SecDevOps y Administración de Redes para Cloud

Herramientas de seguridad en la nube

Índice

Esquema

HERRAMIENTAS DE SEGURIDAD EN LA NUBE

IAM

- Servicio de identidad y autorización de AWS.
- Soporta usuarios y grupos como otros sistemas tradicionales.
- Las cuentas de usuario se pueden destinar:
  - A individuos, con MFA y política de contraseñas.
  - A aplicaciones, con claves secretas complejas y soporte de rotación de claves.

- Los permisos se definen en directivas.
- Una directiva es un documento JSON, que contiene las acciones permitidas o denegadas, y los recursos sobre los que se aplican.
- Las acciones se agrupan en servicios; cada servicio tiene acciones específicas.
- Los recursos se identifican con ARN.
- Tanto recursos como acciones soportan comodines para abarcar varias acciones con una única entrada (`ec2:Describe*`).

- Los roles permiten asignar permisos temporales a un usuario o un recurso.
- Al contrario que las cuentas de usuario, no tienen credenciales asociadas.
- Un usuario puede *impersonarse* con un rol y le aplicarán los permisos del rol, no los del usuario.
- La *impersonación* funciona entre cuentas de AWS, por lo que se puede dar permiso a una organización externa.
- Algunos servicios de AWS también pueden *impersonarse*.
- Los SDKs y awscli pueden *impersonarse* si se ejecutan en una instancia de EC2 sin necesidad de configurar credenciales.

Otras

CloudTrail: registro de todas las acciones recibidas en la API sobre una cuenta.
- Almacena los registros tres meses, pero se pueden guardar más tiempo.

Inspector: análisis de vulnerabilidades de red y de software.
- Requiere un agente para analizar el software.
- Los análisis se pueden programar.
- Los informes contienen sugerencias de resolución.

Detective: análisis de correlación de *logs*.
- Combina *logs* de CloudTrail, flujos de VPC, etc.
- No es intrusivo, trabaja solo sobre *logs* ya emitidos.
- Encuentra correlaciones y compara eventos actuales con una situación habitual.

Ideas clave

7.1. Introducción y objetivos

Los **proveedores de nube** ofrecen, en muchos casos, **herramientas nativas** de seguridad que requieren un aprendizaje intensivo, incluso para administradores con experiencia. Mientras que **VMware vSphere** ofrece un acceso basado en roles tradicional, **AWS IAM** es un sistema construido de cero que soporta una alta granularidad en la selección de permisos y recursos, además de facilitar técnicas como la ***impersonación*** o la distribución de claves de manera automática.

Además del control de permisos, AWS también ofrece **análisis de vulnerabilidades *as a service,*** de forma que el usuario no necesita desplegar **infraestructura adicional** ni configurar herramientas de terceros para evaluar la seguridad de una aplicación.

Los **objetivos** de este tema son:

- Familiarizarse con el modelo de identidad y autorización de AWS.
- Conocer algunas de las herramientas de auditoría y seguridad nativas en AWS.

7.2. IAM

IAM *(Identity and Access Management)* es el servicio de AWS que permite administrar **quién** puede **acceder** a las API y **qué acciones** puede **ejecutar** (Lucifredi y Ryan, 2018). Tener una **política de IAM** bien planificada es una parte importante de la seguridad de AWS. IAM distingue entre **autenticación,** que está basada en usuarios y grupos; y **autorización,** que se basa en las directivas de IAM.

ARN

Para identificar los recursos, entre ellos a los usuarios de IAM, AWS usa los ***Amazon Resource Names*** o ARN. Un ARN es un **identificador único global** que hace referencia a objetos de AWS. La mayoría de los tipos de recursos de AWS tienen ARN, incluidos los objetos de S3 y los roles, usuarios y directivas de IAM. Tienen el siguiente formato:

```
arn:partition:service:region:account-id:resource-type/resource-id
```

Los campos son los siguientes:

`partition`	Una partición es un grupo de regiones de AWS: aws, aws-cn (regiones de China) y aws-us-gov (regiones GovCloud de Estados Unidos). Una cuenta de AWS pertenece a una única partición.
`service`	El nombre que identifica el tipo de servicio que ofrece el recurso, por ejemplo, `s3, iam o ec2`.
`region`	La región donde está ubicado el recurso, por ejemplo, `us-east-2`. Los ARN de algunos tipos de recurso, como los *buckets* de S3 o los objetos de IAM, no incluyen este campo porque se definen de manera global.
`account-id`	El ID de la cuenta de AWS propietaria del recurso. Al igual que la región, se omite en algunos tipos de recursos.
`resource-id`	El identificador del recurso. Puede ser un identificador sencillo o una ruta, por ejemplo, `user/Bob` para un usuario de IAM o `instance/i-1234567890abcdef0` para una instancia de EC2.

Tabla 1. Descripción de campos. Fuente: adaptado de Amazon Web Services, s. f.a.

Directivas de IAM

La idea detrás de IAM es **separar** a los usuarios y grupos de las acciones que necesitan realizar. Para ello, se crean **directivas de IAM,** que son **documentos JSON** que describen qué acciones puede realizar un usuario. Esta política se aplica a los usuarios o grupos, dándoles acceso solo a los **servicios** que especifica el documento.

Un **permiso** es una combinación de **dos elementos:** una **acción** y una **lista de recursos.** AWS verificará si el usuario autenticado puede realizar la acción solicitada en un recurso específico, por ejemplo, ¿se le permite al usuario reiniciar (la acción) una instancia EC2 (el recurso)?

Las acciones se definen como **cadenas de texto** con el formato `servicio:permiso`. Por ejemplo, la acción de reinicio de instancias se define como `ec2:RebootInstances`. Las directivas hacen referencia a **permisos dinámicos** muy granulares en todos los servicios de AWS. El siguiente documento JSON hace uso de esta acción y de `ec2:DescribeInstances` sobre todos los recursos.

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Sid": "VisualEditor0",
        "Effect": "Allow",
        "Action": [
            "ec2:RebootInstances",
            "ec2:DescribeInstances"
        ],
        "Resource": "*"
    }]
}
```

Este ejemplo sirve para describir algunas de las **características** de las **directivas** de IAM:

- Las directivas contienen `statements`, o **declaraciones.** La declaración es la pieza elemental de una directiva, ya que enlaza acciones y recursos.

- El campo `statement` es realmente una **lista de subdocumentos,** por lo que una única directiva puede tener varias declaraciones.

- El campo `effect` indica si se **habilitan** las **acciones** sobre los recursos, con `Allow`, o si se deniegan explícitamente, con `Deny`. Por defecto, un usuario al que no se le aplica ninguna directiva no tendrá permiso para ejecutar ninguna acción sobre ningún recurso. El valor `Deny` en `effect` sirve, por tanto, para denegar permisos específicos que pueden haber sido habilitados por otra directiva.

- El campo `action` especifica las **acciones permitidas o denegadas** sobre los recursos. Puede ser un campo de texto, en cuyo caso, especifica una única acción, o una lista.

- El campo `resource` **especifica el recurso o recursos,** identificados con ARN, sobre los que hace efecto la declaración. Al igual que `action`, puede ser un campo de texto o una lista.

> Comodines
>
> Las acciones admiten comodines en el permiso para reducir el número de permisos que hay que escribir para conseguir un determinado acceso. Por ejemplo, una directiva puede dar acceso completo al servicio EC2 con la acción `ec2:*`. De no usar un comodín, habría que escribir los más de cuatrocientos permisos disponibles en EC2 en el momento de redactar este capítulo (Amazon Web Services, s. f.b). Los comodines se pueden usar también como parte del nombre. Por ejemplo, se puede dar acceso de lectura con una acción `ec2:Describe*`, o dar acceso a los *snapshots,* pero no a otros tipos de recursos con `ec2:*Snapshot`. El campo `resource` también admite comodines; por ejemplo, `arn:aws:s3:::my_corporate_bucket/*` se refiere a todos los elementos contenidos en un *bucket* de S3; y el documento JSON, del ejemplo anterior, abarca todos los recursos disponibles con `"*"`.

Determinar el **conjunto de acciones** que se quieren permitir en una directiva determinada no es simple. Para ayudar en esta tarea, AWS ofrece un **editor visual** de directivas, como muestra la Figura 1. Este editor permite **identificar servicios y permisos** fácilmente. Además, sirve como un buen punto de entrada, ya que, una vez definida una directiva sencilla, es posible personalizarla saltando a la pestaña JSON para editar el documento que acaba de generar el editor visual.

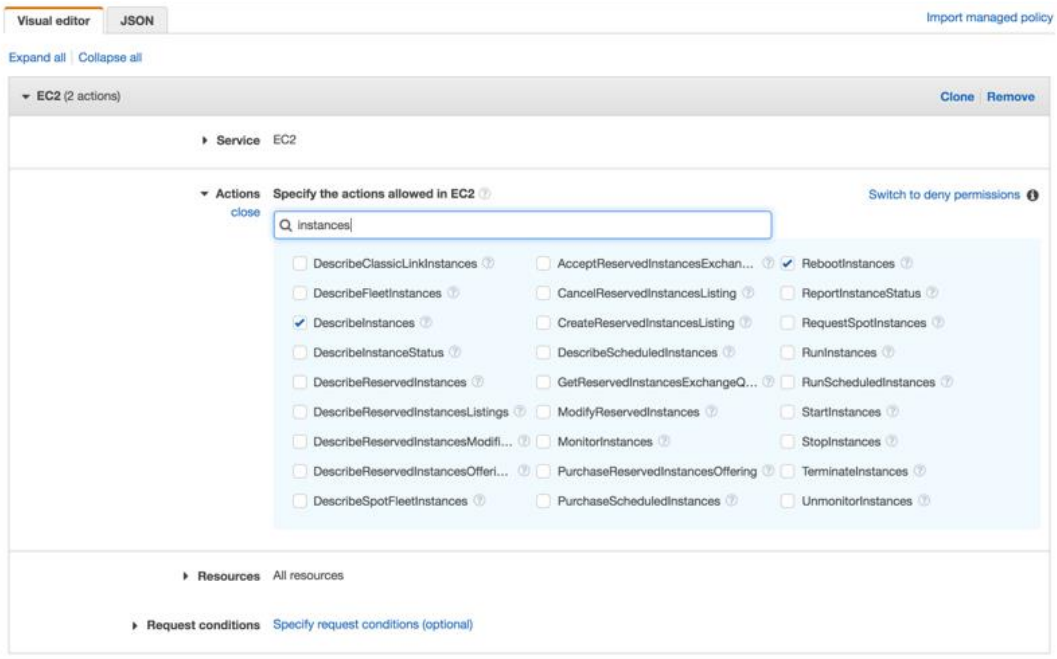

Figura 1. Editor visual de directivas IAM. Fuente: elaboración propia.

No es menos complejo **mapear las interacciones** de múltiples conjuntos de permisos en un solo usuario, y, para ello, se puede hacer uso del **IAM Policy Simulator** (Figura 2). Hay que tener en cuenta que sobre un usuario se pueden aplicar directivas **directamente** o a **través de un grupo,** que se pueden aplicar **múltiples directivas** y que cada directiva puede contener **múltiples declaraciones.** El simulador permite evaluar si un usuario tiene uno o varios permisos concretos sin revisar manualmente todas las opciones posibles.

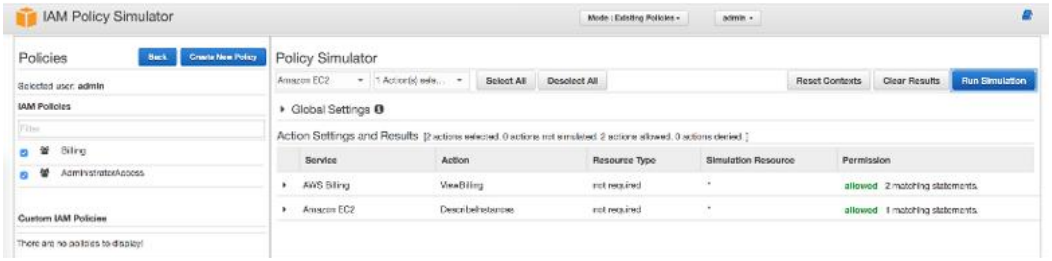

Figura 2. IAM Policy Simulator. Fuente: elaboración propia.

Usuarios y grupos

Un usuario puede ser un humano que **inicia sesión** en la consola de administración web, con un nombre de usuario y contraseña, o un programa que utiliza un conjunto de **credenciales de acceso** para interactuar con las API de AWS. Al usuario se le pueden asignar una o más directivas de IAM, que especificarán las **acciones** que puede realizar.

Al crear un usuario, hay que especificar si podrá iniciar sesión en la consola, o si podrá acceder a las **API** directamente con un **SDK,** o la herramienta de línea de comandos (Figura 3). El acceso a la consola requiere **contraseña** y es el tradicional para seres humanos.

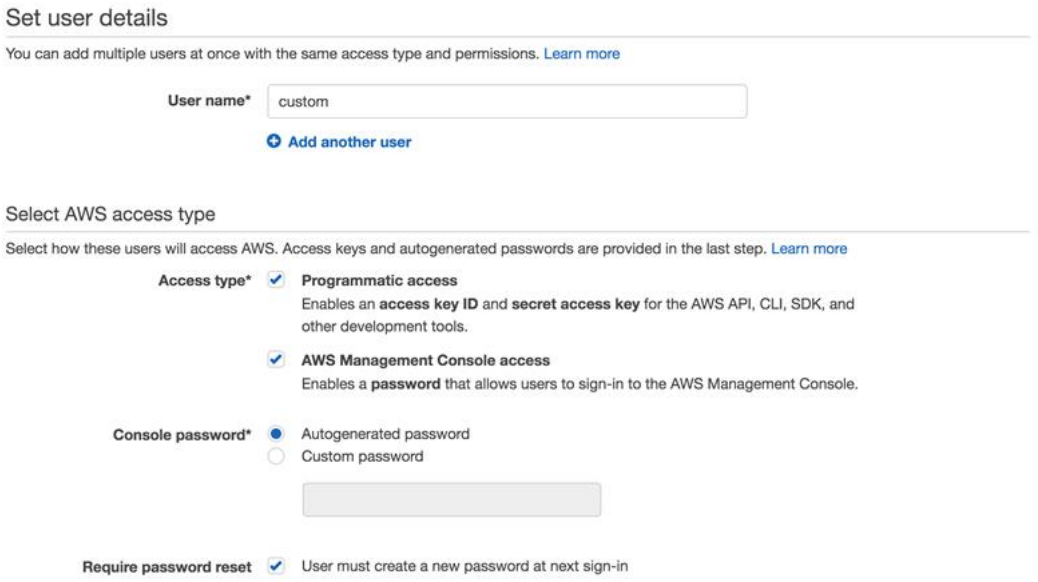

Figura 3. Creación de usuario en IAM. Fuente: elaboración propia.

El **acceso** a las **API** se realiza con una **clave** y un **secreto** adicionales, aleatorios y más complejos que un nombre de usuario y una contraseña normales. Cada usuario puede tener **dos claves activas** (Figura 4). El **objetivo** de estas claves es usarlas en cuentas de servicio, tareas automáticas, etc.

Dado que estas cuentas no pueden cambiar la contraseña por sí mismas, es necesario disponer de, al menos, dos claves para poder **rotarlas.** Este proceso de rotación consiste en los siguientes **pasos:**

- Se crea una pareja de **clave y secreto** para una cuenta de servicio.
- Se **configura** la aplicación, *script,* etc. para usar esta clave. Puede ocurrir que la clave esté en uso en varios sitios, por ejemplo, en varios procesos de una aplicación configurada en alta disponibilidad.
- Tras el **tiempo estipulado** por la **política de seguridad** de la empresa (por ejemplo, a los treinta días), se genera una nueva clave.
- Se **sustituye** la clave original por la nueva en todos los sitios donde se configuró. En este intervalo de tiempo ambas claves son válidas; de lo contrario, unos procesos dejarían de funcionar hasta recibir la nueva clave.
- Una vez distribuida la clave nueva se puede **desactivar** y **borrar** la clave antigua.

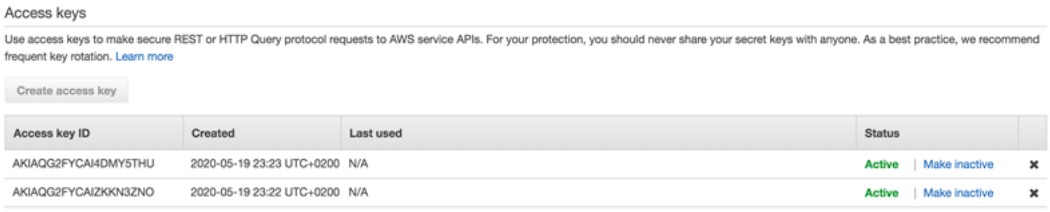

Figura 4. Claves para acceso programático. Fuente: elaboración propia.

Para **facilitar** la administración, los usuarios pueden pertenecer a **grupos.** Cuando se asigna una directiva a un grupo, todos los miembros de ese grupo heredan los **permisos designados** por esa directiva. No es posible anidar grupos.

Roles

La **rotación de claves** puede ser relativamente sencilla cuando se usan en `awscli` (Amazon Web Services, s. f.c), pero se complica notablemente cuando las llamadas a la **API** se llevan a cabo desde un **SDK** integrado en una aplicación corporativa. Es habitual que los clientes de AWS ejecuten aplicaciones en **instancias EC2,** que interactúan con otros servicios de AWS a través de un SDK.

En caso de que haya cientos de instancias EC2 ejecutando la aplicación, ¿cómo se rotan las claves? Si se escriben en una **AMI personalizada,** algo nada recomendable, habría que recrear la AMI y redesplegar las **instancias.** Otra opción es escribir las claves en el *user data*, pero entonces las claves están visibles en la consola. Se podría usar una herramienta de terceros, como Vault, para distribuir las claves. En cualquier caso, el **proceso** no es inmediato y un **error** puede dejar inoperativa la aplicación.

En el siguiente vídeo, titulado **«Herramientas de seguridad en cloud»**, se puede seguir una demostración práctica del uso de roles IAM con instancias EC2.

Accede al vídeo

En resumen, hasta junio de 2012, el proceso de **distribución** de estas claves era doloroso. Los **roles de IAM** eliminan casi por completo estos problemas. Al igual que los usuarios y grupos, los roles de IAM pueden tener una o más **directivas aplicadas.** Estos roles se aplican, entonces, a una instancia en el momento del arranque. **AWS** generará automáticamente la **clave** y el **secreto de acceso**, y los pondrá a disposición de la instancia a través de la API de metadatos (Amazon Web Services, s. f.d).

Estas credenciales se pueden usar para **acceder a los servicios de AWS** con los permisos especificados por las políticas del rol. Los **SDKs de AWS y awscli** están preparados para leer estas claves automáticamente, sin necesidad de que los **desarrolladores** ni los **administradores** se tengan que encargar de indicar configuración alguna. El único requisito es que la aplicación debe ejecutarse en una instancia de EC2. Además, AWS rotará regularmente las claves durante **la vida útil** de la instancia, sin requerir ninguna acción por parte de los administradores.

Los roles se pueden aplicar también a otros **servicios de AWS** (Lambda o API Gateway, por ejemplo), **usuarios** (dentro y fuera de una organización), **aplicaciones móviles** o a **terceras entidades,** para permitir que un proveedor administre ciertos recursos en la cuenta del cliente.

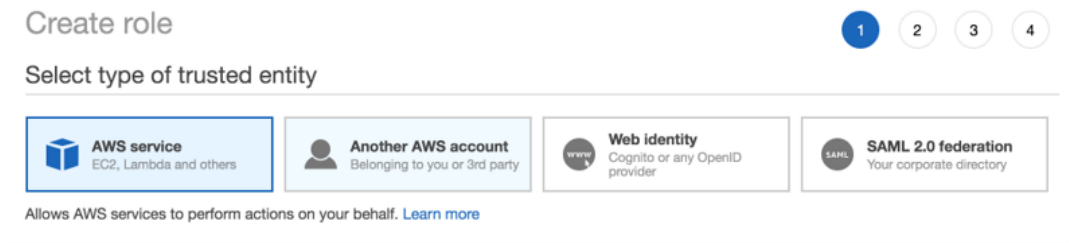

Figura 5. Escenarios en los que es posible asignar un rol de IAM. Fuente: elaboración propia.

El **uso de roles** para dar permisos en otras cuentas ofrece más **flexibilidad** para administrar recursos, presupuestos y credenciales. Es habitual que las organizaciones dispongan de **múltiples cuentas de AWS** para identificar, claramente, los límites de gasto. En estas situaciones no es viable mantener **usuarios** y sus **credenciales** en todas las cuentas.

Para ello, se pueden **asignar roles** de una cuenta A a usuarios de una cuenta B. Cuando un usuario de la cuenta B quiere **operar** sobre recursos de la cuenta A, el usuario asume un rol de la cuenta B. Las acciones que puede llevar a cabo están **delimitadas** por las **directivas aplicadas** a ese rol, no a las aplicadas al usuario. El proceso entre una cuenta `PROD` y una cuenta `TEST` para dar permiso de lectura, sobre instancias de EC2, a un equipo de control de calidad sería el siguiente:

- En la cuenta `PROD`, un administrador crea el rol `ReadEC2` y asigna al rol una directiva con las acciones `ec2:Describe*` y `ec2:Get*` sobre todas las instancias. En el rol se indica el **ID** de la cuenta `TEST`, sin identificar nombres de usuarios concretos. El rol puede requerir opciones adicionales, como que el usuario haya iniciado sesión con ***Multi-Factor Authentication*** (MFA) para poder asumir el rol.

Multi-Factor Authentication (MFA) es el método de inicio de sesión en el que hay que proveer un segundo tipo de credencial, normalmente temporal y dependiente de un dispositivo, además de las credenciales habituales del usuario.

- El administrador **comparte** el **nombre** y el **ARN** del rol y el número de cuenta con los usuarios de la cuenta `TEST`. Para asumir el rol desde la consola web, hay que indicar el nombre del rol y el ID de la cuenta, mientras que, para asumirlo desde `awscli` o desde un SDK, hay que indicar el ARN.

- En la cuenta `TEST`, un administrador crea una **directiva** con el permiso `sts:AssumeRole` y el ARN del rol `ReadEC20` de la cuenta `PROD` como recurso y la asigna a un grupo. A partir de ese momento, los usuarios del grupo pueden asumir el rol. Esta directiva no incluye los permisos sobre las instancias de la cuenta `PROD`, ya que estos están incluidos en la directiva del rol `ReadEC2`.

- El usuario de la cuenta `TEST` solicita el cambio al rol `ReadEC2`:

  - Para acciones en la consola de AWS, el usuario inicia sesión con su usuario y contraseña habituales y selecciona la opción **Switch Role** del menú de usuario. A continuación, indica el **nombre del rol** y el **ID** de la cuenta `PROD`.
  - Para acceder programáticamente, el usuario configura el **SDK** o `awscli` con la clave y secreto (no su contraseña) y, a continuación, ejecuta la función `AssumeRole` con el ARN del rol `ReadEC2`.

- El **servicio AWS STS** devuelve credenciales temporales. Este paso es transparente tanto para el usuario en la consola web como si usa un **SDK.**

- A partir de este momento, el usuario puede ejecutar las acciones permitidas por la directiva del rol `ReadEC2` en la cuenta `PROD`.

En todo este proceso, el administrador de la cuenta `PROD` **no ha facilitado ninguna credencial** a los usuarios de la cuenta `TEST`. Esto es especialmente relevante cuando las dos cuentas implicadas pertenecen a **organizaciones diferentes.** Puede servir, por ejemplo, para dar control sobre una cuenta a una **herramienta de gestión** de ciclo de vida de aplicaciones o de copias de seguridad ofrecida como servicio.

Estas herramientas se **integran** con la API de AWS y usan sus **propias credenciales** para iniciar sesión, relevando al cliente de la rotación de claves o la gestión de usuarios bloqueados o caducados.

7.3. Otras herramientas

CloudTrail

CloudTrail es una **herramienta de auditoría** que registra todas las llamadas a la API en una región específica, o globalmente, **independientemente** de la herramienta que origina la llamada: `awscli`, SDKs, consola e, incluso, otros servicios de AWS (Lucifredi y Ryan, 2018).

Estos registros **mejoran la capacidad** para determinar qué usuario realizó qué acción en un momento dado, y es esencial para reconstruir lo que realmente sucedió en caso de un **incidente de seguridad.** CloudTrail **incluirá** en sus registros todas las llamadas a la API generadas por cualquier servicio de AWS en nombre del usuario. Algunas de estas llamadas pueden haberse realizado automáticamente por otro servicio, bien como **respuesta** a una llamada de usuario o como parte del **funcionamiento habitual.**

Los registros de CloudTrail indican si una llamada API fue generada automáticamente por otro servicio.

Los registros de CloudTrail se puede **visualizar** directamente en la consola web. Este visor de eventos, mostrado en la Figura 6, guarda los registros por un máximo de **tres meses.** Para extender el histórico, hay que crear un *trail*. Este *trail* especifica un *bucket* de destino de S3, donde CloudTrail guardará los registros.

Además de extender el límite de tiempo, los *trails* permiten generar **alarmas** a partir de eventos y realizar búsquedas en el registro con **Athena,** un servicio de AWS que ejecuta búsquedas interactivas en SQL sobre contenido almacenado en S3.

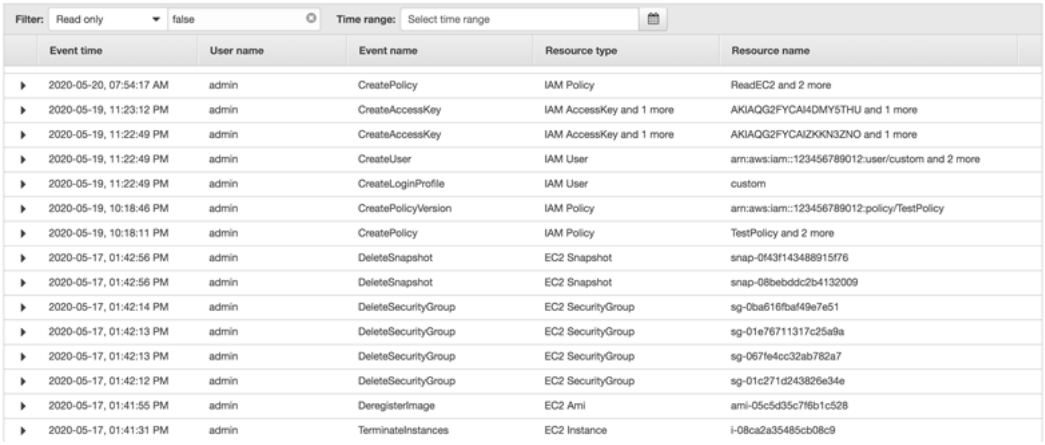

Figura 6. Visor de eventos de CloudTrail. Fuente: elaboración propia.

Inspector

Amazon Inspector es un **escáner de vulnerabilidad** de bajo impacto. Reúne información sobre la red y los procesos de los servidores, la analiza y presenta un **informe,** al usuario, con los **resultados.** Permite analizar el comportamiento de los recursos y ayuda a identificar posibles problemas de seguridad.

Inspector funciona ejecutando **evaluaciones** sobre conjuntos de recursos. Los datos que recopila de los recursos objetivo incluyen **detalles de la comunicación** con los servicios de AWS, uso de **canales seguros, procesos** en ejecución y **tráfico** de red entre los procesos, entre otros. Estos datos se analizan y comparan con un conjunto de **reglas de seguridad.** El informe contiene una **lista** de posibles problemas de seguridad, con un indicador de gravedad.

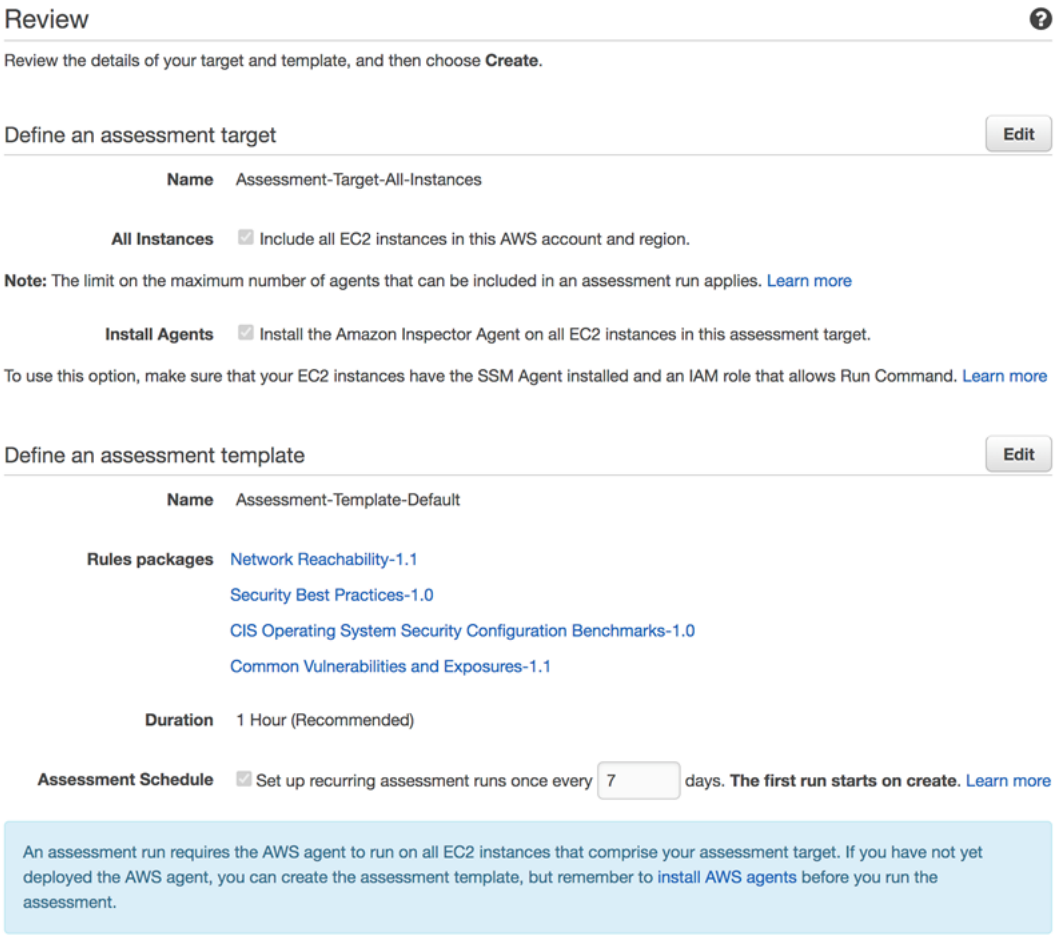

Figura 7. Detalles de evaluación de Inspector. Fuente: elaboración propia.

Incorpora una **biblioteca integrada de reglas e informes.** Esta incluye controles con las mejores prácticas, estándares comunes de cumplimiento y vulnerabilidades. Estos controles ofrecen al usuario **pasos recomendados detallados** para resolver posibles problemas de seguridad.

Inspector ofrece **dos opciones de evaluación:** de red y de equipo.

- Las **evaluaciones de red** comprueban los puertos accesibles desde fuera de la VPC donde residen las instancias objetivo. Además, si el agente está instalado, también comprueba si hay procesos con puertos a la escucha.
- Las **evaluaciones de equipo** buscan software vulnerable, analizan las configuraciones de seguridad y comprueban si se aplican las mejores prácticas.

El **agente** es un **programa** que se instala en las instancias EC2. Es necesario para ejecutar las evaluaciones de equipo y parte de las evaluaciones de red. Se puede instalar **manual o automáticamente** antes de cada evaluación. En este caso, es necesario que las instancias tengan instalado el **agente de SSM** (AWS Systems Manager), otro agente de AWS necesario para la administración remota mediante SSM (SSM es un servicio de administración y operaciones de infraestructura).

El **informe** de una evaluación contiene, entre otros datos, la **gravedad del hallazgo,** una **descripción,** la **regla** que los activó y los **pasos recomendados** para solucionar el problema.

La principal ventaja de Inspector es la **facilidad de ejecución.** Permite delegar, en una tarea automática, la revisión de vulnerabilidades típicas.

Amazon Detective

Amazon Detective es un servicio administrado por AWS que permite a los usuarios **analizar y procesar** cantidades ingentes de ***logs,*** con el objetivo de **buscar causas** e **impactos** de incidentes de seguridad. Se alimenta de *logs* generados por otros servicios de AWS, como GuardDuty (Amazon Web Services, s. f.e), CloudTrail y VPC Flow Logs, por lo que puede **activarse** sin perjuicio de rendimiento de la infraestructura existente. Según AWS, Detective:

> «Utiliza modelos de aprendizaje automático para producir representaciones gráficas del comportamiento de su cuenta y lo ayuda a responder preguntas como "¿es esta una llamada API inusual para este rol?" o "¿se espera este aumento en el tráfico de esta instancia?". No necesita escribir código, configurar o ajustar sus propias consultas» (Stormacq, 2020).

A **alto nivel,** una investigación general, y de Amazon Detective en particular, seguirá estas **fases:**

- **Triaje.** El proceso comienza con un aviso de **posible actividad maliciosa.** Esta información se hace llegar a un analista o ingeniero de seguridad. Uno de estos puntos de entrada puede ser una alerta generada por Amazon GuardDuty, o, simplemente, el ID de la VPC de la que proviene la alerta. El ingeniero determinará si la actividad es genuinamente un riesgo de seguridad o un falso positivo. En caso de considerar la alerta un verdadero positivo, seguirá con la siguiente fase.

- **Alcance.** En este caso, el ingeniero considera el **alcance y posible causa** del incidente. Por ejemplo, deberá averiguar qué sistemas y recursos han sido afectados, dónde se originó el ataque, cuándo empezó, si ya ha terminado y si hay actividades relacionadas con el ataque (si un atacante toma el control de una máquina, es probable que intente extraer datos confidenciales hacia el exterior).

- **Respuesta.** Finalmente, el ingeniero intentará **detener el ataque** y **minimizar el daño.** Además, establecerá medidas y procedimientos para impedir que un ataque similar pueda volver a ocurrir.

En la **segunda etapa** es donde más valor se puede obtener de Detective. A partir del elemento encontrado en la etapa de triaje, Detective muestra información enlazada. Por ejemplo, puede mostrar **llamadas API** sobre un recurso con errores de autenticación, o el origen geográfico de dichas llamadas. Además, permite **comparar** el comportamiento que reflejan los registros actuales con el comportamiento habitual.

Al igual que Inspector, Detective se puede **automatizar** a través de llamadas de API y de SDK.

7.4. Referencias bibliográficas

Amazon Web Services. (S. f.a.). *Amazon Resource Names (ARNs).* https://docs.aws.amazon.com/general/latest/gr/aws-arns-and-namespaces.html

Amazon Web Services. (S. f.b.). *Actions.* https://docs.aws.amazon.com/AWSEC2/latest/APIReference/API_Operations.html

Amazon Web Services. (S. f.c.). *Interfaz de línea de comandos de AWS.* https://aws.amazon.com/es/cli/

Amazon Web Services. (S. f.d.). *Retrieve instance metadata.* https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/instancedata-data-retrieval.html

Amazon Web Services. (S. f.e.). *Amazon GuardDuty.* https://aws.amazon.com/es/guardduty/

Lucifredi, F y Ryan, M. (2018). *AWS System Administration.* O'Reilly Media.

Stormacq, S. (2020, marzo 31). *Amazon Detective - Rapid Security Investigation and Analysis.* Amazon Web Services. https://aws.amazon.com/blogs/aws/amazon-detective-rapid-security-investigation-and-analysis/

A fondo

Introducción a IAM

Simplilearn. (2016, junio 30). *AWS IAM Tutorial | AWS Identity And Access Management | AWS Tutorial | AWS Training | Simplilearn* [Vídeo]. YouTube. https://www.youtube.com/watch?v=3A5hRIT8zdo

Este vídeo resume las principales características de IAM y algunas mejores prácticas para crear recursos. Es muy recomendable como recurso complementario al material de este tema.

Presentación de Detective

AWS Events. (2019, diciembre 10). *AWS re:Invent 2019: [NEW LAUNCH!] Introducing Amazon Detective (SEC312)* [Vídeo]. YouTube. https://www.youtube.com/watch?v=MPQe-4NvesM

AWS presentó Detective durante re:Invent 2019, la feria exclusiva que celebra cada diciembre en Las Vegas. En aquel momento, aún no estaba disponible (no lo estuvo hasta marzo de 2020), pero ya pudieron mostrar una ejecución en vivo del producto.

Introducción a Inspector

AWS Online Tech Talks. (2016, junio 27). *AWS June 2016 Webinar Series - Getting Started with Amazon Inspector* [Vídeo]. YouTube. https://www.youtube.com/watch?v=BksbSqA3j9U

Este vídeo presenta Amazon Inspector y hace una demostración práctica. Es recomendable verlo para entender el potencial, ya que simular vulnerabilidades en un entorno de laboratorio para probar Inspector no es trivial.

Test

1. ¿Qué tipos de análisis de Inspector necesitan agente?

A. Los de red.

B. Ninguno.

C. Todos.

D. Los de *host* y algunos de red.

2. ¿Por qué Detective no es intrusivo?

A. Porque analiza *logs* de otros servicios, pero no actúa sobre las instancias bajo análisis.

B. Es tan intrusivo como Inspector.

C. Porque solo comprueba configuraciones estáticas.

D. Ninguna de las anteriores.

3. ¿Con qué formato se indican los recursos a los que aplica una directiva de IAM?

A. JSON.

B. ARN, con soporte de comodines.

C. ARN, pero no permite comodines.

D. Ninguna de las anteriores.

4. ¿Cuál es la retención de CloudTrail?

A. De 90 días por defecto.

B. Arbitraria, si se definen *trails* para almacenar los registros.

C. Todas las anteriores.

D. Ninguna de las anteriores.

5. ¿Qué permisos por defecto recibe un usuario al que no se le aplica ninguna directiva de IAM?

   A. Ninguno.

   B. De impersonación.

   C. Solo de lectura sobre todos los recursos de la cuenta.

   D. Todos los permisos están habilitados, a menos que los restrinja una política explícitamente.

6. ¿Qué tipos de recursos pueden asumir un rol de IAM?

   A. Usuarios de la propia cuenta.

   B. Un servicio de AWS.

   C. Un usuario de otra cuenta.

   D. Todos los anteriores.

7. ¿Cuál de las siguientes acciones permiten leer los detalles de una instancia de EC2?

   A. `ec2:DescribeInstances`.

   B. `ec2:Describe*`.

   C. `ec2:*`.

   D. Todos los anteriores.

8. ¿Qué valor debe tener el campo `effect` para permitir una acción?

   A. `Allow`.

   B. `Deny`.

   C. `Read`.

   D. `Enabled`.

9. ¿Qué credenciales usa una cuenta con acceso programático?

   A. Usuario y contraseña.

   B. OAuth.

   C. Access key y secret key.

   D. Clave pública y privada.