Implementación de Certificados de Vida Corta y Autenticación de Identidad en Conexiones SSH.

Irene Balaguer Muñoz

Seguridad Informática

Nombre Tutor/a de TF
Jorge Miguel Moneo
Profesor/a responsable de la asignatura
Andreu Pere Isern Deyà

01/2024

Universitat Oberta
de Catalunya

Agradecimientos

A mis padres, a quienes debo un eterno agradecimiento por su amor, paciencia y comprensión. Han sido mi fuente de inspiración y fortaleza durante todos estos años. Su confianza en mí y su apoyo me han animado a seguir adelante.

A Àdria A., cuya presencia en mi vida durante tantos años me ha dado la fuerza que necesitaba. Gracias por creer en mí y por estar siempre dispuesta a ofrecer una palabra de aliento cuando más lo necesitaba, por formar parte de mi vida en todas las etapas importantes.

A María M., tu amistad y apoyo han sido de un valor incalculable para mí. Desde que nos conocimos siempre has estado ahí para escucharme y ofrecerme tu consejo y comprensión. Gracias por escuchar y por tu presencia, han sido un refugio seguro en los momentos más duros.

A todos vosotros, mi más sincero agradecimiento, no faltéis nunca. Sin vuestro apoyo, este viaje no habría sido lo mismo.

A todos los que ya no están…

"Tú siempre estás, aunque no estés"
Gata Cattana, La Escala de Mohs (2016)

FICHA DEL TRABAJO FINAL

Título del trabajo:	*Implementación de Certificados de Vida Corta y Autenticación de Identidad en Conexiones SSH*
Nombre del autor:	*Irene Balaguer Muñoz*
Nombre del consultor/a:	*Jorge Miguel Moneo*
Nombre del PRA:	*Andreu Pere Isern Deyà*
Fecha de entrega (mm/aaaa):	*01/2024*
Titulación o programa:	Grado en Ingeniería Informática
Área del Trabajo Final:	*Seguridad Informática*
Idioma del trabajo:	*Castellano*
Palabras clave	*PKI, Certificados, SSH*

Resumen del Trabajo

En el ámbito de la seguridad informática, el manejo inapropiado de claves SSH representa una vulnerabilidad significativa.

Este trabajo aborda esta problemática presentando una solución basada en la verificación de identidad a través de un Proveedor de Identidad (IdP), en este caso, *Okta*, y la generación de certificados de corta duración por medio de *Smallstep*.

Esta propuesta surge en respuesta a las malas prácticas observadas en empresas, donde las claves SSH, en lugar de ser secretas y únicas, son compartidas entre empleados. La metodología empleada se basó en la integración de las herramientas mencionadas, con el objetivo de reemplazar las claves SSH estáticas por certificados temporales.

Como resultado, se logró establecer un sistema más seguro y trazable de autenticación. Las conclusiones del trabajo subrayan la importancia de adoptar medidas de seguridad más robustas y la efectividad de combinar *Okta* y *Smallstep* en la protección de accesos remotos.

Abstract

In the realm of cybersecurity, improper handling of SSH keys poses a significant vulnerability.

This research addresses this issue by introducing a solution based on identity verification through an Identity Provider (IdP), specifically, *Okta*, and the generation of short-lived certificates using *Smallstep*.

This approach responds to observed malpractices in businesses where SSH

keys, instead of being secret and unique, are shared among employees. The methodology relied on integrating the mentioned tools with the aim of replacing static SSH keys with temporary certificates.

The outcome was the establishment of a more secure and traceable authentication system. The study's conclusions highlight the importance of adopting robust security measures and the effectiveness of combining *Okta* and *Smallstep* in safeguarding remote accesses.

Índice

Lista de figuras

1. Introducción

El panorama actual de la seguridad informática es tan complejo como crucial. Con el aumento de amenazas y la expansión de la infraestructura tecnológica, las organizaciones se encuentran en una carrera constante para proteger sus activos digitales.

En años recientes, hemos sido testigos de un alarmante incremento en la frecuencia y sofisticación de ataques dirigidos a empresas. En enero de 2023, *Twitter* y *T-Mobile* sufrieron importantes brechas de datos, comprometiendo la seguridad de millones de usuarios y exponiendo información crítica. A estos incidentes se sumó, en marzo del mismo año, un ataque similar dirigido a *AT&T* [1].

Estos eventos no solo subrayan la vulnerabilidad de incluso las corporaciones más grandes y consolidadas, sino que también resaltan la necesidad imperante de adoptar medidas de seguridad más robustas y eficientes. Es un hecho bien documentado que la mayoría de los ataques cibernéticos surgen cuando un atacante obtiene credenciales de acceso. Ya sea mediante técnicas de *phishing*, explotación de vulnerabilidades no parcheadas, ataques de fuerza bruta o incluso manipulación social, una vez que los atacantes tienen acceso a credenciales válidas, pueden moverse lateralmente a través de redes, exfiltrar datos y causar daños significativos, desde brechas masivas de datos que afectan a millones de usuarios, hasta ataques de *ransomware* que paralizan operaciones críticas.

Estos ataques no solo generan pérdidas económicas directas, sino que erosionan la confianza del cliente y dañan la reputación corporativa, con repercusiones que pueden perdurar mucho tiempo después de que el incidente inicial haya sido resuelto.

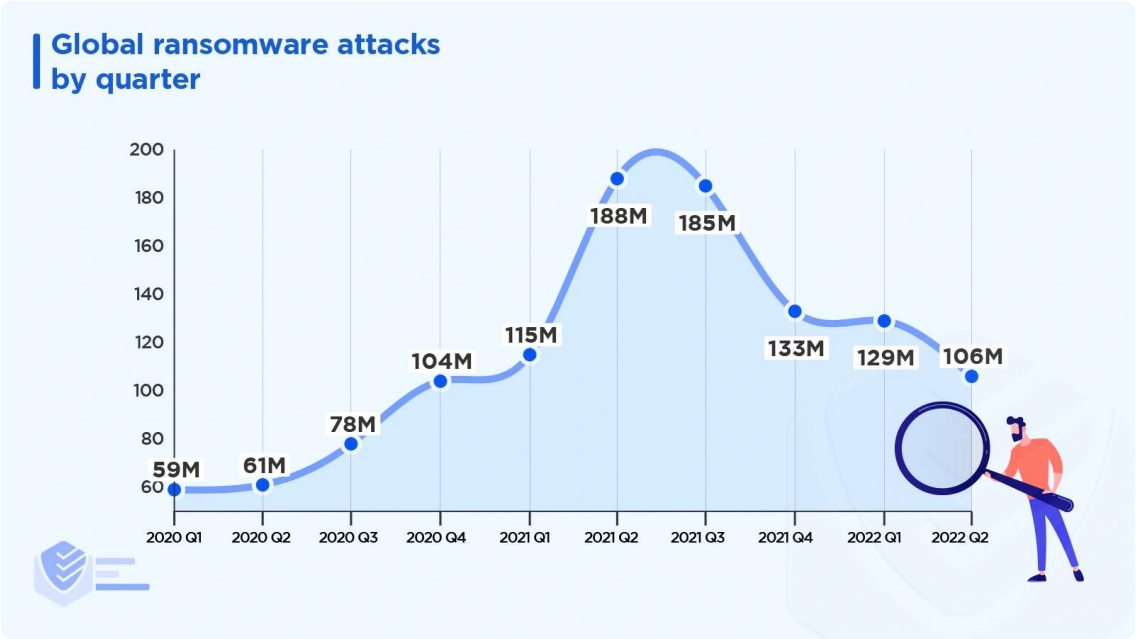


Figura 1. *Ataques ransomware desde Q1 2021 hasta Q2 2022 [2].*

En este contexto, el protocolo SSH emerge como una herramienta esencial, garantizando comunicaciones seguras y cifradas en múltiples entornos. Sin embargo, la fortaleza de este protocolo se ve frecuentemente socavada por prácticas inadecuadas, en particular, la gestión deficiente de las claves SSH.

El acto de compartir claves SSH, aunque pueda parecer una solución práctica a corto plazo, representa un grave riesgo de seguridad. Cada clave SSH debería ser única, secreta y de uso exclusivo. Al compartirlas, no sólo se pierde la capacidad de rastrear y atribuir acciones a individuos específicos, sino que también se eleva el riesgo de que estas claves caigan en manos equivocadas. La falta de trazabilidad significa que es prácticamente imposible determinar quién accedió a un recurso, cuándo lo hizo y qué acciones realizó.

Más allá de la falta de trazabilidad, la retención y uso continuado de claves por empleados que ya no forman parte de la organización es una puerta abierta a potenciales brechas de seguridad. Es una vulnerabilidad que espera ser explotada. Y, para complicar aún más el panorama, las claves SSH, en muchas ocasiones, no se almacenan adecuadamente. La ausencia de mecanismos seguros de almacenamiento y gestión puede llevar a que estas claves se expongan, se pierdan o sean robadas.

Pero, ¿por qué se llega a esta situación? Las razones pueden ser variadas: desde la falta de conciencia sobre la importancia de la gestión adecuada de claves, hasta la

ausencia de herramientas que faciliten su correcto manejo sin entorpecer las operaciones diarias. Lo cierto es que esta problemática es un reflejo de desafíos más amplios en la seguridad informática: la tensión entre la operatividad y la seguridad, y la necesidad de equilibrar ambos aspectos.

Este trabajo se sumerge en esta problemática, buscando no solo diagnosticar y entender sus raíces, sino también proponer y evaluar soluciones concretas. A través de la integración de soluciones de vanguardia, como la autenticación de identidad con un *IdP* y la implementación de certificados de vida corta, buscamos trazar un camino hacia una gestión de claves SSH más segura, eficiente y acorde a las demandas actuales. Es un viaje que nos lleva desde la identificación de fallos en prácticas establecidas hasta la adopción e implementación de innovaciones que pueden redefinir cómo las organizaciones protegen sus activos digitales.

1.1. Contexto y justificación del Trabajo

En la era actual, caracterizada por la presencia de la tecnología en todos y cada uno de los ámbitos en los que nos relacionamos diariamente, la ciberseguridad se ha convertido en un pilar fundamental para la protección de la información y la infraestructura tecnológica en el ámbito laboral. Las operaciones y comunicaciones, esenciales para el buen funcionamiento de las organizaciones, se basan en herramientas y protocolos que, aunque son muy poderosos, pueden verse mermados por prácticas inseguras. El protocolo SSH, un pilar de la seguridad en comunicaciones, es un ejemplo concreto de esto. En este caso en particular, el punto de partida que nos lleva a la creación de este trabajo es la compartición habitual de claves SSH entre empleados, aunque pueda parecer una solución práctica y especialmente rápida, en realidad representa un riesgo significativo. Esta práctica elimina cualquier trazabilidad de acceso y aumenta el riesgo de incursiones no autorizadas, especialmente cuando se trata de entorno productivo y activos desplegados en entornos reales [3].

Como ya se ha mencionado en el apartado de introducción, esta problemática no es simplemente una cuestión teórica o aislada. A nivel global, en años recientes, se ha observado una creciente cantidad de ataques dirigidos a organizaciones [4], lo que refuerza la urgencia de reforzar las medidas de seguridad. Estos ataques no solo tienen implicaciones técnicas, sino que afectan la confianza de clientes, socios y empleados, y pueden tener consecuencias financieras y reputacionales duraderas. Según el reporte de costes debido a brechas de datos realizado por *IBM* y *The*

Ponemon Institute ha aumentado un 2% comparado con el pasado año 2022, este valor se traduce a un total de 4.35 millones de dólares [5].

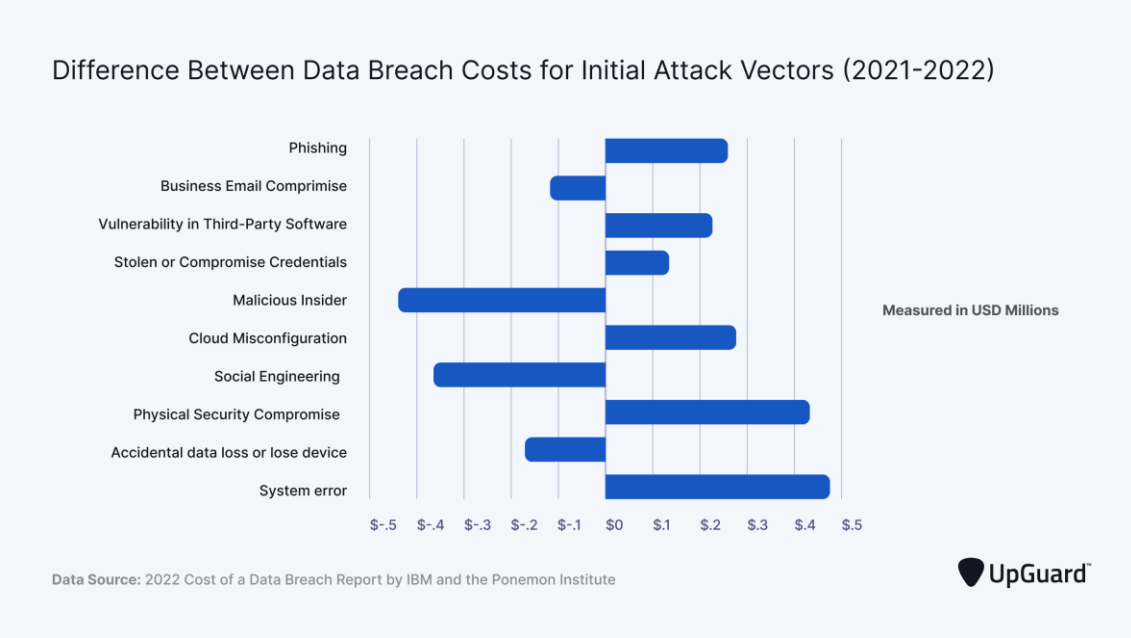


Figura 2. *Diferencia entre costes por brechas de datos según el vector de ataque entre 2021 y 2022 [6].*

Confrontado con esta realidad, y basándonos en las carencias que se encuentran hoy día en el ámbito tecnológico respecto a la seguridad de cualquier tipo de dato sensible para la empresa, se ha propuesto investigar y desarrollar una solución más segura y eficiente. Fue en este punto donde la integración de tecnologías como *Okta*, un reconocido IdP *(Identity Provider)*, y *Smallstep*, una herramienta *open source* especializada en la generación de certificados de vida corta, emergió como una propuesta prometedora.

El objetivo no es meramente presentar un nuevo método o herramienta, sino impulsar un cambio en la mentalidad y cultura organizativa en torno a la seguridad. Es esencial destacar la importancia de una autenticación sólida y la trazabilidad en el acceso a los recursos, y cómo tales prácticas pueden influir directamente en la protección de activos y en la confianza que depositan en nosotros diversas partes interesadas.

1.2. Motivación

Como trabajadora del sector tecnológico y más concretamente en el ámbito de la ciberseguridad he sido testigo directo de las vulnerabilidades que surgen en entornos laborales cuando no se prioriza adecuadamente la seguridad.

La inspiración para mi proyecto de final de grado surge de un desafío que encuentro día a día en mi puesto de trabajo y cada vez en muchas más empresas del sector tecnológico: la gestión de las claves SSH. Tradicionalmente, las claves SSH han sido compartidas entre empleados o reutilizadas a través de múltiples dispositivos, una práctica que, si bien es común, está repleta de riesgos y vulnerabilidades.

El compartir claves SSH puede llevar a una falta de trazabilidad, donde la responsabilidad de las acciones realizadas bajo una clave compartida se vuelve difusa y, por ende, la rendición de cuentas se diluye. Este escenario es un terreno fértil para el abuso de privilegios y la realización de actividades maliciosas sin un rastro claro. Además, una clave única para múltiples dispositivos se convierte en un punto único de fallo; si dicha clave es comprometida, todos los sistemas que dependen de ella quedan expuestos.

Ante esta problemática, mi trabajo se centra en el uso de *Okta*, nuestro proveedor de identidades, y *Smallstep*, una herramienta *open-source* para crear una infraestructura de certificados de vida corta, que no solo erradica la necesidad de compartir claves SSH, sino que también refuerza la seguridad mediante la vinculación directa de la identidad digital del empleado con los permisos de acceso. Este enfoque no solo mejora la seguridad de la información, sino que también facilita la auditoría y el cumplimiento normativo, asegurando que sólo las personas autorizadas puedan acceder a los recursos pertinentes de manera controlada y temporal.

Este proyecto es, en esencia, el reflejo de mi compromiso con la evolución de la ciberseguridad y mi contribución personal hacia un entorno digital más seguro y eficiente. Es una respuesta a los desafíos que enfrento día a día en mi labor, transformando los problemas en oportunidades para innovar y mejorar.

1.3. Objetivos del Trabajo

Dada la creciente preocupación sobre la seguridad en las comunicaciones y el mal manejo de claves SSH en mi ámbito laboral, este trabajo busca integrar soluciones modernas, específicamente utilizando *Okta* como *IdP* y *Smallstep* para certificados de vida corta, con el propósito de ofrecer un sistema de autenticación más robusto y trazable. Por lo tanto, hemos dividido los objetivos de este trabajo en objetivos de aprendizaje y objetivos principales del trabajo.

1.3.1. Objetivos Generales

El objetivo principal de esta investigación es desarrollar e implementar una solución de seguridad integrada que fortalezca la autenticación de identidad y la gestión de certificados en entornos corporativos, específicamente en conexiones SSH. Nuestro enfoque se centra en la integración de la tecnología de *Okta* para la autenticación de identidad y la generación de certificados de vida corta mediante *Smallstep,* una herramienta *open-source*. Esta combinación busca superar los límites de los métodos tradicionales de claves SSH, proporcionando una alternativa más avanzada y segura basada en certificados de vida corta relacionados con la identidad proporcionada por *Okta*.

Este sistema tiene como meta principal mejorar la trazabilidad de las acciones de los usuarios y reducir significativamente los riesgos de seguridad asociados con prácticas inseguras. Mediante pruebas en un entorno controlado, pretendemos demostrar no solo la robustez y eficiencia de nuestra solución, sino también su facilidad de implementación. Además, aspiramos a concienciar sobre la importancia crítica de modernizar los sistemas de autenticación y el rol vital que juegan las organizaciones en la protección de sus activos digitales y usuarios.

Con esta investigación, buscamos no solo abordar un problema actual en el campo de la ciberseguridad, sino también unir teoría y práctica para ofrecer soluciones efectivas a desafíos contemporáneos en este ámbito. Nuestra solución se orienta a mitigar vulnerabilidades comunes en las conexiones SSH, que suelen ser explotadas por actores malintencionados. Estas vulnerabilidades, aunque no detalladas aquí, están relacionadas con deficiencias en los métodos de autenticación y gestión de identidades, así como en la administración de claves y certificados.

1.3.2. Objetivos Parciales

En el marco de nuestro proyecto, nos proponemos alcanzar varios objetivos específicos que son fundamentales para la creación de una solución integral de seguridad. Estos objetivos se orientan tanto al desarrollo técnico como al conocimiento teórico en el campo de la ciberseguridad, buscando aportar innovaciones significativas en la autenticación de identidad y la gestión de certificados. Los objetivos parciales son los siguientes:

1. **Diseñar una Arquitectura de Seguridad Integrada:** Nuestro primer objetivo es desarrollar una arquitectura de seguridad que combine eficazmente las funcionalidades de *Okta* para la autenticación de usuarios y *Smallstep* para la emisión de certificados. Este diseño busca optimizar la gestión de identidades y acceso seguro, creando un marco robusto y flexible para entornos corporativos.
2. **Estudio Avanzado de Sistemas de Autenticación:** Profundizaremos en el conocimiento sobre sistemas de autenticación, centrándonos especialmente en las funciones y capacidades de los Proveedores de Identidad (IdP). Este estudio nos permitirá entender mejor cómo integrar estos sistemas en nuestra solución de manera efectiva.
3. **Explorar la Tecnología de *Smallstep*:** Investigaremos a fondo la tecnología detrás de *Smallstep*, enfocándonos en cómo los certificados de vida corta pueden reforzar la seguridad y trazabilidad en el acceso a sistemas. Este análisis es crucial para entender los beneficios de implementar estos certificados en entornos corporativos.
4. **Implementación de un Prototipo Funcional:** Uno de los pilares de nuestro proyecto es la creación de un prototipo funcional. Este prototipo demostrará la viabilidad técnica de nuestra solución integrada en un ambiente de pruebas, proporcionando una base sólida para futuras implementaciones.
5. **Documentación y Difusión de Hallazgos:** Nos comprometemos a documentar exhaustivamente todos los hallazgos y las mejores prácticas derivadas del desarrollo del proyecto. Esta documentación servirá como un recurso valioso para futuras investigaciones y aplicaciones en el campo.
6. **Análisis de Casos de Brechas de Seguridad:** Finalmente, analizaremos casos recientes de brechas de seguridad para identificar patrones y áreas de mejora en sistemas de autenticación. Este análisis nos ayudará a afinar nuestra solución y asegurar que aborde eficazmente los desafíos actuales en la ciberseguridad.

Cada uno de estos objetivos contribuirá a construir una solución integral y avanzada para la autenticación de identidad y la gestión de certificados en el ámbito corporativo, enfocándose en la mejora continua de la seguridad de las conexiones SSH.

1.4. Impacto en sostenibilidad, ético-social y de diversidad

El presente trabajo ha sido diseñado y desarrollado teniendo en cuenta los principios éticos, de sostenibilidad y de diversidad. Aunque no está exento de desafíos y áreas de mejora, su objetivo central es mejorar la seguridad y la trazabilidad en las comunicaciones, lo que tiene amplias implicaciones positivas en términos de responsabilidad social y comportamiento ético. La reflexión en torno a estas dimensiones ha sido esencial en todas las etapas del proyecto, desde su concepción hasta su implementación basándonos en los Objetivos de Desarrollo Sostenible (ODS) de las Naciones Unidas [7].

1.4.1. Dimensión de Sostenibilidad:

El Trabajo de Final de Grado (TFG) propuesto para la mejora y fortalecimiento del protocolo SSH a través de la integración con *Okta* y *Smallstep* tiene varias implicaciones en términos de sostenibilidad:

- **Impacto Positivo:** La digitalización y automatización de autenticación y autorización reduce el consumo de recursos físicos, apoyando así a los ODS 12 *(Responsible consumption and production)* y ODS 9 *(Industry, innovation and infrastructure)*.
- **Impacto Negativo:** El consumo energético de servidores y sistemas necesarios para la implementación podría incrementar, aunque de manera marginal. Es relevante para el ODS 7 *(Affordable and clean energy).*

1.4.2. Comportamiento ético y responsabilidad social (RS):

La seguridad en las comunicaciones y la autenticación no es solo una cuestión técnica, sino que tiene un fuerte componente ético y de responsabilidad social:

- **Impacto Positivo:** El proyecto contribuye al ODS 16 (*Peace, justice and strong institutions*) al garantizar comunicaciones más seguras y reducir riesgos de brechas de seguridad. Evitando brechas de seguridad se protege la privacidad y los datos, evitando posibles daños a individuos y organizaciones.

- **Impacto Negativo:** Un uso incorrecto o no autorizado del sistema propuesto podría facilitar comportamientos poco éticos o maliciosos, aunque el diseño está orientado a minimizar este riesgo.

1.4.3. Diversidad y derechos humanos:

Aunque a primera vista puede parecer que un proyecto sobre protocolos de seguridad y autenticación no tenga un impacto directo en cuestiones de género o diversidad, debemos adoptar una perspectiva amplia:

- **Impacto Positivo:** El sistema al ser inclusivo en su diseño, no discrimina en función del usuario, apoyando el ODS 10 *(Reduced inequalities*). Además, se ha prestado especial atención en garantizar que la solución sea accesible a personas con diversidad funcional, alineándose con principios de diseño universal.
- **Impacto Negativo:** No se identifican impactos negativos directos en esta dimensión, siempre y cuando se sigan las buenas prácticas de diseño y se evite reforzar sesgos o estereotipos en la implementación.

1.5. Enfoque y método seguido

El propósito de este estudio es abordar un estudio exhaustivo de las estrategias de autenticación y autorización en el contexto de la seguridad informática sobre el protocolo SSH. Se contempla realizar una evaluación rigurosa de las opciones disponibles, para después determinar la idoneidad de diversas herramientas y técnicas en la consolidación de un sistema seguro.

En este sentido, se valorará la incorporación de prácticas y metodologías estándar en el sector, tales como el uso de protocolos seguros de comunicación y la aplicación de modelos de amenazas actualizados. Se examinarán marcos de trabajo como OWASP para el desarrollo seguro de aplicaciones y se contrastarán con los estándares de ciberseguridad para la gestión de seguridad de la información.

Se realizará un análisis de riesgos, con especial enfoque en el proceso de autenticación y la gestión del protocolo SSH y se implementarán soluciones líderes como *Okta* y *Smallstep*, evaluando su integración dentro de una arquitectura que promueva la defensa en profundidad y la minimización de privilegios. Se explorará la

viabilidad técnica de adaptar estos productos a un entorno de pruebas realizado con la ayuda de *Amazon Web Services (AWS)*.

Se efectuará una revisión de la solución a integrar mediante una prueba de concepto y se buscará determinar la facilidad de detección y corrección de vulnerabilidades, así como la eficiencia en la gestión de identidades y el control de accesos.

Para asegurar un proceso ético y legal, cualquier prueba implicará únicamente sistemas y dispositivos voluntarios, y se llevará a cabo en un entorno controlado, sin comprometer la privacidad o la integridad de los datos.

1.6. Planificación del Trabajo

La naturaleza compleja y detallada de nuestro trabajo, que se centra en la seguridad informática mediante la integración de *Okta* y *Smallstep*, requiere de una estructura y un seguimiento meticuloso. Para ello, se ha diseñado un Plan de Trabajo, que no solo servirá como guía durante las diferentes etapas del proyecto, sino que también permitirá anticipar posibles desafíos y ajustar el rumbo cuando sea necesario.

El Plan de Trabajo se ha dividido en varias etapas y tareas, cada una con objetivos claros y fechas específicas para su realización. La idea es tener una visión global del proyecto y poder distribuir de manera eficiente el tiempo y los recursos. Las etapas han sido establecidas en base a las entregas requeridas y los hitos claves del proyecto.

En el Anexo 8.1, presentamos el diagrama de Gantt de cada una de las entregas, que ilustra de manera gráfica y cronológica las tareas y etapas establecidas o posibles cambios en nuestro Plan de Trabajo. Este diagrama será una herramienta vital para monitorizar el progreso del proyecto, asegurando que permanecemos en el camino correcto y cumpliendo con los plazos estipulados.

1.7. Análisis de Riesgos

La implementación de un sistema de autenticación y autorización basado en la integración de *Okta* y *Smallstep* para mejorar la seguridad SSH tiene múltiples ventajas, pero, como cualquier proyecto tecnológico, no está exento de riesgos. A continuación, se detalla un análisis de los principales riesgos asociados:

1. **Integración Técnica Inadecuada:**
   - **Descripción:** Dificultades técnicas al intentar integrar *Okta* con *Smallstep*, debido a incompatibilidades o limitaciones no anticipadas.
   - **Mitigación:** Realizar pruebas de integración preliminares y estudios técnicos detallados antes de la implementación completa.
2. **Fallas de Seguridad:**
   - **Descripción:** Posibilidad de vulnerabilidades de seguridad que podrían surgir de la integración, o del uso inadecuado del sistema.
   - **Mitigación:** Realizar evaluaciones y auditorías de seguridad periódicas. Asegurarse de que las implementaciones estén actualizadas y seguir las mejores prácticas de seguridad.
3. **Tiempo de Inactividad No Planificado:**
   - **Descripción:** Interrupciones del servicio debido a actualizaciones, fallas o problemas imprevistos.
   - **Mitigación:** Establecer protocolos de respaldo y recuperación, así como sistemas de monitorización para detectar y resolver problemas rápidamente.
4. **Resistencia al Cambio por Parte de los Usuarios:**
   - **Descripción:** Los empleados o usuarios del sistema pueden resistirse a adoptar el nuevo método de autenticación.
   - **Mitigación:** Ofrecer capacitación y recursos educativos. Comunicar los beneficios y la necesidad del cambio para la seguridad organizacional.
5. **Costos Inesperados**:
   - **Descripción:** Inicialmente no se plantea ningún costo de la implementación, mantenimiento o resolución de problemas, pero podría surgir algún gasto inesperado.
   - **Mitigación:** Establecer un presupuesto detallado y mantener un fondo de reserva para contingencias.

6. **Desactualización Tecnológica:**
    - **Descripción:** Los productos o herramientas utilizadas podrían quedar obsoletos o ser reemplazados por alternativas más eficientes.
    - **Mitigación:** Realizar revisiones periódicas de la tecnología utilizada y estar dispuesto a adaptarse a nuevas soluciones si resultan ser más beneficiosas.
7. **Problemas de Escalabilidad:**
    - **Descripción:** El sistema podría no ser capaz de manejar un aumento en el número de usuarios o en la demanda de autenticación.
    - **Mitigación:** Diseñar la implementación con escalabilidad en mente y monitorizar activamente la demanda del sistema.
8. **Riesgos Legales y de Cumplimiento:**
    - **Descripción:** Cambios en las regulaciones de privacidad y seguridad pueden afectar la implementación.
    - **Mitigación:** Mantenerse informado sobre las regulaciones relevantes y estar dispuesto a hacer ajustes según sea necesario.

La identificación y mitigación proactiva de estos riesgos será crucial para el éxito del proyecto y para garantizar que el sistema no solo sea técnico sino también operacionalmente robusto.

1.8. Breve sumario de productos obtenidos

El proyecto aborda la integración de *Okta* y *Smallstep* para fortalecer la seguridad en conexiones SSH, basándose en la identidad del empleado. A lo largo del desarrollo, se han generado diversos productos esenciales para comprender, implementar y presentar la solución propuesta:

- **Plan de Trabajo:** Descripción y planificación de tareas, sirviendo como guía cronológica para el desarrollo del proyecto.
- **Memoria Final:** Documento detallado que abarca todas las fases del proyecto, desde la definición del problema hasta las conclusiones.
- **Revisión Bibliográfica:** Recopilación de fuentes y estudios consultados, esencial para situar el proyecto en el contexto actual.
- **Documentación Técnica:** Conjunto de documentos que detallan el funcionamiento, tecnologías y configuración del sistema propuesto.

- **Prototipo del Sistema:** Desarrollo técnico que integra *Okta* y *Smallstep*, demostrando la viabilidad de la solución.
- **Vídeo Resumen:** Contenido audiovisual que presenta de manera concisa el proyecto y su impacto.
- **Presentación del Proyecto:** Síntesis visual y oral diseñada para comunicar los resultados y logros del proyecto.
- **Defensa del Proyecto:** Preparación y justificación del trabajo ante una junta de evaluación.

1.9. Breve descripción de los otros capítulos de la memoria

A lo largo de esta memoria, se desplegará un viaje detallado por el mundo de la autenticación, específicamente centrándose en las implicaciones del protocolo SSH y la necesidad crítica de solucionar los desafíos asociados con claves. Inicialmente, nos adentraremos en la esencia teórica del SSH y su autenticación, desgranando sus características y vulnerabilidades intrínsecas. Se dará especial importancia al estado actual del arte, explorando investigaciones y desarrollos recientes que aborden problemas similares, proporcionando así un marco contextual para nuestro proyecto.

Siguiendo este cimiento teórico, la memoria se centrará en Proveedores de Identidad, particularmente destacando la configuración y potencial de *Okta* en garantizar autenticaciones robustas. Este recorrido teórico se complementa con una introducción detallada sobre los certificados de corta duración, identificando a *Smallstep* como una herramienta esencial en esta propuesta.

Posteriormente, el enfoque se traslada del diseño a la acción, describiendo el proceso de integración y desarrollo con *Okta* y *Smallstep*. Esta fase de implementación es crucial y se complementa con un protocolo de pruebas riguroso, cuyos resultados iniciales ofrecen una visión preliminar del desempeño del sistema y las posibles áreas de mejora.

Finalizando, la memoria culmina con una evaluación meticulosa, contrastando la eficacia de la implementación con los objetivos propuestos al inicio. A través de una discusión reflexiva, se consideran las limitaciones, se comparan los hallazgos con trabajos similares y se resaltan los aprendizajes adquiridos. Esta retrospección no solo

sirve como culminación del proyecto, sino que también sienta las bases para futuras investigaciones y mejoras en el ámbito de la seguridad en autenticación.

2. Estado del Arte

Con el crecimiento exponencial de las redes y la computación en la nube, la necesidad de garantizar conexiones seguras entre dispositivos y servidores se ha vuelto imperativa. En este contexto, el protocolo SSH ha emergido como una solución estándar para garantizar comunicaciones seguras.

En este estado del arte vamos a explicar en qué punto se encuentra el protocolo SSH en términos de autenticación a nivel académico, revisando trabajos previos al estudio y revisando lo que pueden aportarnos a nuestra investigación, por ello, en nuestro caso se dan unas definiciones de qué es SSH, cómo funciona y qué soluciones existen para la autenticación segura evitando el uso de claves.

2.1. ¿Qué es el protocolo SSH?

Secure Shell (SSH) es un protocolo de uso general que ha ganado amplia aceptación en la comunidad tecnológica gracias a su enfoque en proporcionar comunicaciones seguras sobre canales inseguros. Es ampliamente utilizado para la administración de sistemas, acceso a terminales remotos y para la transferencia de archivos en redes. El protocolo de autenticación SSH, como se define en el RFC4251 [8], es un componente esencial de esta arquitectura, diseñado específicamente para autenticar un cliente hacia un servidor.

El protocolo de autenticación SSH es fundamental para garantizar que las comunicaciones realizadas a través de SSH sean seguras y confiables. A través de este protocolo, las organizaciones pueden proteger sus sistemas y datos de accesos no autorizados, garantizando que solo las entidades verificadas puedan acceder a recursos protegidos.

En la práctica, esto significa que cuando un administrador intenta acceder a un servidor remoto a través de SSH, el protocolo de autenticación garantizará que dicho administrador es la persona correcta al verificar sus credenciales antes de conceder acceso. Esta capa de seguridad es esencial para evitar ataques, como intentos de acceso por fuerza bruta o usurpación de identidad.

2.2. Orígenes del SSH

El protocolo SSH *(Secure Shell)* surgió como una solución a los problemas de seguridad inherentes a los protocolos de red existentes en la década de 1990. Fue desarrollado por Tatu Ylönen, un investigador de la Universidad de Tecnología de Helsinki en Finlandia, como respuesta directa a un ataque de sniffing de contraseñas en la red de su universidad. Ylönen, que no tenía experiencia previa en criptografía, se dedicó a aprender y desarrollar un programa que pudiera garantizar comunicaciones seguras a través de Internet. Después de tres meses de trabajo, presentó la primera versión de SSH en el verano de 1995 [9].

El protocolo SSH se diseñó inicialmente para ser una alternativa más segura al protocolo TELNET, que había sido desarrollado a finales de la década de 1960 y carecía de las medidas de seguridad necesarias para las comunicaciones en un mundo cada vez más conectado a través de Internet. Con la capacidad de operar servicios de red de forma segura sobre una red no segura, SSH abordó no solo el problema del sniffing de contraseñas, sino también otros problemas de ciberseguridad significativos de la época.

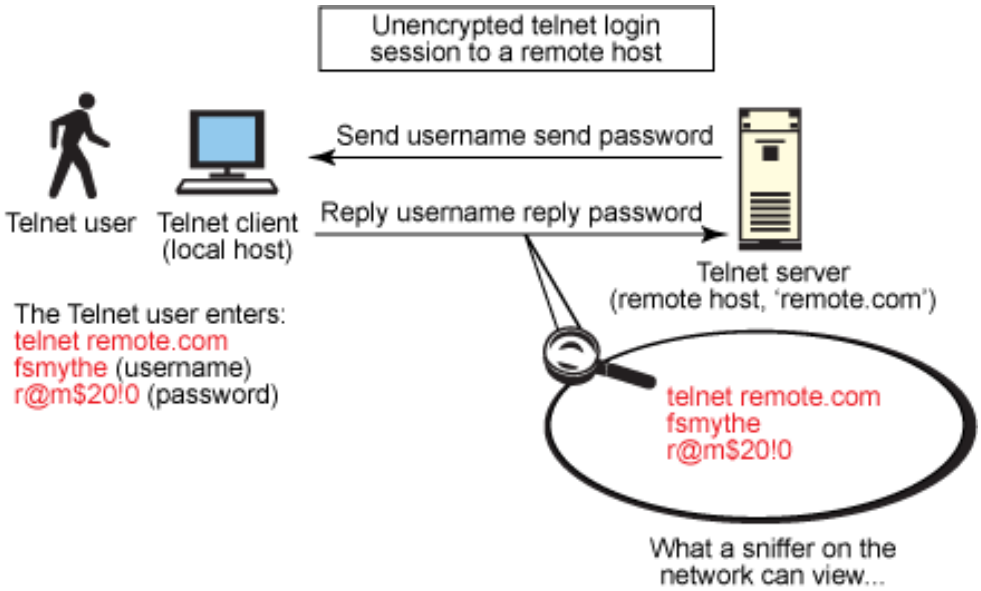


Figura 3. *Sesiones del protocolo Telnet [10].*

La adopción del protocolo SSH fue rápida y amplia, debido a su facilidad de uso y a que no requería de una infraestructura centralizada. Esta popularidad llevó a Ylönen a fundar *SSH Communications Security Corp* a finales de 1995 para dar soporte comercial al protocolo. A pesar de los desafíos iniciales, la empresa logró atraer a

grandes clientes y se convirtió en un líder del mercado para SSH empresarial, especialmente con el desarrollo de Tectia, un producto que permitía transferencias de archivos seguras y acceso remoto.

Con el tiempo, el protocolo SSH evolucionó a SSH-2, una versión más nueva que fue adoptada como RFC *(Request for Comments)* por la *Internet Engineering Task Force* (IETF) en 2006. SSH-2 incluía muchos algoritmos nuevos y no era compatible con su predecesor SSH-1. La necesidad de mantener SSH libre y accesible llevó a los desarrolladores de *OpenBSD* a crear *OpenSSH*, una implementación gratuita del protocolo SSH-2 que soporta tanto SSH-1 como SSH-2, manteniendo el compromiso con la seguridad y la accesibilidad.

Estos desarrollos no sólo respondieron a necesidades técnicas sino también a preocupaciones políticas, ya que en aquel momento existía un gran debate sobre la criptografía, especialmente en Europa, donde Estados Unidos estaba presionando para limitar el cifrado por razones de ciberseguridad. Ylönen tuvo motivaciones políticas significativas para publicar SSH como freeware, ya que consideraba que no era adecuado que una sola parte tuviera toda la información y la capacidad de eliminar sistemas a su antojo.

En resumen, el protocolo SSH fue desarrollado para abordar vulnerabilidades críticas en la seguridad de las redes informáticas y evolucionó en respuesta a la necesidad de mantener Internet como un espacio seguro y abierto. Su origen y desarrollo reflejan la intersección entre la innovación tecnológica, las necesidades de seguridad de la información y el contexto político global de la época.

2.3. Actualidad del SSH

El *Secure Shell* (SSH) ha evolucionado significativamente desde su creación. Lo que comenzó como una solución para proporcionar comunicaciones cifradas seguras en un entorno de red no seguro ha crecido y se ha adaptado a los cambiantes paisajes de la ciberseguridad. SSH no es solo una herramienta para administradores de sistemas; se ha convertido en una parte esencial de la infraestructura de seguridad de la información en organizaciones de todos los tamaños.

Durante años, la autenticación basada en contraseña y clave pública ha sido el pilar de SSH. Aunque efectivos, ambos métodos tienen sus limitaciones en términos de

seguridad y administración. La autenticación basada en contraseña es susceptible a ataques de fuerza bruta, mientras que la administración de claves públicas en entornos grandes puede ser un desafío.

Aunque SSH fue concebido como una solución segura para la administración remota y la transferencia de datos, recientes investigaciones han identificado vulnerabilidades en su esquema de autenticación y cifrado. En particular, el paradigma "Encode-then-Encrypt-and-MAC" [11], que SSH emplea para cifrar y autenticar mensajes, ha demostrado poseer debilidades que pueden ser explotadas por actores maliciosos. Esta situación subraya la necesidad de revisar y adaptar las implementaciones de seguridad tradicionales a las amenazas contemporáneas.

En el panorama actual, hay una tendencia creciente hacia la autenticación basada en certificados en combinación con redes Zero Trust [12] en el cual también trabajan con la misma herramienta planteada para este proyecto, *Smallstep* como el generador de certificados que identifican al usuario realizando una conexión SSH. Estas redes operan bajo el principio de "nunca confiar, siempre verificar". En lugar de confiar en cualquier entidad dentro de la red, cada solicitud de acceso se verifica rigurosamente antes de ser otorgada. La autenticación basada en certificados se integra perfectamente con este modelo. Los certificados, que son emitidos por una Autoridad de Certificación (CA), actúan como un sello de aprobación que verifica la identidad de un cliente o servidor.

Adicionalmente, el auge de los Proveedores de Identidad (IdP) ha reforzado este cambio. Los IdP, como *Okta,* en el caso de esta investigación, son capaces de actuar como intermediarios que verifican a los usuarios y les otorgan las credenciales adecuadas. En el contexto de SSH, esto puede traducirse en la emisión de certificados temporales que ofrecen una ventana de tiempo limitada para el acceso, reforzando aún más la seguridad.

En resumen, el estado actual del SSH refleja un esfuerzo continuo para adaptarse y responder a los desafíos de seguridad emergentes. Con el auge de las redes Zero Trust y la autenticación basada en certificados, SSH continúa siendo relevante y esencial, garantizando comunicaciones seguras en la era moderna.

2.4. El futuro del protocolo SSH

El futuro del protocolo SSH se enfoca en la criptografía post-cuántica, centrada en el desarrollo de algoritmos criptográficos resistentes a ataques de computadoras cuánticas, es fundamental en la era de la computación cuántica emergente. Estos algoritmos están diseñados para ser seguros tanto contra computadoras cuánticas como clásicas, abordando la vulnerabilidad de los sistemas de criptografía de clave pública actuales frente a estas nuevas tecnologías. Con la creciente posibilidad de que las computadoras cuánticas puedan romper los esquemas de criptografía tradicionales, la criptografía post-cuántica busca garantizar la confidencialidad y la integridad de las comunicaciones digitales en el futuro [13].

Con los avances en la computación cuántica, la criptografía de clave pública utilizada en SSH para el intercambio de claves y las firmas digitales acabará quedando en desuso, lo que subraya la necesidad de avanzar hacia algoritmos cuántico-seguros. El intercambio de claves (KEX) es fundamental para establecer un canal seguro y debe ser una de las principales prioridades en la transición hacia la seguridad cuántica (QS). La migración a algoritmos de KEX cuántico-seguros es crítica, ya que las computadoras cuánticas avanzadas podrían desencriptar las sesiones actuales en el futuro [14].

La tarea de conceder acceso a través de claves SSH es el segundo componente crítico afectado por la criptografía post-cuántica. Los servidores SSH verifican la identidad de la entidad solicitante mediante funciones criptográficas, otorgando acceso solo al portador de la clave privada correcta. En un mundo post-cuántico, se espera que los actores maliciosos sean capaces de derivar claves privadas a partir de sus contrapartes públicas, lo que hace imperativo la sustitución de las claves SSH actuales por algoritmos de firma de clave seguros contra amenazas cuánticas.

El camino hacia la agilidad criptográfica incluye evaluación, adopción y migración de la criptografía clásica a la cuántico-segura. Esto implica primero evaluar el entorno SSH actual, luego adoptar el uso de algoritmos QS junto con los clásicos, y finalmente migrar por completo a algoritmos QS para preservar la confidencialidad e integridad de la comunicación.

Los estudios y referencias sobre la adaptación de SSH a la criptografía post-cuántica y la gestión de certificados e identidades son esenciales para esta transición.

2.5. Arquitectura del SSH

En el contexto de la seguridad informática, el protocolo *Secure Shell* (SSH) ha emergido como una solución robusta y omnipresente para la gestión segura de sistemas remotos. Este capítulo se adentra en el protocolo SSH, desentrañando los entresijos de su arquitectura y los mecanismos que lo han convertido en un estándar de seguridad.

Desde sus orígenes como un reemplazo seguro para protocolos no cifrados hasta su posición actual como un pilar de las operaciones de seguridad en red, SSH ha evolucionado para enfrentar un panorama de amenazas en constante cambio. Exploraremos la estructura operativa de SSH, sus métodos de autenticación y cifrado, y cómo estos aspectos se combinan para ofrecer una solución de confianza para la administración remota y la transferencia de datos. Además, discutiremos los desafíos inherentes a la gestión de claves y cómo las mejoras y extensiones han fortalecido la funcionalidad y la seguridad del protocolo a lo largo de los años.

La comprensión de estos fundamentos es crucial para apreciar las soluciones modernas de gestión de identidades y acceso, que serán discutidas en los siguientes capítulos de este trabajo.

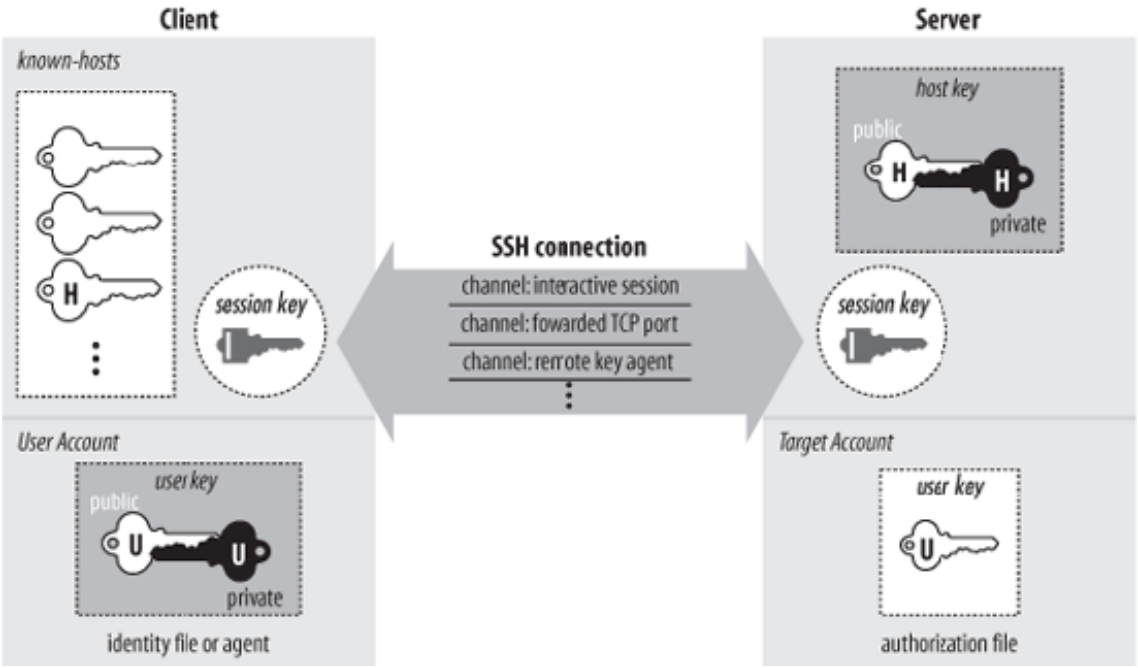


Figura 4. *Arquitectura SSH [15].*

El protocolo *Secure Shell* (SSH) se ha establecido como el estándar para la administración remota segura y la transferencia de datos. Su arquitectura se basa en una serie de principios de diseño enfocados en la autenticación, la confidencialidad, la integridad y la disponibilidad de la información que trataremos en el siguiente apartado.

2.5.1. Principios de la Seguridad

El *Secure Shell* no solo proporciona un medio para acceder de manera remota a un sistema, sino que también implementa una serie de principios de seguridad críticos para garantizar la autenticación [16], integridad, confidencialidad y disponibilidad [17] de las comunicaciones. Estos principios de seguridad, que son fundamentales en cualquier protocolo de comunicaciones, juegan un papel esencial en el diseño y la operación de SSH. Entre ellos, los más destacados son:

- **Autenticación:** Asegura que las partes involucradas en una comunicación son realmente quienes dicen ser. En SSH, esto se logra típicamente a través de métodos como contraseñas o claves SSH, pero existen muchos otros métodos que trataremos en el apartado 2.7 de este estudio.
- **Confidencialidad:** Se refiere a la protección de la información de ser accedida por entidades no autorizadas. SSH utiliza cifrado para garantizar que los datos transmitidos sean inaccesibles para observadores externos.
- **Integridad:** Garantiza que los datos no sean alterados, ya sea accidentalmente o de manera malintencionada, durante la transmisión. SSH emplea mecanismos como los códigos de autenticación de mensajes para verificar que los datos recibidos son los que se enviaron originalmente.
- **Disponibilidad:** SSH contribuye a la disponibilidad garantizando un acceso remoto seguro y fiable al sistema. Sin embargo, es importante aclarar que la disponibilidad a la que contribuye SSH se refiere específicamente a la disponibilidad del propio acceso remoto facilitado por SSH, no necesariamente a la disponibilidad general del servicio al que se accede. En otras palabras, mientras SSH mejora la fiabilidad y seguridad del canal de acceso remoto, no tiene control sobre la disponibilidad del sistema o servicio subyacente. Si hay un problema de disponibilidad en el sistema o servicio, sería independiente de las capacidades de SSH. Por lo tanto, SSH aporta a la propia disponibilidad de su conexión segura, pero no afecta directamente a la disponibilidad del servicio en sí.

Estos principios son la base de la seguridad en las comunicaciones digitales y son esenciales para comprender la importancia y la eficacia del protocolo SSH en entornos seguros de red.

2.5.2. Componentes del protocolo SSH

En su núcleo, SSH opera mediante una arquitectura cliente-servidor, donde el cliente inicia una sesión segura al servidor, que responde con los métodos de autenticación disponibles.
Los componentes clave de SSH incluyen:

- **Cliente SSH:** La interfaz utilizada por el usuario para conectar con un servidor remoto. Los clientes SSH están disponibles para una variedad de sistemas operativos y suelen ofrecer una línea de comandos para la ejecución de operaciones remotas.
- **Servidor SSH:** El sistema que acepta las conexiones de los clientes SSH. Requiere la configuración apropiada de las políticas de seguridad y autenticación para permitir accesos remotos seguros.
- **Protocolo de Transporte SSH (SSH-TRANS):** Se encarga del intercambio inicial de claves y de la configuración de los parámetros de la sesión segura, como la elección del algoritmo de cifrado [18].
- **Protocolo de Autenticación SSH (SSH-USERAUTH)**: Define el proceso mediante el cual el servidor autentica al cliente, que puede incluir métodos como contraseñas, claves públicas o métodos multi-factor [19].
- **Protocolo de Conexión SSH (SSH-CONN):** Facilita la comunicación segura después de que se ha establecido la autenticación, permitiendo la ejecución de comandos y la transferencia de archivos [20].

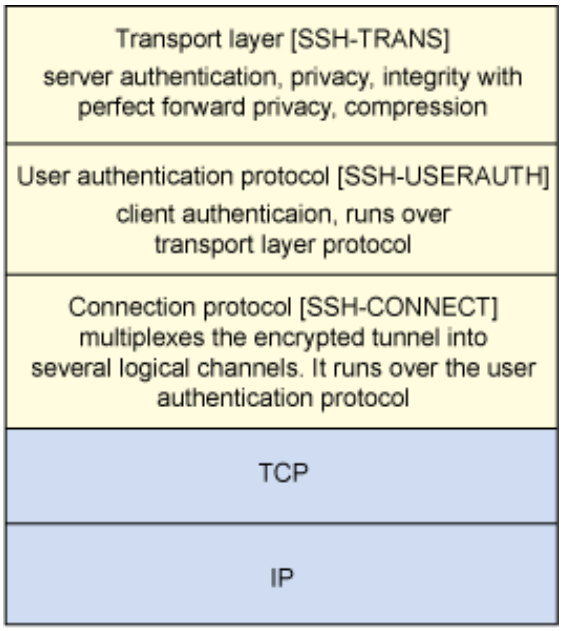


Figura 5. *Capas lógicas del protocolo SSH [10].*

2.5.3. Tipos de cifrado

Una de las características más importantes de SSH es su capacidad para establecer conexiones cifradas, garantizando así la confidencialidad y la integridad de los datos transmitidos.

Para lograr esto, SSH emplea varios tipos de cifrado y técnicas de *hashing* [21], cada uno con un rol específico en el proceso de establecimiento y mantenimiento de la conexión segura del que hablaremos en el apartado 2.6. A continuación, exploraremos los métodos principales de cifrado y hashing utilizados en SSH, los cuales son esenciales para comprender cómo este protocolo protege las comunicaciones entre el cliente y el servidor.

- **Encriptación Simétrica:** SSH utiliza una clave única para cifrar y descifrar datos entre dos máquinas. Este método es rápido y eficiente en recursos, siendo usado en cada sesión de SSH.

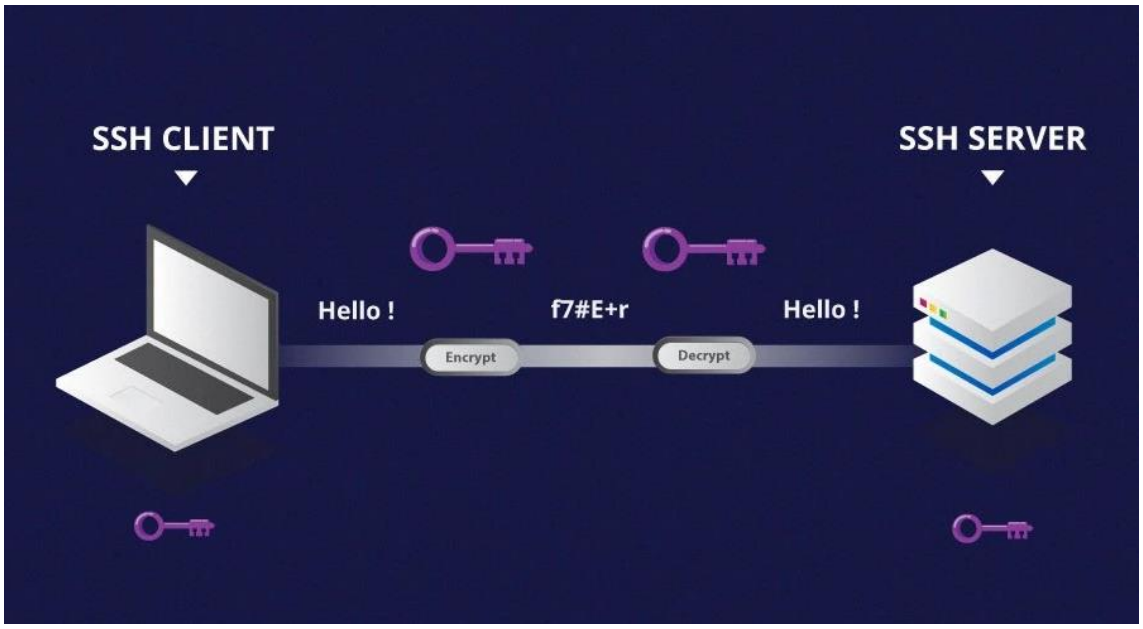


Figura 6. *Ejemplo de encriptación simétrica [22].*

- **Encriptación Asimétrica:** Durante el proceso de autenticación, SSH emplea claves asimétricas temporales (pública y privada) para intercambiar las claves simétricas. Estas claves asimétricas son matemáticamente relacionadas y permiten un cifrado seguro.

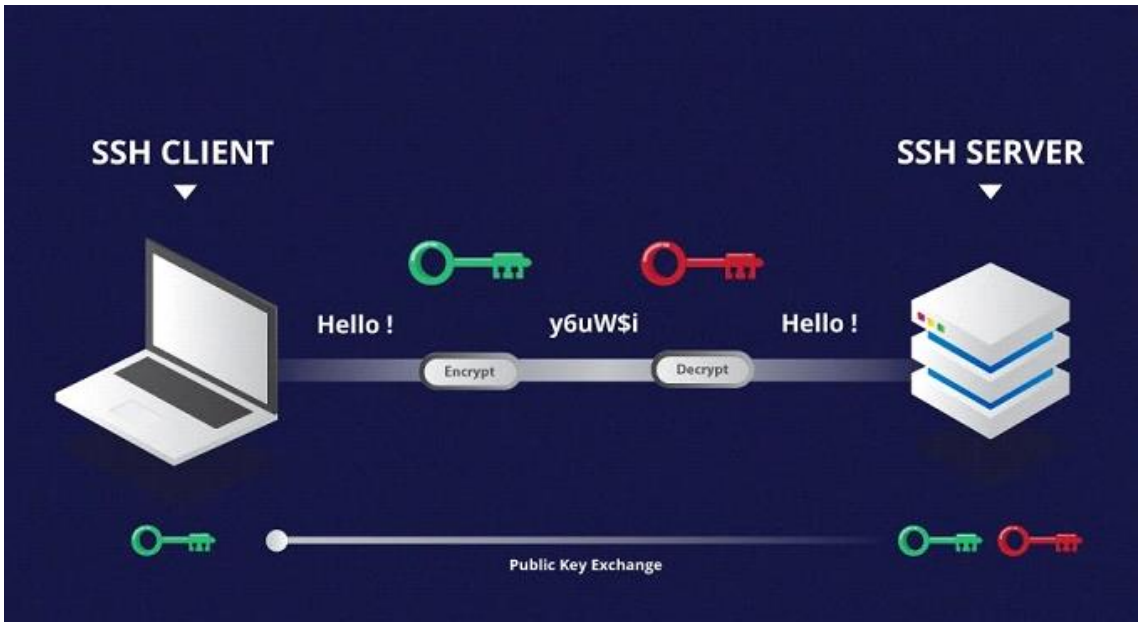


Figura 7. *Ejemplo de encriptación asimétrica [22].*

- **Hashing:** SSH utiliza algoritmos de hashing, como MAC (Message Authentication Code) y HMAC (Hashed Message Authentication Code), para verificar la integridad y el origen de los paquetes de datos. El hashing crea un

valor único a partir de los datos, permitiendo que la máquina receptora confirme que los datos no se han alterado durante la transferencia.

Figura 8. *Ejemplo de encriptación asimétrica [22].*

2.5.4. Seguridad y Gestión de Claves

La gestión de claves es un pilar fundamental en la seguridad del protocolo SSH. Este proceso no solo facilita el establecimiento de una conexión segura entre el cliente y el servidor, sino que también asegura la autenticación y la integridad de los datos a lo largo de toda la sesión SSH.

El manejo adecuado de las claves implica varios pasos cruciales y herramientas específicas que juntos conforman un robusto sistema de seguridad. Desde el intercambio inicial de claves para cifrar la sesión hasta la implementación de políticas de seguridad para el manejo de estas claves, cada aspecto juega un rol vital en la protección de las comunicaciones.

A continuación, examinaremos los componentes clave de la gestión de claves en SSH, incluyendo el intercambio de claves, el uso de claves asimétricas, los agentes de claves y las políticas de seguridad que rigen su uso, proporcionando así una visión completa de cómo SSH mantiene las conexiones seguras y eficientes:

- **Intercambio de Claves:** Al iniciar una sesión SSH, el cliente y el servidor realizan un intercambio de claves, un proceso mediante el cual se establece un

secreto compartido utilizado para cifrar la sesión. Este proceso utiliza algoritmos de intercambio de claves como *Diffie-Hellman* [23].

- **Claves Asimétricas:** Aunque el uso de un par de claves asimétricas —una clave privada que el cliente guarda en secreto y una clave pública que se puede compartir libremente— es frecuente para la autenticación de usuarios, SSH también admite otras formas como la autenticación basada en contraseñas y métodos de autenticación basados en desafíos. Esta diversidad de opciones permite a los usuarios y administradores de sistemas seleccionar el método de autenticación que mejor se adapte a sus necesidades de seguridad y conveniencia.
- **Agentes de Claves:** Son aplicaciones que se ejecutan en el cliente y almacenan las claves privadas utilizadas para la autenticación automática. Los agentes de claves permiten un uso más seguro y conveniente de las claves sin necesidad de ingresar contraseñas repetidamente.
- **Políticas de Seguridad:** La configuración del servidor SSH puede ser ajustada para requerir ciertos parámetros de seguridad, como tamaños mínimos de clave y tipos de algoritmos de cifrado permitidos, para mejorar la postura de seguridad general.

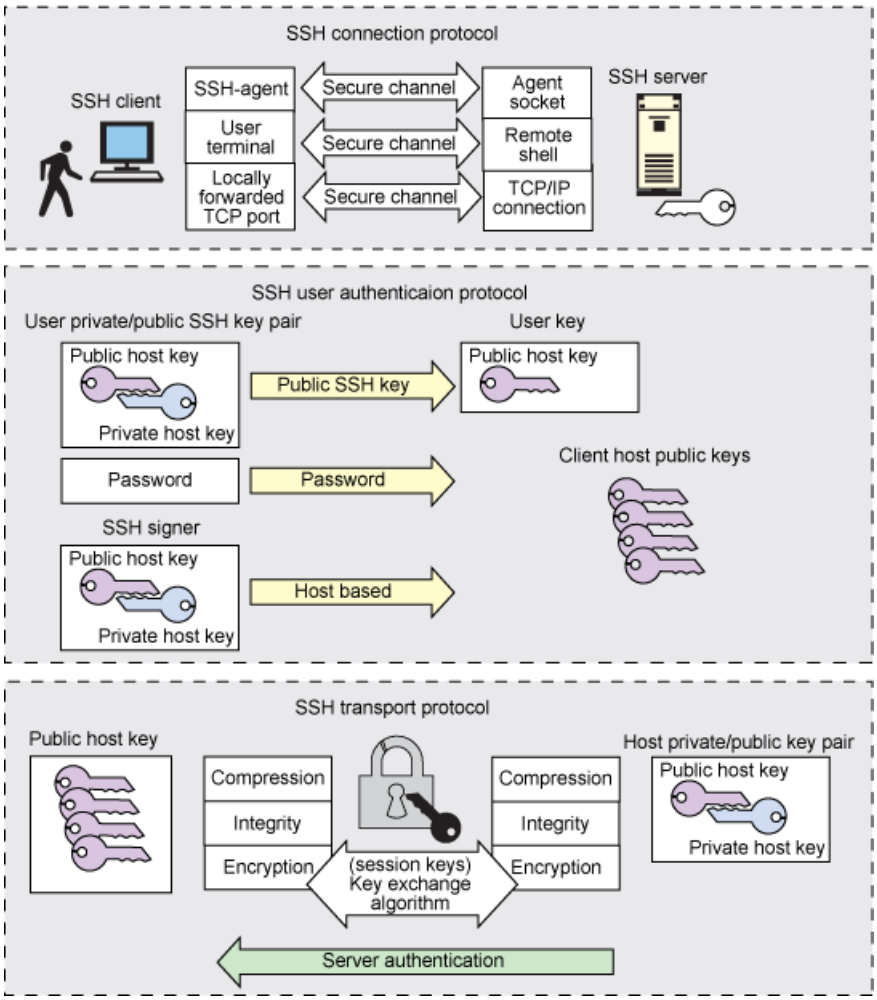


Figura 9. *Diagrama de transacción de llaves privadas y públicas del protocolo SSH [10].*

La seguridad en SSH no es solo una función de los algoritmos y herramientas utilizadas, sino también de cómo se configuran y se gestionan. Una política de seguridad bien definida y aplicada es esencial para proteger contra ataques comunes, como el *spoofing* de *IP*, el secuestro de sesiones y los ataques de fuerza bruta.

2.6. ¿Cómo funciona?

Para garantizar comunicaciones seguras en canales potencialmente inseguros, el protocolo *Secure Shell* (SSH) utiliza múltiples componentes. Uno de estos es el protocolo de autenticación SSH, definido en el RFC4251. Este capítulo profundizará en la mecánica y el flujo de trabajo de este protocolo, proporcionando una comprensión clara de cómo asegura las conexiones SSH.

A continuación, definimos las fases del protocolo general:

1. **Establecimiento de la Conexión**
    a. **Modelo Cliente-Servidor:** SSH utiliza un modelo cliente-servidor y opera generalmente en el puerto TCP 22.
    b. **Escucha del Servidor:** El servidor escucha las conexiones entrantes en este puerto y prepara una conexión segura.
    c. **Inicio de la Conexión por el Cliente:** El cliente inicia un *handshake* TCP con el servidor para establecer una conexión segura.
2. **Negociación de la Encriptación de la Sesión**
    a. **Presentación de Protocolos de Encriptación:** El servidor muestra al cliente los protocolos de encriptación y sus versiones.
    b. **Acuerdo y Verificación:** Si el cliente tiene un par compatible, se llega a un acuerdo y se inicia la conexión. El cliente verifica la autenticidad del servidor usando su clave pública.
    c. **Algoritmo *Diffie-Hellman*:** Se utiliza este algoritmo para crear una clave simétrica compartida para la sesión.
3. **Autenticación del Usuario**

    Este paso del protocolo describe los dos usos más comunes de autenticación en el protocolo SSH y que queremos mejorar en esta investigación. Más adelante, en el apartado 2.7 encontraremos todos los tipos de autenticación existentes en el protocolo SSH y que cada uno de ellos sigue un flujo diferente para su autenticación.

    a. **Introducción de Credenciales:** El usuario introduce su nombre de usuario y contraseña, que se transmiten de forma segura. Estas credenciales se envían cifradas usando la clave de sesión establecida durante la fase de negociación de la encriptación. El servidor SSH recibe y descifra la contraseña y la compara con la almacenada en sus registros. Si coinciden, el usuario se autentica.
    b. **Uso de Pares de Claves SSH:** Se utilizan pares de claves SSH (claves asimétricas) en lugar de contraseñas para una mayor seguridad. En la autenticación por clave pública, el usuario debe haber registrado previamente su clave pública en el servidor. Durante el proceso de autenticación, el servidor crea un mensaje o desafío criptográfico y lo envía al cliente. El cliente utiliza su clave privada para firmar este mensaje y lo envía de vuelta al servidor. El servidor usa la clave pública del usuario para verificar la firma. Si la firma es válida, demuestra que el usuario posee la clave privada correspondiente y se autentica.

Durante todo el proceso de autenticación, se utiliza un mecanismo de hashing (como *HMAC* del inglés *Hash-Based Message Authentication Codes*) para asegurar la integridad y autenticidad de los mensajes intercambiados.

Esto garantiza que los mensajes no han sido alterados y provienen de una fuente auténtica.

4. **Inicio de la Sesión SSH**
    a. **Acceso Concedido:** Tras la autenticación exitosa, se concede al usuario el acceso SSH al servidor.
    b. **Entorno de Shell Seguro:** El servidor abre el entorno de shell correcto para el usuario autenticado.

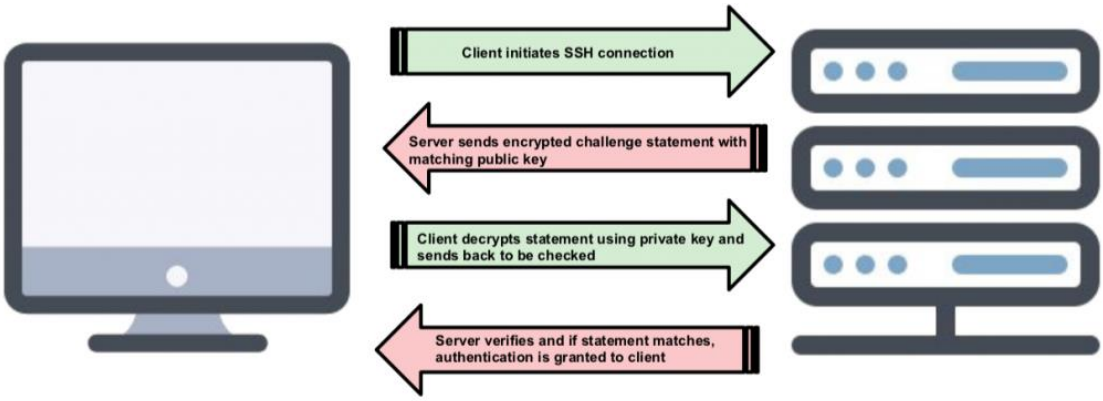


Figura 10. *Ejemplo de autenticación SSH a través de claves públicas [24].*

2.7. ¿Qué tipos de autenticaciones existen?

SSH admite una variedad de métodos de autenticación, lo que lo hace altamente adaptable a diferentes requisitos y escenarios de seguridad. Cada uno de estos métodos tiene sus propias ventajas y desventajas, así como consideraciones de seguridad específicas. A continuación, se exploran algunos de los métodos más comunes:

La **autenticación por contraseña** Es el método más básico y ampliamente utilizado. Implica que el usuario proporcione una contraseña que se verificará contra una almacenada en el servidor. Es fácil de implementar y utilizar, pero, por otro lado, es vulnerable a ataques de fuerza bruta, ingeniería social, etc...

La **autenticación de clave pública** se basa en un par de claves, una privada y una pública. Los clientes envían un mensaje cifrado con su clave privada, y el servidor verifica el mensaje utilizando la clave pública correspondiente del cliente. Aunque es mucho más seguro que la autenticación basada en contraseña y resistente contra la

mayoría de los ataques requiere una gestión adecuada de las claves, y la seguridad puede verse comprometida si se pierde la clave privada o se expone.

La **autenticación basada en certificados** es similar a la autenticación de clave pública, pero en lugar de una clave pública, el cliente presenta un certificado firmado por una Autoridad de Certificación (CA) de confianza. Aporta un nivel adicional de confianza ya que los certificados son emitidos y firmados por CAs, pero depende de la infraestructura de la CA y requiere la gestión de certificados.

La **autenticación *host-based*** permite a un cliente autenticarse basándose en la identidad del host desde el que se conecta, siendo útil para escenarios donde las máquinas específicas tienen permisos definidos. Sin embargo, si el host se ve comprometido, un atacante podría obtener acceso.

La **autenticación multi-factor** combina dos o más métodos de autenticación, como una contraseña seguida de un código de autenticación de un solo uso. Aunque aumenta significativamente la seguridad al requerir múltiples formas de verificación, puede ser menos conveniente para el usuario debido a los pasos adicionales.

Finalmente, la **autenticación basada en desafío-respuesta** implica que el servidor presenta un desafío, como una pregunta o un valor aleatorio, y el cliente debe proporcionar una respuesta adecuada. Aumenta la interactividad en el proceso de autenticación, complicando los ataques automáticos. No obstante, si las preguntas o desafíos se vuelven predecibles, la seguridad puede verse comprometida.

2.8. Retos en la gestión de acceso SSH

La gestión del acceso SSH es un componente crítico en la infraestructura de TI de cualquier organización. Con el crecimiento exponencial en el número de servidores, tanto en entornos de nube como en centros de datos tradicionales, los retos en la gestión de accesos SSH se han magnificado. Este segmento profundiza en los problemas comunes asociados con el acceso SSH y las implicaciones de no abordar estos retos de manera eficaz.

2.8.1. Ineficiencias y riesgos centrados en la gestión del protocolo SSH

La gestión del acceso SSH (*Secure Shell*) incluye varias ineficiencias y riesgos que las organizaciones deben conocer para asegurar la seguridad de sus sistemas. La falta de supervisión central de las identidades y configuraciones de acceso SSH, a menudo manejadas por administradores individuales, puede resultar en riesgos de seguridad significativos:

- **Servidores SSH No Aprobados**: Permitir el acceso al servidor SSH en sistemas donde no es necesario puede ampliar la superficie de ataque.
- **Software SSH Desactualizado**: El software SSH no actualizado puede exponer sistemas a vulnerabilidades y compromisos potenciales.
- **Configuración SSH Vulnerable**: Configuraciones inseguras pueden abrir sistemas a ataques más amplios.
- **Políticas de Acceso Uniforme:** Una política de 'talla única' para el acceso SSH no toma en cuenta el principio de privilegio mínimo necesario, permitiendo que los usuarios tengan más acceso del requerido, lo que aumenta el riesgo de daño si se comprometen las credenciales.

2.8.2. Ineficiencias y riesgos centrados en la gestión de claves

La gestión ineficaz del acceso SSH puede resultar de varias prácticas deficientes, tales como:

- **Uso Compartido de Claves:** La práctica de compartir claves SSH entre múltiples usuarios es notoriamente insegura. Compromete la capacidad de auditar quién hizo qué y cuándo, creando un serio vacío en la trazabilidad y la responsabilidad.
- **Rotación Inadecuada de Claves:** La ausencia de un proceso de rotación regular de claves puede hacer que las claves comprometidas pasen desapercibidas, manteniendo abiertas las puertas para posibles brechas de seguridad.

- **Compromiso de Claves Privadas**: Si las claves privadas no se gestionan de forma segura, pueden comprometerse, permitiendo a los atacantes autenticarse en cuentas donde se confía en la clave privada.
- **Gestión Descentralizada:** La falta de un sistema centralizado para gestionar las claves SSH puede llevar a inconsistencias y a la dificultad de hacer cumplir políticas de seguridad coherentes a través de toda la organización.

En conclusión, aunque SSH es una herramienta esencial para la comunicación segura, requiere una gestión diligente y evaluaciones regulares para mitigar riesgos como el acceso no autorizado, claves comprometidas y configuraciones no seguras. Sin una administración adecuada, el protocolo SSH, en lugar de ser un elemento de fortaleza en la seguridad, puede transformarse en un punto vulnerable crítico de la infraestructura.

2.8.3. Respuestas a la gestión deficiente del protocolo SSH

Para abordar los problemas derivados de una mala gestión del protocolo SSH [25], enfocándonos exclusivamente en errores de configuración, vulnerabilidades y ataques directos al protocolo, se pueden implementar las siguientes medidas:

- **Control Riguroso de Accesos:** Para evitar el acceso no autorizado y limitar la superficie de ataque, es crucial establecer una lista blanca de servidores SSH. Esto significa que solo los servidores verificados y considerados seguros pueden recibir conexiones SSH. Adicionalmente, se deben realizar auditorías de seguridad de manera periódica para identificar y deshabilitar cualquier servidor SSH que opere fuera de los parámetros establecidos.
- **Mantenimiento y Actualizaciones de Software:** La actualización y el mantenimiento del software SSH son fundamentales para la seguridad del sistema. Automatizar este proceso asegura que los parches de seguridad se apliquen tan pronto como estén disponibles. Además, el uso de herramientas de escaneo de vulnerabilidades de manera regular ayuda a identificar configuraciones potencialmente inseguras que necesitan corrección.
- **Fortalecimiento de la Configuración:** Desarrollar guías para una configuración segura de SSH y utilizar herramientas de gestión de configuración de sistemas para aplicar y monitorear estas configuraciones es vital. Es importante prohibir prácticas que puedan comprometer la seguridad,

como el reenvío de puertos y el acceso root remoto. Implementar Firewalls de Aplicaciones Web (WAF) proporciona una capa de seguridad adicional, supervisando y bloqueando el tráfico malicioso dirigido a los servidores SSH.

Estas prácticas ayudarán a reducir la superficie de ataque y mejorar la postura de seguridad frente a vulnerabilidades inherentes al protocolo SSH.

2.8.4. Respuestas a la gestión deficiente de claves

Para contrarrestar estos problemas, las organizaciones están adoptando diversas estrategias de mejora:

- **Sistemas Centralizados de Gestión de Claves:** La centralización del manejo de claves SSH permite una administración y revocación eficientes de las credenciales, así como una visibilidad completa de todas las claves en uso dentro de la empresa.
- **Automatización de la Rotación de Claves:** La implementación de sistemas que automáticamente roten las claves SSH a intervalos regulares ayuda a mitigar los riesgos asociados con claves estáticas y potencialmente comprometidas.
- **Políticas de Acceso Diferenciadas:** Adoptar un enfoque granular para las políticas de acceso que se alinea con el principio de privilegio mínimo necesario, asignando a los usuarios solo los derechos que necesitan para realizar sus tareas.
- **Certificados de Vida Corta:** La utilización de certificados de vida corta para la autenticación SSH representa un avance significativo, ya que estos certificados tienen un periodo de validez limitado y se revocan automáticamente, lo que disminuye considerablemente la ventana de explotación para los atacantes.
- **Herramientas de Gestión de Identidades y Accesos (IAM):** Las soluciones IAM modernas ofrecen un marco robusto para la asignación y revocación dinámica de privilegios, integrando con frecuencia funcionalidades avanzadas como la autenticación multi-factor y el análisis de comportamiento para una seguridad reforzada.

La adopción de estas soluciones no solo mejora la seguridad, sino que también facilita el cumplimiento de regulaciones y la realización de auditorías efectivas. Al abordar proactivamente los desafíos en la gestión de acceso SSH, las organizaciones pueden

asegurar sus operaciones críticas y proteger sus activos digitales de amenazas internas y externas.

3. Preparación técnica

En este capítulo estableceremos las bases teóricas y prácticas necesarias para la implementación de una solución robusta de gestión de certificados SSH de vida corta.

La preparación técnica se centrará en la infraestructura de clave pública (PKI), crucial para la emisión y gestión de certificados digitales, y se explicará cómo *Smallstep* se utiliza para establecer y operar una Autoridad Certificadora (CA) de confianza.

Además, se presentará *Okta* como el Proveedor de Identidad (IdP) seleccionado, desglosando su papel en la autenticación de identidades y cómo se integra con *Smallstep* para vincular de manera segura la identidad de un empleado con sus correspondientes certificados SSH. Se abordará la teoría subyacente de los certificados de vida corta y su importancia en la seguridad moderna, destacando cómo reducen el riesgo de compromiso en comparación con las claves SSH tradicionales y de larga duración.

Por último, se explorará la integración entre *Okta* y *Smallstep* en el contexto de una infraestructura de PKI, ilustrando cómo estas tecnologías pueden trabajar conjuntamente para proporcionar una solución de autenticación segura, eficiente y escalable. La discusión incluirá detalles técnicos sobre la configuración, el proceso de emisión de certificados, y la automatización de la gestión de la identidad y acceso en entornos de TI dinámicos.

Este capítulo sentará las bases necesarias para comprender las decisiones de diseño y las consideraciones de implementación que se detallarán en los capítulos subsiguientes, asegurando una comprensión integral del proyecto y su contexto dentro de la seguridad informática y la gestión de identidades.

3.1. Introducción a *Okta*

Okta es una plataforma de gestión de identidades y acceso basada en la nube que se ha convertido en un pilar para las empresas que buscan simplificar y asegurar el acceso de usuario a aplicaciones y servicios tanto en la nube como en locales. Como líder en el campo de la identidad como servicio (IDaaS), *Okta* proporciona una solución unificada que abarca la autenticación, el inicio de sesión único (SSO), la gestión de identidades y acceso, así como la integración de directorios.

En su núcleo, *Okta* opera como un Proveedor de Identidad (IdP), centralizando y manejando las credenciales de usuario y los perfiles de identidad en una plataforma segura y fácilmente accesible. Al actuar como un intermediario entre el usuario y las aplicaciones, *Okta* garantiza que solo los usuarios autenticados y autorizados puedan acceder a los recursos de TI críticos. Esto no solo mejora la seguridad, sino que también aumenta la eficiencia al reducir la carga administrativa asociada con la gestión de múltiples identidades y contraseñas.

Okta implementa una variedad de opciones de autenticación, desde métodos tradicionales basados en contraseñas hasta enfoques modernos como la autenticación multifactor (MFA), que ofrece una capa adicional de seguridad requiriendo dos o más factores de verificación antes de conceder el acceso. Esto es esencial para proteger contra el robo de credenciales y los ataques de phishing.

Okta también es conocido por su capacidad para proporcionar Inicio de Sesión Único (SSO), un servicio que permite a los usuarios acceder a múltiples aplicaciones y servicios con una única autenticación. SSO no solo mejora la experiencia del usuario eliminando la necesidad de recordar múltiples contraseñas, sino que también reduce significativamente el riesgo de ataques de phishing y otras amenazas de seguridad cibernética, ya que minimiza el número de veces que los usuarios deben ingresar sus credenciales. Para conseguir este propósito se utilizan métodos estándar de la industria como SAML y *OpenID Connect* (OIDC) que explicaremos en los siguientes

subapartados detallando concretamente el caso de OIDC ya que será el método elegido para el desarrollo de nuestra infraestructura.

3.1.1. SAML

El Lenguaje de Marcado para Confirmaciones de Seguridad (SAML por sus siglas en inglés *Security Assertion Markup Language*) es un estándar abierto que se utiliza para el intercambio de datos de autenticación y autorización entre dos partes, usualmente entre un proveedor de identidad (IdP) y un proveedor de servicio (SP). SAML facilita el intercambio de afirmaciones de seguridad, que son declaraciones que utilizan los proveedores de servicios para tomar decisiones de control de acceso gracias a aserciones SAML, es decir, mensajes que incluyen la información necesaria para que un proveedor de servicios confirme la identidad del usuario [26].

El proceso de autenticación SAML generalmente involucra las siguientes etapas:

1. El usuario intenta acceder a un servicio (SP, del inglés *Service Provider*) desde un navegador.
2. El SP genera una solicitud SAML y la envía al navegador.
3. El navegador redirige al usuario al IdP con la solicitud SAML.
4. El IdP autentica al usuario. Una vez autenticado, el IdP genera una respuesta SAML y la envía de vuelta al navegador.
5. El navegador del usuario envía una petición POST al servicio de consumo de afirmaciones *(Assertion Consumer Service)* del proveedor de servicio, incluyendo la respuesta SAML obtenida previamente.
6. El servicio de consumo de afirmaciones procesa esta respuesta, establece un contexto de seguridad en el proveedor de servicio y redirige al navegador hacia el recurso deseado.
7. El navegador solicita nuevamente el recurso objetivo al proveedor de servicio.
8. Como ya existe un contexto de seguridad, el proveedor de servicio proporciona el recurso solicitado al navegador del usuario.

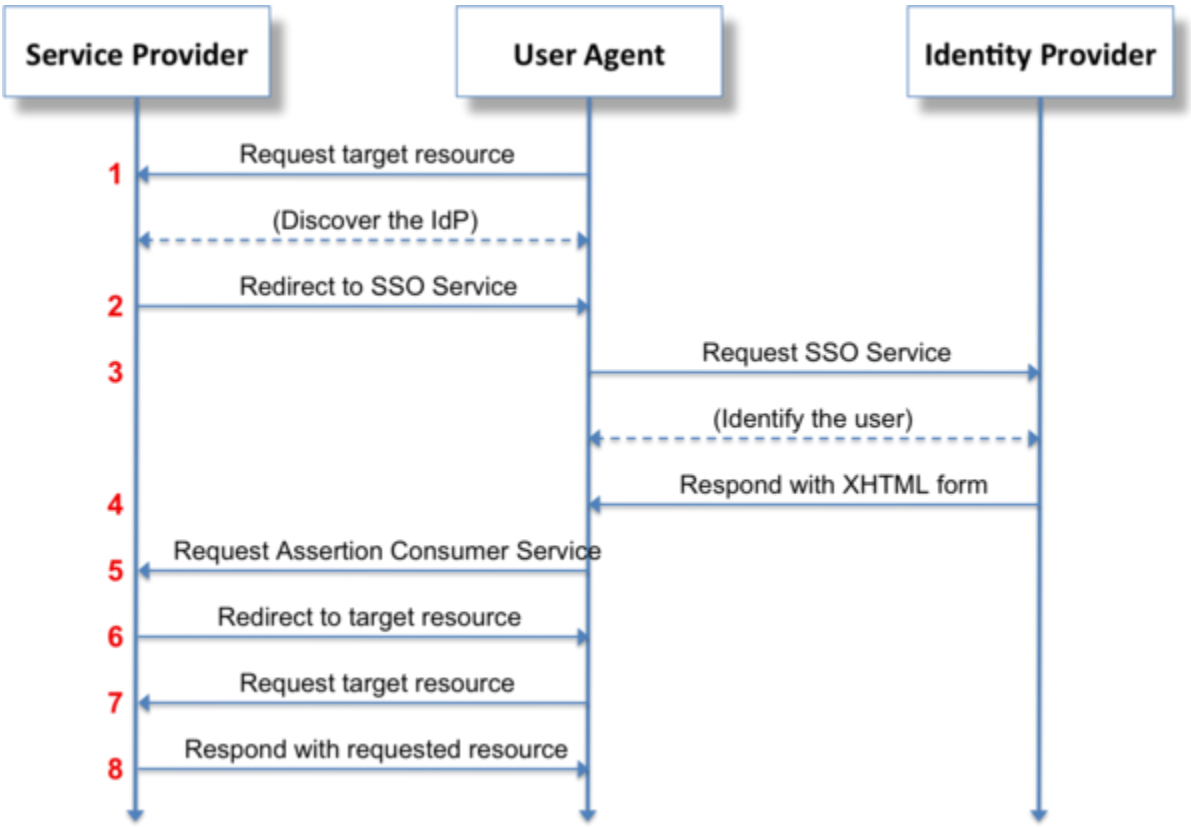


Figura 11. *Single sign-on utilizando SAML en un navegador web [27].*

SAML no especifica el método de autenticación en el IdP, lo que permite utilizar métodos como nombre de usuario y contraseña, autenticación multifactor, o incluso servicios de directorio como LDAP o *Active Directory*.

3.1.2. OIDC

OpenID Connect (OIDC) es un protocolo de autenticación basado en el marco de especificaciones de *OAuth 2.0* definidas en el RFC6749 [28] y RFC6750 [29]. Su propósito es simplificar la manera de verificar la identidad de los usuarios basándose en la autenticación realizada por un Servidor de Autorización, en el contexto del trabajo, *Okta*, y obtener información del perfil del usuario de manera interoperable y similar a REST.

OIDC permite a los desarrolladores de aplicaciones y sitios web iniciar flujos de inicio de sesión y recibir afirmaciones verificables sobre los usuarios y características opcionales como la encriptación de datos de identidad, descubrimiento de Proveedores de OpenID y cierre de sesión [30].

El flujo del protocolo OIDC es el siguiente:

1. El usuario intenta iniciar sesión contra la aplicación y se le redirige al proveedor OIDC y le proporciona un *client ID* único y propio de esa aplicación.
2. El proveedor de *OpenID* autentica y autoriza al usuario para esa aplicación concreta.
3. Se codifican los detalles del usuario en un *id_token (JWT)* que contiene información del usuario y firma que se le proporciona a una página de redirección configurada en el proveedor de *OpenID*.
4. El proveedor confirma el *id_token* desencriptándolo gracias al uso de una clave pública.

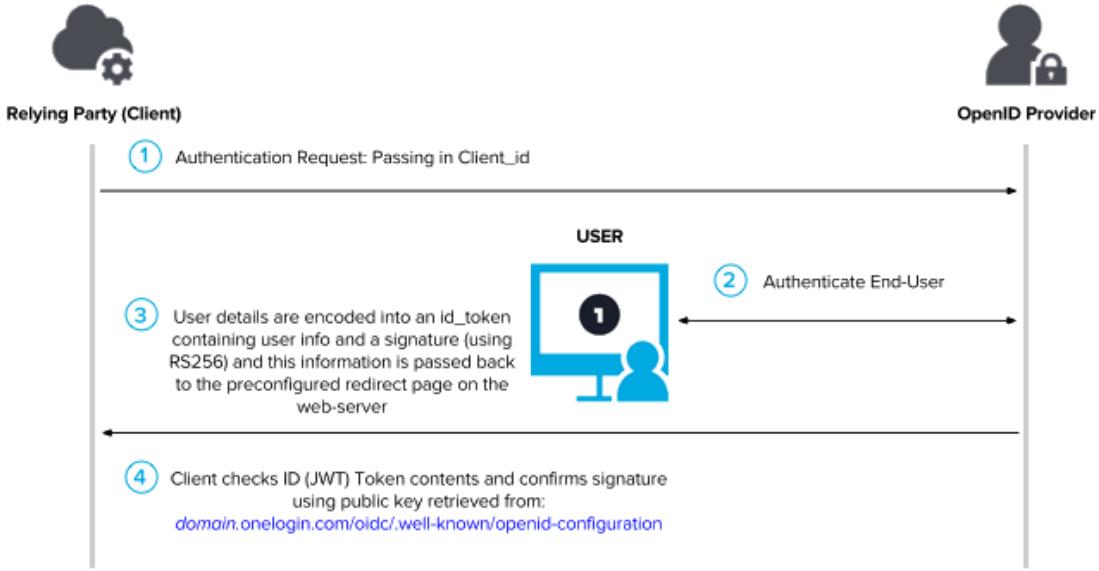


***Figura* 12.** *Flujo de autenticación OIDC [31].*

3.2. Introducción a *Smallstep*

Smallstep es un proveedor de una plataforma de código abierto diseñada para generar y operar Autoridades de Certificación (CAs). La plataforma se compone de dos herramientas principales: step-ca, que mantiene la cadena de certificados y sirve a provisionadores como ACME, X5C, OIDC, etc…, y step, una herramienta de línea de comandos (CLI) que interactúa con esa cadena de certificados para acciones como generar un certificado [32].

En el contexto de la Infraestructura de Clave Pública (PKI), *Smallstep* permite:

- **Encriptación:** Uso de pares de claves públicas y privadas para asegurar datos y firmar comunicaciones.
- **Identidad:** Extensión de claves con metadatos para proporcionar identidad; la clave pública se convierte en un certificado.
- **Confianza:** Establecimiento de confianza mediante la firma criptográfica de certificados por una CA raíz confiable.
- **Política:** Gestión de la confianza de los certificados con restricciones como períodos de validez y las capacidades otorgadas a los certificados firmados.

Para la implementación de este proyecto utilizaremos la versión de código abierto de *Smallstep* (*step-ca* y *step-cli*), con las cuales se pueden crear CAs raíz e intermedias que facilitan la automatización del proceso de solicitud y emisión de certificados, similar a lo que se ofrece con *Let's Encrypt*. Estos certificados pueden ser utilizados para una variedad de propósitos, incluyendo la autenticación segura de usuarios y dispositivos en una red, o, como en el caso de este estudio, el acceso SSH a servidores o *hosts*.

3.3. Introducción a *Amazon Web Services* (AWS)

Amazon Web Services (AWS) es una plataforma de servicios de computación en la nube que proporciona una amplia gama de servicios basados en la infraestructura de Amazon. Estos servicios incluyen almacenamiento, bases de datos, análisis, redes, móvil, herramientas de desarrollo, herramientas para Internet de las cosas (IoT), seguridad, y mucho más.

AWS permite a las empresas y a los desarrolladores alquilar capacidad de computación, almacenamiento y otros recursos, ofreciendo así una alternativa escalable y económica a la construcción de una infraestructura física propia.

AWS se caracteriza por su flexibilidad, escalabilidad, y por su modelo de pago por uso, que permite a los usuarios pagar solo por los recursos que realmente utilizan. Esta flexibilidad se extiende a una amplia gama de aplicaciones, desde páginas web y aplicaciones móviles hasta procesamiento de datos a gran escala y aplicaciones empresariales.

Dentro de AWS, uno de los servicios más fundamentales es *Amazon Elastic Compute Cloud* (EC2) y es el que utilizaremos para desarrollar este estudio. Amazon EC2 proporciona capacidad de computación en la nube, permitiendo a los usuarios ejecutar aplicaciones en la infraestructura de Amazon reduciendo así los costos de *hardware*.

EC2 ofrece una amplia variedad de tipos de instancias optimizadas para diferentes casos de uso, que van desde aplicaciones web y móviles hasta aplicaciones empresariales y de procesamiento de datos a gran escala donde podemos lanzar tantos servidores virtuales como necesitemos, configurar la seguridad, las redes y administrar el almacenamiento.

En el siguiente apartado hablaremos en profundidad de este servicio.

3.3.1. *Elastic Compute Cloud* (EC2)

Amazon EC2 (*Elastic Compute Cloud*) es uno de los servicios más importantes y utilizados dentro de *Amazon Web Services*. Su principal función es proporcionar capacidad de procesamiento escalable en la nube. Esto permite a los usuarios alquilar máquinas virtuales (instancias), configurarlas y controlarlas de acuerdo con sus necesidades.

A continuación, explicaremos las características de este servicio en profundidad:

- **Tipos de Instancias:** EC2 ofrece una variedad de tipos de instancias, cada una optimizada para diferentes cargas de trabajo. Esto incluye instancias optimizadas para computación, memoria, almacenamiento y más. En nuestro caso, utilizaremos una máquina t2.micro ya que se trata de un servidor virtual gratuito [33].
- **Escalabilidad y Elasticidad:** EC2 proporciona la capacidad de escalar los recursos arriba o abajo rápidamente, lo que es ideal para manejar cambios en los requisitos de carga de trabajo.
- **Seguridad y Control:** Los usuarios tienen control completo sobre sus instancias. Pueden utilizar grupos de seguridad y redes virtuales para controlar el acceso a las instancias en una red a través de los *Security Groups* y *Amazon Virtual Private Cloud* (VPC) respectivamente. En este caso, los *Security Groups* garantizarán que el acceso a la máquina solo ocurre desde la IP pública de mi *router* garantizando así que ningún otro tráfico está permitido.
- **Almacenamiento:** EC2 permite integrar diferentes tipos de almacenamiento, incluyendo EBS (*Elastic Block Store*) y almacenamiento efímero, para nuestro caso una máquina con 8gb de almacenamiento será suficiente.
- **Precios Flexibles:** EC2 ofrece varias opciones de precios, como precios bajo demanda, reservados y de *spot*, que permiten a los usuarios optimizar costos según sus necesidades. Como bien hemos mencionado en las anteriores características, el uso de nuestra instancia EC2 no tendrá ningún coste, pero una implementación incorrecta o un uso excesivo podría implicar costes en el desarrollo.

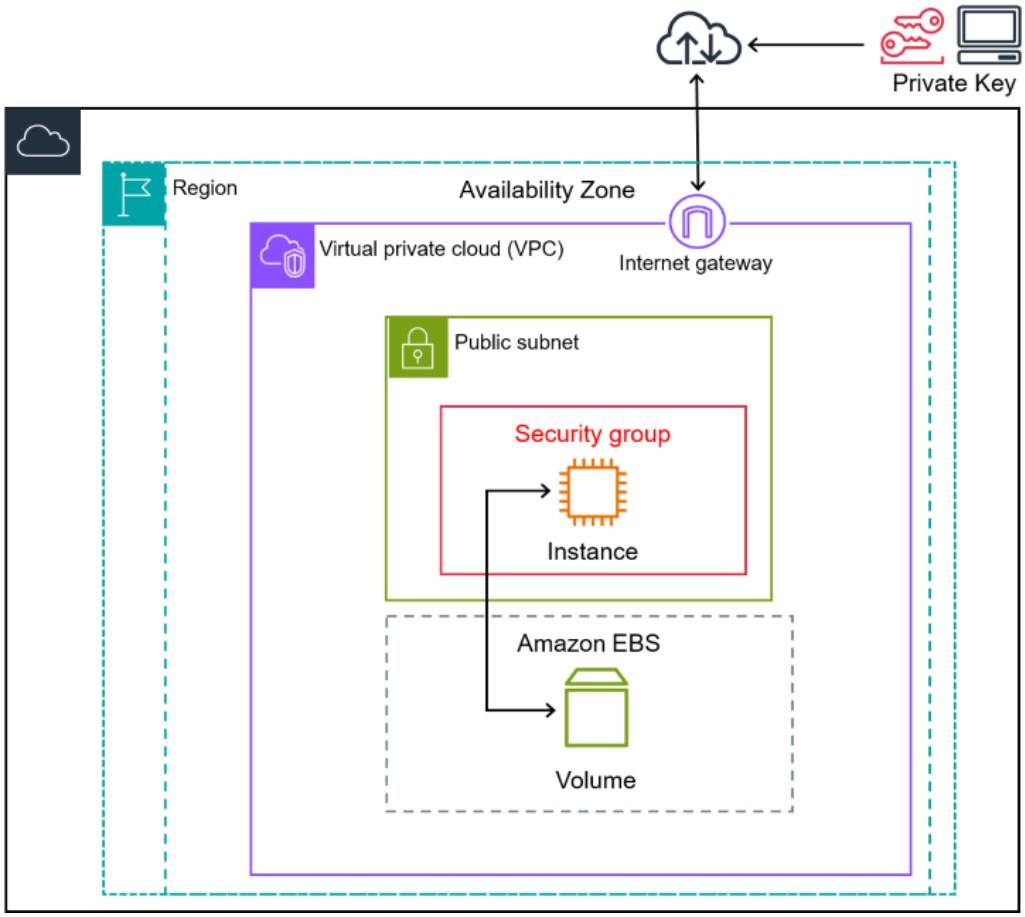


Figura 13. *Estructura básica de una instancia EC2 [34].*

3.4. Infraestructura de clave pública (PKI)

La Infraestructura de Clave Pública o en inglés *Public Key Infraestructure* (PKI) es esencial para la gestión de la seguridad a través de la encriptación. Usa un par de claves, una pública y otra privada, para encriptar y desencriptar mensajes, y es aplicable a personas, dispositivos y aplicaciones. PKI surgió en los años 90 para administrar claves de encriptación mediante la emisión y gestión de certificados digitales, que verifican al propietario de una clave privada y la autenticidad de esa relación para mantener la seguridad [35].

PKI utiliza algoritmos criptográficos para encriptar y desencriptar mensajes. Incluye la encriptación simétrica, donde se usa la misma clave para encriptar y desencriptar, y la asimétrica, que usa un par de claves distintas, una privada y una pública.

La encriptación asimétrica permite acciones como firmas digitales y es fundamental para la seguridad de PKI. La *Public Key Infrastructure* resuelve el problema de saber si una clave pública realmente pertenece a la persona que se cree mediante certificados

digitales que confirman la identidad de los propietarios de claves privadas y las claves públicas correspondientes.

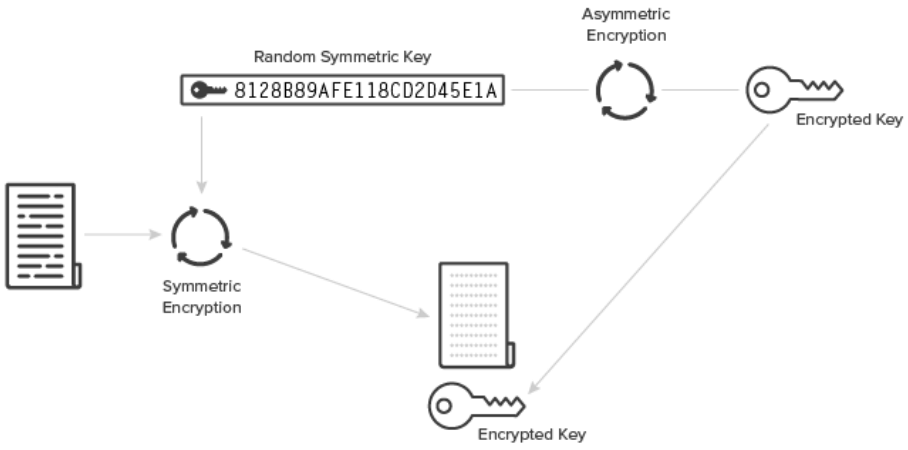


Figura 14. *Uso de claves simétricas y asimétricas [35].*

Los certificados SSH y los X.509 comparten el principio de asociar claves públicas con identidades y la verificación de dicha asociación por una entidad de confianza. Sin embargo, se diferencian en su estructura y uso específico dentro de la seguridad digital:

- **Certificados X.509:** Son ampliamente utilizados en PKI para establecer conexiones seguras, como las de HTTPS. Los certificados X.509 incluyen metadatos extensos sobre el titular, un período de validez, y están firmados por una CA, lo cual permite establecer una cadena de confianza desde la CA raíz hasta el certificado final [36].
- **Certificados SSH:** Están optimizados para la autenticación en el protocolo SSH. No tienen una cadena de confianza tan compleja y suelen ser más simples en contenido, focalizándose en la identidad del usuario o el host y la clave pública correspondiente.

Ambos tipos de certificados utilizan un par de claves asimétricas y ambos necesitan de una entidad de confianza que será nuestra Autoridad de Certificación (CA) la que respalde la relación entre la clave pública y la identidad del titular del certificado. Sin embargo, para el desarrollo de la infraestructura nos enfocaremos en los certificados SSH ya que están diseñados específicamente para el contexto de autenticación SSH y

será nuestro caso de estudio, mientras que los certificados X.509 tienen aplicaciones más amplias y formales en la seguridad de redes y en internet como, por ejemplo, TLS/SSL.

Además de los certificados digitales, existen varios servicios adicionales que son esenciales para garantizar la seguridad y eficacia del sistema. Estos servicios, que complementan y refuerzan la funcionalidad de los certificados, incluyen las Listas de Revocación de Certificados (CRL), el Sellado de Tiempo y las Autoridades de Registro (RA). Cada uno de estos servicios desempeña un rol único y crucial en la gestión de la confianza y la autenticidad dentro de una PKI.

- **Listas de Revocación de Certificados (CRL):** Son esenciales en cualquier PKI. Estas listas contienen los certificados que han sido revocados antes de su fecha de expiración. La revocación puede deberse a diversas razones, como compromiso de la clave privada o cambios en la información del sujeto. Las CRL permiten a los usuarios verificar que un certificado sigue siendo válido y confiable [37].
- **Sellado de Tiempo:** Este es un servicio proporcionado por una Autoridad de Sellado de Tiempo (TSA) dentro de una PKI. Consiste en asociar una fecha y hora confiables a un documento o dato digital. Esto es crucial para demostrar que cierta información existía en un momento específico y no ha sido alterada desde entonces [38].
- **Autoridad de Registro (RA):** Funciona como un intermediario entre el usuario y la Autoridad de Certificación (CA). La RA es responsable de verificar la identidad y otros atributos del solicitante antes de que la CA emita un certificado. Este proceso garantiza que los certificados se otorguen a las entidades correctas y refuerza la confiabilidad de toda la infraestructura [39].

3.4.1. Autoridad de Certificación (CA)

Una CA es una entidad que almacena, firma y emite certificados digitales. Un certificado digital acredita la propiedad de una clave pública por parte del sujeto nombrado en el certificado, permitiendo a otras partes (partes confiantes) confiar en las firmas o declaraciones hechas sobre la clave privada que corresponde a la clave pública certificada. Las CAs actúan como terceros de confianza, fiables tanto para el sujeto propietario del certificado como para la parte que confía en el certificado.

3.4.2. CA Intermedia

Una CA intermedia es una CA que ha sido certificada por una CA raíz. La CA intermedia puede emitir certificados a otros, y esta jerarquía ayuda a establecer una cadena de confianza. Una CA raíz firma el certificado de la CA intermedia, lo que permite a los usuarios finales confiar en los certificados emitidos por la CA intermedia debido a su relación con la CA raíz de confianza.

3.4.3. Estructura de un certificado

La estructura básica de un certificado es relativamente simple y contiene los siguientes componentes:

- **Clave Pública:** Que puede ser distribuida y compartida.
- **Nombre del Sujeto:** La entidad o persona a la que se emite el certificado.
- **Firma Digital:** Realizada por la entidad emisora para vincular la clave pública al nombre del sujeto.
- **Emisor (o Autoridad de Certificación):** La entidad que firma y emite el certificado.
- **Sujeto:** La entidad nombrada en el certificado.

Además, los certificados pueden contener metadatos adicionales, como indicaciones de si el sujeto del certificado es una CA, si su clave pública debe usarse para firmar o encriptar y además disponen de una fecha de expiración a partir de la cual el certificado deja de ser válido [40].

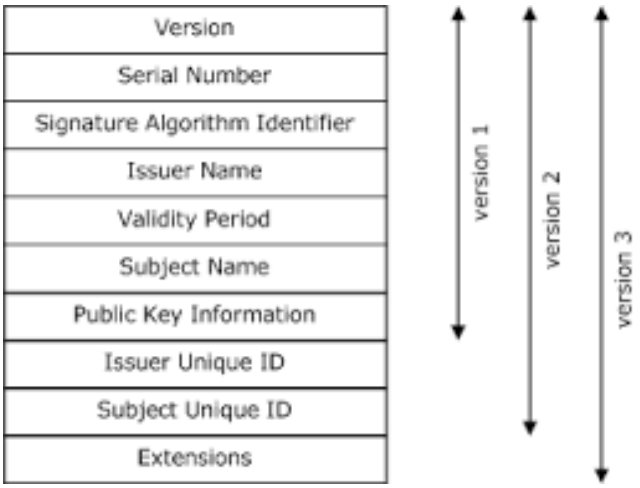


Figura 15. *Estructura de un certificado X509 [41].*

3.4.4. Relación entre SSH y Certificados

La interacción entre el protocolo SSH y los certificados digitales, en el marco de la Infraestructura de Clave Pública (PKI), se centra en cómo se pueden integrar estos elementos para mejorar la seguridad en las comunicaciones en red. Mientras que SSH emplea sus propios métodos para la autenticación de usuarios y la seguridad de la conexión, la inclusión de servicios de PKI como Listas de Revocación de Certificados (CRL), Sellado de Tiempo, y Autoridades de Registro (RA) podría proporcionar capas adicionales de seguridad y gestión.

1. **Listas de Revocación de Certificados (CRL) en SSH:** En un entorno SSH, las CRL podrían utilizarse para garantizar que las claves SSH revocadas no se utilicen para acceder a los sistemas. La integración de CRL mejoraría la gestión de las claves SSH, permitiendo una respuesta rápida en caso de compromiso de seguridad.
2. **Sellado de Tiempo en SSH:** El servicio de Sellado de Tiempo podría ser aplicado a las sesiones SSH para proporcionar una marca temporal verificable de las acciones realizadas. Esto es especialmente útil para auditorías y para garantizar la integridad de los registros de acceso y transacciones realizadas a través de SSH.
3. **Autoridad de Registro (RA) y su Relevancia en SSH:** Integrar una RA en el proceso de autenticación de SSH permitiría una verificación más exhaustiva de las identidades antes de la emisión de claves SSH. Esto fortalecería la seguridad al asegurar que las claves se otorguen solo a entidades debidamente autorizadas y verificadas.

Aunque los certificados SSH son distintos de los certificados X.509 utilizados en PKI, la relación entre ellos se centra en su propósito común de asociar claves públicas con identidades.

Los certificados SSH, están optimizados para la autenticación en el protocolo SSH, podrían beneficiarse de la integración con los servicios de PKI para mejorar la gestión de identidades y la seguridad de la autenticación.

A pesar de sus diferencias estructurales, tanto los certificados SSH como los X.509 podrían ser parte de un sistema más amplio de gestión de seguridad, donde las

funcionalidades de PKI complementen y refuercen las capacidades de autenticación y seguridad de SSH.

La integración de estos servicios de PKI en el contexto de SSH representa un enfoque avanzado para la seguridad de las comunicaciones en red, donde la eficiencia de SSH en la conexión segura se combina con la robustez y la gestión centralizada de identidades y claves que ofrece PKI.

3.5. Flujo de autenticación entre *Okta* y *Smallstep*

En el contexto de este trabajo, la implementación de OIDC para la autenticación y la generación de certificados de vida corta para autenticarnos contra un *host* a través del protocolo SSH se llevaría a cabo de la siguiente manera:

1. *Okta* actúa como el Proveedor de *OpenID*, autenticando a los usuarios y proporcionando tokens de identidad y acceso.
2. *Smallstep* se integra con *Okta* para recibir estos tokens y, en base a ellos, emitir certificados SSH para la autenticación.
3. Cuando un usuario intenta acceder a un host vía SSH, *Smallstep* valida el token de identidad de *Okta* para asegurarse de que el usuario está autenticado y tiene el acceso permitido.
4. Una vez verificada la autenticación, *Smallstep* genera un certificado SSH de vida corta que el usuario puede utilizar para iniciar sesión en el host de manera segura y sin necesidad de contraseñas tradicionales.
5. Este proceso mejora la seguridad al minimizar el tiempo durante el cual un certificado es válido, reduciendo así la ventana de oportunidad para ataques.

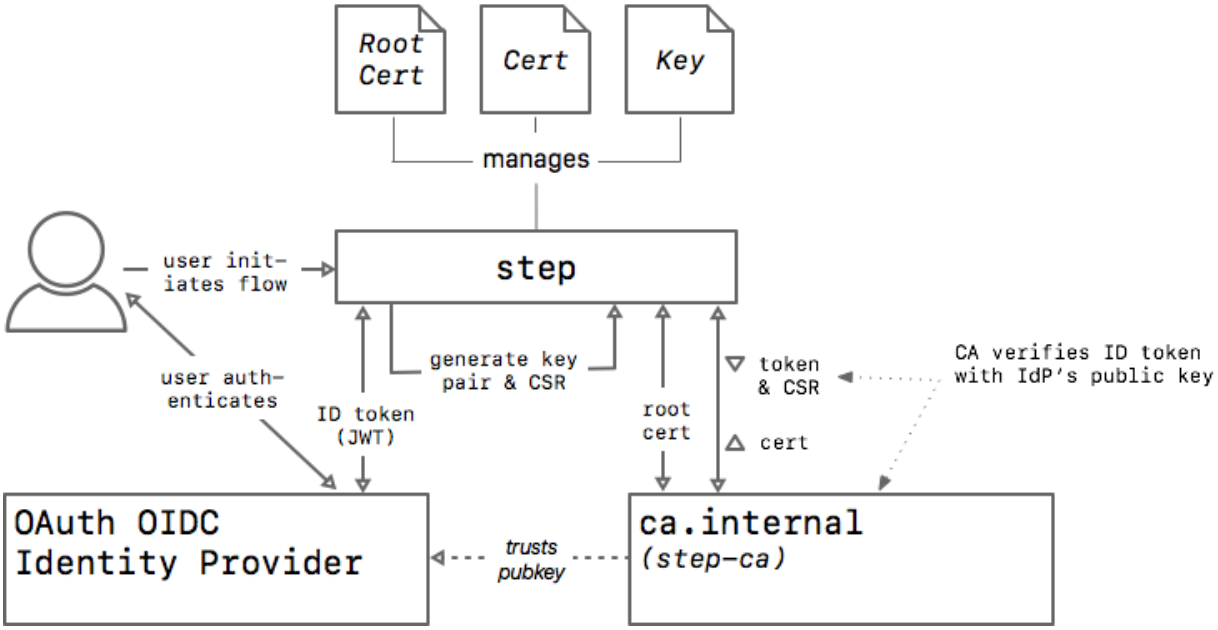


Figura 16. *Diagrama de autenticación con Smallstep y SSO [42].*

Okta y OIDC proporcionan una base segura y estandarizada para la autenticación de usuarios, mientras que *Smallstep* aprovecha esta autenticación para simplificar y asegurar el acceso SSH, creando un proceso de autenticación eficiente y moderno. Este sistema no solo fortalece la seguridad al reemplazar las contraseñas estáticas con certificados de vida corta, sino que también simplifica la gestión de accesos y mejora la eficiencia operativa.

4. Desarrollo de la Infraestructura

En este apartado nos centraremos en el desarrollo, diseño e implementación de una infraestructura robusta, segura y escalable, aprovechando tecnologías de vanguardia y prácticas modernas de ingeniería de software.

A lo largo de esta sección, se detallarán los pasos y consideraciones tomadas en cada etapa del desarrollo de la infraestructura, desde la elección de la tecnología y la planificación inicial hasta la implementación y configuración de los servicios.

Para ilustrar de manera efectiva la arquitectura y los componentes clave del sistema que hemos desarrollado, se incluye a continuación un diagrama detallado de la infraestructura. Este diagrama no solo proporciona una visión general visual de cómo se estructura y organiza la infraestructura, sino que también destaca las interconexiones y flujos de datos entre las diferentes tecnologías utilizadas: *Smallstep*, *Okta* y *AWS EC2*.

El diagrama ha sido diseñado para ser claro y comprensible, asegurando que los lectores puedan apreciar fácilmente la disposición y el papel de cada componente dentro del sistema. Asimismo, este visual sirve como una herramienta esencial para explicar los procesos subyacentes y las decisiones de diseño tomadas durante el desarrollo de la infraestructura.

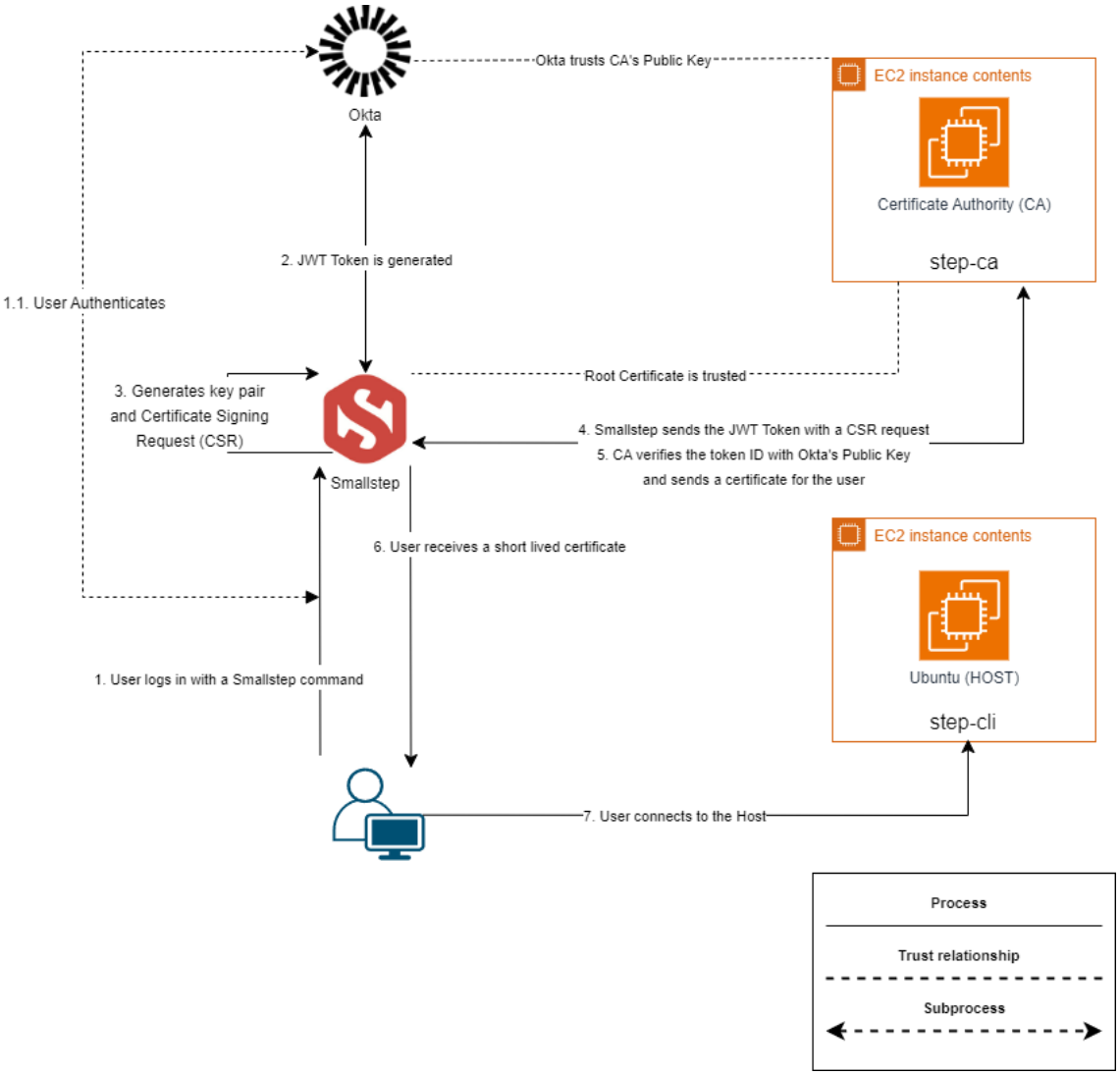


Figura 17. *Diagrama de funcionamiento de la infraestructura.*

Este diagrama será la base sobre la cual discutiremos los aspectos más detallados de la implementación en las secciones subsiguientes, proporcionando un marco de referencia visual que complementa y enriquece la explicación textual.

4.1. Entornos EC2

Para poder empezar el desarrollo práctico, primero debemos levantar el entorno virtual en el que tendremos localizadas nuestra autoridad de certificación y el dispositivo al que queremos conectarnos a través del protocolo SSH (*host*).

Como hemos mencionado brevemente en la introducción técnica, concretamente en el apartado 3.4.1. *Elastic Compute Cloud* (EC2) nuestras instancias EC2 serán de *tier* gratuito, utilizaremos instancias de tipo t2.micro con una CPU y 1gb de memoria,

además de esto dispondrán de 8gb de almacenamiento. La máquina escogida a simular será un *Ubuntu Server 22.04 LTS*.

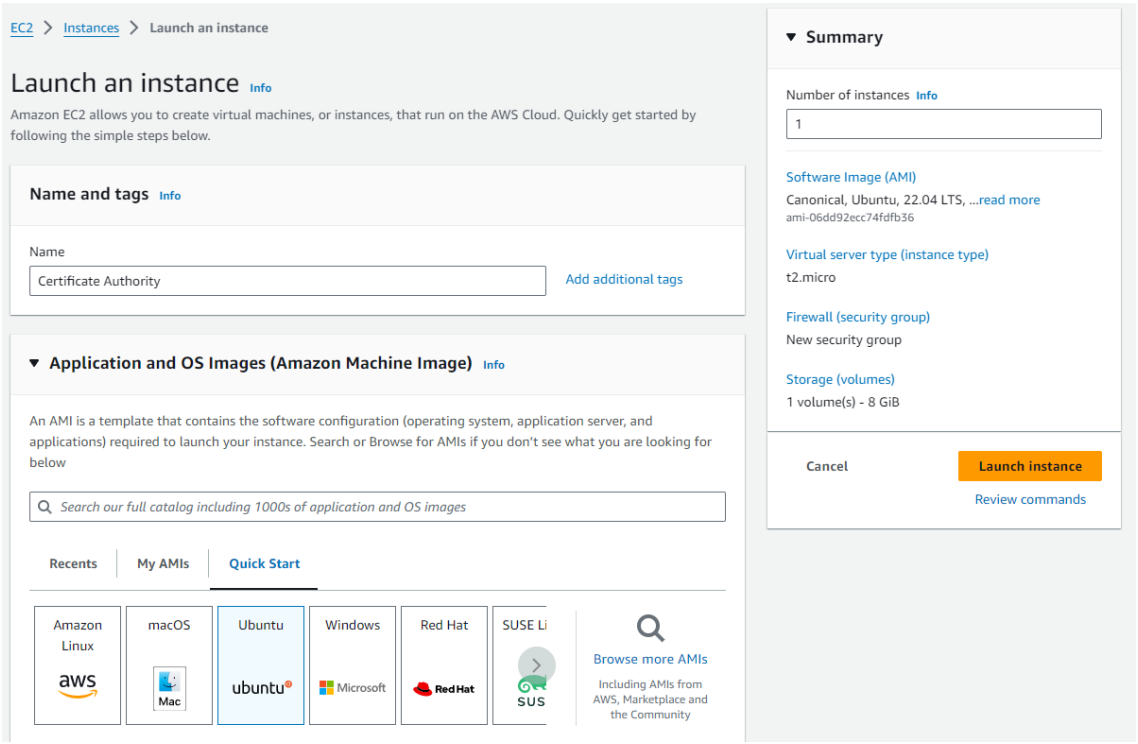


Figura 18. *Lanzamiento de una instancia EC2.*

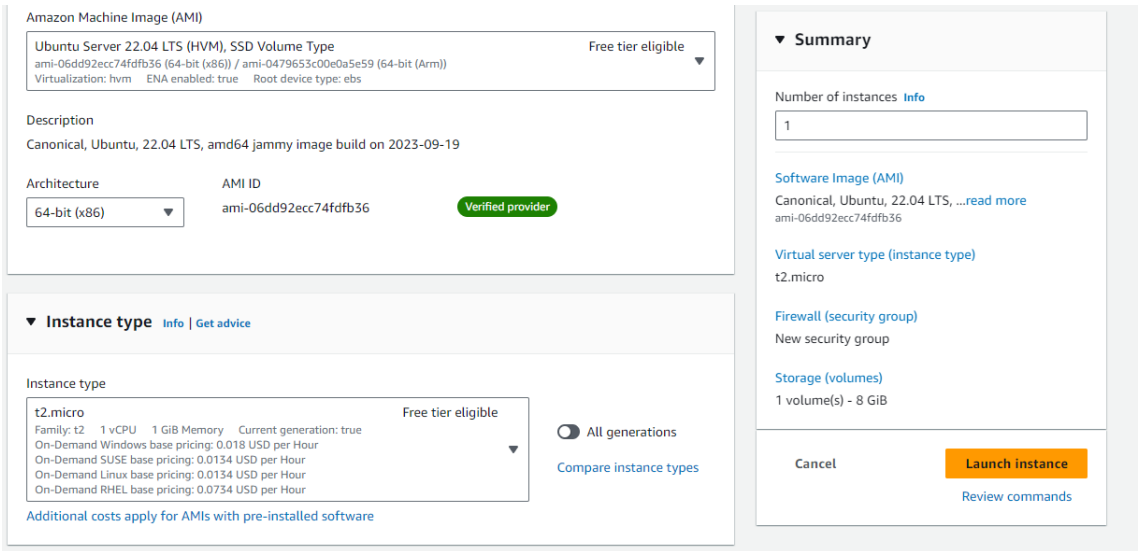


Figura 19. *Tipo de máquina escogida*

A continuación podemos ver que AWS nos obliga a escoger un par de llaves para poder hacer login a través de SSH, es decir, se basa en una autenticación por claves, pero no utilizaremos este método.

Seguidamente, escogemos una VPC con acceso a Internet, se trata de unas VPC preconfiguradas en mi entorno de trabajo, simplemente nos garantizan acceso a Internet. Además de eso, escogemos asignarle una IP pública a la máquina para poder conectarnos a ella.

Para finalizar, definimos un *Security Group*, en él, configuramos el acceso desde cualquier punto de Internet, ya que las conexiones vendrán desde el lugar donde un usuario quiera hacer SSH a la Autoridad de Certificación, desde los *hosts* que deseen solicitar un *root certificate* y finalmente la petición desde cualquier IP pública de *Okta*, que será quien realice la relación de confianza con nuestra entidad certificadora para poder gestionar las peticiones de firma de certificados siempre que vengan desde nuestra instancia de *Okta*.

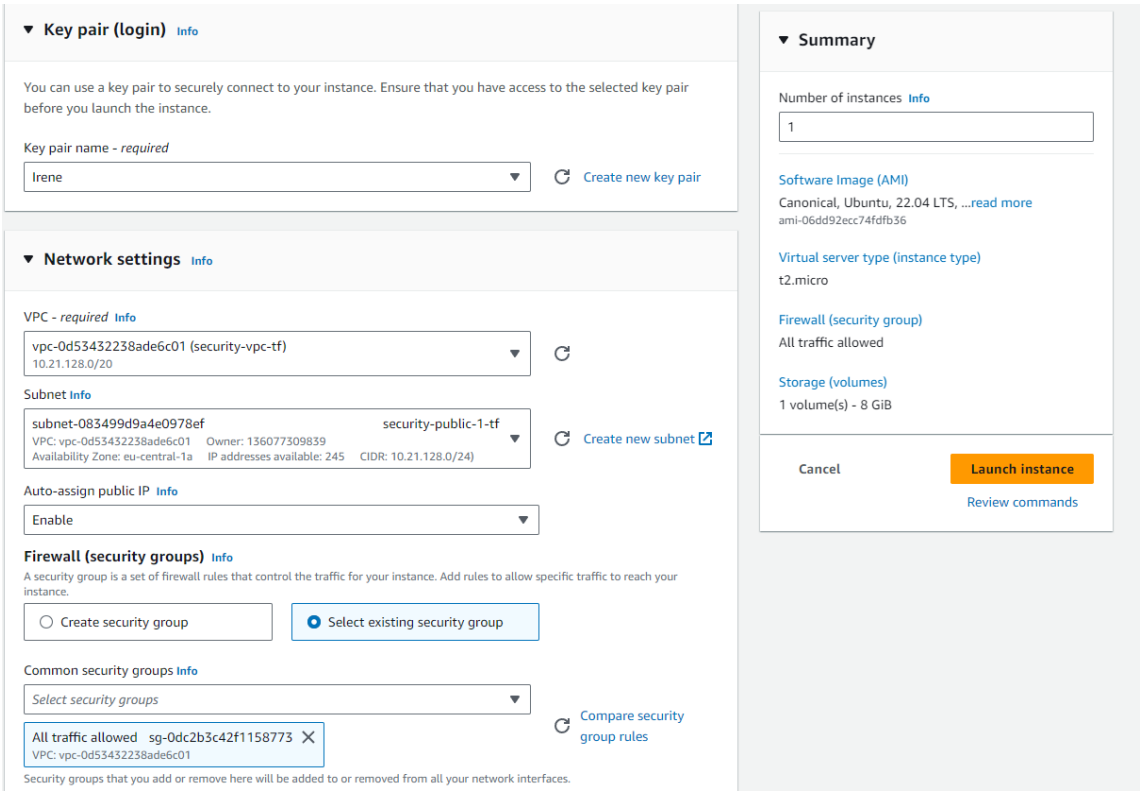


Figura 20. *Configuración de red y seguridad de la instancia.*

Finalmente, para no asumir costes extra el espacio que necesitamos de almacenamiento lo dejaremos en 8gb ya que nos será más que suficiente para la práctica.

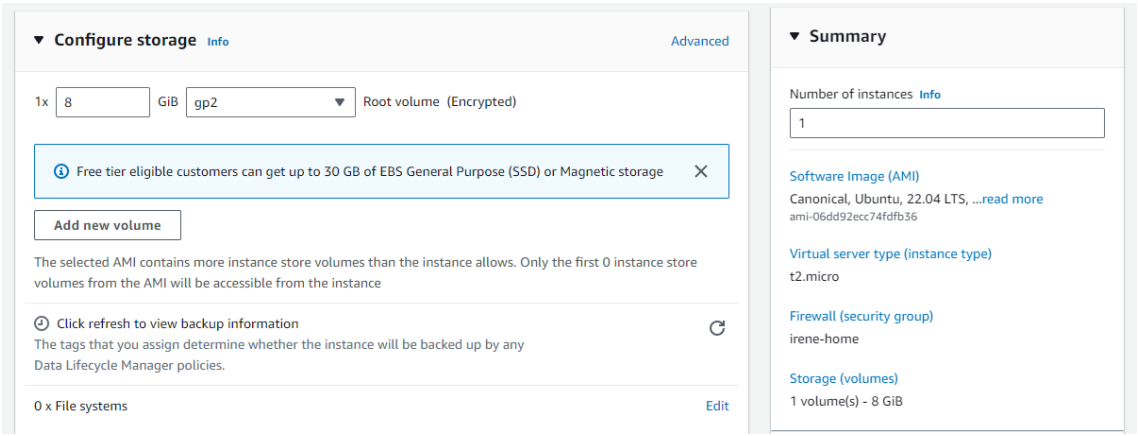


Figura 21. *Selección del almacenamiento de la instancia.*

En el caso del *host*, realizaremos el mismo proceso. El *host* a excepción de la entidad certificadora, no necesitará recibir conexiones de nuestro *IdP*, simplemente las recibirá de la CA y el cliente que desee conectarse a ella.

4.1.1. Configuración de la CA

Para facilitar el despliegue de la Autoridad de Certificación, hemos realizado un *script* con el que configuraremos todos los *provisioners*, es decir, todas aquellas aplicaciones que deban ser capaces de emitir certificados firmados por nuestra CA.

Se puede apreciar el código en el Anexo 8.2 con el que desplegaremos nuestra entidad certificadora.

En el apartado 4.3. Configuración en *Okta,* podemos ver los detalles de configuración de la integración entre la herramienta *open-source Smallstep* y *Okta*.

Nos conectaremos a la instancia a través de la funcionalidad EC2 *Instance Connect*:

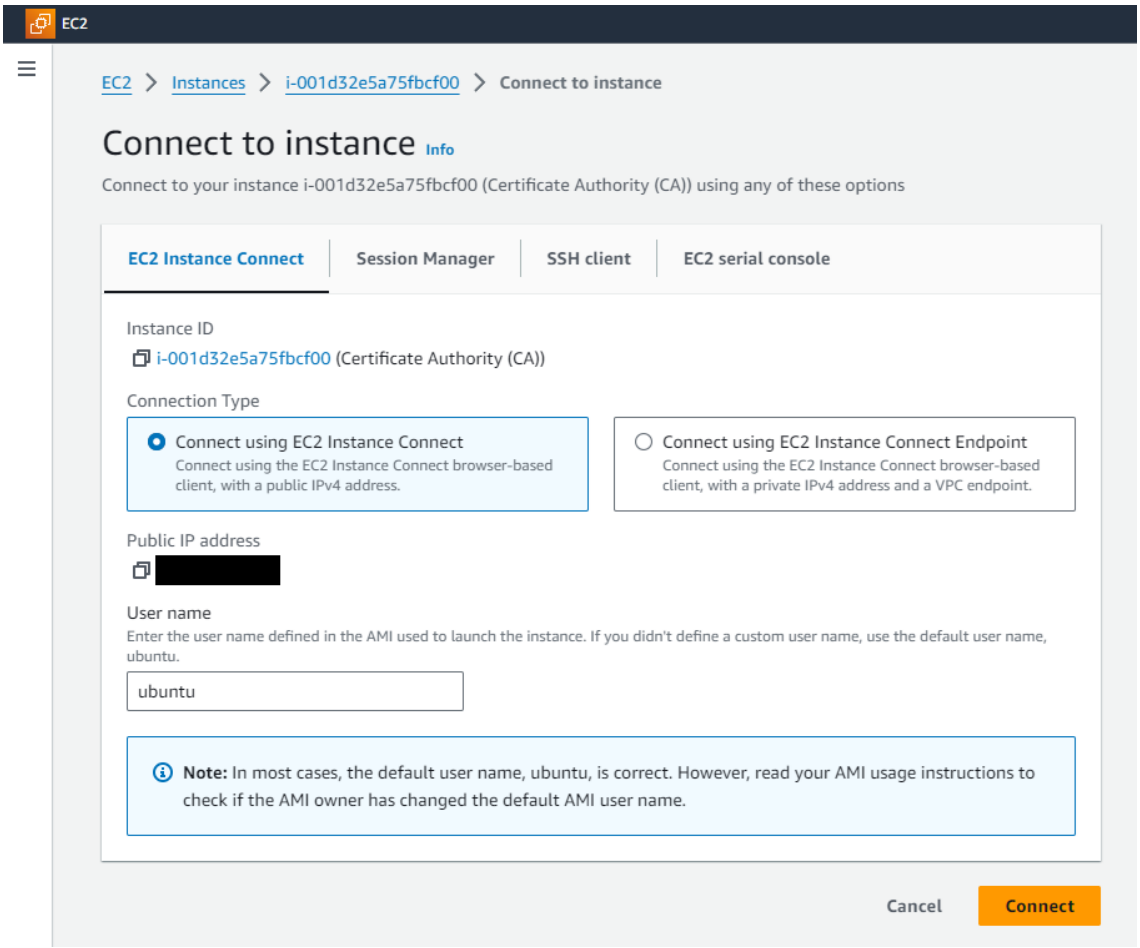


Figura 22. *Conexión a la Autoridad de Certificación.*

Una vez dentro, crearemos el *script* mencionado iniciando sesión como el usuario *root*, el usuario con más privilegios del sistema, esto es debido a que, como se comenta en el código del *script*, es necesario que el servicio que gestiona la CA siga activo siempre.

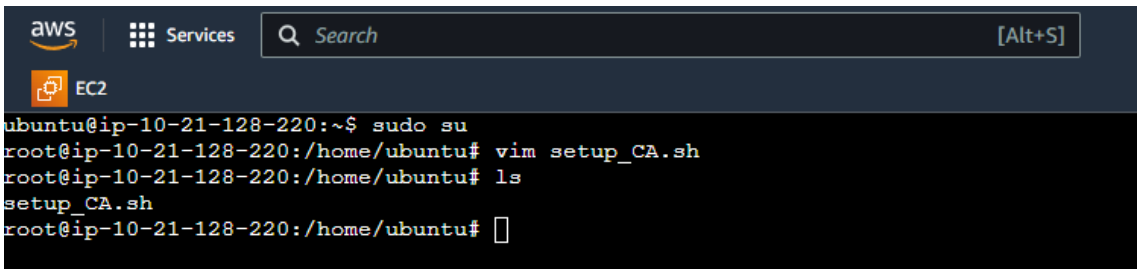


Figura 23. *Creación del fichero con el script.*

Seguidamente, lo convertiremos en un archivo ejecutable y lanzaremos su ejecución:

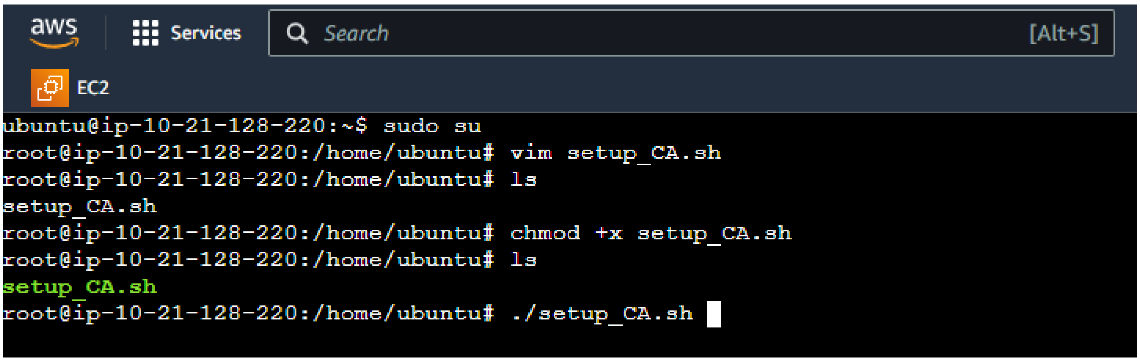


Figura 24. *Ejecución del script.*

Aquí vemos como la ejecución fue exitosa, debemos almacenar de forma segura el *Root Fingerprint*, lo utilizaremos en los pasos siguientes:

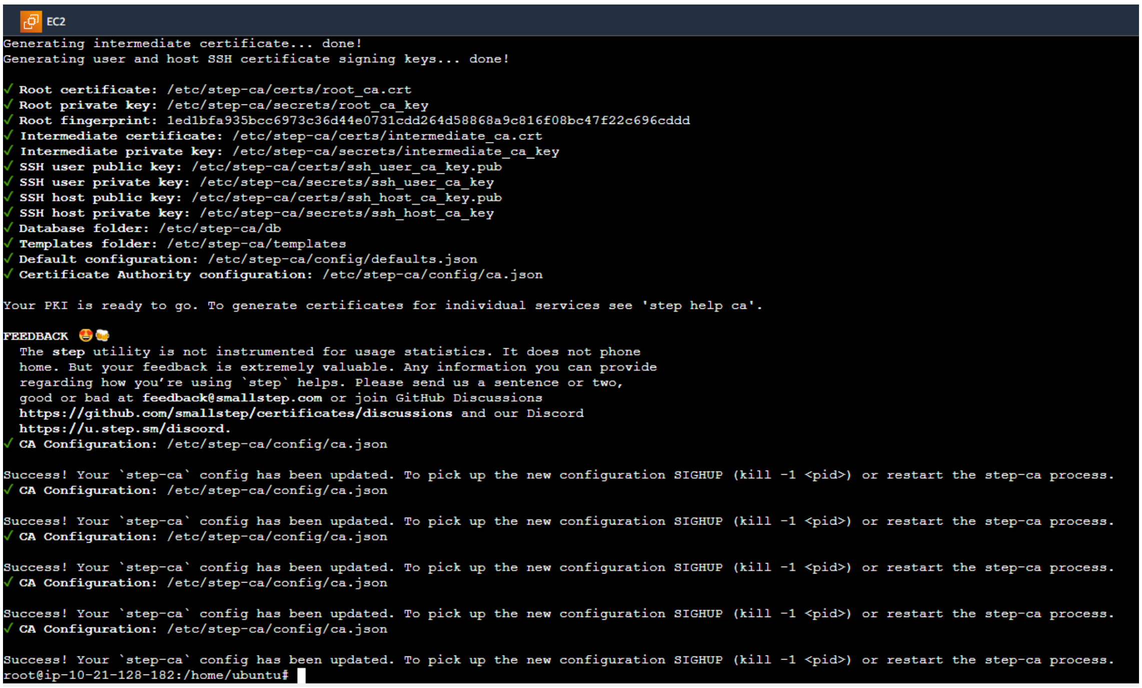


Figura 25. *Resultado de la ejecución exitoso.*

4.1.2. Configuración del *host*

Como ya hemos mencionado, el *host* se despliega de la misma forma que la entidad certificadora, por lo que no incluiremos la parte del despliegue en una instancia EC2.

Para el despliegue de los *hosts* hemos realizado otro *script* que se encargará, dado un *Root Fingerprint* de vincularla con la autoridad de certificación creada en el paso anterior.

En el Anexo 8.3 podemos apreciar el código de configuración que utilizaremos para el *host*.

Una vez ejecutado este *script* dentro de nuestro *host*, debemos firmar el certificado del *host* utilizando alguno de los provisioners aceptados, en este caso, firmaremos con el JWT Token de la cuenta de administrador configurada por defecto y de forma obligatoria en la CA:

```
root@ip-10-21-128-150:~/.step/certs# cd
root@ip-10-21-128-150:~# vim setup_host.sh
root@ip-10-21-128-150:~# chmod +x setup_host.sh
root@ip-10-21-128-150:~# ./setup_host.sh
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0
100 11.9M  100 11.9M    0     0  18.4M      0 --:--:-- --:--:-- --:--:--  124M
(Reading database ... 64729 files and directories currently installed.)
Preparing to unpack step-cli_0.24.4_amd64.deb ...
Unpacking step-cli (0.24.4) over (0.24.4) ...
Setting up step-cli (0.24.4) ...
✓ Would you like to overwrite /root/.step/certs/root_ca.crt [y/n]: y
The root certificate has been saved in /root/.step/certs/root_ca.crt.
✓ Would you like to overwrite /root/.step/config/defaults.json [y/n]: y
The authority configuration has been saved in /root/.step/config/defaults.json.
Certificate /root/.step/certs/root_ca.crt has been installed.
X.509v3 Root CA Certificate (ECDSA P-256) [Serial: 8958...5210]
  Subject:     Certificate Authority Root CA
  Issuer:      Certificate Authority Root CA
  Valid from:  2023-12-03T20:16:25Z
          to:  2033-11-30T20:16:25Z
Use the arrow keys to navigate: ↓ ↑ → ←
What provisioner key do you want to use?
  ▸ irene.balaguer@[redacted] (JWK) [kid: fny6Nz89mZu08-Z-Su3h0K7BQNXSNIzXMFkfozENH9I]
    sshpop (SSHPOP)
    Okta (OIDC) [client: 0oa7253hg6prVEJoN697]
    Amazon Web Services (AWS)
↓   SSHPOP (SSHPOP)
```

Figura 26. *Firma del certificado del host utilizando el usuario administrador.*

Finalmente, el resultado es el siguiente:

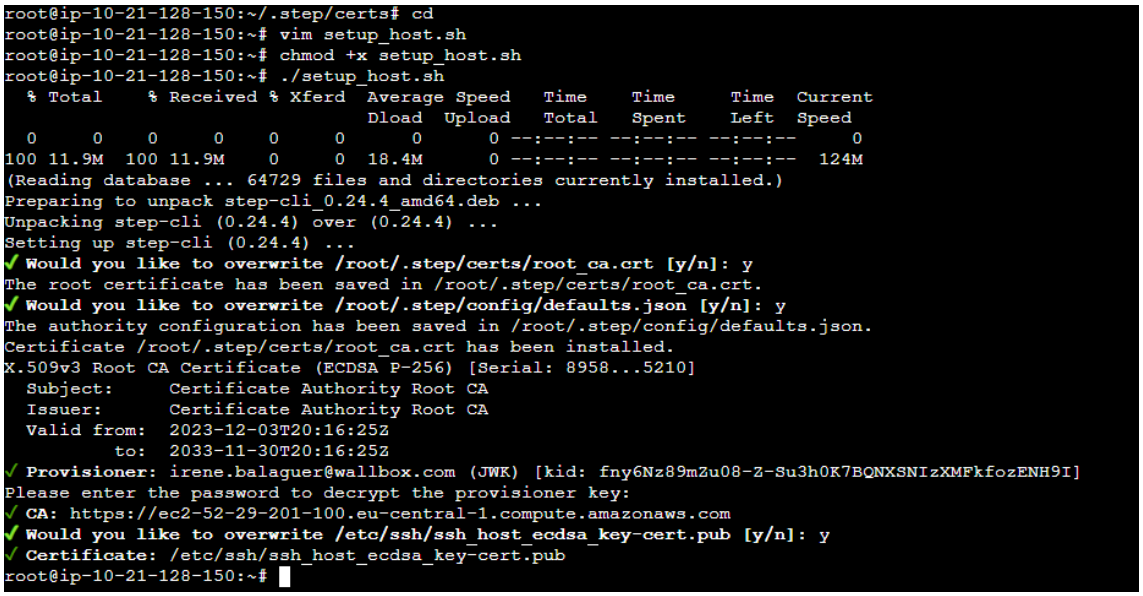


Figura 27. *Resultado de la ejecución del script en el host exitosa.*

4.2. Configuración en *Okta*

Para poder configurar la aplicación en *Okta* seguiremos la documentación proporcionada por *Smallstep* [43].

Esta configuración será la que permitirá a la Autoridad de Certificación (CA) que hemos creado en AWS verificar que un usuario es quien dice ser y generar el certificado firmado por ella a nombre del usuario.

Como hemos comentado en el apartado de la descripción técnica y conocimiento necesario para realizar la práctica, esta integración se basa en OIDC *OpenID Connect*, es decir, se basa en una autorización por *tokens OAuth 2.0* a través de *API endpoints*.

Estos son los parámetros que necesitamos introducir, podemos encontrarlos también en el *script* de configuración de la CA:

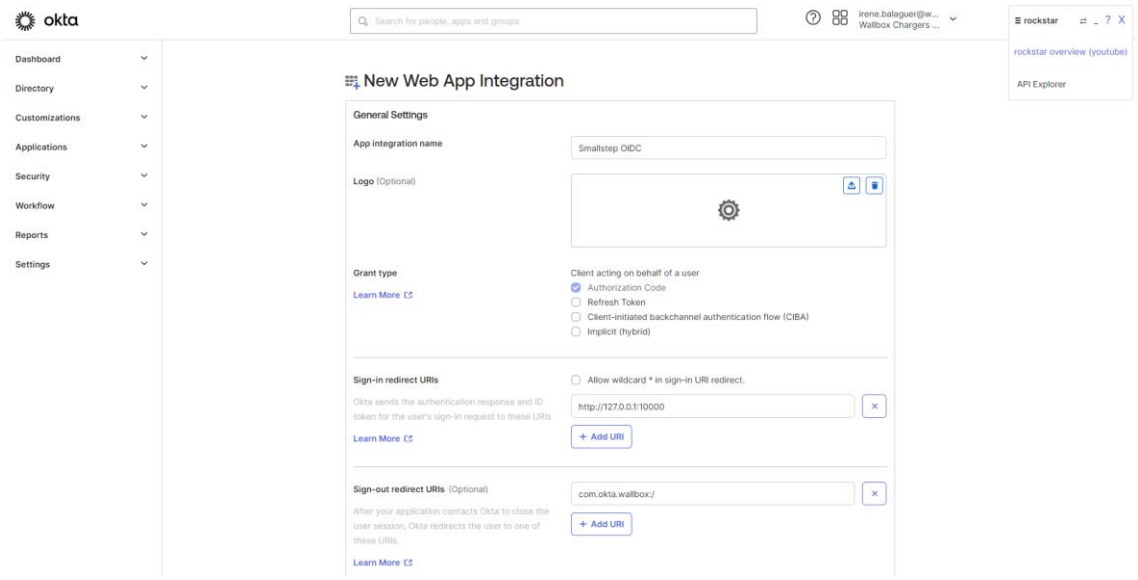


Figura 28. *Creación de la aplicación OIDC en Okta.*

La creación de esta aplicación nos genera un *Client ID* que es público y un *Client Secret*, la forma con la que la CA creará una relación de confianza con *Okta* y viceversa.

5. Pruebas Finales

En la fase final de este proyecto, se realizará una serie de pruebas críticas para evaluar la eficacia y seguridad del protocolo SSH (*Secure Shell*) en un entorno de red real. Estas pruebas implican una conexión remota desde un PC con sistema operativo Windows a un host remoto. Este host está vinculado a una Autoridad de Certificación (CA), lo cual es esencial para garantizar una comunicación segura y autenticada.

De acuerdo con los objetivos definidos en la investigación, en este apartado trataremos de demostrar la implementación efectiva y la funcionalidad del protocolo SSH en un escenario práctico. Esto incluye verificar la autenticidad de la conexión a través de la CA, asegurar la integridad de los datos transmitidos y evaluar la facilidad de uso del sistema desde una perspectiva del usuario final.

Dado que la preparación del entorno se ha realizado en el apartado 4 junto con el Desarrollo de la Infraestructura, en estas pruebas se observará como con un ordenador *Windows* estableceremos una conexión SSH contra un *host Ubuntu* verificando nuestra identidad con *Okta* e inspeccionando el certificado emitido por la Entidad Certificadora a nuestro nombre.

Para poder conectarnos al *host* desde nuestro entorno principal primero deberemos vincular dicho entorno a la entidad certificadora creada:

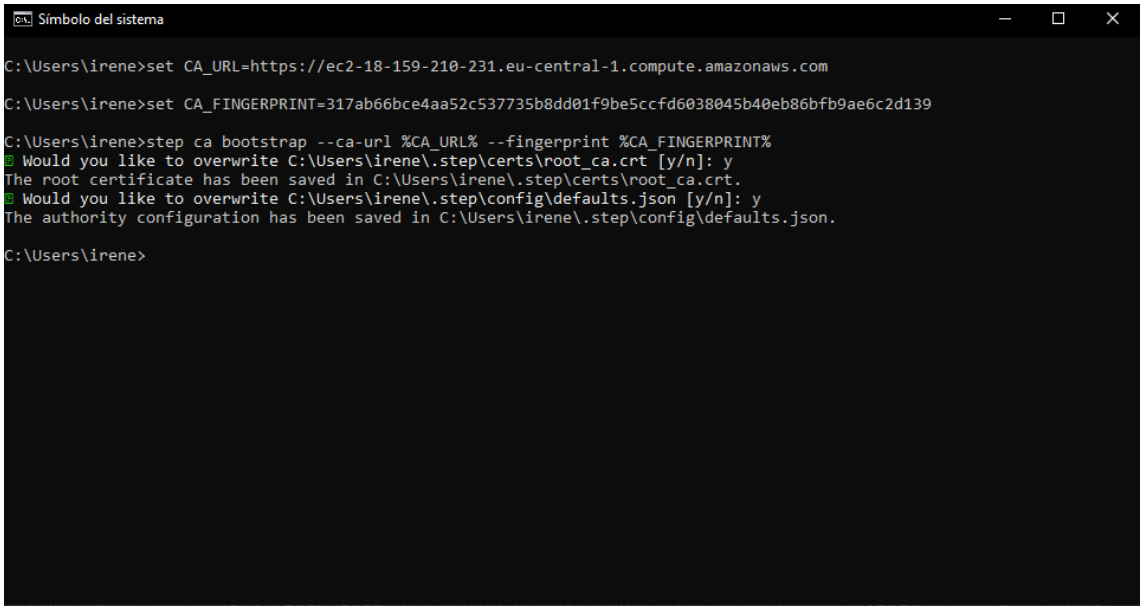


Figura 29. *Vinculación del entorno principal a la Entidad Certificadora (CA).*

Seguidamente, deberemos iniciar sesión con nuestro usuario de *Okta*:

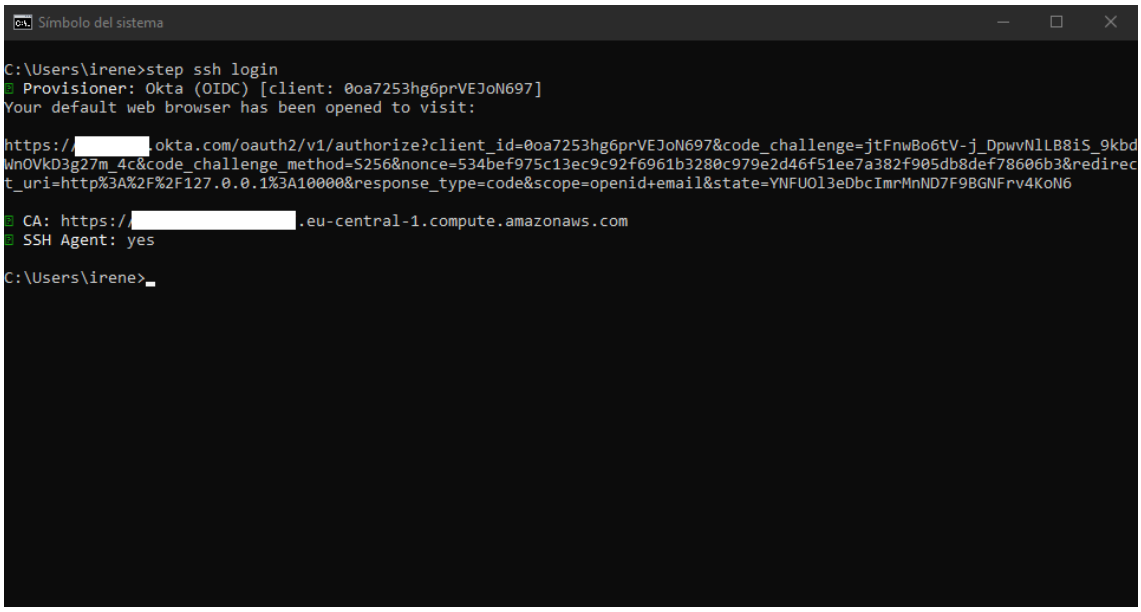


Figura 30. *Login con el servicio step.*

Se nos abrirá en el navegador una pestaña pidiendonos acceso a nuestra cuenta de *Okta:*

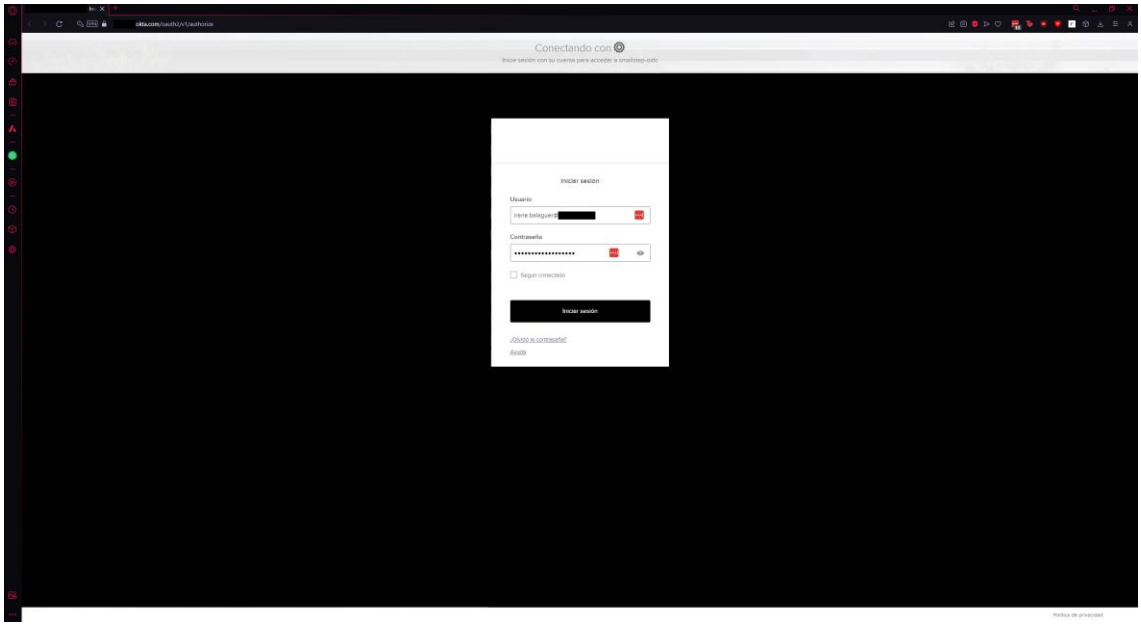


Figura 31. *Inicio de sesión en Okta.*

Una vez pongamos nuestras credenciales, el acceso será exitoso:

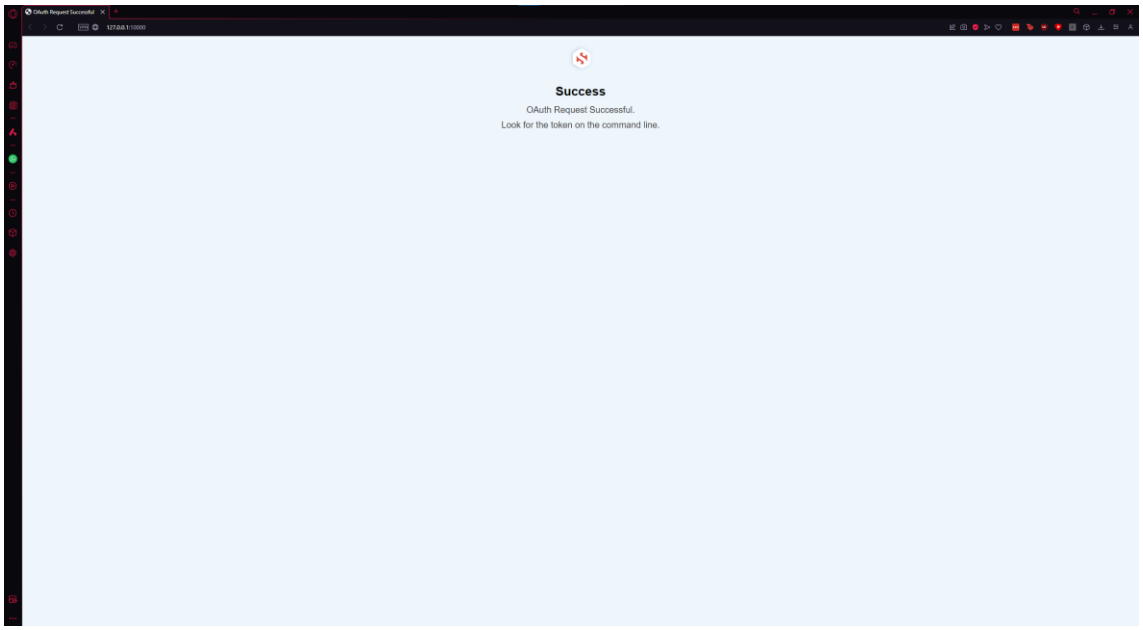


Figura 32. *Inicio de sesión exitoso.*

Una vez tengamos nuestra sesión iniciada, podemos inspeccionar el certificado emitido por la Entidad Certificadora usando nuestra identidad de *Okta:*

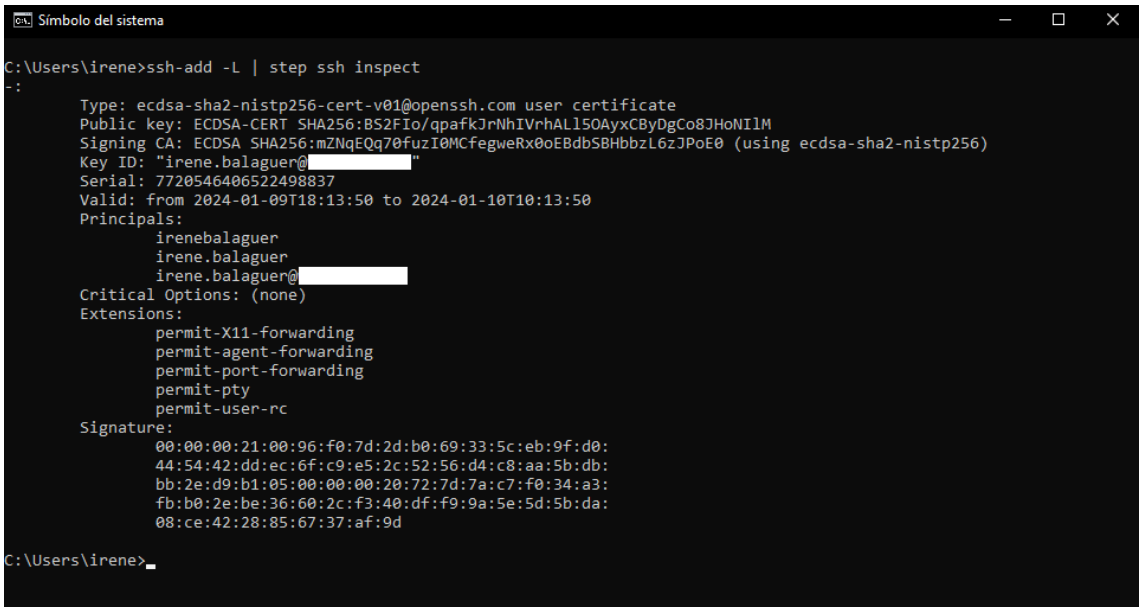


Figura 33. *Certificado emitido para la identidad verificada con Okta.*

Como observamos en la figura este es el certificado generado para nuestro usuario y en la que podemos ver las partes generales de un certificado descritas en el apartado 3.4.3. La parte que nos interesa es la que describe a los *Principals*, es decir, los usuarios autenticados para iniciar sesión en la máquina *Host*, en este caso: irenebalaguer, irene.balaguer e irene.balaguer@[REDACTED].com.

A continuación, con el comando *step ssh hosts* observamos los dispositivos disponibles para conectarnos:

Figura 34. *Hosts disponibles.*

El último paso de esta prueba es realizar la conexión y probar que funciona correctamente, esto se realiza con el comando *step ssh [REDACTED].hopto.org* con el que nos conectaremos a la instancia *host* definida en AWS:

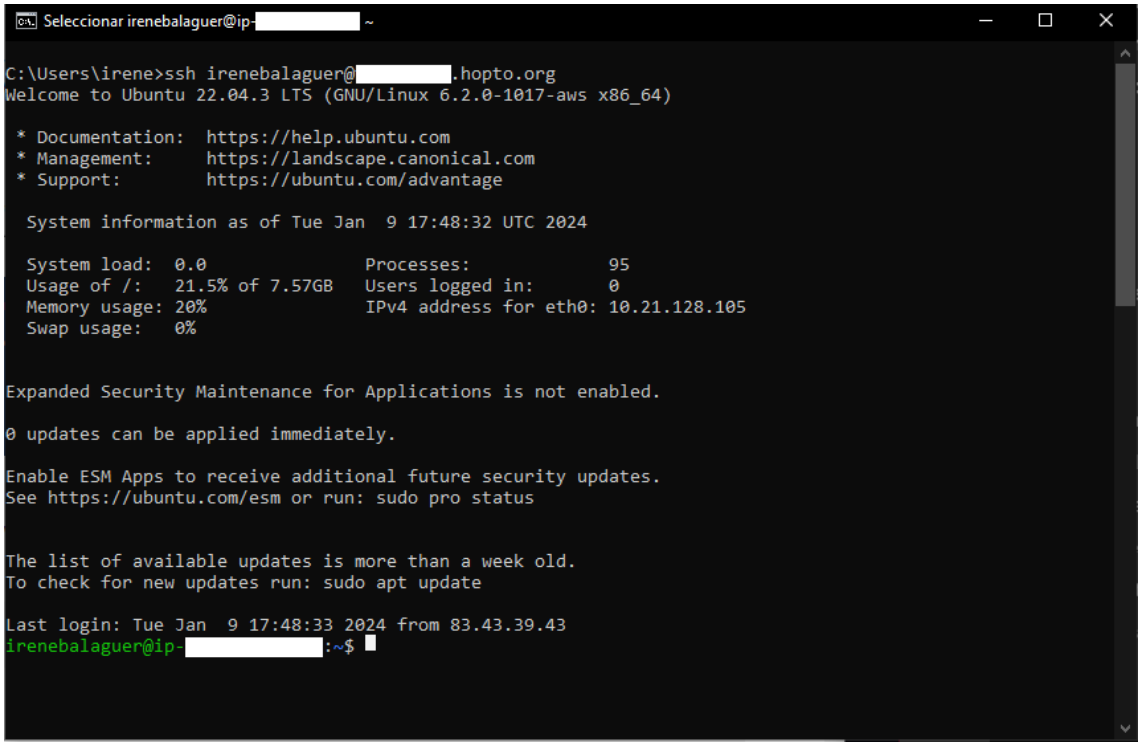


Figura 35. *Inicio de sesión a través de SSH a la instancia Host.*

Como prueba extra, mostramos como se puede ver la conexión dentro del *Host* para poder auditar quien accede:

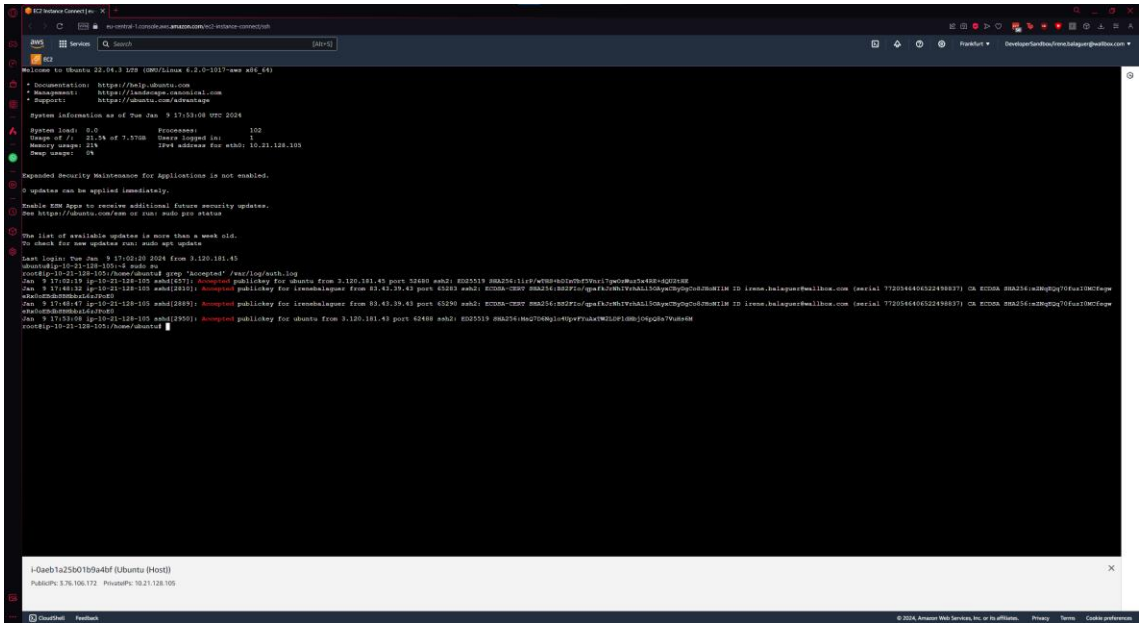


Figura 36. *Inicios de sesión del usuario irenebalaguer.*

6. Conclusiones y trabajos futuros

En el transcurso de este trabajo, se ha logrado establecer un sistema de autenticación más seguro y trazable, destacando la importancia de adoptar medidas de seguridad robustas y la efectividad de la combinación de *Okta* y *Smallstep* para la protección de accesos remotos. Este logro resalta no solo la realización de los objetivos planteados inicialmente, como se detalla en la sección 1.3, sino también la importancia de una metodología y planificación cuidadosas, reflejada en el enfoque y método seguido y en la planificación detallada con un diagrama de Gantt.

Un aspecto crucial del sistema es su escalabilidad, robustez y seguridad. La solución diseñada es escalable, permitiendo su expansión para acomodar un número creciente de usuarios y escenarios de uso, aunque requiere de trabajo manual como por ejemplo la creación previa de usuarios en la instancia *host*.

Su robustez se evidencia en la capacidad de manejar eficazmente las diversas amenazas y desafíos de seguridad, adaptándose a las necesidades cambiantes de un entorno de seguridad en constante evolución ya que la única forma de acceder a estos *hosts* es a través de un certificado emitido por la entidad certificadora, la cual confía únicamente en los dominios especificados en la configuración.

La seguridad del sistema se ha reforzado mediante la implementación de soluciones líderes en el mercado, evaluaciones periódicas y la adopción de las mejores prácticas del sector, como se menciona en la identificación y mitigación de los riesgos asociados con la implementación del sistema.

En términos de impacto, el proyecto consideró cuidadosamente los aspectos ético-sociales, de sostenibilidad y de diversidad, alineándose con los principios de responsabilidad social y comportamiento ético. La mitigación proactiva de riesgos y la planificación detallada han sido fundamentales para asegurar que el sistema no solo sea técnico sino también operacionalmente robusto.

Mirando hacia el futuro, hay varias áreas que ofrecen oportunidades para la investigación y el desarrollo continuos.

Podemos destacar las siguientes investigaciones futuras:

- Creación automática de usuarios en los *hosts* basándonos en la procedencia de la solicitud del inicio de sesión
- La exploración de nuevas integraciones y tecnologías, por ejemplo, servidores ACME para garantizar la creación de certificados SSH de entornos no procedentes del *Cloud*.
- La expansión del alcance del sistema actual donde en entornos grandes deberíamos tener un sistema de respaldo que garantice el acceso en caso de la caída de la entidad certificadora principal
- Investigación en criptografía post-cuántica y como mejoraría la tecnología actual
- Desarrollo de estrategias de formación y adopción más efectivas para facilitar la transición hacia sistemas más seguros.

En conjunto, el proyecto ha demostrado ser un esfuerzo exitoso en mejorar la seguridad de la autenticación y autorización, marcando el camino para futuras innovaciones y mejoras en el campo de la seguridad informática.

7. Glosario

PKI (*Public Key Infrastructure*): Sistema de roles, políticas y procedimientos necesarios para crear, administrar, distribuir, usar, almacenar y revocar certificados digitales y gestionar criptografía de clave pública.

SSH (*Secure Shell*): Protocolo de red que permite la comunicación segura entre sistemas a través de una red no segura, utilizando técnicas de criptografía fuerte.

IdP (Proveedor de Identidad): Servicio que autentica la identidad de usuarios y dispositivos en el ámbito digital, generalmente en sistemas de autenticación y autorización.

SAML (*Security Assertion Markup Language*): Estándar de intercambio de datos de autorización y autenticación, que permite a los proveedores de servicios y de identidad compartir información de manera segura.

OIDC (*OpenID Connect*): Capa de identidad simple basada en el protocolo OAuth 2.0, utilizada para la autenticación en aplicaciones modernas.

CA (*Certificate Authority*): Entidad responsable de emitir y gestionar certificados de seguridad digitales, parte esencial de la PKI.

MAC (*Message Authentication Code*): Pequeño fragmento de información utilizado para autenticar un mensaje y proporcionar integridad y autenticación.

HMAC (*Hashed Message Authentication Code*): Tipo específico de MAC que involucra una función hash criptográfica y una clave secreta.

TCP (*Transmission Control Protocol*): Protocolo de red fundamental que permite la transmisión confiable de datos entre sistemas informáticos.

RFC (*Request for Comments*): Serie de documentos formales que describen, especifican, ayudan en el desarrollo y establecen estándares de Internet y protocolos relacionados.

***Diffie-Hellman*:** Algoritmo de intercambio de claves que permite a dos partes que no se han conocido antes establecer un secreto compartido a través de un canal inseguro.

TELNET (*Teletype Network*): Protocolo de red anticuado utilizado para proporcionar comunicación de texto bidireccional interactiva.

IETF (*Internet Engineering Task Force*): Organización que desarrolla y promueve estándares voluntarios de Internet, enfocada en los aspectos técnicos de la infraestructura de Internet.

***OpenSSH*:** Conjunto de herramientas de conectividad de cliente y servidor que ofrecen sesiones de comunicaciones seguras a través de redes inseguras.

Tectia: Solución de software comercial que ofrece servicios de cifrado para transferencias de datos y administración remota a través de redes.

***Zero Trust*:** Enfoque de seguridad de red que no asume confianza inherente a ningún elemento, insistiendo en la verificación continua de todas las solicitudes de acceso.

***Hashing*:** Proceso de conversión de una entrada de cualquier longitud en una cadena de longitud fija, utilizando una función hash.

Fuerza Bruta: Método de prueba y error para descubrir información sensible como contraseñas o claves de cifrado.

***Spoofing*:** Acto de disfrazar la comunicación de un dispositivo, red o usuario, haciéndola parecer como si procediera de una fuente legítima.

TCP/IP (*Transmission Control Protocol/Internet Protocol*): Conjunto de protocolos de comunicación que interconectan dispositivos de red en Internet.

Credenciales: Información o datos utilizados para autenticar la identidad de un usuario ante un sistema.

Modelo Cliente-Servidor: Arquitectura de red donde múltiples clientes solicitan y reciben servicios de un servidor centralizado.

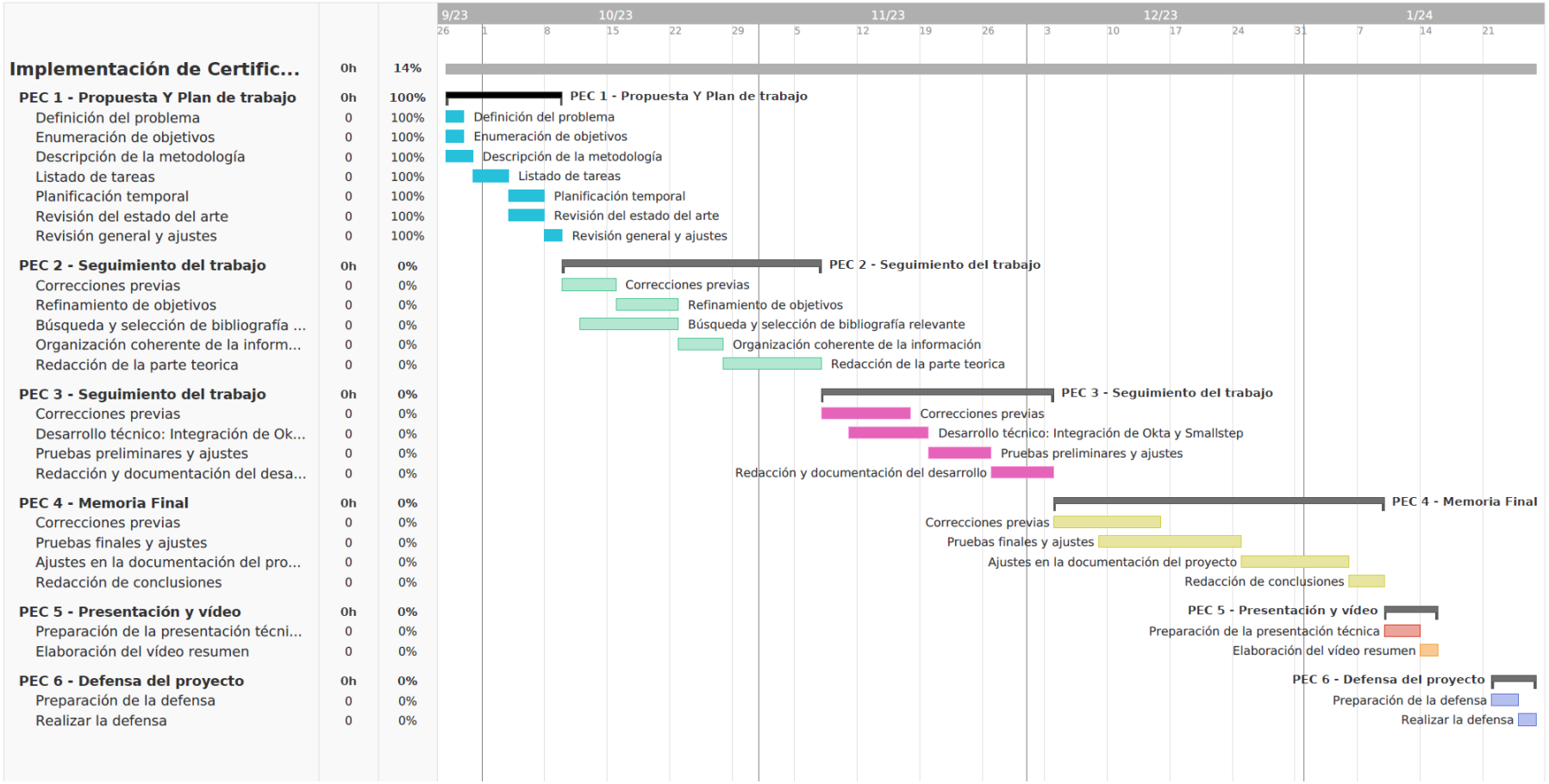
***Freeware*:** Software disponible para su uso sin costo, pero que a menudo tiene limitaciones en comparación con versiones pagadas.

Criptografía: Ciencia y arte de cifrar y descifrar información, asegurando la confidencialidad y autenticidad en la comunicación digital.

Encriptación: Proceso de convertir información o datos en un código para prevenir el acceso no autorizado.

8. Anexos

8.1. Diagrama de Gantt



8.2. Código de configuración CA

```
#!/bin/bash
#
# This script will launch and configure a step-ca SSH Certificate Authority with OIDC,
# ACME, AWS provisioners and SSHPOP
#
# Variables we will use for our setup.

OIDC_CLIENT_ID="0oa7253hg6prVEJoN697" # from Okta
OIDC_CLIENT_SECRET="[REDACTED]" # from Okta
ALLOWED_DOMAIN="[REDACTED].com"
CA_NAME="Certificate Authority"
ROOT_KEY_PASSWORD="TFG"
EMAIL="irene.balaguer@[REDACTED]"
LISTEN_ADDRESS=":10000"

TOKEN=`curl -X PUT "http://169.254.169.254/latest/api/token" -H "X-aws-ec2-metadata-token-ttl-seconds: 21600"`

# Get network data from the current AWS EC2 instance.
LOCAL_HOSTNAME=`curl -H "X-aws-ec2-metadata-token: $TOKEN" http://169.254.169.254/latest/meta-data/local-hostname`
LOCAL_IP=`curl -H "X-aws-ec2-metadata-token: $TOKEN" http://169.254.169.254/latest/meta-data/local-ipv4`
PUBLIC_HOSTNAME=`curl -H "X-aws-ec2-metadata-token: $TOKEN" http://169.254.169.254/latest/meta-data/public-hostname`
PUBLIC_IP=`curl -H "X-aws-ec2-metadata-token: $TOKEN" http://169.254.169.254/latest/meta-data/public-ipv4`
AWS_ACCOUNT_ID=`curl -H "X-aws-ec2-metadata-token: $TOKEN" http://169.254.169.254/latest/dynamic/instance-identity/document | grep accountId | awk '{print $3}' | sed  's/"//g' | sed 's/,//g'`

# Default configuration set by Okta's documentation
OPENID_CONFIG_ENDPOINT="https://[REDACTED].Okta.com/.well-known/openid-configuration"

# Download step-ca this tool will be able to setup the CA
```

```
curl -sLO https://github.com/Smallstep/certificates/releases/download/v0.24.2/step-ca_0.24.2_amd64.deb

# Install step-ca
dpkg -i step-ca_0.24.2_amd64.deb

# Download step-cli this tool will be used to work with the CA and hosts
curl -sLO https://github.com/Smallstep/cli/releases/download/v0.24.4/step-cli_0.24.4_amd64.deb

# Install step-cli
dpkg -i step-cli_0.24.4_amd64.deb

# All your CA config and certificates will go into $STEPPATH.
export STEPPATH=/etc/step-ca

# Create /etc/step-ca folder
mkdir -p $STEPPATH

# Modify permissions of the folder
# (7) The owner will have full permissions, read, write and execution.
# (0) No group member has read, write, or execute permissions
# (0) No permissions for anyone outside the owner and group.
chmod 700 $STEPPATH

# Save the $ROOT_KEY_PASSWORD in /etc/step-ca/password.txt, only the root user Will be able to see it or use it.
echo $ROOT_KEY_PASSWORD > $STEPPATH/password.txt

# Add a service to systemd for our CA, this will be used to run our CA on the background.
cat<<EOF > /etc/systemd/system/step-ca.service
[Unit]
Description=step-ca service
After=network.target
StartLimitIntervalSec=0
[Service]
Type=simple
Restart=always
```

```
RestartSec=1
User=root
Environment=STEPPATH=/etc/step-ca
ExecStart=/usr/bin/step-ca ${STEPPATH}/config/ca.json --password-
file=${STEPPATH}/password.txt
[Install]
WantedBy=multi-user.target
EOF

# Set up our basic CA configuration and generate root keys, this setup will allow us to have
an email with admin permissions that will be always aloe to generate a certificate for
himself
step ca init --ssh --name="$CA_NAME" \
    --dns="$LOCAL_IP,$LOCAL_HOSTNAME,$PUBLIC_IP,$PUBLIC_HOSTNAME" \
    --address=":443" --provisioner="$EMAIL" \
    --password-file="$STEPPATH/password.txt"

# Add the Okta OAuth provisioner, for user certificates, this configuration is done via the
documented process by Smallstep and Okta
step ca provisioner add Okta --type=oidc --ssh \
   --client-id="$OIDC_CLIENT_ID" \
   --client-secret="$OIDC_CLIENT_SECRET" \
   --configuration-endpoint="$OPENID_CONFIG_ENDPOINT" \
   --domain="$ALLOWED_DOMAIN" \
   --listen-address="$LISTEN_ADDRESS"

# Add the AWS provisioner, for host bootstrapping
step ca provisioner add "Amazon Web Services" --type=AWS --ssh \
   --aws-account=$AWS_ACCOUNT_ID

# The sshpop provisioner lets hosts renew their ssh certificates
step ca provisioner add SSHPOP --type=sshpop --ssh

# Add ACME provisioner in case we want to generate X.509 certificates
step ca provisioner add ACME --type=acme

# Add X5C Provisioner, will turn certificates from ACME into SSH public and private keys
```

```
step ca provisioner add x5c-testsecurity --type=X5C --x5c-roots
$STEPPATH/certs/root_ca.crt

# Use Okta (OIDC) as the default provisioner in the end user's ssh configuration template.
sed -i 's/\%p$/%p --provisioner="Okta"/g' /etc/step-ca/templates/ssh/config.tpl

# Restart the service we created to apply the new settings
service step-ca start

# Appends the line export STEPPATH=$STEPPATH to the /root/.profile file. This ensures
that the STEPPATH environment variable, which points to a directory (possibly /etc/step-
ca), is set every time the root user logs in.
echo "export STEPPATH=$STEPPATH" >> /root/.profile
```

8.3. Código de configuración del *host*

```
#!/bin/bash
#
# This script will get an SSH host certificate from our CA and add a weekly
# cron job to rotate the host certificate. It should be run as root.
#

CA_URL="https://[REDACTED].compute.amazonaws.com"

# Obtain your CA fingerprint by running this on your CA:
#   # step certificate fingerprint $(step path)/certs/root_ca.crt
CA_FINGERPRINT="1ed1bfa935bcc6973c36d44e0731cdd264d58868a9c816f08bc47f22c
696cddd"

STEPCLI_VERSION="0.24.4"

curl -LO https://github.com/Smallstep/cli/releases/download/v${STEPCLI_VERSION}/step-
cli_${STEPCLI_VERSION}_amd64.deb
dpkg -i step-cli_${STEPCLI_VERSION}_amd64.deb

# Configure `step` to connect to & trust our `step-ca`.
# Pull down the CA's root certificate
step ca bootstrap --ca-url $CA_URL \
                  --fingerprint $CA_FINGERPRINT
```

```
# Install the CA cert for validating user certificates (from /etc/step-
ca/certs/ssh_user_key.pub` on the CA).
step ssh config --roots > $(step path)/certs/ssh_user_key.pub

# This helps us avoid a potential race condition / clock skew issue
# "x509: certificate has expired or is not yet valid: current time 2020-04-01T17:52:51Z is
before 2020-04-01T17:52:52Z"
sleep 1

step certificate install $(step path)/certs/root_ca.crt

# When using ACME provisioner we need to generate SSH keys from a X5C Certificate

# step ca certificate [DOMAIN] internal.crt internal.key
# step ssh certificate --x5c-cert internal.crt --x5c-key internal.key [DOMAIN] internal


# Ask the CA to exchange our instance token for an SSH host certificate
# Use JWT provisioner to sign an existing key
# We have used www.noip.com to actually have a domain linked to our Public IPv4 address
so we can know which host is being bootstrapped.
step ssh certificate --host --sign \
  [DOMAIN] ssh_host_ecdsa_key.pub

# Create the user needed to log in the host instance
sudo adduser --quiet --disabled-password --gecos '' irenebalaguer

# Configure and restart `sshd` this config allows Smallstep to use and trust the signed keys
by our CA's root certificate.
tee -a /etc/ssh/sshd_config > /dev/null <<EOF
# SSH CA Configuration
# This is the CA's public key, for authenticatin user certificates:
TrustedUserCAKeys $(step path)/certs/ssh_user_key.pub
# This is our host private key and certificate:
HostKey /etc/ssh/ssh_host_ecdsa_key
HostCertificate /etc/ssh/ssh_host_ecdsa_key-cert.pub
EOF
```

```
# Restart ssh service to apply the changes
service ssh restart

# Now add a weekly cron script to rotate our host certificate.
cat <<EOF > /etc/cron.weekly/rotate-ssh-certificate
#!/bin/sh
export STEPPATH=/root/.step
cd /etc/ssh && step ssh renew ssh_host_ecdsa_key-cert.pub ssh_host_ecdsa_key --force
2> /dev/null
exit 0
EOF

# Modify permissions of the file
(7) The owner will have full permissions, read, write and execution.
(5) Group members have read and execute permissions
(5) Anyone outside the group or owner have read and execute permissions.
chmod 755 /etc/cron.weekly/rotate-ssh-certificate
```

9. Bibliografía

[1] Neil Ford, “List of Data Breaches and Cyber Attacks in 2023”, Octubre 05, 2023. (Última consulta: 06/10/2023). Disponible en: List of Data Breaches and Cyber Attacks in 2023

[2] Antivirus Guide, “Ransomware Statistics 2023”, (Última consulta: 06/10/2023). Disponible en: Ransomware Statistics 2023

[3] Gradient Technologies, “SSH Access Controls”, Noviembre 06, 2023. (Última consulta: 16/10/2023). Disponible en: What are the risks of using SSH keys?

[4] Antonio Rentero, “Los ciberataques continuarán aumentando durante 2023”, Marzo 02, 2023 (Última consulta: 07/10/2023). Disponible en: Los ciberataques continuarán aumentando durante 2023

[5] IBM, “Cost of a Data Breach Report 2023”, Julio 2023. (Última consulta: 17/10/2023). Disponible en: Cost of a Data Breach Report 2023

[6] UpGuard, “Cost of Data Breach 2023”, (Última consulta: 17/10/2023). Disponible en: Cost of Data Breach 2023

[7] Naciones Unidas, “Objetivos de Desarrollo Sostenible”, Septiembre, 2015 (Última consulta: 07/10/2023). Disponible en: Objetivos de Desarrollo Sostenible

[8] Ylonen T, Lonvick C, eds. “The *Secure Shell* (SSH) Protocol Architecture”, Enero, 2006. (Última consulta: 07/10/2023). Disponible en: The *Secure Shell* (SSH) Protocol Architecture

[9] SSH.COM. "SSH History - Part 1". 2023 (Última consulta: 20/10/2023). Disponible en: SSH History - Part 1

[10] IBM Developer, “Enhancing SSH Security”, (Última consulta: 20/10/2023). Disponible en: Enhancing SSH Security

[11] Mihir Bellare, Tadayoshi Kohno, Chanathip Namprempre. “Breaking and Provably Repairing the SSH Authenticated Encryption Scheme: A Case Study of the Encode-then-Encrypt-and-MAC Paradigm” 21 Marzo, 2004 (Última consulta: 09/10/2023). Disponible en: Breaking and Provably Repairing the SSH

[12] Cristina-Raluca Iță, Rodica-Claudia Constantinescu, Alexandru Vlădescu, Bogdan Alexandrescu. “Security in remote access, based on zero trust model concepts and SSH authentication with signed certificates”, 02 Marzo, 2023 (Última consulta: 09/10/2023). Disponible en: Security in remote access, based on zero trust model concepts and SSH authentication with signed certificates

[13] Wikipedia, "Criptografía postcuántica", 11 Marzo, 2023 (Última consulta: 20/11/2023). Criptografía Postcuántica

[14] Uitto M. "SSH in the World of Post-Quantum Cryptography (PQC)". Enero 18, 2023 (Última consulta: 03/11/2023). Disponible en: SSH in the World of Post-Quantum Cryptography (PQC)

[15] O'Reilly Media, “SSH, The *Secure Shell*: The Definitive Guide”, (Última consulta: 03/11/2023). Disponible en: SSH, The *Secure Shell*: The Definitive Guide

[16] Universitat Carlemany. “Qué es la seguridad informática: principios, tipos, ejemplos y más”, 13 Julio, 2023 (Última consulta: 16/11/2023). Qué es la seguridad informática: principios, tipos, ejemplos y más

[17] Ciberseguridad de la Universidad Pontificia Comillas, "Confidencialidad, Integridad y Disponibilidad", 8 Mayo, 2023 (Última consulta: 20/11/2023). Confidencialidad, Integridad y Disponibilidad

[18] Ylonen T, Lonvick C, eds. “The *Secure Shell* (SSH) Transport Layer Protocol”. Enero 2006. (Última consulta: 20/10/2024). Disponible en: The *Secure Shell* (SSH) Transport Layer Protocol

[19] Ylonen T, Lonvick C, eds. “The *Secure Shell* (SSH) Authentication Protocol”. Enero 2006. (Última consulta: 20/10/2024). Disponible en: The *Secure Shell* (SSH) Connection Protocol

[20] Ylonen T, Lonvick C, eds. “The *Secure Shell* (SSH) Connection Protocol”. Enero 2006. (Última consulta: 20/10/2024). Disponible en: The *Secure Shell* (SSH) Connection Protocol

[21] Phoenix Nap, "How Does SSH Work?”, 17 Diciembre, 2020 (Última consulta: 23/11/2023). How does SSH work

[22] Hostinger, “SSH Tutorial: How Does SSH Work”, (Última consulta: 22/11/2023). Disponible en: SSH Tutorial: How Does SSH Work

[23] Florian Bergsma, Benjamin Dowling, Florian Kohlar, Jörg Schwenk, Douglas Stebila. " Multi-ciphersuite security of the *Secure Shell* (SSH) protocol ". Junio 05, 2020 (Última consulta: 20/10/2023). Disponible en: Multi-ciphersuite security

[24] RunCloud, “Why Authentication Using SSH Public Key is Better than Using Password and How Do They Work?”, Noviembre 28, 2018 (Última consulta: 20/10/2023). Disponible en: Why Authentication Using SSH Public Key is Better

[25] TechTarget, "6 SSH best practices to protect networks from attacks", 16 de abril de 2021 (Última consulta: 23/11/2023). 6 SSH Best practices to protect networks from attacks

[26] Wikipedia. "Security Assertion Markup Language". Septiembre 25, 2023 (Última consulta: 25/10/2023). Disponible en: Security Assertion Markup Language

[27] OneLogin Developers, “OpenID Connect Overview”, (Última consulta: 25/10/2023). Disponible en: OpenID Connect Overview

[28] Dick Hardt. “The OAuth 2.0 Authorization Framework”. Octubre, 2012. (Última consulta: 25/10/2023). Disponible en: The OAuth 2.0 Authorization Framework

[29] Michael B. Jones, Dick Hardt. “The OAuth 2.0 Authorization Framework: Bearer Token Usage”. (Última consulta: 25/10/2023). Disponible en: The OAuth 2.0 Authorization Framework: Bearer Token Usage

[30] OpenID Foundation. "How Connect Works". Fecha de creación no especificada (Última consulta: 30/10/2023). Disponible en: How Connect Works

[31] OneLogin Developers, “OpenID Connect Overview”, (Última consulta: 02/11/2023). Disponible en: OpenID Connect Overview

[32] *Smallstep* Labs Inc. "Step-ca". 2023 (Última consulta: 02/11/2023). Disponible en: Step-ca

[33] Amazon Web Services, "Computación: tipos de instancias de Amazon EC2", sin fecha de última modificación especificada (Última consulta: 23/11/2023). Tipos de Instancias

[34] Amazon Web Services, "Conceptos de Amazon EC2",(Última consulta: 23/11/2023). Disponible en: Conceptos de Amazon EC2

[35] Keyfactor. "What is PKI?". 2023 (Última consulta: 05/11/2023). Disponible en: What is PKI?

[36] Sharon Boeyen, Stefan Santesson, Tim Polk, Russ Housley, Stephen Farrell, David Cooper. "Internet X.509 Public Key Infrastructure Certificate and

Certificate Revocation List (CRL) Profile". Mayo 2008 (Última consulta: 07/11/2023). Disponible en: Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile

[37] Wikipedia, "Lista de revocación de certificados", última edición el 13 de agosto de 2023 (Última consulta: 23/11/2023). Lista de revocación de certificados

[38] Tecalis, "Sellado de tiempo: qué es el sello o timestamp y sus usos", 2 de octubre de 2023 (Última consulta: 23/11/2023). Sellado de tiempo

[39] Wikipedia, "Autoridad de registro", última edición el 5 de noviembre de 2023 (Última consulta: 23/11/2023). Autoridad de registro

[40] Mike Malone. "Everything PKI". Junio 07, 2023 (Última consulta: 07/11/2023). Disponible en: Everything PKI

[41] CodeProject, "X.509 SSL Certificates With Custom Extensions", (Última consulta: 07/11/2023). Disponible en: X.509 SSL Certificates With Custom Extensions

[42] *Smallstep* Labs Inc., "Configuring **step-ca** Provisioners: OAuth/OIDC Single Sign-On", 2023. Disponible en: Configuring **step-ca** Provisioners

[43] *Smallstep*.com, "*Okta* Quickstart", 2023 (Última consulta: 23/11/2023). *Okta*: Step by step instructions