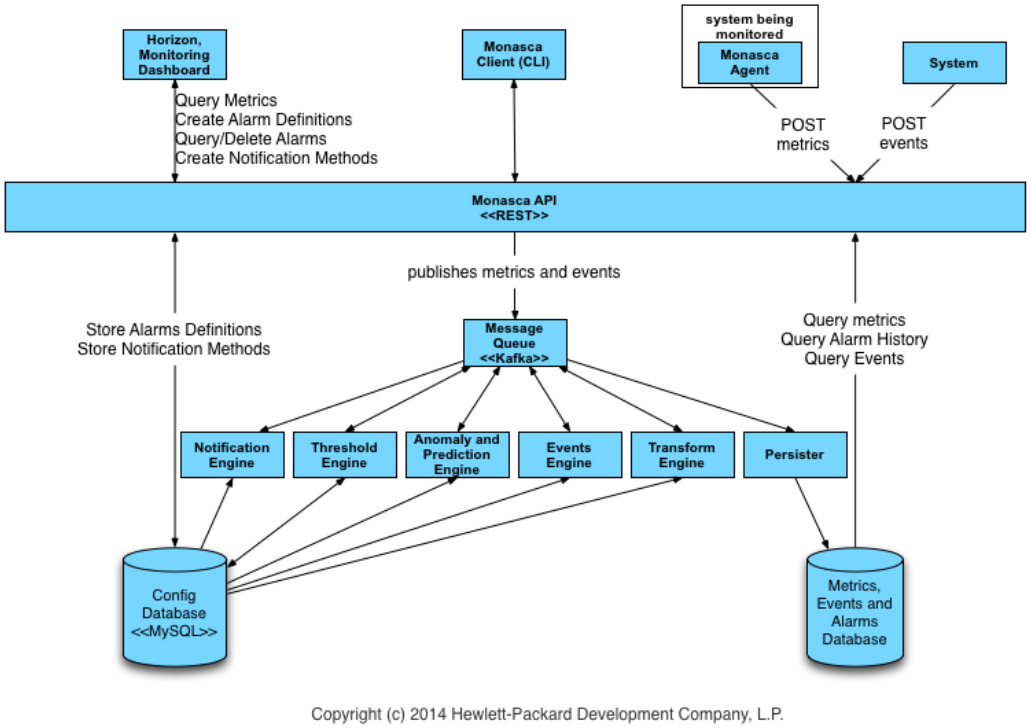

Figura 3.6: Infraestructura de monasca

agentes de Monasca.

Como se puede ver en la figura 3.6, la estructura de Monasca se basa en dos bases de datos que almacenan la configuración y métricas, eventos y alarmas respectivamente, los seis componentes en fila sobre estas son los programas encargados de comprobar que las alarmas configuradas no se sobrepasan y de almacenar las métricas y alarmas generadas. Sobre estos se encuentra la cola de mensajes gestionada por Kafka[1] y toda la comunicación entre Kafka, el dashboard, el cliente de linea de comandos, los agentes, el sistema y las bases de datos se hacen a través de la API REST[2] de monasca. A parte de esto Monasca brinda más funcionalidades que no aparecen en la figura antes referenciada.

[1]Apache Kafka es un proyecto de intermediación de mensajes, que proporciona alto rendimiento y baja latencia para la manipulación de fuentes de datos en tiempo real[42]

[2]Se traduce como interfaz de programación de aplicaciones de transferencia de estado representacional, es un estilo de arquitectura de software para sistemas en internet. Es un conjunto de rutinas que provee acceso a funciones de un software web mediante el protocolo HTTP

Figura 3.7: Dashboard de Grafana

Grafana En la instalación de Monasca mediante DevStack se añade al conjunto de herramientas el software de monitorización Grafana, una interfaz web para observar las métricas que se almacenan en la base de datos de Monasca. Esta interfaz web se puede acceder, por lo general, desde el puerto 3000 de donde se encuentra la interfaz de Horizon (servicio que provee a OpenStack de interfaz web), la url para acceder a grafana tendría la forma: `http://192.168.0.100:3000`. Un ejemplo de la interfaz se puede ver en la figura 3.7. La mayoría de gráficas se encuentran pregeneradas, pero el modo de añadir nuevas gráficas es muy intuitivo y la mayoría de los campos son autocompletados.

Logs Monasca utiliza el complemento `monasca-log-api`, es una Api REST que colecciona los registros de los distintos servicios de OpenStack y son registrados por los agentes. Los agentes que se utilizan pueden ser logstash (uno o varios archivos de registro) o beaver (únicamente un archivo de registro) que monitorizan los archivos, añaden metadatos, se autentifican en keystone y mandan los datos al gestor de logs; el gestor publica los registros en Kafka, un gestor de mensajes utilizado por Monasca; una vez los logs se encuentran en Kafka son transformados para su almacenamiento, publicación y utilización en métricas[43]. El principal problema de este complemento es que, al encontrarnos en DevStack, los logs de las unidades de systemd no pueden ser accedidos por los agentes de registro.

Pero las métricas de Monasca son bastante pobres para una investigación forense, como mucho se podrían detectar los datos de red para detectar una situación de DoS o DDoS, que a efectos del uso de la red sistema son prácticamente lo mismo. Por ello se decide incluir los complementos Ceilometer y Ceilosca

Ceilometer Se trata de una utilidad de monitorización muy similar a Monasca, más potente en cuanto a las métricas que puede obtener, pero carece de una api tan potente y acceso desde el dashboard. De entre las métricas de Ceilometer se destaca `identity.authenticate.failure` que como el nombre indica se produce con intentos de autenticación erróneos, que puede ser útil en un intento de ataque por fuerza bruta.

Ceilosca El nombre es el resultado de la mezcla de las palabras Ceilometer y Monasca, crea una pipeline[3] entre Ceilometer y Monasca, no es un programa en sí mismo si no que es un conjunto de archivos que modifican el funcionamiento de Ceilometer para que sus metricas vayan a parar a Monasca.

Congress

Como se vio en el punto 2.3.3, Congress es un plugin dedicado a auditar políticas. Congress se basa en una serie de mecanismos: una serie de drivers para cada servicio con el que interactúe y un gestor de políticas. Los drivers conectan los servicios de OpenStack con el gestor de políticas; recogen el estado del servicio y lo envían al gestor en forma de tablas. Por su parte el gestor se encarga de comprobar que una serie de reglas (políticas) se cumplen pudiendo este actuar de tres formas distintas: proactivamente, evitando que la política llegue a ser violada; reactivamente, actúa cuando la regla es rota, o no actuando, sirviendo como monitor de políticas.

En este proyecto se ha decidido que Congress audite en modo monitor una política que prohíbe que los usuarios de un proyecto se puedan conectar a la red de otro proyecto salvo que ambos proyectos estén de acuerdo en compartir sus redes. Es decir, si un usuario malintencionado se conecta a una red de un proyecto que no le pertenece (y no ha solicitado compartir dicho recurso), Congress mandará una alerta.

[3]Es una cadena de procesos, la salida de un proceso va a la entrada del siguiente

Para la redacción de políticas Congress utiliza Datalog, un lenguaje declarativo[4] derivado de SQL[5] y lógica de primer orden[6] en el que se relacionan las tablas generadas por los drivers y se realizan operaciones sobre estas para rellenar una tabla de errores con los datos de las tablas de los drivers.

Defínción de políticas en Congress Una política describe como deberían funcionar los servicios en la nube individualmente o en conjunto. Más concretamente, decide qué estados de la nube están permitidos y cuales no. Para ello el lenguaje de Congress no es una sucesión de acciones que se tienen que tomar, si no un conjunto de estados almacenados en tablas. Dos de estas tablas, la de error y warning son las que indican que estados son los que prohíbe la política. Un ejemplo de definición de una tabla de estados sería:

```
error(port_id, ip1, ip2) :-
  port(port_id, ip1),
  port(port_id, ip2),
  not equal(ip1, ip2);
```

Como se puede ver es una tabla que muestra errores, en la primera linea se indican el nombre y los campos de la tabla y se pone un símbolo que indica que comienza la definición. Como se puede observar en la definición se toman valores de la tabla llamada "port" buscando puertos con más de una dirección IP y se comprueba que no tenga asociadas más de una dirección IP.

3.1.4. Actores implicados

La instalación de DevStack con los complementos actuales deja un sistema con 3 proyectos principales y varios usuarios, estos son:

- Admin: en este proyecto se encuentra el usuario admin, que como indica su nombre es el administrador del sistema y muchas de las funcionalidades de control sólo se pueden acceder desde los usuarios en este proyecto.

[4]La programación declarativa es un estilo de programación en el que el programador especifica qué debe computarse más bien que cómo deben realizarse los cómputos.[44]

[5]SQL es un lenguaje específico para la gestión de bases de datos relacionales.

[6]Una lógica de primer orden, también llamada lógica predicativa, lógica de predicados o cálculo de predicados, es un sistema formal diseñado para estudiar la inferencia en los lenguajes de primer orden. Los lenguajes de primer orden son, a su vez, lenguajes formales con cuantificadores que alcanzan solo a variables de individuo, y con predicados y funciones cuyos argumentos son solo constantes o variables de individuo.[45]

- Service: en este proyecto se encuentra un usuario por cada servicio que existe salvo los usuarios generados por Monasca que se encuentran en su propio proyecto.
- Mini-mon: es el proyecto de los usuarios de Monasca y es desde el único que se puede acceder a las métricas y configurar las alarmas de este servicio.

Para finalizar la configuración se crean dos proyectos nuevos con un usuario cada uno: alice, del proyecto "defender-users" y eve, del proyecto "evil-users". El nombre es bastante claro, alice se trata de un usuario bienintencionado en la nube, al que hay que proteger, y eve es el usuario malintencionado, que intentará aprovecharse de los fallos del sistema para atacar a otros usuarios o al mismo sistema.

3.2. Preparación del demostrador

Durante la preparación del demostrador se implementó el sistema hasta el punto en el que se encontraría justo antes de ser publicado para producción por lo tanto en esta sección se hablará de: la creación de la máquina virtual, la instalación del sistema base y OpenStack y la configuración que se aplicará a los distintos componentes.

3.2.1. Especificaciones de la máquina virtual

El primer paso para la creación del entorno es la especificación de las características de la máquina virtual, para ello se utiliza el asistente de VirtualBox, este paso es bastante sencillo, se escoge utilizar 8GB de memoria RAM y 4 procesadores, se indica que el sistema operativo va a ser un Linux-Ubuntu de 64 bits y que se va a utilizar un disco virtual de 60GB de tamaño fijo, lo que hace que el sistema sea algo más rápido. Tras especificar el hardware virtual se configura la red en modo adaptador puente, y se le añade una imagen ISO del instalador del sistema operativo Ubuntu server 16.04, tener una interfaz gráfica lo único que hará es consumir recursos, lo cual, no es interesante.

3.2.2. Instalación del sistema

Tras esto se arranca la máquina virtual y se sigue el proceso de instalación, durante el cual se configuran una serie de parámetros:

- La red se configura con IP estática, en este caso 192.168.0.115/24.
- Se instala el servidor SSH para, desde el primer momento, poder establecer una sesión con la máquina virtual.
- Se instala el servidor mail para que Monasca pueda enviar notificaciones a través de email.

Cuando el sistema operativo está configurado se procede a la instalación de DevStack según está indicado en su documentación[46]. Este proceso es básicamente copiar y pegar los comandos indicados.Dependiendo de los complementos del sistema que se quieran instalar se necesitará configurar el archivo local.conf que se encuentra en el Apéndice I. Que a continuación se describe.

local.conf

El archivo local.conf es un archivo tipo INI[7] modificado que introduce un sistema de meta-secciones en cabeceras que lleva información sobre que archivos han de ser modificados[12].

A continuación se distinguen las distintas partes del archivo utilizado

```
[[local|localrc]]
```

Esta línea es la cabecera de una sección, indica que se describe la composición del archivo .localconf.auo que almacena todas las configuraciones indicadas por el usuario, no se utiliza ninguna otra cabecera en el archivo, pero se pueden especificar otras opciones.

```
ADMIN_PASSWORD=1234
DATABASE_PASSWORD=$ADMIN_PASSWORD
RABBIT_PASSWORD=$ADMIN_PASSWORD
SERVICE_PASSWORD=$ADMIN_PASSWORD
MYSQL_PASSWORD=$ADMIN_PASSWORD
SERVICE_TOKEN=111222333444
```

Esta es la configuración obligatoria de DevStack, se le dan contraseñas a distintos servicios de autenticación de la plataforma, la contraseña de MySQL y el token de servicio no son necesarios en una configuración mínima.

[7]Los archivos INI son archivos de configuración estandarizados de manera no oficial que sirven para varios sistemas operativos y software

```
LOGFILE=$DEST/logs/stack.sh.log
LOGDIR=$DEST/logs
LOG_COLOR=False

USE_PYTHON3=True
PIP_UPGRADE=True

HOST_IP=192.168.0.115
```

Las tres primeras líneas indican como se almacena el log ya que de no hacerlo se mostraría por pantalla y en caso de error la información de pantalla no siempre es suficiente, por lo tanto resulta bastante útil. Las dos siguientes indican que se fuerza el uso de Python en su versión 3, ya que por defecto se utiliza la 2 y además se obliga a utilizar una versión actualizada del gestor de paquetes de Python, pip; sin los cuales el programa de instalación daba muchos problemas con los plugins que se decidieron instalar. En la línea final se especifica la IP del equipo, ya que de no hacerlo suele dar problemas, a pesar de que el manual del indique que no es necesario

```
enable_plugin congress https://github.com/openstack/
    congress
enable_plugin monasca-api https://github.com/openstack/
    monasca-api
enable_plugin ceilometer https://github.com/openstack/
    ceilometer stable/rocky
enable_plugin ceilosca https://github.com/openstack/
    monasca-ceilometer stable/rocky

enable_service ceilosca
```

Se especifican los plugins que se quieren instalar y el repositorio desde donde se instalan y finalmente se ejecuta el comando `./stack.sh` que tras un rato dejará el sistema operativo con OpenStack instalado, el resultado del script tras casi 2 horas de ejecución se puede ver en la figura 3.8.

3.2.3. Preparación de los complementos

Con el sistema ya instalado se comprueba que funciona, creando una instancia de CirrOS, una imagen Linux mínima, preparada por defecto para utilizar en OpenStack como máquina de pruebas. Cuando se ha comprobado que un usuario puede crear sus propias instancias se procede a eliminar todas las redes, usuarios y proyectos que OpenStack crea a modo de demostración y desde el software de virtualización se toma una instantánea del sistema

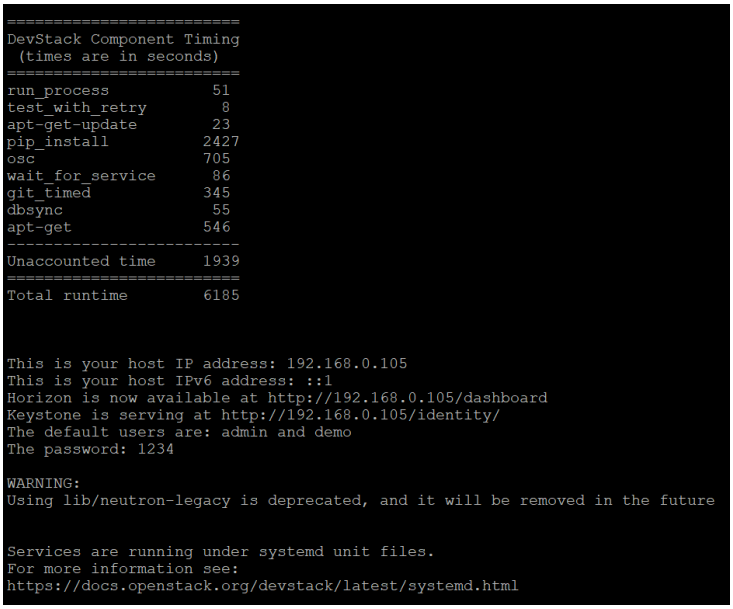

Figura 3.8: Resultado de la instalación de OpenStack mediante DevStack

para poder apagar la máquina virtual sin ocasionar problemas en el entorno. Tras esto se procede a configurar las redes y usuarios de OpenStack así como los plugins que se añadieron.

Configuración de OpenStack

Lo primero que se realizó fue la eliminación desde la interfaz de usuario de OpenStack el "router1" y la red "private", que pertenecen al proyecto demo; a continuación se eliminaron los usuarios demo y alt-demo y los proyectos con el mismo nombre.

Seguidamente se crearon los proyectos defender_users y evil_users, el primero de los proyectos tendrá como usuario a "alice", el segundo a "eve". Y finalmente se descargaron los ficheros rc[8] de ambos usuarios y del administrador. Un ejemplo del archivo rc se puede encontrar en el Apéndice II

Dado que la creación de redes dio problemas cuando se ejecutó desde horizon, la interfaz web, la creación de las redes para cada proyecto se realizó desde la consola de la máquina en la que se encuentra instalado OpenStack.

[8]El fichero rc es un archivo que establece las variables de entorno en un sistema linux para que, desde consola, se pueda interactúar con los distintos servicios de OpenStack

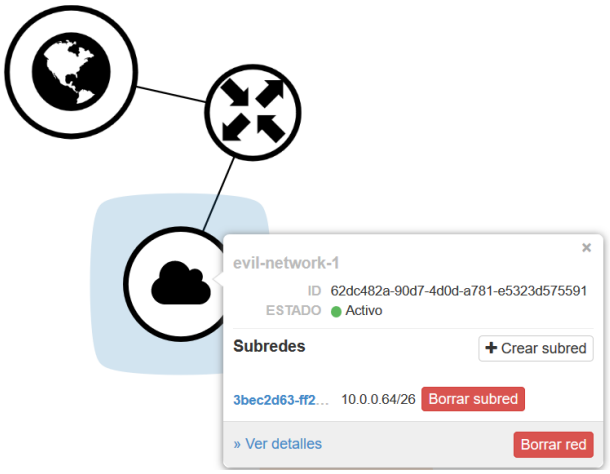

Figura 3.9: Red del usuario malintencionado

Para la creación de la red de usuarios atacantes se utilizaron los siguientes comandos:

```
. evil_users-openrc.sh
openstack network create evil-network-1
openstack subnet create evil-subnet-1.1 --network evil-
    network-1 --subnet-range 10.0.0.64/26
openstack router create evil-router-1
openstack router set evil-router-1 --external-gateway
    public
openstack router add subnet evil-router-1 evil-subnet
    -1.1
```

Y para la de usuarios bienintencionados:

```
. defender_users-openrc.sh
openstack network create defender-network-1 --share
openstack subnet create defender-subnet-1.1 --network
    defender-network-1 --subnet-range 10.0.0.0/26
openstack router create defender-router-1
```

```
openstack router set defender-router-1 --external-
    gateway public
openstack router add subnet defender-router-1 defender-
    subnet-1.1
```

Como se puede ver en la primera linea de la creación de la red de usuarios bienintencionados `defender-network-1`, se utiliza la opción `--share`, que es la única configuración que permite que una red pueda ser compartida por varios tennant. La topología generada con los comandos se puede ver en la figura 3.9 para la primera sucesión de comandos y en la figura 3.10 para la segunda.

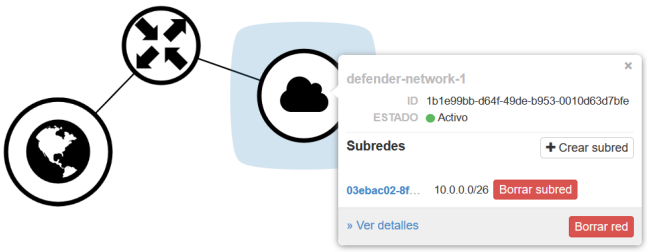

Figura 3.10: Red del usuario bienintencionado

Configuración de Congress

La política que se implementa se encuentra en el Apéndice III donde viene descrita a través de los comentarios del archivo. A continuación se explica el funcionamiento de cada una de las reglas:

project_groups_by_name Esta tabla define una serie de grupos de proyectos que pueden compartir redes y a qué grupo pertenece cada proyecto, es definida por el creador de la política, el grupo 1 es propiedad del proyecto service, el 2 defender_users y el 3 de evil_users y admin pertence a todos. La tabla resultante se muestra en el cuadro 3.1.

project_group_by_id Una tabla similar a `project_groups_by_name` pero a través del driver de keystone se cambia el nombre del proyecto por su ID.

Proyecto	Grupo
admin	1
admin	2
admin	3
service	1
defender_users	2
evil_users	3

Cuadro 3.1: project_groups_by_name

same_group Relaciona los ID de proyectos colocando por parejas, en cada fila, aquellos que se encuentran en el mismo grupo. La tabla resultante se muestra en el cuadro 3.2. Se puede ver como admin se encuentra compartiendo grupo con el resto de proyectos y el resto de proyectos se encuentran aislados unos de otros.

Proyecto A	Proyecto B
admin_project_id	service_project_id
admin_project_id	defender_users_project_id
admin_project_id	evil_users_project_id

Cuadro 3.2: project_groups_by_name

unexpected_server_to_port Almacena servidores conectados a puertos que no pertenezcan a su grupo de proyectos (por ejemplo una máquina del proyecto service conectada a un puerto perteneciente a evil_users), del driver de Neutron toma todos los puertos conectados a equipos virtuales y con el driver de Nova lee el ID de proyecto del equipo, comprueba si los ID del puerto y del equipo pertenecen a la misma fila de la tabla `same_group`.

unexpected_server_to_network Almacena servidores conectados a redes que no pertenezcan a su grupo de proyectos funciona de manera similar a `unexpected_server_to_port`, salvo que en vez de fijarse en puertos se fija en la red en la que se encuentra el puerto para decidir si la máquina virtual está conectada y si debiera estarlo.

warnig y error Estas tablas almacenan qué datos de otras tablas deben de ser mostrados como warning y error, en este caso la tabla de error se

llena con datos de `unexpected_server_to_network`, pues, por ejemplo, un equipo conectado a una red que no le pertenece puede empezar a distribuir código malicioso o leer mensajes que están siendo enviados. En el caso de `unexpected_server_to_port` se marca como una alarma (warning) esta situación se dará en casos en que 3 tennants se encuentren en el mismo grupo y el tenant A reserva un puerto para el tenant B y el tenant C se conecta a este, en este momento el tenant A pudiera pensar que los datos que recibe de la dirección que reservó para el tenant B son realmente del tenant B y el tenant C comience a enviar datos falsos incriminando al tenant B.

Cuando el fichero con la política fue correctamente creado se subió a openstack, se cargaron las variables de entorno del usuario admin, en cualquiera de los proyectos[9] utilizando los siguientes comandos, suponiendo que el archivo rc del administrador se llame `admin_openrc.sh` y el archivo de política sea `sniffing_machines.yaml`:

```
. admin_openrc.sh
openstack congress policy create-from-file
    sniffing_machines.yaml
```

Configuración de Monasca

Se ejecuta el comando `monasca metric-list` desde el usuario mini-mon que se encarga de las métricas del sistema, para conocer de que datos se dispone en un primer momento, el resultado, en parte, se muestra en la figura 3.11. Las métricas obtenidas se refieren principalmente a estado y uso del sistema, con métricas referentes al estado de servidores, uso de los procesadores y consumo de datos principalmente. Para ello se ha instalado el plugin llamado Ceilosca, que configura Ceilometer como fuente de datos de Monasca.

Estos datos, pueden llegar a ser útiles en una investigación forense, pero se prefiere investigar sobre otra fuente de métricas del sistema, Ceilometer. Esta fuente, entre otras medidas, recaba información de keystone sobre intentos de inicio de sesión en el grupo de métricas identity.authenticate [47]. En el repositorio de Ceilosca [48] se indican los pasos a seguir para la configuración de métricas.

[9]Congress obliga a que sea el usuario admin quien decida qué políticas son utilizadas y monitoriza que dichas políticas son cumplidas, aunque no necesita que este se encuentre trabajando en el proyecto admin

```
stack@OpenStack:/etc/ceilometer$ monasca metric-list
+--------------------------------+-------------------------------------+
| name                           | dimensions                          |
+--------------------------------+-------------------------------------+
| cpu.frequency_mhz              | service: monitoring                 |
|                                | hostname: OpenStack                 |
| cpu.idle_perc                  | service: monitoring                 |
|                                | hostname: OpenStack                 |
| cpu.idle_time                  | service: monitoring                 |
|                                | hostname: OpenStack                 |
| cpu.percent                    | service: monitoring                 |
|                                | hostname: OpenStack                 |
| cpu.stolen_perc                | service: monitoring                 |
|                                | hostname: OpenStack                 |
| cpu.system_perc                | service: monitoring                 |
|                                | hostname: OpenStack                 |
| cpu.system_time                | service: monitoring                 |
|                                | hostname: OpenStack                 |
| cpu.user_perc                  | service: monitoring                 |
|                                | hostname: OpenStack                 |
| cpu.user_time                  | service: monitoring                 |
|                                | hostname: OpenStack                 |
| cpu.wait_perc                  | service: monitoring                 |
|                                | hostname: OpenStack                 |
| cpu.wait_time                  | service: monitoring                 |
|                                | hostname: OpenStack                 |
```

Figura 3.11: Principio de la lista de métricas que, por defecto, se obtienen tras la instalación del sistema

En primer lugar se configuran los archivos encontrados en */etc/ceilometer*, el archivo `monasca_field_definitions.yaml` incluye las métricas y campos que son recogidos por Ceilometer y los trata para que puedan ser leídos por Monasca, en este caso se prueba con:

```
identity.authenticate:
        - outcome
        - initiator_name
        - initiator_host_addr
```

Se supone que *outcome* variará entre éxito y fracaso,e `initiator_name` e `initiator_host_addr` serán el usuario que se intenta conectar y la dirección IP desde la que se conecta. A continuación se modifica `pipeline.yaml` para que los datos generados en Ceilometer se manden a Monasca. Posteriormente se reinician los servicios de Ceilometer. Y por último se ejecuta el comando `monasca metric-list --dimensions datasource=ceilometer` que dará la lista de métricas que Ceilometer está mandando a la base de datos, el resultado del comando se muestra en la figura 3.12.

Como se puede ver solo se está almacenando la métrica `image.size`, esto se puede deber a que aún no se ha registrado ningún otro dato de ninguna métrica, se introducen varios éxitos y fallos en autenticación para que se comiencen a introducir métricas. Tras la introducción de credenciales erróneas las métricas no son añadidas, se estudian los logs de Ceilometer y Monasca y no se encuentra ningún indicio, se desconoce el error ocurrido.

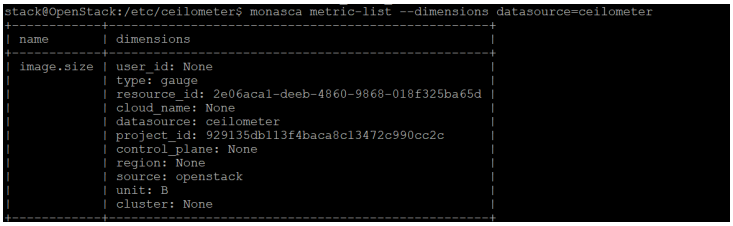

Figura 3.12: Métricas recogidas por Ceilometer y enviadas a Monasca

Por lo que se estudiará el complemento mediante métricas introducidas por el usuario. Se introduce una métrica llamada *test* y varios valores, mediante el comando `monasca metric-create test <valor>`.

Alarmas y Notificaciones Las alarmas y notificaciones de Monasca pueden ser gestionadas desde el dashboard, entrando como un usuario del proyecto mini-mon desde la subpestaña *alarm definitions*, en la pestaña *monitoring* se puede acceder al sistema de creación de métricas. Como se puede ver en la figura 3.13 se crea una alarma que se activará cuando la métrica test sobrepase el valor 200. En la sección detalles se especifica el nombre de la alarma y su severidad, en este caso se le llama *Test-alarm* y su severidad es baja, la severidad se puede establecer en baja, media, alta y crítica; en cuanto a la notificación, dado que aún no se ha establecido ningún método de notificación se deja por defecto (sin notificación).

Las notificaciones en Monasca son un método, a parte de la interfaz de usuario, para que un servicio pueda enviar una alarma. Actualmente hay tres métodos de notificación: email, webhook[10] y pagerduty[11]. En este caso simplemente se creará una notificación por email a una dirección generada en la página web `24hour.email`[12]. La configuración de la notificación se puede ver en la figura 3.14.

Finalmente se añade el método de notificación a la alarma que se creó antes y el sistema queda completamente configurado.

[10]Los weebhooks son llamadas http que envían un mensaje a distintos servicios web.

[11]Pagerduty es una plataforma SaaS para respuesta a incidentes en el campo de las tecnologías de la información.

[12]La página 24hour.email, al igual que muchas otras, se trata de un servicio de mail efímero en la que la bandeja de entrada solo funciona durante un tiempo limitado

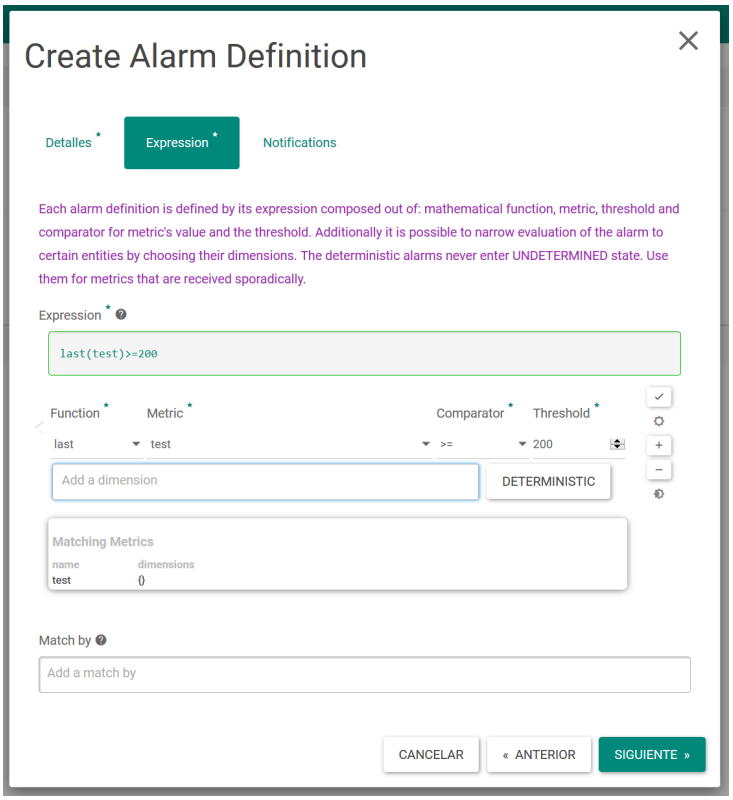

Figura 3.13: Definición de una alarma sobre la métrica *test*

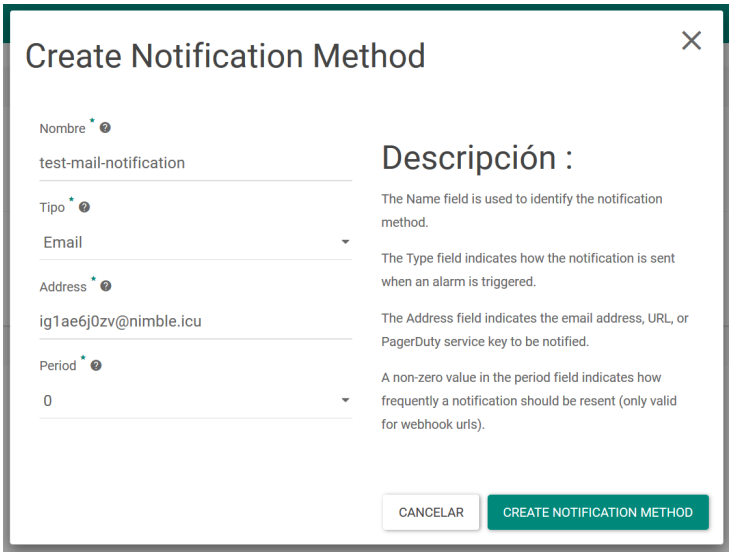

Figura 3.14: Definición de una notificación en Monasca

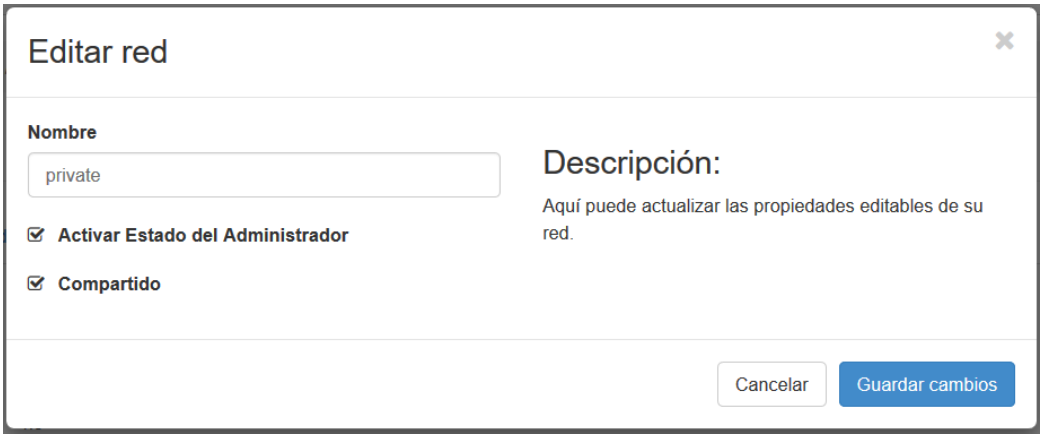

Figura 3.15: La red de usuarios bienintencionados se encuentra en modo compartido

3.3. Generación de evidencias

Con el sistema completamente configurado se procedió a la generación de usa serie de evidencias. En primer lugar se procede a simular el ataque MitM (Man in the middle), este tipo de ataque previsiblemente se generará una alarma en el dashboard de congress y dejará un rastro en los archivos de registro de neutron (redes) y nova (computación). Dado que no existe un archivo propiamente dicho, ya que DevStack no tiene la misma estructura que un entorno real; para ello se utilizará el programa `journalctl`. Posteriormente se generarán datos de la métrica test que sobrepasen el nivel de alarma y se estudiarán las alarmas generadas.

3.3.1. Ataque MitM en una red virtual

Como se ve en el apartado 3.2.3 la red del tenant de usuarios bienintencionados se encuentra en modo compartido con el fin de que usuarios del proyecto admin puedan desplegar instancias en ella. En la figura 3.15 se puede ver que la red se encuentra en modo compartido; por lo tanto entrado en la interfaz web desde el tennant de usuarios maliciosos se puede ver la otra red, como en la figura 3.16. No sólo se puede conocer dicha red si no que cualquier usuario, sea del tennant que sea, puede desplegar una instancia en ella. Pero al haber creado la política con Congress se monitoriza qué tennats se pueden haber conectado a redes de otros tennants.

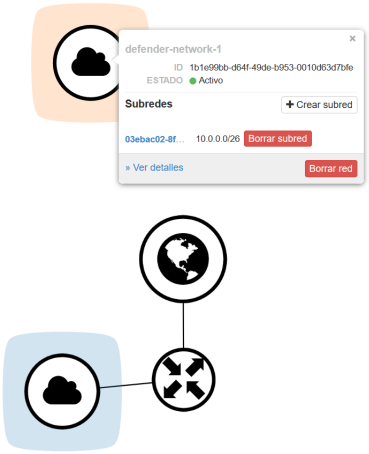

Figura 3.16: La red compartida se ve desde otros tennats

Desde la interfaz, conectado como el usuario eve del tennant evil_users se crea una máquina virtual en la red que no pertenece al tennant. La máquina desplegada se puede ver en la figuras 3.17 y 3.18. En este caso se utiliza una imagen de CirrOS preconfigurada en la instalación de OpenStack, esta imagen principalmente se utiliza para probar que la configuración del sistema es correcta.

Finalmente se accede a los logs[13] de nova y neutron mediante el uso de los comandos `sudo juornalctl -u devstack@n-*`, para Nova, y `sudo journalctl -u devstack@q-*`, para Neutron, cuyo resultado se encuentra en los apéndices IV y V respectivamente y serán analizados en la siguiente sección. En el caso de un entorno de OpenStack real la incautación de estos archivos se hubiese realizado tras obtener la alarma, en este caso se realiza antes porque estos archivos no son almacenados de manera permanente.

[13]Debido a la longitud de los archivos se ha decidido mostrar únicamente las líneas en las que aparecen las evidencias, con el fin de reducir el tamaño del apéndice.

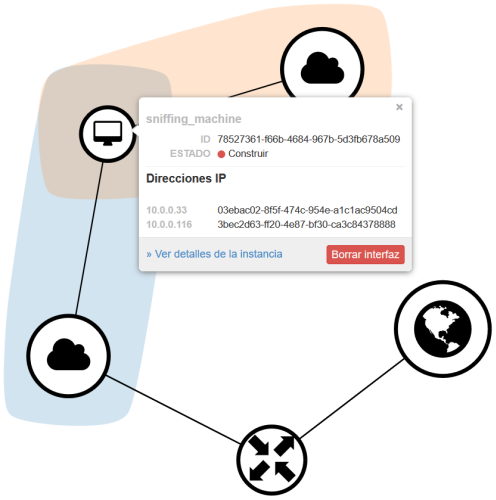

Figura 3.17: El usuario malicioso despliega una máquina en una red de otro proyecto

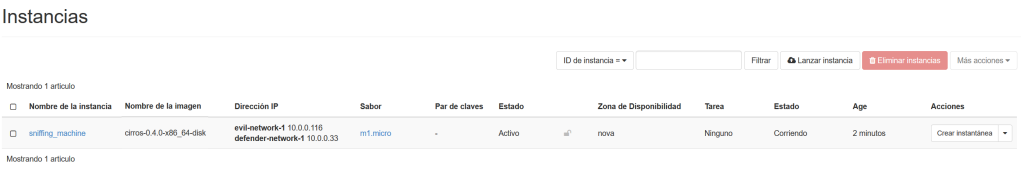

Figura 3.18: Máquina desplegada por el usuario malicioso

3.3.2. Generación de alarmas en Monasca

En un principio se pretendía utilizar el programa THC-Hydra, disponible en la suite básica de Kali Linux[49]. Hydra es un programa para realizar ataques de fuerza bruta a varios protocolos de nivel de aplicación, entre los que se encuentra HTTP[50], que es el utilizado por el dashboard de OpenStack, para ello se habría replicado un intento de login capturado por otro programa cambiando la contraseña en cada intento al usuario admin. Esto hubiese generado una gran cantidad de medidas de la métrica `identity.authenticate` que se intentó configurar en el punto 3.2.3 pero dado que la métrica no se consiguió crear, se paso al estudio de la métrica artificial "test".

Dado que solo se tiene la métrica llamada test para analizar se crean usa serie de medidas mediante el comando `monasca metric-create test <valor>`, como se hizo en el apartado anterior. Además se extraen los logs de keystone de una serie de intentos de autenticación erróneos, introducidos desde la página de identificación del dashboard para ver qué evidencias se pueden extraer, este archivo se encuentra en el apéndice VI.

3.4. Detección, preservación y análisis de evidencias

Tras la generación de las evidencias para los distintos servicios de OpenStack se pasará a la detección, preservación y análisis de estas, se pretende hallar una alarma en el dashboard de Congress de que ha habido una máquina conectada a una red en la que no debería estar y se mostrarán otras evidencias de este ataque que se pueden obtener desde la interfaz web; dado que esta información solo se puede hallar hasta que se elimina la máquina virtual, se analizarán los logs de Nova y Neutron para encontrar evidencias permanentes y replicables.

En el caso de Monasca, como no se pudo realizar la implementación inicial, se estudiará la generación de las alarmas y posteriormente los logs generados en Keystone después de realizar un intento fallido de autenticación. En el caso de que se pudiese haber utilizado Monasca para su detección en los logs hubiese aparecido el mismo registro replicado tantas veces como el ataque de fuerza bruta hubiese fallado.

Figura 3.19: Alarma generada en el dashboard de Congress

3.4.1. Evidencias en Congress

Tras la creación de la máquina que intenta emular un ataque del tipo MitM se encuentran 2 evidencias desde la interfaz web, una primera en la figura 3.19, es el dashboard de Congress en el que se muestra la violación de la política; como se vio en la sección 3.2.3 y en apéndice III el orden de los datos que se muestran son:

1. Nombre del proyecto que posee la máquina
2. ID del proyecto que posee la máquina
3. Nombre del proyecto dueño de la red
4. ID del proyecto dueño de la red
5. Nombre la máquina
6. ID de la máquina

Por otro lado desde la subpestaña “compute” en la pestaña de administración se puede acceder a los datos de la máquina haciendo click sobre su nombre, el cual se ha obtenido mediante la alarma anterior. Desde ahí se puede obtener la fecha de creación y el nombre de su creador, como se puede ver en la figura 3.20. Además, desde la pestaña anterior, se puede obtener un snapshot de la máquina para su posterior análisis.

Estas evidencias, en general, se pueden considerar una primera aproximación del proceso forense, presentan una serie de problemas: en primer lugar si el usuario malintencionado borra la máquina antes de que se detecte la incidencia por parte de un administrador las alarmas generadas en Congess

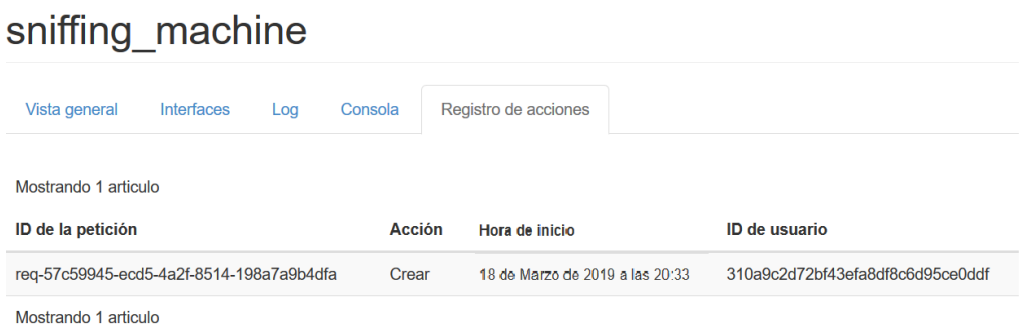

Figura 3.20: Registro de creación en Horizon

desaparecen junto a la información de la subpestña "compute", por otro lado no se puede generar una instantánea si se ha borrado. Para solucionar los anteriores problemas se plantea la inclusión de un registro de alarmas en Congress[36], en el caso del registro de creación se subsana con la información que se obtiene de los logs de Nova y Neutron y por su parte para la toma de snapshots se puede incluir un complemento anteriormente presentado, freezer, que permite automatizar la toma de instantáneas.

3.4.2. Evidencias en Monasca

Para acceder a las evidencias en Monasca se tiene que acceder desde un usuario del grupo mini-mon, después de la introducción de las métricas se comprueba la subpestaña de alarmas como se ve en la figura 3.21 se ha generado una alarma de severidad baja; para más información se accede al historial de la alarma que se muestra en la figura 3.22. Posteriormente se accede al servidor de correo y se comprueba si ha llegado el email que había sido configurado, no se encuentra ningún email. Aun no habiendo una notificación por email la alarma ha sido configurada correctamente, por lo que se supone que el problema se halla en el plugin monasca-notification. Para certificar que el fallo es del sistema y no de la configuración del mail se prueba a lanzar un webhook a `https://www.webhook.site/`, una página que ofrece una interfaz temporal para probar webhooks. Esta última opción sigue sin funcionar, por lo tanto se supone que el problema es del complemento.

En esta alarma se puede utilizar para encontrar la evidencia en el registro, pues cada incidencia se almacena con la fecha en la que se ha producido y se puede visionar con el momento en formato UTC o según la hora del

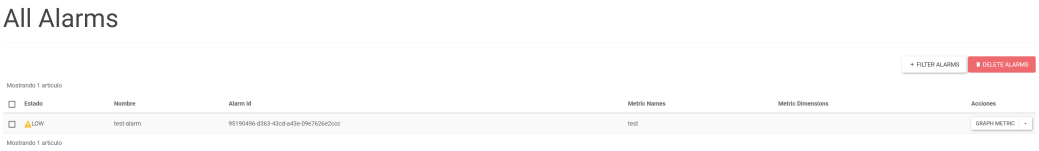

Figura 3.21: Alarma activada

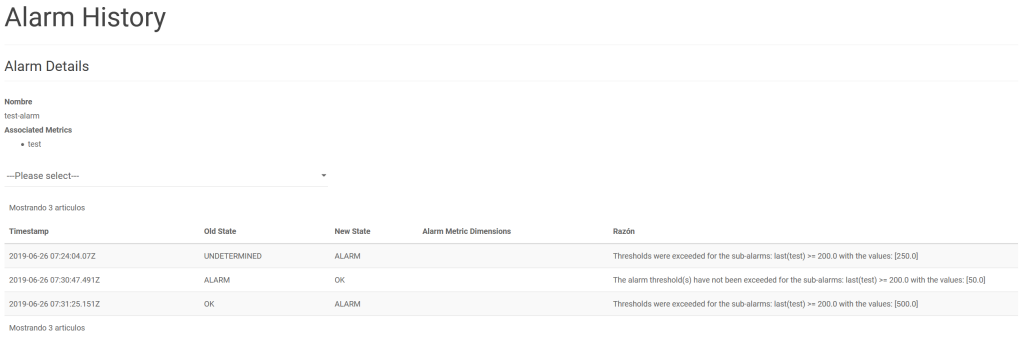

Figura 3.22: Historial de los estados de la alarma

explorador utilizado. La notificación mediante email no ha sido generada, posiblemente por errores en el plugin comentado anteriormente; pero esto no es tan importante, en comparación con que la alarma no hubiese podido llegar a generarse.

3.4.3. Evidencias en los registros

Finalmente se analizan los archivos de registros obtenidos mediante el programa `journalctl`, en estos registros se pretende encontrar la información válida para un proceso forense que pueda ser llevado ante un tribunal, por lo tanto lo que se espera, al menos, de estos archivos es que mantengan su integridad y que su adquisición sea replicable.

En este caso no se puede conseguir ninguno de estos requisitos, dado que los registros son almacenados de manera volátil y OpenStack carece de los medios para validar la integridad de estos logs, por ejemplo se propone el uso de árboles de Merkle[14][30] para mantener su integridad. Si se utiliza OpenStack sin DevStack se almacenan los logs en archivos en el disco duro.

[14]Los árboles de Merkle, Merkle Trees en inglés, es una estructura de datos para mantener la integridad de logs de apis y cortafuegos. Cada entrada del log se almacena con su hash y estos hashes se van agrupando en días, meses, años, instancias y usuarios y se almacena el hash de la concatenación de sus hashes.

Logs de Nova

Los logs de nova es de donde más información se ha podido extraer para el ataque MitM en el que el usuario malintencionado "eve" despliega una instancia en la red del tennant de "alice". En la primera entrada del log de nova, en el apéndice IV se ve la petición de la máquina virtual; de esto se encarga DevStack@n-api.service que es la unidad de systemd encargada de la api de Nova, de este log solamente se puede sacar qué usuario (y a que tennant pertenece) ha solicitado la creación de la instancia y el id de la instancia. En este caso el tennat Evil-users con el usuario eve solicita crear la máquina virtual con ID 060a9ccd-d723-4cd7-a5e6.

Las dos siguientes son algo menos interesantes, indican la creación de los puertos de la instancia, como se puede ver sigue perteneciendo a eve y mantiene el mismo ID.

El último log, sin lugar a dudas el más interesante, registra la conexión a las redes de la máquina virtual, de este log se extrae toda la información necesaria, salvo el usuario que solicita crear la máquina virtual. En primer lugar ovs_interfaceid indica que interfaz de la instancia se conecta a la red, por su parte tenant_id indica el tenant poseedor de la red, como se puede observar aparece dos veces con dos valores distintos, por lo tanto se puede observar que el equipo se encuentra conectado a dos redes de distintos tenants.

Con esta información se puede seguir el rastro que deja eve de la creación de su instancia malintencionada. En primer lugar se puede ver en el cuarto log que hay un equipo conectado a dos redes de dos tennats distintos, a partir de esto se obtiene el ID de la instancia. Con la ID de la instancia se puede buscar cualquiera de los tres primeros logs mostrados en los que se ve el nombre de usuario y proyecto al que pertenece el dueño de la instancia.

Logs de Neutron

Los logs extraídos de Neutron son menos interesantes. Los registros más importantes que se han podido sacar son las solicitudes y respuestas de conexión de un interfaz a una red. El primero y el tercero del apéndice son la petición que realiza "eve" a neutron para que su instancia conecte un interfaz a una red. Se puede observar usuario y tenant que realiza la solicitud, el id de la instancia, el id del tenant al que pertenece la instancia y el id de la red al que se quiere conectar.

Por su parte el segundo y cuarto registro son la confirmación de que la interfaz se ha conectado satisfactoriamente. Se puede observar a falta del usuario y el tenant que realiza la solicitud: el id de la instancia, la subred a la que está conectado el id del puerto y la dirección ip que le ha otorgado el DHCP.

Logs de Keystone

En los registros de keystone no se puede sacar una gran cantidad de información sobre un ataque de fuerza bruta contra el usuario admin.Se puede distinguir entre dos tipos de solicutudes: la autenticación fallida, correspondiente a los cuatro primeros registros, y los de usuario bloqueado, que se corresponden con los 4 últimos.

Dela inspección de estos archivos de registro no se puede obtener mucha información, lo que sí se puede deducir es que OpenStack tiene mecanismos de protección contra ataques de fuerza bruta. Si se entra en el archivo situado en /etc/keystone/keystone.conf se puede encontrar unas lineas que indican:

```
[security_compliance]
unique_last_password_count = 2
lockout_duration = 10
lockout_failure_attempts = 2
```

Lo que se corresponde con los logs registrados, los dos primeros intentos de acceso se corresponden con autenticación fallida y los siguientes se corresponden con usuario bloqueado. Por lo tanto no se ha podido encontrar ninguna evidencia del ataque de fuerza bruta, salvo que se ha producido; pero se ha encontrado que OpenStack ya posee una función para repeler este tipo de ataques.

4. Conclusiones y líneas futuras

La computación en la nube y su oferta de servicios supone un tremendo ahorro tanto para particulares como para empresas. Las bondades de esta tecnología son las suficientes como para virar el avance de la computación tradicional hacía la migración de tareas a la nube.

Desde el surgimiento de los primeros servicios de computación en la nube se ha evolucionado en gran medida, tanto en la diversificación de medios para virtualizar servidores a través de internet como en la tecnología para poder asegurarlos. La tecnología se ha liberalizado, habiendo varias soluciones de código abierto, con multitud de características distintas.

La investigación en el estudio de medidas de seguridad ha avanzado en gran medida y desde la publicación del reporte del N.I.S.T. [23] existen soluciones teóricas para muchos de los problemas; aunque, desgraciadamente, existen aún carencias en la implementación de estas medidas y en los medios para que los proveedores, desarrolladores e investigadores puedan formarse en el campo de las técnicas forenses y de la seguridad en general.

A lo largo de la descripción del demostrador se han utilizado dos complementos para la plataforma OpenStack que ayudan a solucionar las problemática de detección e identificación en entornos con una gran cantidad de datos que gestionar. Monasca, aunque no se ha conseguido mostrar, pretendía identificar ataques de fuerza bruta contra el servicio de autenticación (keystone); que de otra manera se debería de haber identificado mediante un exhaustivo estudio de los logs. Aún así Monasca se ha mostrado como efectiva herramienta de monitorización de recursos y con una correcta configuración del servicio de telemetría (ceilometer) se puede conseguir un servicio de alarmas a nivel de proveedor bastante completo. En el caso de Congress se ha conseguido identificar un equipo que podía estar recibiendo datos de manera ilegítima de una red virtualizada. Para hacerlo sin este complemento hubiera sido necesario comprobar tennant por tennant las redes y las máquinas que

posee y las interfaces de red de dichas máquinas, tarea que en un entorno tan elástico es necesario automatizar.

En un principio también se pretendió utilizar la herramienta de logs de Monasca, pero dado las características de la implementación mediante DevStack esto no fue posible, por lo que finalmente se accedió a los registros de las unidades systemd. La api para logs podría ser una útil herramienta para la colección de evidencias, pero aún se carece de medios de análisis eficaces para la gran cantidad de datos que se generan en la nube.

El proyecto, en general, interesante para asignaturas del grado en Ingeniería de Tecnologías de Telecomunicación de la rama de telemática y el grado en Ingeniería Informática. Especialmente el funcionamiento de la aplicación monasca, en aquellas asignaturas relacionadas con gestión. El sistema, en general, puede ser utilizado como un entorno de simulación de redes de computadores.

En cuanto a posibles mejoras a lo expuesto en este proyecto y otras líneas a seguir en el campo de las técnicas forenses en entornos cloud podrían ser las siguientes:

1. Realizar la implementación del mismo entorno de pruebas sobre un entorno OpenStack sobre varios servidores, simulados o no, para así poder utilizar mayor cantidad de funcionalidades de los servicios.

2. Desarrollar nuevas normas de detección de ataques de red mediante el uso de Congress. Ya sea mediante los drivers ya existentes o la creación de un nuevo driver para algún otro servicio.

3. Utilizar Monasca como servicio de monitorización de instancias para los usuarios. Crear uno o varios programas ejecutados en las instancias, que envíen datos a la api de Monasca, un usuario cualquiera puede leer los datos enviados desde su tennant, o utilizar la api de logs para el mismo objetivo (centralizar los logs de las instancias de un tennant).

4. Explorar otras herramientas y entornos para la detección de ataques y colección de evidencias en entornos cloud.

Bibliografía

[1] A. Regalado, "Who coined 'cloud computing'?," *technologyreview.com*, 2011. Disponible en `https://www.technologyreview.com/s/425970/who-coined-cloud-computing/`.

[2] E. Schmidt, "Conversation with Eric Schmidt hosted by Danny Sullivan," in *Search engine strategies conference*, 2006. Disponible en `http://www.google.com/press/podium/ses2006.html`.

[3] Amazon, "Announcing Amazon Elastic Compute Cloud (Amazon EC2)," 2006. Disponible en `https://aws.amazon.com/es/about-aws/whats-new/2006/08/24/announcing-amazon-elastic-compute-cloud-amazon-ec2---beta/`.

[4] B. Sosinsky, *Cloud computing bible*. Wiley Publishing, Inc., 2011.

[5] R. L. Krutz and R. D. Vines, *Cloud security: a comprehensive guide to secure cloud computing*. Oxford University Press, 2008.

[6] Gartner, "Gartner says cloud computing will be as influential as e-business," *Stamford Conn.*, 2008. Disponible en `https://www.gartner.com/newsroom/id/707508`.

[7] P. Mell and T. Grance, "The NIST definition of cloud computing," *NIST*, 2011. Disponible en `https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-145.pdf`.

[8] T. Erl, *Cloud computing: concepts, technology & architecture*. Arcitura Education, Inc., 2013.

[9] I. Ramírez, "Máquinas virtuales: qué son, cómo funcionan y cómo utilizarlas," *Xataka*, 2016. Disponible en `https://www.xataka.com/especiales/maquinas-virtuales-que-son-como-funcionan-y-como-utilizarlas`.

[10] M. T. Jones, "La anatomía de un hipervisor linux," *IBM*, 2009. Disponible en http://web.archive.org/web/20160806114016/http://www.ibm.com/developerworks/ssa/library/l-hypervisor/index.html.

[11] Wikipedia, "Licencia de código abierto," 2018. Disponible en https://es.wikipedia.org/wiki/Licencia_de_c%C3%B3digo_abierto.

[12] OpenStack, *OpenStack Stein Documentation*, 2019. Disponible en https://docs.openstack.org/stein/.

[13] New Media Rights, "Open source licensing guide," 2015. Disponible en https://www.newmediarights.org/open_source/new_media_rights_open_source_licensing_guide.

[14] CloudStack, *CloudStack Documentation*, 2017. Disponible en http://docs.cloudstack.apache.org/en/latest/index.html.

[15] GNU, "Gnu licenses," 2007. Disponible en https://www.gnu.org/licenses/gpl-3.0.html.

[16] S. J. Vaughan-Nichols, "Eucalyptus 3.3 arrives with additional amazon cloud features," 2013. Disponible en https://www.zdnet.com/article/eucalyptus-3-3-arrives-with-additional-amazon-cloud-features/.

[17] D. Chappell, "Introducing the windos azure platform," 2009. Disponible en http://davidchappell.com/writing/white_papers/Windows_Azure_Platform_v1.3--Chappell.pdf.

[18] G. Palmer, "A roadmap for digital forensic research," *Digital forensic research workshop*, 2001. Disponible en http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.14.8953&rep=rep1&type=pdf.

[19] A. Eleyan and D. Eleyan, "Forensic Process as a Service (FPaaS) for cloud computing," *European intelligence and security informatics conference*, 2015. Disponible en https://ieeexplore.ieee.org/document/7379741/.

[20] S. Alharbi, J. Weber-Jahnke, and I. Traore, "The proactive and reactive digital forensics investigation process: A systematic literature review," *International journal of security and its applications*, 2011. Disponible en https://www.uvic.ca/engineering/ece/isot/assets/docs/IJSIA.pdf.

[21] B. Martini and K. K. R. Choo, "Cloud forensic technical challenges and solutions: a snapshot," *IEEE Cloud Computing*, 2014. Disponible en https://ieeexplore.ieee.org/document/7057622.

[22] K. Ruan, J. Carthy, T. Kechadi, and M. Crosbie, "Cloud forensics: an overview," *Centre for Cybercrime Investigation*, 2011. Disponible en https://www.researchgate.net/publication/229021339_Cloud_forensics_An_overview.

[23] J. Dykstra, L. Gowen, R. Jackson, O. S. Reemelin, E. F. Rojas, K. Ruan, M. Salim, K. E. Stavinoha, L. P. Taylor, and K. R. Zatyko, "Cloud Computing Forensic Science Challenges," tech. rep., NIST Cloud Computing Forensic Science Working Group Information Technology Laboratory, 2014. Disponible en https://csrc.nist.gov/csrc/media/publications/nistir/8006/draft/documents/draft_nistir_8006.pdf.

[24] Y. Lei and Y. Cui, "Research on live forensics in cloud environment," 2013. Disponible en https://pdfs.semanticscholar.org/fc64/01e7c217d7ec0d7d1dac2f3552e81367ab64.pdf.

[25] S. Alqahtany, N. Clarke, S. Furnell, and C. Reich, "Cloud Forensics: A Review of Challenges, Solutions and Open Problems," 2015. Disponible en https://ieeexplore.ieee.org/document/7149635.

[26] J. Carles, "¿qué es y para que sirve un sandbox en la informática?," 2017. Disponible en https://geekland.eu/que-es-y-para-que-sirve-un-sandbox/.

[27] J. Li, X. Chen, Q. Huang, and D. S. Wong, "Digital provenance: Enabling secure data forensics in cloud computing," 2013. Disponible en http://faratarjome.ir/u/media/shopping_files/store-EN-1486561013-8283.pdf.

[28] J. Dykstra and A. T. Sherman, "Acquiring forensic evidence from infrastructure-as-a-service cloud computing: Exploring and evaluating tools, trust, and techniques," 2012. Disponible en https://www.sciencedirect.com/science/article/pii/S1742287612000266.

[29] M. Damshenas, A. Dehghantanha, R. Mahmoud, and S. bin Shamsuddin, "Forensics investigation challenges in cloud computing environments," 2012. Disponible en https://ieeexplore.ieee.org/document/6246092.

[30] J. Dykstra and A. T. Sherman, "Design and implementation of FROST - digital forensic tools for the OpenStack cloud computing platform," 2013. Disponible en https://www.dfrws.org/sites/default/files/session-files/paper-design_and_implementation_of_frost_-_digital_forensic_tools_for_the_openstack_cloud_computing_platform.pdf.

[31] G. Geethakumari and A. Belorkar, "Regenerating cloud attack scenarios using LVM2 based system snapshots for forensic analysis," 2012. Disponible en https://www.researchgate.net/publication/275405569_Regenerating_Cloud_Attack_Scenarios_using_LVM2_based_System_Snapshots_for_Forensic_Analysis.

[32] J. Bongertz, "Wireshark vs. "The Cloud": Capturing packets in virtual environments," 2015. Disponible en https://sharkfestus.wireshark.org/assets/presentations15/34.pdf.

[33] OpenStack, *Ceilometer Documentation*, 2017. Disponible en https://docs.openstack.org/ceilometer/latest/.

[34] OpenStack, *Monasca Documentation*, 2018. Disponible en https://wiki.openstack.org/wiki/Monasca.

[35] The Snort Project, *Snort user manual*, 2019. Disponible en http://manual-snort-org.s3-website-us-east-1.amazonaws.com/snort_manual.html.

[36] OpenStack, *Congress documentation*, 2017. Disponible en https://wiki.openstack.org/wiki/Congress.

[37] OpenStack, *Freezer documentation*, 2015. Disponible en https://wiki.openstack.org/wiki/Freezer.

[38] OpenStack, "Install openstack services," 2019. Disponible en https://docs.openstack.org/install-guide/openstack-services.html.

[39] OpenStack, *Using Systemd in DevStack*, 2018. Disponible en https://docs.openstack.org/devstack/latest/systemd.html.

[40] OpenStack, "Packstack repository," 2019. Disponible en https://github.com/redhat-openstack/packstack.

[41] OpenStack, "Devstack rocky documentation," 2019. Disponible en https://docs.openstack.org/devstack/rocky/index.html.

[42] E. Mouzakitis, "What is Kafka?," 2016. Disponible en https://www.datadoghq.com/blog/monitoring-kafka-performance-metrics/.

[43] Hewlett Packard Enterprise Development LP, *Monasca/Logging*, 2019. Disponible en https://wiki.openstack.org/wiki/Monasca/Logging.

[44] P. J. Iranzo, "Programación declarativa," 2007. Disponible en http://www.inf-cr.uclm.es/www/pjulian/teaching/sl_apPD.pdf.

[45] S. Blackburn, *The Oxford Dictionary of Philosophy*. Wiley Publishing, Inc., 2010.

[46] OpenStack, *DevStack Installation Guide*, 2017. Disponible en https://docs.openstack.org/devstack/latest/guides/single-vm.html.

[47] OpenStack, "Telemetry measurements," 2019. Disponible en https://docs.openstack.org/ceilometer/pike/admin/telemetry-measurements.html.

[48] Ceilosca developers, "Ceilosca sample local.conf," 2018. Disponible en https://github.com/openstack/monasca-ceilometer/blob/master/devstack/sample-local.conf.

[49] Offesive Security, "Kali linux tools listing," 2019. Disponible en https://tools.kali.org/tools-listing.

[50] THC, "Hydra package description." Disponible en https://tools.kali.org/password-attacks/hydra.

Apéndice I - local.conf

```
[[local|localrc]]
ADMIN_PASSWORD=1234
DATABASE_PASSWORD=$ADMIN_PASSWORD
RABBIT_PASSWORD=$ADMIN_PASSWORD
SERVICE_PASSWORD=$ADMIN_PASSWORD
MYSQL_PASSWORD=$ADMIN_PASSWORD
SERVICE_TOKEN=111222333444

LOGFILE=$DEST/logs/stack.sh.log
LOGDIR=$DEST/logs
LOG_COLOR=False

USE_PYTHON3=True
PIP_UPGRADE=True

HOST_IP=192.168.0.105

enable_plugin congress https://github.com/openstack/
    congress
enable_plugin monasca-api https://github.com/openstack/
    monasca-api
enable_plugin ceilometer https://github.com/openstack/
    ceilometer stable/rocky
enable_plugin ceilosca https://github.com/openstack/
    monasca-ceilometer stable/rocky

enable_service ceilosca

enable_service rabbit
enable_service mysql
enable_service key
```

```
enable_service tempest
```

Apéndice II - Archivo rc (admin-openrc.sh)

```
#!/usr/bin/env bash
# To use an OpenStack cloud you need to authenticate
    against the Identity
# service named keystone, which returns a **Token** and
    **Service Catalog**.
# The catalog contains the endpoints for all services
    the user/tenant has
# access to - such as Compute, Image Service, Identity,
    Object Storage, Block
# Storage, and Networking (code-named nova, glance,
    keystone, swift,
# cinder, and neutron).
#
# *NOTE*: Using the 3 *Identity API* does not
    necessarily mean any other
# OpenStack API is version 3. For example, your cloud
    provider may implement
# Image API v1.1, Block Storage API v2, and Compute API
    v2.0. OS_AUTH_URL is
# only for the Identity API served through keystone.

export OS_AUTH_URL=http://192.168.0.105/identity/v3

# With the addition of Keystone we have standardized on
    the term **project**
# as the entity that owns the resources.

export OS_PROJECT_ID=2fdec5068eeb4f3f9e5f22a6a8bbc531
```

```
export OS_PROJECT_NAME="admin"

export OS_USER_DOMAIN_NAME="Default"
if [ -z "$OS_USER_DOMAIN_NAME" ];
  then unset OS_USER_DOMAIN_NAME;
fi
export OS_PROJECT_DOMAIN_ID="default"
if [ -z "$OS_PROJECT_DOMAIN_ID" ];
  then unset OS_PROJECT_DOMAIN_ID;
fi
# unset v2.0 items in case set
unset OS_TENANT_ID
unset OS_TENANT_NAME
# In addition to the owning entity (tenant), OpenStack
    stores the entity
# performing the action as the **user**.
export OS_USERNAME="admin"
# With Keystone you pass the keystone password.
echo "Please enter your OpenStack Password for project
    $OS_PROJECT_NAME as user $OS_USERNAME: "
read -sr OS_PASSWORD_INPUT
export OS_PASSWORD=$OS_PASSWORD_INPUT
# If your configuration has multiple regions, we set
    that information here.
# OS_REGION_NAME is optional and only valid in certain
    environments.
export OS_REGION_NAME="RegionOne"
# Don't leave a blank variable, unset it if it was
    empty
if [ -z "$OS_REGION_NAME" ];
  then unset OS_REGION_NAME;
fi
export OS_INTERFACE=public
export OS_IDENTITY_API_VERSION=3
```

Apéndice III - politica.yaml

```
---
name: SniffingMachines
description: >
  Identifica maquinas en redes de proyectos a los que
     no pertenecen, a menos que dichos proyectos se
     configuren como parte del mismo grupo, entonces se
      espera inter-operabilidad.
rules:
  -
    comment: >
      Se define al proyecto admin como perteneciente a
         todos los grupos de proyectos.
    rule: >
      project_groups_by_name(1, 'admin')
  -
    rule: >
      project_groups_by_name(2, 'admin')
  -
    rule: >
      project_groups_by_name(3, 'admin')
  -
    comment: >
      Se define al proyecto service como perteneciente
         al primer grupo.
    rule: >
      project_groups_by_name(1, 'service')
  -
    comment: >
      Se define al proyecto defender_users como
         perteneciente al segundo grupo.
    rule: >
```

```
    project_groups_by_name(2, 'defender_users')
-
  comment: >
    Se define al proyecto evil_users como
      perteneciente al tercer grupo.
  rule: >
    project_groups_by_name(3, 'evil_users')
-
  comment: >
    Transforma la tabla project_groups_by_name en
      project_groups_by_id
  rule: >
    project_groups_by_id(group_id, project_id) :-
      project_groups_by_name(group_id, project_name),
      keystonev3:projects(name=project_name, id=
        project_id)
-
  comment: >
    Define una tabla de grupos de proyectos, entre
      los que se espera inter-operabilidad.
  rule: >
    same_group(project_a, project_b) :-
      project_groups_by_id(group_id, project_a),
      project_groups_by_id(group_id, project_b)
-
  comment: >
    Identifica VMs conectadas a puertos que no
      pertenecen a su grupo de proyectos.
  rule: >
    unexpected_server_to_port(server_project_id,
      port_project_id, server_id, server_name) :-
      neutronv2:ports(id=port_id, tenant_id=
        port_project_id, network_id=network_id,
        device_id=server_id),
      nova:servers(id=server_id, name=server_name,
        tenant_id=server_project_id),
      not same_group(port_project_id,
        server_project_id)
-
  comment: >
    Identifica VMs conectadas a redes que no
```

```
      pertenecen a su grupo de proyectos.
  rule: >
    unexpected_server_to_network(server_project_id,
      network_project_id, server_id, server_name) :-
      neutronv2:ports(id=port_id, network_id=
        network_id, device_id=server_id),
      nova:servers(id=server_id, name=server_name,
        tenant_id=server_project_id),
      neutronv2:networks(id=network_id, tenant_id=
        network_project_id),
      not same_group(server_project_id,
        network_project_id)
-
  comment: >
    Advierte sobre las VMs pertenecientes a
      unexpected_server_to_port.
  rule: >
    warning(server_project_name, server_project_id,
      port_project_name, port_project_id,
      server_name, server_id) :-
      unexpected_server_to_port(server_project_id,
        port_project_id, server_id, server_name),
      keystonev3:projects(name=server_project_name,
        id=server_project_id),
      keystonev3:projects(name=port_project_id, id=
        port_project_name)
-
  comment: >
    Marca un error en las VMs pertenecientes a
      unexpected_server_to_network.
  rule: >
    error(server_project_name, server_project_id,
      network_project_name, network_project_id,
      server_name, server_id) :-
      unexpected_server_to_network(server_project_id,
        network_project_id, server_id, server_name)
        ,
      keystonev3:projects(name=server_project_name,
        id=server_project_id),
      keystonev3:projects(name=network_project_name,
        id=network_project_id)
```

Apéndice IV - Log de Nova

```
jun 30 14:28:17 OpenStack devstack@n-api.service
    [26055]: DEBUG nova.compute.api [None req-cd261f50-
    e8a2-41d4-a35b-9ce1b4862ed6 evil-users eve] [
    instance: 060a9ccd-d723-4cd7-a5e6-3f590bc145fb]
    block_device_mapping [BlockDeviceMapping(
    attachment_id=<?>,boot_index=0,connection_info=None,
    created_at=<?>,delete_on_termination=False,deleted
    =<?>,deleted_at=<?>,destination_type='volume',
    device_name=None,device_type=None,disk_bus=None,
    guest_format=None,id=<?>,image_id='123d2dd1-0221-41
    f7-bdba-c5d7e726a6e2',instance=<?>,instance_uuid
    =<?>,no_device=False,snapshot_id=None,source_type='
    image',tag=None,updated_at=<?>,uuid=<?>,volume_id=
    None,volume_size=1,volume_type=None)] {{(pid=26058)
    _bdm_validate_set_size_and_instance /opt/stack/nova/
    nova/compute/api.py:1465}}

jun 30 14:28:30 OpenStack nova-compute[2676]: DEBUG
    nova.network.neutronv2.api [None req-cd261f50-e8a2
    -41d4-a35b-9ce1b4862ed6 evil-users eve] [instance:
    060a9ccd-d723-4cd7-a5e6-3f590bc145fb] Successfully
    created port: 15f1b97d-def9-4b18-b751-7c51e5d8cd69
    {{(pid=2676) _create_port_minimal /opt/stack/nova/
    nova/network/neutronv2/api.py:544}}

jun 30 14:28:33 OpenStack nova-compute[2676]: DEBUG
    nova.network.neutronv2.api [None req-cd261f50-e8a2
    -41d4-a35b-9ce1b4862ed6 evil-users eve] [instance:
    060a9ccd-d723-4cd7-a5e6-3f590bc145fb] Successfully
    created port: 57377e4a-25f8-463d-902b-4e27a9c4df38
```

{{(pid=2676) _create_port_minimal /opt/stack/nova/nova/network/neutronv2/api.py:544}}

jun 30 14:28:49 OpenStack nova-compute[2676]: DEBUG nova.network.base_api [None req-cd261f50-e8a2-41d4-a35b-9ce1b4862ed6 evil-users eve] [instance: 060a9ccd-d723-4cd7-a5e6-3f590bc145fb] Updating instance_info_cache with network_info: [{"profile": {}, "details": {"bridge_name": "br-int", "datapath_type": "system", "port_filter": true, "ovs_hybrid_plug": false}, "qbg_params": null, "vnic_type": "normal", "type": "ovs", "preserve_on_delete": false, "meta": {}, "active": false, "devname": "tap15f1b97d-de", "ovs_interfaceid": "15f1b97d-def9-4b18-b751-7c51e5d8cd69", "address": "fa:16:3e:3a:96:98", "network": {"id": "62dc482a-90d7-4d0d-a781-e5323d575591", "meta": {"physical_network": null, "injected": false, "tenant_id": "bc4cbec89d664578823d06a5c149d131", "mtu": 1450, "tunneled": true}, "bridge": "br-int", "label": "evil-network-1", "subnets": [{"version": 4, "cidr": "10.0.0.64/26", "routes": [], "dns": [], "gateway": {"address": "10.0.0.65", "version": 4, "meta": {}, "type": "gateway"}, "ips": [{"floating_ips": [], "address": "10.0.0.79", "version": 4, "meta": {}, "type": "fixed"}], "meta": {"dhcp_server": "10.0.0.66"}}]}, "id": "15f1b97d-def9-4b18-b751-7c51e5d8cd69", "qbh_params": null}, {"profile": {}, "details": {"bridge_name": "br-int", "datapath_type": "system", "port_filter": true, "ovs_hybrid_plug": false}, "qbg_params": null, "vnic_type": "normal", "type": "ovs", "preserve_on_delete": false, "meta": {}, "active": false, "devname": "tap57377e4a-25", "ovs_interfaceid": "57377e4a-25f8-463d-902b-4e27a9c4df38", "address": "fa:16:3e:69:aa:c1", "network": {"id": "1b1e99bb-d64f-49de-b953-0010d63d7bfe", "meta": {"physical_network": null, "injected": false, "tenant_id": "063671b275cc43348863ebd3d12d7a11", "mtu": 1450, "tunneled": true}, "bridge": "br-int", "label": "defender-network-1", "subnets": [{"version

```
": 4, "cidr": "10.0.0.0/26", "routes": [], "dns":
[], "gateway": {"address": "10.0.0.1", "version": 4,
 "meta": {}, "type": "gateway"}, "ips": [{"
floating_ips": [], "address": "10.0.0.54", "version
": 4, "meta": {},
```

Apéndice V - Log de Neutron

```
jun 30 14:28:26 OpenStack neutron-server[26607]: DEBUG
   neutron.api.v2.base [req-cd261f50-e8a2-41d4-a35b-9
   ce1b4862ed6 req-c9f82173-5eef-4d15-aa50-1949a9305cb7
    evil-users eve] Request body: {'port': {'
   security_groups': ['867de525-9843-414c-9465-78
   baba17e762'], 'admin_state_up': True, 'device_id':
   '060a9ccd-d723-4cd7-a5e6-3f590bc145fb', 'tenant_id':
    'bc4cbec89d664578823d06a5c149d131', 'network_id':
   '62dc482a-90d7-4d0d-a781-e5323d575591'}} {{(pid
   =26732) prepare_request_body /opt/stack/neutron/
   neutron/api/v2/base.py:715}}

jun 30 14:28:30 OpenStack neutron-dhcp-agent[28007]:
   INFO neutron.agent.dhcp.agent [-] Trigger
   reload_allocations for port admin_state_up=True,
   allowed_address_pairs=[], binding:host_id=, binding:
   profile=, binding:vif_details=, binding:vif_type=
   unbound, binding:vnic_type=normal, created_at
   =2019-06-30T12:28:27Z, description=, device_id=060
   a9ccd-d723-4cd7-a5e6-3f590bc145fb, device_owner=,
   extra_dhcp_opts=[], fixed_ips=[{'ip_address':
   '10.0.0.79', 'subnet_id': '3bec2d63-ff20-4e87-bf30-
   ca3c84378888'}], id=15f1b97d-def9-4b18-b751-7
   c51e5d8cd69, mac_address=fa:16:3e:3a:96:98, name=,
   network_id=62dc482a-90d7-4d0d-a781-e5323d575591,
   port_security_enabled=True, project_id=
   bc4cbec89d664578823d06a5c149d131, revision_number=1,
    security_groups=['867de525-9843-414c-9465-78
   baba17e762'], status=DOWN, tags=[], tenant_id=
   bc4cbec89d664578823d06a5c149d131, updated_at
```

=2019-06-30T12:28:28Z

jun 30 14:28:30 OpenStack neutron-server[26607]: DEBUG neutron.api.v2.base [req-cd261f50-e8a2-41d4-a35b-9ce1b4862ed6 req-7e7c8781-4df5-4c01-9b57-ed946afb5f6c evil-users eve] Request body: {'port': {'security_groups': ['867de525-9843-414c-9465-78baba17e762'], 'admin_state_up': True, 'device_id': '060a9ccd-d723-4cd7-a5e6-3f590bc145fb', 'tenant_id': 'bc4cbec89d664578823d06a5c149d131', 'network_id': '1b1e99bb-d64f-49de-b953-0010d63d7bfe'}} {{(pid=26732) prepare_request_body /opt/stack/neutron/neutron/api/v2/base.py:715}}

jun 30 14:28:33 OpenStack neutron-dhcp-agent[28007]: INFO neutron.agent.dhcp.agent [-] Trigger reload_allocations for port admin_state_up=True, allowed_address_pairs=[], binding:host_id=, binding:profile=, binding:vif_details=, binding:vif_type=unbound, binding:vnic_type=normal, created_at=2019-06-30T12:28:32Z, description=, device_id=060a9ccd-d723-4cd7-a5e6-3f590bc145fb, device_owner=, extra_dhcp_opts=[], fixed_ips=[{'ip_address': '10.0.0.54', 'subnet_id': '03ebac02-8f5f-474c-954e-a1c1ac9504cd'}], id=57377e4a-25f8-463d-902b-4e27a9c4df38, mac_address=fa:16:3e:69:aa:c1, name=, network_id=1b1e99bb-d64f-49de-b953-0010d63d7bfe, port_security_enabled=True, project_id=bc4cbec89d664578823d06a5c149d131, revision_number=1, security_groups=['867de525-9843-414c-9465-78baba17e762'], status=DOWN, tags=[], tenant_id=bc4cbec89d664578823d06a5c149d131, updated_at=2019-06-30T12:28:32Z

Apéndice VI - Log de Keystone

```
jun 30 16:22:21 OpenStack devstack@keystone.service
   [3523]: DEBUG keystone.server.flask.
   request_processing.req_logging [None req-d14ea651
   -817f-4dea-81e2-941af90a190f None None]
   REQUEST_METHOD: 'POST' {{(pid=3527) log_request_info
    /opt/stack/keystone/keystone/server/flask/
   request_processing/req_logging.py:27}}

jun 30 16:22:21 OpenStack devstack@keystone.service
   [3523]: DEBUG keystone.server.flask.
   request_processing.req_logging [None req-d14ea651
   -817f-4dea-81e2-941af90a190f None None] SCRIPT_NAME:
    '/identity' {{(pid=3527) log_request_info /opt/
   stack/keystone/keystone/server/flask/
   request_processing/req_logging.py:28}}

jun 30 16:22:21 OpenStack devstack@keystone.service
   [3523]: DEBUG keystone.server.flask.
   request_processing.req_logging [None req-d14ea651
   -817f-4dea-81e2-941af90a190f None None] PATH_INFO:
   '/v3/auth/tokens' {{(pid=3527) log_request_info /opt
   /stack/keystone/keystone/server/flask/
   request_processing/req_logging.py:29}}

jun 30 16:22:21 OpenStack devstack@keystone.service
   [3523]: WARNING keystone.server.flask.application [
   None req-d14ea651-817f-4dea-81e2-941af90a190f None
   None] Authorization failed. La solicitud que ha
   hecho requiere autenticacion. from 192.168.0.105:
   keystone.exception.Unauthorized: La solicitud que ha
    hecho requiere autenticacion.
```

```
jun 30 16:22:25 OpenStack devstack@keystone.service
   [3523]: DEBUG keystone.server.flask.
   request_processing.req_logging [None req-8a6b3786-
   dcac-43ba-a4c8-58da7a990c50 None None]
   REQUEST_METHOD: 'POST' {{(pid=3526) log_request_info
    /opt/stack/keystone/keystone/server/flask/
   request_processing/req_logging.py:27}}

jun 30 16:22:25 OpenStack devstack@keystone.service
   [3523]: DEBUG keystone.server.flask.
   request_processing.req_logging [None req-8a6b3786-
   dcac-43ba-a4c8-58da7a990c50 None None] SCRIPT_NAME:
   '/identity' {{(pid=3526) log_request_info /opt/stack
   /keystone/keystone/server/flask/request_processing/
   req_logging.py:28}}

jun 30 16:22:25 OpenStack devstack@keystone.service
   [3523]: DEBUG keystone.server.flask.
   request_processing.req_logging [None req-8a6b3786-
   dcac-43ba-a4c8-58da7a990c50 None None] PATH_INFO: '/
   v3/auth/tokens' {{(pid=3526) log_request_info /opt/
   stack/keystone/keystone/server/flask/
   request_processing/req_logging.py:29}}

jun 30 16:22:25 OpenStack devstack@keystone.service
   [3523]: WARNING keystone.server.flask.application [
   None req-8a6b3786-dcac-43ba-a4c8-58da7a990c50 None
   None] Authorization failed. The account is locked
   for user: 310a9c2d72bf43efa8df8c6d95ce0ddf. from
   192.168.0.105: keystone.exception.AccountLocked: The
    account is locked for user: 310
   a9c2d72bf43efa8df8c6d95ce0ddf.
```