ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO

EVALUACIÓN DE SISTEMAS DE AUTORIZACIÓN EN PLATAFORMAS CLOUDING A TRAVÉS DEL PROTOCOLO OAUTH 2.0 QUE PERMITA MEJORAR EL CONTROL DE ACCESO A RECURSOS WEB

GUILLERMO VALENCIA PETROCHE

Trabajo de Titulación modalidad Proyectos de Investigación y Desarrollo, presentado ante el Instituto de Posgrado y Educación Continua de la ESPOCH, como requisito parcial para la obtención del grado de

MAGÍSTER EN SEGURIDAD TELEMÁTICA

RIOBAMBA - ECUADOR
Septiembre, 2021

ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO

CERTIFICACIÓN:

EL TRIBUNAL DE TRABAJO DE TITULACIÓN CERTIFICA QUE:

El **Trabajo de Titulación modalidad Proyectos de Investigación y Desarrollo**, determinado: Evaluación de sistemas de autorización en plataformas clouding a través del protocolo OAuth 2.0 que permita mejorar el control de acceso a recursos web, de responsabilidad del señor Guillermo Valencia Petroche, ha sido prolijamente revisada y se autoriza su presentación.

Tribunal:

Ing. Luis Eduardo Hidalgo Almeida; PhD.
PRESIDENTE

LUIS EDUARDO HIDALGO ALMEIDA
Firmado digitalmente por LUIS EDUARDO HIDALGO ALMEIDA
Nombre de reconocimiento (DN): c=EC, o=BANCO CENTRAL DEL ECUADOR, ou=ENTIDAD DE CERTIFICACION DE INFORMACION-ECIBCE, l=QUITO, serialNumber=0000445780, cn=LUIS EDUARDO HIDALGO ALMEIDA
Fecha: 2021.09.09 06:22:32 -05'00'

Ing. Washington Gilberto Luna Encalada; PhD.
DIRECTOR

Firmado electrónicamente por:
WASHINGTON GILBERTO LUNA ENCALADA

Ing. Diego Fernando Veloz Chérrez; Mag.
MIEMBRO

Firmado electrónicamente por:
DIEGO FERNANDO VELOZ CHERREZ

Ing. Antonio Manuel Meneses Freire; PhD.
MIEMBRO

MANUEL ANTONIO MENESES FREIRE
Firmado digitalmente por MANUEL ANTONIO MENESES FREIRE
Fecha: 2021.09.07 10:07:24 -05'00'

Riobamba, septiembre 2021

DERECHOS INTELECTUALES

Yo, Guillermo Valencia Petroche, declaro que soy responsable de las ideas, doctrinas y resultados expuestos en el **Trabajo de Titulación modalidad Proyectos de Investigación y Desarrollo**, y que el patrimonio intelectual generado por la misma pertenece exclusivamente a la Escuela Superior Politécnica de Chimborazo.

Guillermo Valencia Petroche
C.C.: 1802251759

DECLARACIÓN DE AUTENTICIDAD

Yo, Guillermo Valencia Petroche, declaro que el presente **Trabajo de Titulación modalidad Proyectos de Investigación y Desarrollo**, es de mi autoría y que los resultados del mismo son auténticos y originales. Los textos constantes en el documento que provienen de otra fuente están debidamente citados y referenciados.

Como autor, asumo la responsabilidad legal y académica de los contenidos de este proyecto de investigación de maestría.

Guillermo Valencia Petroche
C.C.: 1802251759

DEDICATORIA

A mi familia que siempre estuvo pendiente del total desarrollo del programa de maestría y a la Escuela Politécnica del Chimborazo.

Guillermo Valencia P.

AGRADECIMIENTO

Mi total agradecimiento a la Escuela Superior Politécnica de Chimborazo por permitirme ser parte de esta gran comunidad universitaria. En particular a CEDIA por facilitar su infraestructura.

The authors would like to thank the Corporación Ecuatoriana para el Desarrollo de la Investigación y Academia- CEDIA for their contribution in innovation, through the CEPRA projects, especially the project CEPRA-XIV-2020-08-RVA "Tecnologías Inmersivas Multi-Usuario Orientadas a Sistemas Sinérgicos de Enseñanza-Aprendizaje "; also the Escuela Superior Politécnica de Chimborazo, for the support for the development of this work

Guillermo Valencia P.

TABLA DE CONTENIDO

RESUMEN XVII

ABSTRACT.....XVIII

CAPÍTULO I.....1

1. INTRODUCCIÓN.....1

1.1. Antecedentes.....1

1.2. Planteamiento del problema.....1

1.2.1. Situación problemática.....1

1.2.2. Formulación del problema.....2

1.2.3. Sistematización del problema.....2

1.3. Justificación de la investigación.....3

1.4. Objetivos de la investigación.....3

1.4.1. Objetivo general.....3

1.4.2. Objetivos específicos.....3

1.5. Planteamiento de la hipótesis.....4

CAPÍTULO II.....5

2. MARCO DE REFERENCIA.....5

2.1. Antecedentes del problema.....5

2.2. Bases teóricas.....6

2.2.1. Cloud Computing.....6

2.2.2. Cloud Computing Service Models.....6

2.2.3. Cloud Computing Deployment Models.....7

2.2.4. Características de Cloud Computing.....8

2.2.5. Virtualización de Hardware.....8

2.2.6. Hipervisor.....9

2.2.7. Clasificación de los Hipervisores.....10

2.2.8. Contenedores *11*
2.2.9. Contenedores vs VMs *12*
2.2.10. Vagrant *12*
2.2.11. HyperText Transfer Protocol (HTTP) *13*
2.2.12. Método HTTP *14*
2.2.13. Introducción a Application Program Interface (API) *15*
2.2.14. Introducción a REST APIs *15*
2.2.15. Seguridad en REST APIs *15*
2.2.16. Acceso básico de autenticación *16*
2.2.17. OAuth 2.0 ..... *16*
2.2.18. Introducción del protocolo OAuth *16*
2.2.19. Roles de un modelo QAuth 2.0 ..... *18*
2.2.19.1. Servidor de Autorización *18*
2.2.19.2. Token de acceso *18*
2.2.19.3. Cliente *18*
2.2.19.4. Token de actualización *18*
2.2.19.5. Servidor de recursos *18*
2.2.19.6. Propietario del recurso *18*
2.2.20. Flujo del protocolo abstracto QAuth 2.0 ..... *19*
2.2.21. Registro de la aplicación *20*
2.2.22. Tipos de otorgamiento de la autorización *20*
2.2.23. Análisis de vulnerabilidades de OAuth 2.0 ..... *21*
2.2.23.1. Vulnerabilidades *21*
2.2.23.2. Ataques *22*
CAPÍTULO III **24**
3. DISEÑO DE LA INVESTIGACIÓN **24**
3.1. Tipo y diseño de la investigación **24**
3.2. Métodos y técnicas de investigación **24**
3.3. Enfoque de la investigación **25**

3.4. Alcance de la investigación **25**
3.5. Población de estudio **25**
3.6. Unidad de análisis **25**
3.7. Selección de la muestra **25**
3.8. Tamaño de la muestra **25**
3.9. Técnicas de recolección de datos **26**
3.10. Instrumentos para la recolección de datos **26**
3.11. Instrumentos para procesar datos recopilados **27**
3.12. Variables e indicadores **27**
3.13. Operacionalización conceptual de variables **27**
3.14. Operacionalización metodológica de variables **27**
3.15. Procesamiento y análisis **27**
3.16. Definición de las variables **28**
3.17. Diseño de escenarios **28**
3.17.1. Escenario base de Servidor de Recursos ***29***
3.17.1.1. Heroku *29*
3.17.1.2. Amazon AWS *30*
3.17.1.3. Motor de base de datos PostgreSQL *30*
3.17.1.4. Sistema de Inventario *30*
3.17.2. Escenario para análisis de acceso a recursos a través de autenticación básica ***30***
3.17.2.1. Equipo cliente con Vagrant *31*
3.17.2.2. Aplicación cliente para petición de recursos *31*
3.17.2.3. Repositorio de control de versiones GitHub *31*
3.17.3. Escenario para análisis de acceso a recursos a través de autorización con token de acceso ***32***
3.17.3.1. Equipo cliente con Vagrant *32*
3.17.3.2. Aplicación cliente para petición de recursos *32*
3.17.3.3. CEDIA *33*
3.17.3.4. Docker *33*
3.17.3.5. Motor de base de datos SQLite *34*

3.17.3.6. Repositorio de control de versiones GitHub 34
3.18. Plan de pruebas para acceso a recursos compartidos 34
3.18.1. Medir el rendimiento y el alcance con la aplicación cliente 34
3.18.2. Analizar las vulnerabilidades y ethical hacking 34
3.19. Plan para el análisis de las variables 36
3.19.1. Implementación del código para el análisis de las variables en R 37
CAPÍTULO IV 38
4. RESULTADOS Y DISCUSIÓN 38
4.1. Presentación de resultados 38
4.1.1. Tiempo de respuesta 38
4.1.1.1. Escenario con autenticación básica 38
4.1.1.2. Escenario con autorización por token 40
4.1.2. Autenticación y autorización 43
4.1.3. Roles y alcance de acceso a recursos compartidos 45
4.1.4. Cifrado de tráfico y encriptación 46
4.1.5. Alertas y monitoreo en recursos compartidos 48
4.1.6. Vulnerabilidades y mecanismos de protección frente a amenazas 49
4.1.6.1. Análisis de vulnerabilidades con autenticación básica 50
4.1.6.2. Análisis de vulnerabilidades con autorización por token 51
4.2. Análisis de interpretación 53
4.3. Comprobación estadística de la hipótesis 54
CAPÍTULO V 56
5. PROPUESTA DISEÑO E IMPLEMENTACIÓN 56
5.1. Sistema de autorización OAuth 2.0 56
5.1.1. Introducción 56
5.1.2. Escenario implementado 56
5.1.3. Requerimientos 57
5.1.4. Instalación 57
5.1.5. Clases implementadas 59

5.1.6. Verificar el servicio 59

5.1.7. Configuraciones adicionales 60

CONCLUSIONES 62

RECOMENDACIONES 63

BIBLIOGRAFÍA

ANEXOS

ÍNDICE DE TABLAS

Tabla 3-1. Operacionalización de variables 27

Tabla 3-2. Operacionalización metodológica de variables 27

Tabla 3-3. Variables para análisis de datos 28

Tabla 4-1. Tabla de tiempo de respuesta 42

Tabla 4-2. Tabla de escala de valores 42

Tabla 4-3. Tabla de ponderación tiempo 42

Tabla 4-4. Tabla de autenticación y autorización básica 43

Tabla 4-5. Tabla de autenticación y autorización por token 43

Tabla 4-6. Tabla de mecanismos utilizados en autenticación y autorización 44

Tabla 4-7. Tabla de escala de valores 44

Tabla 4-8. Tabla de ponderación autenticación autorización 45

Tabla 4-9. Tabla de roles y alcance con autenticación básica 45

Tabla 4-10. Tabla de roles y alcance con autorización por token 45

Tabla 4-11. Tabla de mecanismos utilizados en roles y alcance 46

Tabla 4-12. Tabla de escala de valores 46

Tabla 4-13. Tabla de ponderación roles y alcance 46

Tabla 4-14. Tabla de intercambio de mensajes en autenticación básica 46

Tabla 4-15. Tabla de intercambio de mensajes con autenticación por token 47

Tabla 4-16. Tabla de mecanismos utilizados en cifrado y encriptación 47

Tabla 4-17. Tabla de escala de valores 47

Tabla 4-18. Tabla de ponderación cifrado y tráfico 48

Tabla 4-19. Tabla de alertas y monitoreo con autenticación básica 48

Tabla 4-20. Tabla de alertas y monitoreo con autorización por token 48

Tabla 4-21. Tabla de instrumentos utilizados en alertas y monitoreo 48

Tabla 4-22. Tabla de escala de valores.....49

Tabla 4-23. Tabla de ponderación alertas y monitoreo 49

Tabla 4-24. Vulnerabilidades según OWAST 49

Tabla 4-25. Vulnerabilidades autenticación básica OWAST ZAP 50

Tabla 4-26. Vulnerabilidades detectadas con autenticación básica 51

Tabla 4-27. Vulnerabilidades Autorización por token OWAST ZAP 51

Tabla 4-28. Vulnerabilidades detectadas con autorización por token.....52

Tabla 4-29. Tabla de mecanismos de seguridad implementados 52

Tabla 4-30. Tabla de escala de valores.....53

Tabla 4-31. Tabla de ponderación vulnerabilidades.....53

Tabla 4-32. Tabla de matriz de impacto 53

ÍNDICE DE FIGURAS

Figura 2-1. Cloud service models 7

Figura 2-2. Virtualización 9

Figura 2-3. Contenedor 11

Figura 2-4. Contenedor vs VM 12

Figura 2-5. Vagrant 13

Figura 2-6. Protocolo HTTP 14

Figura 2-7. Token de acceso 17

Figura 2-8. Flujo de protocolo 19

Figura 2-9. OAuth en uso 21

Figura 3-1. Servidor de Recursos en Heroku 29

Figura 3-2. Recursos compartidos con autenticación básica 31

Figura 3-3. Recursos compartidos con token de acceso 32

Figura 3-4. Recursos de CEDIA 33

Figura 3-5. Owasp zap 35

Figura 3-6. Owasp zap configuración 35

Figura 3-7. Owasp zap autoscan 36

Figura 3-8. Creación del marco de datos 37

Figura 3-9. Marco de datos de análisis 37

Figura 4-1. Histograma de densidad en función del tiempo 38

Figura 4-2. Cuantil en función del tiempo 39

Figura 4-3. Test Shapiro y Ks 39

Figura 4-4. Histograma de densidad en función del tiempo 40

Figura 4-5. Cuantil en función del tiempo 40

Figura 4-6. Test Shapiro AWS 41

Figura 4-7. Comparativa en función del tiempo 42

Figura 4-8. Análisis por comando curl en el servidor de recursos 50

Figura 4-9. Análisis por comando curl en el servidor de acceso 52

Figura 4-10. Test Chi-cuadrado 54

Figura 4-11. Valor crítico 54

Figura 5-1. Sistema implementado 56

Figura 5-2. Clases implementadas 59

Figura 5-3. Token de acceso del servidor de acceso 59

Figura 5-4. Verificar token de acceso del servidor 60

Figura 5-5. Control de acceso del servidor 60

Figura 5-6. Diagrama de tipos de acceso OAuth2 61

ÍNDICE DE ANEXOS

ANEXO A. Configuración de archivo Vagrant

ANEXO B. Archivo de configuración de clase AWS y test de muestra

ANEXO C. Configuración de tablas en la base de datos de control de acceso Oauth2

ANEXO D. Datos para estudio de autenticación básica en función del tiempo

ANEXO E. Datos para estudio de autorización por token en función del tiempo

ANEXO F. Informe OWASP ZAP de autenticación básica

ANEXO G. Informe OWASP ZAP de autorización por token

RESUMEN

El objetivo de este trabajo de investigación fue desarrollar un sistema de autorización en plataformas clouding a través del protocolo OAuth 2.0 que permita mejorar el control de acceso a recursos web mediante la implementación de servicios REST API para que, se acoplen en el actual modelo de negocio en el intercambio de recursos de manera confiable, segura y que, se adaptan a cada necesidad de organizaciones, instituciones, empresas públicas o privadas, para lo cual en el desarrollo de la investigación, se utilizaron los métodos inductivos-deductivos con un enfoque cualitativo, para realizar el estudio de accesos, se aplicaron intercambio de peticiones y respuestas a los servidores de recursos web en la cloud computing, evidenciando un considerable mejoramiento en tiempos de respuesta, autenticación, autorización, roles, cifrado, encriptación, alertas, monitoreo y protección frente a amenazas, a la vez que se pudo determinar los posibles riesgos que puedan poner en peligro la autenticación, autorización y a las propiedades de integridad de sesión en el intercambio de mensajes con lo que, se tomó al sistema implementado como patrón de diseño de Sistemas de Token de Seguridad (STS) para ejecutar procedimientos que permiten mitigar las potenciales amenazas y fortaleciendo los procesos desde el punto de vista de la Seguridad Telemática.

Palabras Clave: < SEGURIDAD TELEMÁTICA >, <CLOUD COMPUTING>, <RECURSOS WEB>, <REST API>, < SISTEMAS DE TOKEN DE SEGURIDAD >

ABSTRACT

The objective of this research was to develop an authorization system on cloud platforms through the OAuth 2.0 protocol that allows to improve access control to web resources by implementing RESTAPI services. So that, they are coupled in the current business model in the exchange of resources in a reliable, secure and adaptable way to every need of organizations, institutions, public or private enterprises. For which in the development of the research, inductive-deductive methods were used with a qualitative approach, to carry out the study of accesses, exchange of requests and responses were applied to the servers of web resources in the cloud computing, evidencing a considerable improvement in response times, authentication, authorization, roles, encryption, alerts, monitoring and threat protection. At the same time that it was possible to determine the possible risks that may endanger the authentication, authorization and session integrity properties in the exchange of messages with which, the system implemented was taken as a design pattern of Security Token Systems (STS) to execute procedures that allow to mitigate potential threats and strengthening the processes from the point of view of Telematics Security.

.

KEYWORDS: <CLOUD COMPUTING>, < RESOURCES WEB>, <REST API>, <SECURITY TOKEN SYSTEMS>, < TELEMATICS SECURITY >

CAPÍTULO I

1. INTRODUCCIÓN

1.1. Antecedentes

Cada dispositivo electrónico inteligente conectado a la red global llamada Internet tiene acceso a la Cloud computing, lo que permite compartir información entre estos dispositivos a través de recursos informáticos. Este intercambio de información puede ser pública o privada lo que conlleva a determinar el alcance de acceso a los datos por medio de métodos de autenticación y autorización.

Con la aparición de grandes empresas proveedoras de servicios y recursos computacionales (cloud computing) como son: Amazon Web Services, Windows Azure, Google App Engine Cloud, IBM SmartCloud, Oracle Cloud, entre otros, que han facilitado a los usuarios finales utilizar las nuevas tecnologías sin la necesidad de un vasto conocimiento y reduciendo costos de infraestructura.

La demanda de acceso a datos remotos ha crecido exponencialmente haciendo que a los usuarios les resulte difícil mantener cuentas diferentes en cada uno de los servicios que utilizan. La solución a este problema es el framework OAuth 2.0, que hace uso de una sola cuenta para identificar a los usuarios a través de los diferentes servicios sin compartir o transferir contraseñas.

El objetivo de este documento es evaluar los sistemas de autorización en plataformas clouding a través del protocolo OAuth 2.0, que permita mejorar el control de acceso a recursos web y explorar las posibles soluciones que permita alcanzar los niveles de seguridad deseados.

1.2. Planteamiento del problema

1.2.1. Situación problemática

En el modelo tradicional de autenticación cliente servidor, el cliente ingresa sus credenciales en un formulario de inicio para solicitar un recurso de acceso restringido o protegido al servidor. Los nombres de usuario y la contraseña suelen ser el mínimo común denominador para autenticación y autorización en la web.

Sin embargo, solicitar al usuario su contraseña tiene varios efectos secundarios como son: disminución de la confiabilidad, sensibilidad del usuario al phishing, acceso total a los recursos, fiabilidad limitada, dificultad para implementar una autenticación más fuerte (Richer, 2017).

Para compartir recursos restringidos las aplicaciones de terceros solicitan acceso a estos recursos y el propietario comparte sus credenciales con las aplicaciones, lo que genera problemas y limitaciones que, se indican a continuación:

- Se requieren aplicaciones de terceros para almacenar las credenciales del propietario para uso futuro, generalmente una contraseña en texto plano.

- Se requiere que los servidores soporten la autenticación de contraseña, a pesar de las debilidades de seguridad inherentes a las contraseñas.

- Las aplicaciones de terceros obtienen un acceso total al recurso protegido del propietario, por lo que, el propietario del recurso no puede restringir ni el tiempo de duración de acceso ni limitar el alcance.

- Los propietarios de recursos no pueden revocar el acceso a un tercer individuo sin revocar el acceso a todos los terceros y debe hacerlo cambiando la contraseña del tercero.

1.2.2. Formulación del problema

¿La evaluación de sistemas de autorización en plataformas clouding mediante la implementación del protocolo OAuth 2.0, mejorará el control de acceso a recursos web de manera que sean universales, de fácil ingreso, seguros, y que prevengan los accesos no autorizados?

1.2.3. Sistematización del problema

- ¿Qué características debe tener el protocolo de autorización OAuth 2.0, para que un sistema permita asegurar el control de acceso?

- ¿Qué vulnerabilidades pueden afectar a la seguridad de la información en protocolos de autorización?

- ¿Qué mecanismo de autorización se implementará en plataformas clouding para mejorar los procesos convencionales?

- ¿Cuáles son los resultados de la evaluación de flujos de autorización en plataformas clouding?

1.3. Justificación de la investigación

En la actualidad la demanda de acceso a datos remotos en varios servicios, se ha incrementado de manera exponencial, por lo que, a los usuarios les resulta difícil mantener diferentes cuentas en cada servicio que utilizan (Siriwardena, 2019).

El nombre de usuario y contraseña usan un enmascaramiento de solicitud simple, pero esto conlleva a un riesgo de seguridad inaceptable. En contraste con esto, OAuth promueve un modelo de privilegios mínimos, lo que permite a un usuario otorgar acceso limitado a sus aplicaciones y datos mediante la emisión de un token con capacidad limitada (Boyd, 2012).

A medida que los datos migran a la nube, los servicios de cloud computing tienen mayor intercambio de operaciones con dispositivos inteligentes como son teléfonos inteligentes y tabletas (O'Raw, 2015).

Un protocolo universal es requerido para que estos servicios y dispositivos puedan acceder a estos datos sin comprometer la seguridad y la integridad. OAuth se aleja del paradigma vulnerable de nombre de usuario contraseña y ha adoptado el uso de una ficha de portador.

Muchas prácticas comunes y modelos de acceso aceptados no abordan los desafíos importantes inherentes al diseño de sistemas para arquitecturas de computadoras modernas. Para tener éxito, el control de acceso a los sistemas debe hacer frente a un entorno hostil en Internet.

El acceso no autorizado a datos sensibles puede ocurrir en entornos seguros como centros de datos administrados con cuidado, e incluso en arquitecturas virtualizadas, por lo que, es necesario la implementación de controles de acceso con modelos bien establecidos.

1.4. Objetivos de la investigación

1.4.1. Objetivo general

Evaluar sistemas de autorización en plataformas clouding dentro de un ambiente de pruebas a través del protocolo OAuth 2.0 que permita mejorar el control de acceso a recursos web.

1.4.2. Objetivos específicos

- Fundamentar teóricamente los protocolos de autorización.
- Identificar y evaluar vulnerabilidades en procesos de autorización de plataformas clouding.

- Diseñar e implementar mecanismos de autorización con el objetivo de mejorar los accesos a recursos web en un entorno de plataformas clouding a través del protocolo OAuth 2.

- Evaluar el mejoramiento de los flujos de autorización implementados en recursos web de plataformas clouding dentro de un ambiente de pruebas.

1.5. Planteamiento de la hipótesis

La evaluación de sistemas de autorización en plataformas clouding a través del protocolo OAuth 2.0 mejorará el control de acceso a recursos web.

CAPÍTULO II

2. MARCO DE REFERENCIA

2.1. Antecedentes del problema

La gran mayoría de servicios web utilizan arquitectura monolítica, en donde el proceso de autenticación y autorización, se realiza una sola vez al inicio de sesión. Esto permite el acceso a todos los recursos del sistema o en el mejor de los casos a parte del sistema. Si otro usuario requiere cierta información, se le debe obligatoriamente proporcionar un usuario y clave con la confianza de que va a darle el uso adecuado.

El presente proyecto toma como referencia artículos científicos afines a la propuesta que respaldan la presente investigación.

- Determinar las vulnerabilidades conocidas, cómo operan y ponen en peligro la autenticación, la autorización y las propiedades de una sesión íntegra. Dado que el marco de OAuth 2.0 es ampliamente adoptado, se debe apoyar la adopción urgente de esta propuesta técnica para abordar cada tema conocido (Argyriou, 2017).

- En un modelo tradicional de autenticación cliente-servidor, el cliente presenta su ID de credencial y contraseña para que la autenticación acceda a los recursos almacenados en el servidor. A medida que se desarrolló Internet, diferentes servicios web han cooperado para crear servicios de mash-up. En este caso, el propietario de un recurso debe ser autenticado por el servidor que almacena el recurso, y una aplicación web de terceros debe estar autorizada para su acceso al recurso (Shernan, 2015).

- Una aplicación accede a recursos protegidos por un servidor de recursos en nombre del propietario del recurso, al consumir un token de acceso emitido por el servidor de autorización. OAuth 2.0 involucra cuatro roles. El propietario, que es un host que actúa en nombre de un usuario final y puede otorgar acceso a recursos protegidos. El servidor de recursos, que es un servidor de almacenamiento de recursos protegidos y consume tokens de acceso proporcionados por un servidor de autorización. El Cliente, que es una aplicación donde, se ejecuta en un servidor para realizar solicitudes en nombre del propietario del recurso. El Servidor de Autorización, que genera tokens de acceso para el cliente después de autenticar al propietario del recurso y obtener su autorización (Seitz, 2018).

OAuth 2.0 fue estandarizado en 2012 por Internet Engineering Task Force (IETF) RFC 6749 como una organización de normalización de código abierto, cuyo objetivo principal es contribuir a la tecnología del internet.

2.2. Bases teóricas

2.2.1. Cloud Computing

Cloud computing hace referencia a la asignación remota de recursos en la nube. Según el Instituto Nacional de Estándares y Tecnología (NIST), define formalmente a la computación en la nube como lo siguiente:

"La computación en la nube es un modelo para permitir el acceso ubicuo y conveniente a un conjunto compartido de recursos informáticos configurables como son: redes, servidores, almacenamiento, aplicaciones y servicios, que pueden aprovisionarse y liberarse rápidamente con un mínimo de esfuerzo de gestión o de interacción con el proveedor de servicios" (NIST, 2020).

2.2.2. Cloud Computing Service Models

Los proveedores de infraestructura de computación en la nube ofrecen diversos servicios que dan aprovisionamiento básico y la liberación de recursos. La mayoría de estos modelos de servicios de infraestructura, se encuentran en una de las siguientes categorías:

- **Infrastructure as a Service (IaaS):** Los recursos como servidores, redes y almacenamiento son ofrecidos como servicios, estos recursos actualmente pueden ser virtualizados sin que el usuario vea la diferencia. Iaas es una forma de renta computacional donde los pagos están relacionados de acuerdo a su uso. El software de administración del Iaas aprovisiona recursos de acuerdo a como se necesitan, además son escalables dinámicamente (Hill, 2012).

- **Platform as a Service (PaaS):** Van en paralelo con el web hosting, en la que dispone de un completo aprovisionamiento de software de desarrollo y pruebas para aplicaciones dentro de la nube. Similar al IaaS, los recursos son dinámicamente escalables. PaaS es buena opción si su ambiente de desarrollo de aplicaciones concuerda con el proveedor del servicio en la nube o si se desea experimentar con los nuevos productos y servicios. Un potencial inconveniente es la carencia de portabilidad que ata al desarrollador al conjunto de herramientas provisionadas por el proveedor del servicio en la nube (Hill, 2012).

- **Software as a Service (SaaS)**: En algunas ocasiones SaaS es el camino más fácil de cloud computing. Se puede experimentar con algún software disponible en la plataforma por un tiempo limitado. Si es necesario se continúa con el servicio con un plan de pago. El software escala automáticamente al número de usuarios disponibles y los datos también son respaldados. SaaS ofrece los beneficios de negocios como son los Customer Relationship Management (CRM) SaaS, que ofrece un nivel empresarial sin inversión inicial y de problemas de infraestructura, donde, se debe enfocar solo en el flujo del negocio. Se crea entonces un alto nivel organizacional denominado business process management (BPM) (Hill, 2012).

Figura 2-1. Cloud service models
Fuente: (Hill, 2012)
Elaborado por: Guillermo Valencia, 2020

2.2.3. Cloud Computing Deployment Models

- **Pública**: Es pública para todos los usuarios en general y es administrado por una organización. La organización puede ser académica, de negocios o gubernamentales. El propietario es el encargado de administrar la infraestructura en la nube.

- **Privada:** Estos servicios informáticos se ofrecen a través de una red interna privada para pocos usuarios y no para el público en general. También se la conoce como nube corporativa.

- **Híbrida:** Este tipo de infraestructura en la nube es utilizada en situaciones particulares. Puede utilizar la nube pública para aspectos de negocios, mientras que para manejar datos sensibles puede utilizar nube privada.

- **Comunitaria:** Formada por múltiples organizaciones que comparten una común infraestructura de la nube.

- **Distribuida:** Compuesta por sistemas distribuidos conectados a una simple red.

- **Múltiple:** Una organización utiliza múltiples nubes públicas en su carga de trabajo típicamente para evitar bloqueos del vendedor.

- **Diversos:** Las organizaciones utilizan varios proveedores de servicios públicos para apalancar servicios específicos de cada proveedor (LinuxFoundationX, 2020).

2.2.4. Características de Cloud Computing

Entre las características principales de cloud computing tenemos:

- **Fácil acceso a recursos:** Los usuarios pueden acceder a los servicios de la nube desde cualquier lugar y dispositivo, tan pronto como estén conectados al proveedor de servicios en la nube.

- **Velocidad y agilidad:** Los recursos de la nube se aprovisionan con unos pocos clics, lo que ahorra tiempo y proporciona agilidad. También se amplían o reducen con facilidad, aumentando o disminuyendo el número de instancias de funcionamiento de un recurso concreto como respuesta a la demanda actual o en función de la demanda futura proyectada. Una característica adicional es la elasticidad, que permite que algunas instancias de recursos de nubes aumenten o disminuyan de tamaño, también en función de la demanda.

- **Mantenimiento:** El proveedor de servicio es el encargado de mantener los equipos encendidos, actualizados y conectados, sin interferencia del usuario del servicio.

- **Costo:** No hay necesidad de adquirir infraestructura física, el usuario alquila el servicio de infraestructura de acuerdo a las necesidades del negocio.

- **Fiabilidad:** Los recursos pueden estar hospedados en diferentes data centers, lo que incrementa la confiabilidad, resiliencia y alta disponibilidad (LinuxFoundationX, 2020).

2.2.5. Virtualización de Hardware

La virtualización de hardware es una tecnología de desarrollo que explota el continuo incremento del poder del procesador, habilitando instancias virtuales de hardware que se ejecutan sobre infraestructuras físicas. Estas tecnologías han permitido a organizaciones como centro de datos mejorar la administración y manejo de sus propios recursos con la construcción de varias capas virtuales de hardware a través de numerosas máquinas físicas (Hill, 2012).

La virtualización puede alcanzar diferentes capas de hardware y de software, como la unidad central de proceso (CPU), disco duro, memoria RAM, el sistema de archivos, etc. La creación de máquinas virtuales (VMs) después de emular componentes esenciales de hardware que soportan la instalación de Sistemas Operativos (OS) invitados. Una VM representa una colección aislada de recursos emulados, comportándose como un sistema físico actual (LinuxFoundationX, 2020).

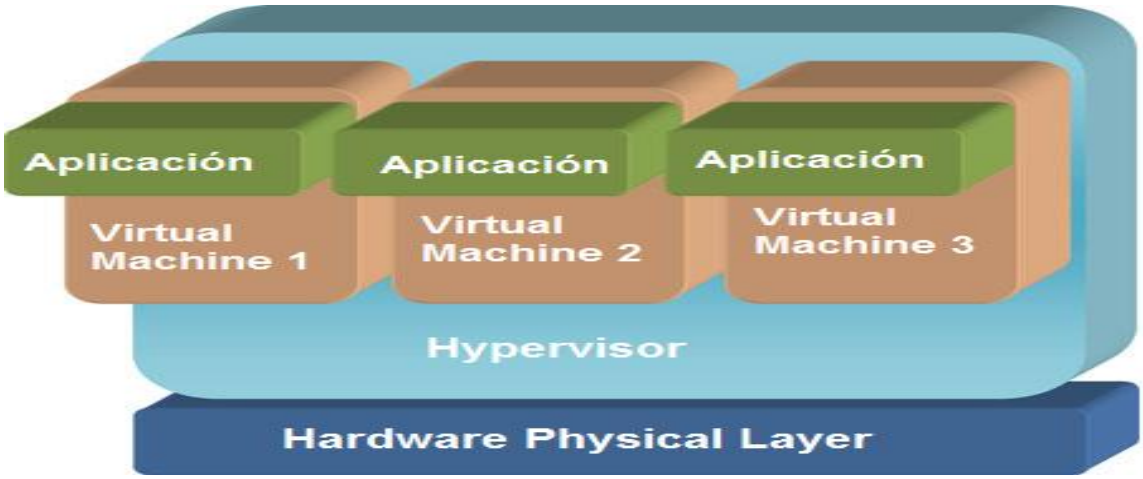

Figura 2-2. Virtualización
Fuente: (Hill, 2012)
Elaborado por: Guillermo Valencia, 2020

2.2.6. Hipervisor

El Hipervisor es una pieza de software de capas de creación de múltiples ambientes virtuales operacionales aislados, cada recurso emulado es hecho disponible para el sistema operativo invitado (LinuxFoundationX, 2020).

Los hipervisores habilitan la virtualización de hardware de computadora como el CPU, disco, memoria RAM, red, etc. y permiten la instalación de VMs invitados en su nivel más alto. Se puede crear múltiples VMs invitadas con diferentes sistemas operativos en un simple hipervisor. Por ejemplo, se puede crear una máquina nativa Linux como host, corriendo en un bare-metal, y después configurar un hipervisor tipo-2 para crear varias máquinas virtuales con diferentes sistemas operativos.

La virtualización de hardware es soportada por los modernos CPUs, y esta es la característica que permite al hipervisor virtualizar el hardware físico de un equipo, además se puede compartir múltiples recursos con múltiples sistemas invitados de forma eficiente y segura. Los modernos CPUs permiten la virtualización anidada, es decir, VMs dentro de una VM (LinuxFoundationX, 2020).

2.2.7. Clasificación de los Hipervisores

- **Type-1 hypervisor** (bare-metal nativo) corre directamente en el tope del hardware de la máquina física, sin la necesidad de un sistema operativo.

  Ejemplos de type-1 hypervisors:

  - IBM z/VM
  - Microsoft Hyper-V
  - Nutanix AHV
  - Oracle VM Server for SPARC
  - Oracle VM Server for x86
  - VMware ESXi
  - AWS Nitro
  - Xen.

- Type-2 hypervisor (equipo) corre en tope de los sistemas operativos. Típicamente son para usuarios finales.

  Ejemplos de type-2 hypervisors:

  - VirtualBox
  - VMware Player
  - VMware Workstation
  - Parallels Desktop for Mac.

Existen también hipervisores que pueden estar en las dos categorías, pueden ser hipervisores de tipo-1 y tipo-2. Tienen los módulos de kernel Linux que actúan como hipervisores tipo-1 y tipo-2 al mismo tiempo.

Ejemplos de hipervisores tipo-1 y tipo-2 son:

- KVM
- Bhyve (LinuxFoundationX, 2020).

2.2.8. Contenedores

Un contenedor es una unidad estándar de software que empaqueta el código y todas sus dependencias para que la aplicación, se ejecute de forma rápida y segura de un ambiente computacional a otro (Docker, 2020).

Generalmente el reto de cualquier aplicación se encuentra en la portabilidad, la necesidad de trabajar consistentemente sobre múltiples plataformas de hardware, como son VMs, data centers, nube pública y privada, todos en ambientes de desarrollo en producción (LinuxFoundationX, 2020).

Independientemente de la plataforma, el uso de tecnología de contenedores como por ejemplo Docker, la aplicación y sus dependencias están empaquetados en una caja contenedora. Esta caja contenedora puede ser transportada a cualquier plataforma y podrá correr idénticamente en cada una.

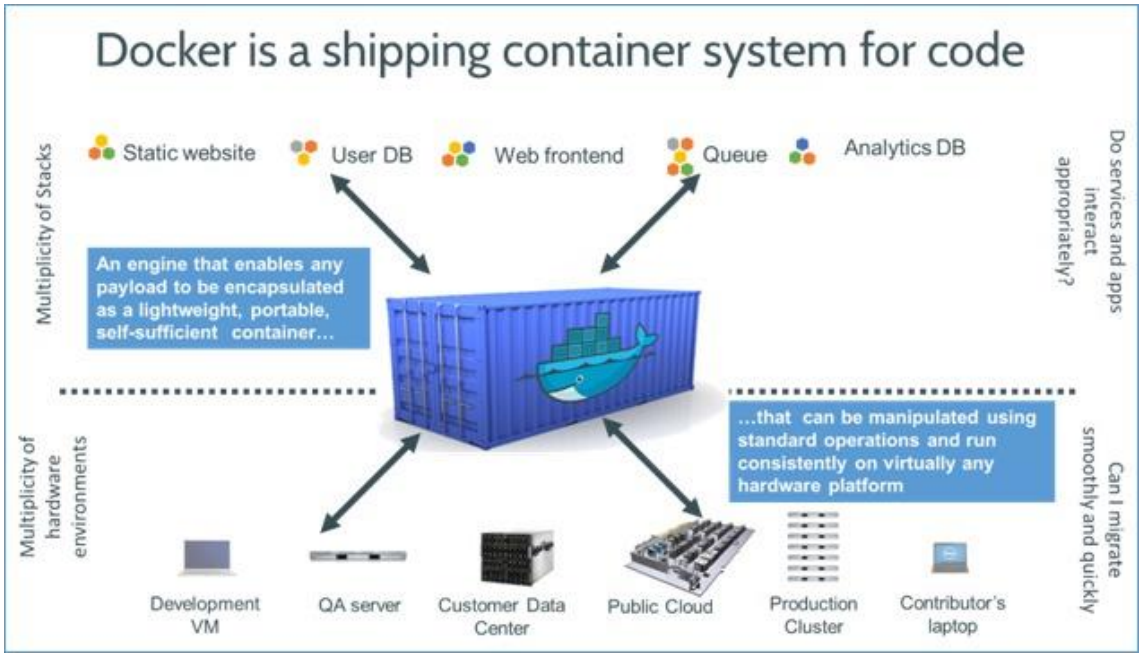

Figura 2-3. Contenedor
Fuente: (LinuxFoundationX, 2020)

El código origen de la aplicación y todas sus dependencias y librerías están referenciadas como (imagen). La instancia que hace referencia a esta imagen se la conoce como (contenedor). Se pueden correr múltiples contenedores de la misma imagen.

2.2.9. Contenedores vs VMs

Una máquina virtual se ejecuta en el tope de un hipervisor, que virtualiza un sistema de hardware emulando CPU, memoria y recursos de hardware de red donde el sistema operativo invitado, se ejecuta en el tope de ellos. Diferentes clases de sistemas operativos invitados pueden ejecutarse en el tope de un hipervisor. Cada VM incluye una copia del sistema operativo lo que las hace lentas en el arranque.

En contraste de VMs, los contenedores se ejecutan directamente como procesos sobre el sistema operativo del host. Los contenedores ocupan menos espacio que las VMs y puede manejar más aplicaciones y requieren menos VMs y SOs (Docker, 2020).

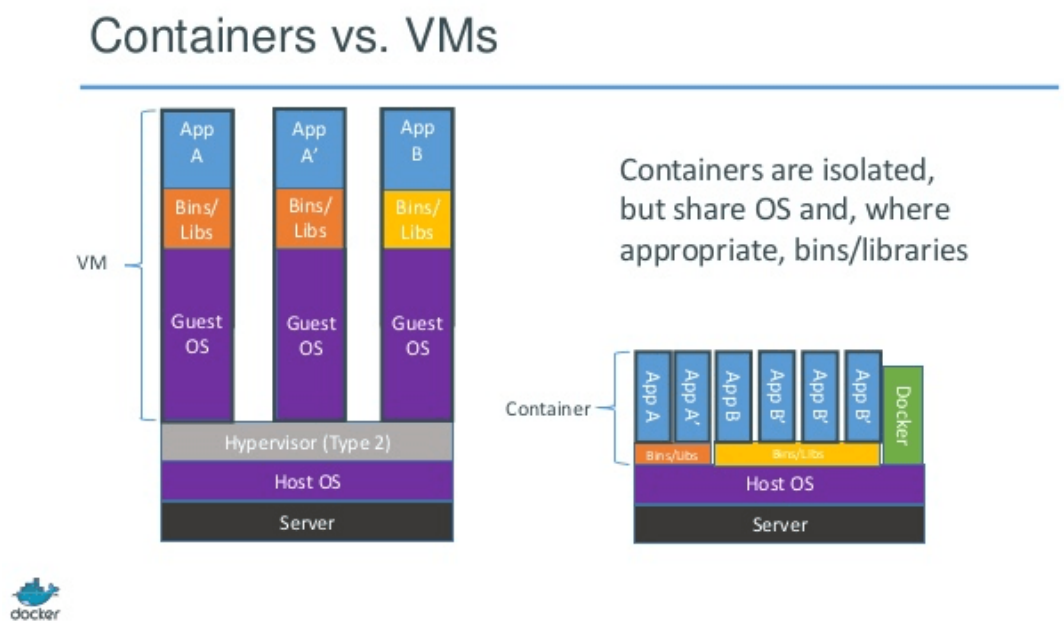

Figura 2-4. Contenedor vs VM
Fuente: (Docker, 2020)

2.2.10. Vagrant

Vagrant es una herramienta que permite crear y manipular entornos de VM en un simple flujo de trabajo enfocado en la automatización, disminuyendo el tiempo de configuración del entorno de desarrollo y mejorando la producción (HashiCorp, 2020).

Es configurado a través del archivo Vagrantfile, que contiene información almacenada en texto plano. Vagrantfile puede ser utilizado para configurar una simple máquina o múltiples máquinas.

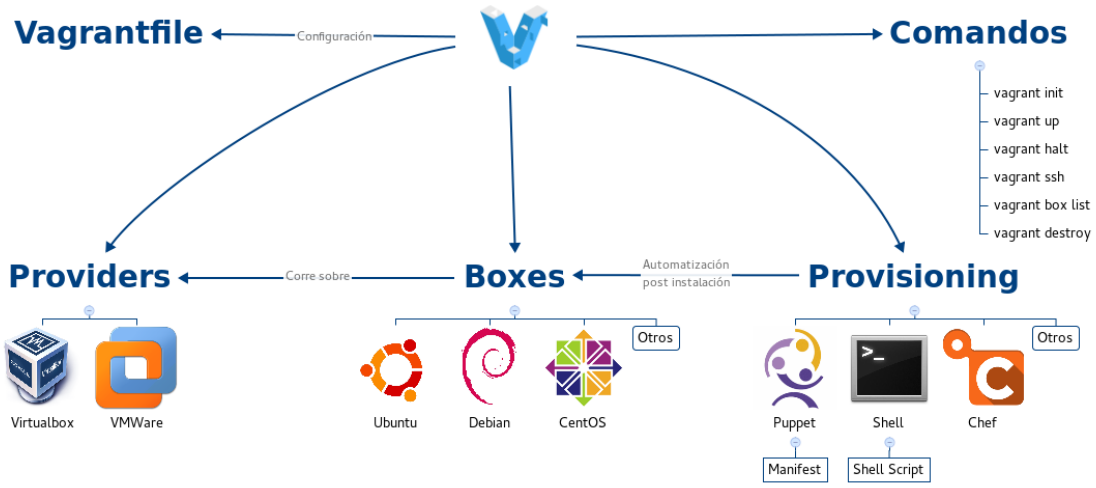

Figura 2-5. Vagrant
Fuente: (Tomas, 2014)

2.2.11. HyperText Transfer Protocol (HTTP)

Es el protocolo de comunicación que permite la transferencia de información entre dos entidades o nodos a través de mensajes formateados y con reglas preestablecidas.

Entre las partes principales de un mensaje tenemos:

- **URL:** es un localizador de recurso uniforme, contiene el destino del mensaje como una cadena de caracteres. Como ejemplo: http://myserver.com/greeting?name=Alex.

- **Body:** usualmente está presente en la respuesta del mensaje, aunque el mensaje de solicitud también puede contenerlo. En el cuerpo del mensaje puede contener código HTML para representar la página. Puede contener texto en cualquier formato, imágenes, JSON, etc.

- **Headers:** son los metadatos que el receptor necesita para entender el contenido del mensaje. La cabecera consiste en un arreglo de un par de clave y valor. Ejemplo:

  Accept: text/html

  Cookie: name=Alex

- **Código de estado:** este código está presente en la respuesta. Identifica el estado de la respuesta en código numérico para que el navegador y otras herramientas puedan reaccionar, por ejemplo

  200: la solicitud fue exitosa.

  401: no autorizado; el usuario no tiene permisos para ver el recurso.

  404: página no encontrada.

  500: error interno del servidor (Lopez, 2016).

2.2.12. Método HTTP

Son las acciones que se deben ejecutar en los mensajes. Los más comunes son GET, POST, DELETE, PUT, o OPTION.

GET: solicita información al receptor de algún recurso específico. Estas solicitudes solo deben recuperar datos y no deben tener otro efecto.

POST: envía datos en el cuerpo del mensaje para que sean procesados por el recurso identificado de la URI, por lo general actualiza datos recibidos.

DELETE: borra el recurso que se ha especificado.

PUT: envía datos al servidor para que sean actualizados.

OPTION: retorna los métodos HTTP que el servidor soporta para un URL específico y comprueba su funcionalidad.

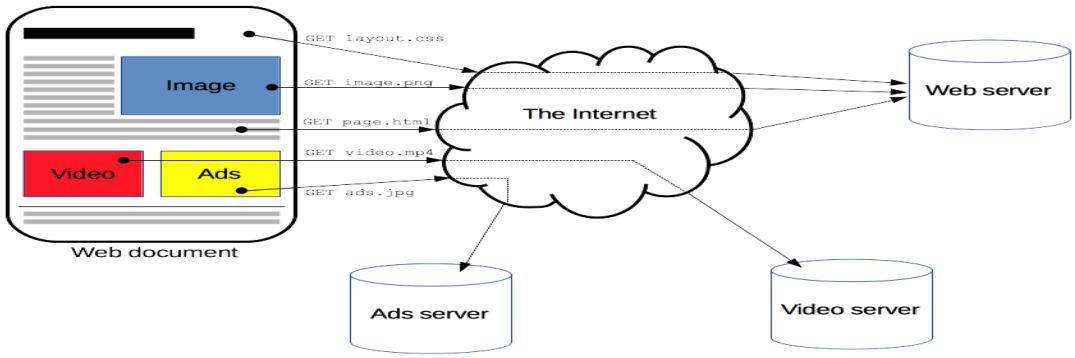

Figura 2-6. Protocolo HTTP
Fuente: (Mozilla, 2020)

2.2.13. Introducción a Application Program Interface (API)

Su objetivo es proveer de una interfaz para que otros programas puedan enviar comandos que puedan ejecutar algún proceso dentro de la aplicación y posiblemente que retornen algún tipo de salida. Existen virtualmente APIs para todo lo que a computación se refiere. Las APIs las encontramos, por ejemplo, cuando alguna aplicación intenta comunicarse con el Sistema Operativo, o cuando una aplicación móvil provee de GUI para interactuar con el usuario. (Lopez, 2016).

2.2.14. Introducción a REST APIs

Este tipo específico de APIs usa el protocolo HTTP para comunicarse, siendo las más utilizadas en aplicaciones web, al igual que estas, el cliente envía una petición HTTP y el servidor replica con una respuesta HTTP. La diferencia es que las REST APIs hacen uso extensivo de los códigos de estado HTTP para entender el tipo de respuesta, y en lugar de retornar recursos HTML con CSS o javascript, la respuesta usa JSON, XML, o cualquier otro formato con solo información, y no una interfaz gráfica de usuario (Lopez, 2016).

2.2.15. Seguridad en REST APIs

REST APIs es una herramienta que permite a los desarrolladores recuperar y actualizar datos del servidor, lo que implica una gran responsabilidad en el diseño, para mantener los datos seguros como sea posible.

Similar a las aplicaciones web, hay dos conceptos de seguridad que se manejan:

- **Autenticación:** es el proceso para verificar la identidad de un usuario y determinar si este es quien dice ser. La autenticación es típicamente realizada solicitando al usuario por un username y un password (Boyd, 2012).

- **Autorización:** es el proceso para verificar que el usuario dispone de los permisos para ejecutar una determinada acción. Este generalmente requiere primero la validación del usuario (autenticación), para luego determinar si el actual usuario está autorizado y finalmente acceder a los datos y servicios que le han sido otorgados (Boyd, 2012).

2.2.16. Acceso básico de autenticación

En el entorno web el cliente adiciona la información del usuario en la cabecera de cada solicitud, que es, el username y el password. El problema de que esta información es solo codificada usando código de cifrado BASE64, pero no encriptado, lo que hace extremadamente fácil para el atacante decodificar la cabecera y obtener el password en texto plano, por lo que es recomendable usar HTTPS (Lopez, 2016).

2.2.17. OAuth 2.0

El marco de referencia de autorización OAuth 2.0 habilita a las aplicaciones de terceros, para tener un acceso limitado a un servicio HTTP, ya sea dado por el propietario del recurso mediante una interacción de aprobación entre el propietario del recurso y el servicio HTTP, o permitiendo a aplicaciones de terceros obtener acceso en su propio nombre (Hardt, 2012).

2.2.18. Introducción del protocolo OAuth

OAuth es un estándar de autorización emergente que fue adoptado por el creciente número de sitios web como Twitter, Facebook, Google, Yahoo!, Netflix, Flickr, y muchos otros proveedores de recursos y redes sociales. Tiene especificación open-web, para que, organizaciones puedan acceder a recursos protegidos entre distintos sitios web. Esto se logra dando permisos de acceso de contenido protegido a los usuarios en a las aplicaciones de terceros sin tener que proveer de credenciales a la aplicación (Kiani, 2016).

OAuth 2.0 define un marco de trabajo de acceso seguro de aplicación para recursos protegidos a través de Application Programming Interfaces (APIs) típicamente de RESTful. Existen tres participaciones primarias en el flujo de OAuth. Esto permite al cliente (una aplicación que solicita información) enviar una consulta API hacia un recurso de servidor (RS), la aplicación hospeda la información deseada, a lo que, el RS puede autentificar el mensaje que fue enviado por el cliente. El cliente autentifica el RS a través de la inclusión de un token de acceso en su mensaje API, un token previamente proveído para el cliente por el servidor de autorización (AS). En estos escenarios OAuth con la API protege el acceso hacia los atributos identificados, puede ser el caso de que el token de acceso solo podría ser emitido por el AS después de que el usuario haya dado explícitamente dado consentimiento para acceder a esos atributos (Ping Identity, 2012).

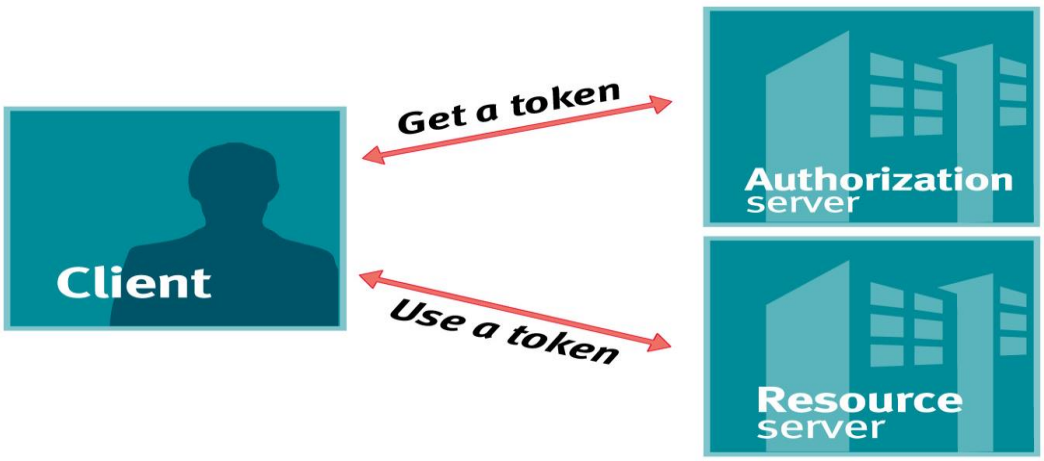

Figura 2-7. Token de acceso
Fuente: (Ping Identity, 2012)

OAuth 2.0 incluye:

1. Un mecanismo de redireccionamiento web con el cual un propietario del recurso pueda delegar autorizaciones de acceso a sus recursos (por ejemplo, su perfil) a algún sitio de terceros.

2. Otra variedad de mecanismos con los que el propietario de recursos puede delegar sus atributos de identidad mantenidos en algún sitio para aplicaciones de cliente como son computadoras de escritorio, celulares, y en el internet de las cosas (IoT).

3. Un modelo de servicio de token de seguridad restringido (STS) notablemente diseñado en torno a los principios REST en lugar de la mensajería SOAP. El STS admite la emisión de tokens y actualización.

4. Un conjunto de mecanismos para REST-based HTTP APIs, con las que se puede hacer:

    - Proteger los atributos de identidad de los propietarios de recursos para los cuales el propietario del recurso requiere consentimiento.

    - Proteger los atributos de identidad de los propietarios de recursos para lo cual el consentimiento del propietario del recurso está implícito.

    - Proteger los datos específicos del propietario del recurso, por lo tanto, no se requiere consentimiento (Ping Identity, 2012).

2.2.19. Roles de un modelo QAuth 2.0

2.2.19.1. Servidor de Autorización

Actor que emite tokens de acceso y tokens de actualización a los clientes en nombre de los servidores de recursos.

2.2.19.2. Token de acceso

El token de acceso es una cadena cifrada que se utiliza como credencial para acceder a recursos restringidos. El estándar que utiliza el token de acceso es Json Web Token (JWT) que tiene como formato una cadena de tres strings separados por un punto, cada string representa header, payload y signature. El token es generado en el lado del Servidor de Acceso y almacenado en el lado del cliente.

2.2.19.3. Cliente

Es la aplicación que desea acceder a un recurso protegido por el servidor de recursos e interactúa con un servidor de autorización para obtener tokens de acceso.

2.2.19.4. Token de actualización

Token de larga duración que el cliente puede intercambiar con el servidor de autorización para obtener un nuevo token de acceso con las mismas autorizaciones adjuntas. Estos permiten obtener tokens de acceso nuevos sin cambiar las autorizaciones del propietario del recurso.

2.2.19.5. Servidor de recursos

Actor encargado de proteger los recursos y los pone a disposición de los clientes autenticados y autorizados.

2.2.19.6. Propietario del recurso

Es un actor típicamente humano que controla el acceso a clientes de un recurso en particular hospedado en el servidor de recursos. Un propietario de recurso especifica qué autorizaciones se deben hacer al servidor de autorización. Las autorizaciones luego se manifiestan en un token de acceso emitido al cliente en cuestión (Ping Identity, 2012).

2.2.20. Flujo del protocolo abstracto QAuth 2.0

Figura 2-8. Flujo de protocolo
Fuente: (DigitalOcean, 2020)

El flujo abstracto de OAuth 2.0 describe la interacción entre los roles e incluye los siguientes pasos:

(1) La aplicación hace una solicitud de autorización para acceder al recurso del propietario. La autorización puede hacerse directamente al propietario del recurso, o solicitar la autorización al Servidor de Recursos.

(2) La aplicación obtiene la autorización de acceso con las credenciales que identifican al propietario del recurso. El alcance permitido para generar la autorización de acceso a los recursos va a depender del modelo que utilice el cliente y los tipos permitidos por el Servidor de Autorizaciones.

(3) La aplicación solicita un token de acceso autenticándose con el servidor de autorización y presentando la autorización de concesión.

(4) Una vez autenticada y validada la aplicación cliente por parte del Servidor de Autorización, este generará un token de acceso.

(5) La aplicación cliente hace una petición de acceso del recurso protegido al Servidor de Recursos adjuntando el token de acceso.

(6) Una vez validado el token de acceso por el Servidor de Recursos, se permite el acceso al recurso protegido.

El flujo de este proceso va a depender del tipo de autorización que esté en uso, este flujo es tomado como idea general (DigitalOcean, 2020).

2.2.21. Registro de la aplicación

Antes de usar OAuth, se debe registrar la aplicación con el Servidor de Autorización a través de un formulario de registro en la API del desarrollador del sitio web del servicio, en el que, se proporciona información como:

- El nombre de cada aplicación
- La dirección URI del sitio web de la aplicación
- Redireccionamiento de la aplicación

Se redireccionará al usuario una vez autorizada o negada la solicitud, para que la aplicación maneje los códigos de autorización o tokens de acceso (DigitalOcean, 2020).

Una vez registrada la aplicación, el servicio emitirá las credenciales a la aplicación cliente en forma de:

Client ID: especificado como client_id cuando interactúa con el servidor de recursos.

Client Secret: especificado como client_secret cuando intercambia un código de autorización para un token de acceso y un token de actualización de acceso usado en el lado del servidor del flujo web de la aplicación (Boyd, 2012).

2.2.22. Tipos de otorgamiento de la autorización

Dependen del método utilizado por la aplicación y de los tipos de autorización soportados por la API. OAuth 2.0 define cuatro tipos de autorización, dependiendo de cada caso:

- **Código de autorización:** este tipo utiliza aplicaciones en el Servidor de Acceso.
- **Implícito:** es utilizado en aplicaciones móviles, web, IoT del lado del usuario.
- **Permiso en base a contraseña del propietario del recurso:** este tipo de permiso es usado únicamente en clientes confiables.

- **Credenciales del cliente:** permite a una aplicación obtener un token de acceso para el propietario del recurso por el cliente. Este tipo de permiso es adecuado para aplicaciones que necesitan APIs de acceso como los servicios de almacenamiento o base de datos (DigitalOcean, 2020).

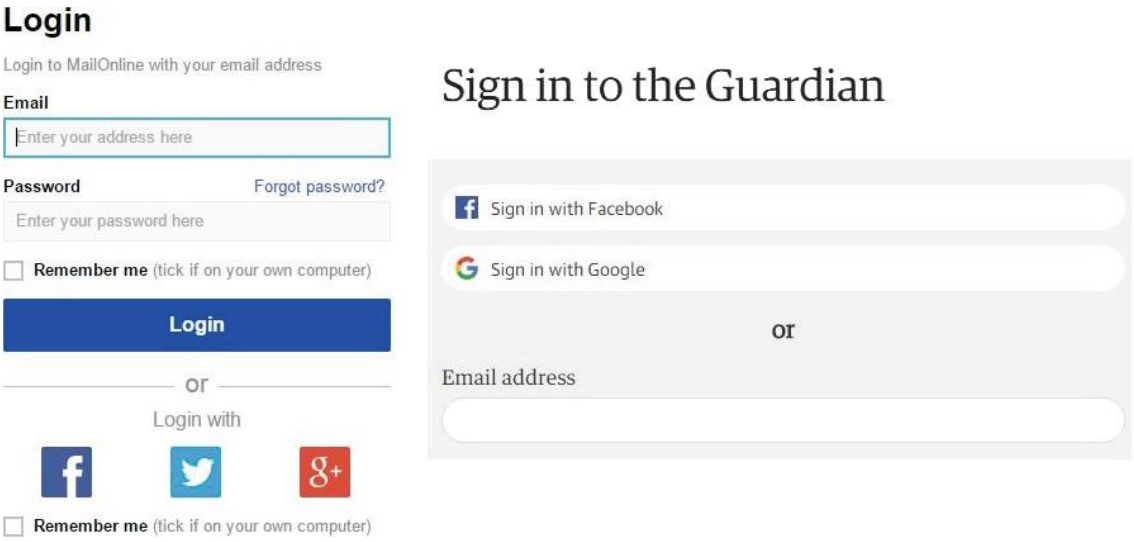

Figura 2-9. OAuth en uso
Fuente: (Argyriou, 2017)

2.2.23. Análisis de vulnerabilidades de OAuth 2.0

Muchos sitios web utilizan OAuth como proveedor de acceso a recursos externos protegidos, por mencionar algunos, Google, Facebook, Paypal, Amazon, etc. Cada sitio cuenta con varios probadores de identidad, los cuales, podrían tener errores de código aumentando las probabilidades de fallos en la seguridad de la comunicación.

2.2.23.1. Vulnerabilidades

Protocolo HTTP: la principal causa de vulnerabilidad de OAuth viene del protocolo HTTP subyacente debido a que no impone políticas estrictas de seguimiento a los métodos del protocolo HTTP, además, intenta evitar utilizar solicitudes HTTPs y utiliza el canal abierto.

Almacenamiento de credenciales: cuando el cliente es una aplicación de escritorio y las credenciales son almacenadas en el lado del cliente, en este caso, el responsable de la seguridad de las mismas es el ambiente de escritorio y del usuario. Esto lo hace susceptible a muchos tipos de ataques.

Invocación de servicios respaldados: la gran variedad de servicios y soluciones para diferentes tipos de uso, hace que los desarrolladores web sin un conocimiento profundo del tema implementen un sistema complejo o sobre dimensionado, lo que puede generar incuestionables agujeros de seguridad.

Paquetes sin soporte: OAuth proporciona librerías estándares, pero también existen librerías de terceros que se pueden utilizar en donde no se garantiza su fiabilidad y lo que es peor quedar obsoletas (Argyriou, 2017).

2.2.23.2. Ataques

Los siguientes ataques están basados en malas configuraciones de autenticación entre los propietarios del recurso (RO) y el proveedor de identidad (IdP). El código de autorización y los tipos de permisos implícitos son susceptibles a estos ataques.

HTTP 307 Redirect Attack: el objetivo de este ataque es aprender la credencial de un usuario legítimo. El atacante corre un servidor de recursos maliciosos y se basa en el mal uso del código de estado 307, utilizado por el servidor de recursos.

Ataque: el usuario trata de conectarse a un RO del atacante, luego el usuario es redirigido hacia un idP para autenticarse en una solicitud POST para enviar sus credenciales. El IdP verifica cual credencial es legítima y cuál es la réplica enviando un HTTP 307 redirect status. Como el código 307 es usado para redireccionamiento, el navegador del usuario ejecutará una solicitud POST al URI de redireccionamiento con el mismo cuerpo que, se envió anteriormente al IdP que aún contiene su credencial.

Corrección: cambiar el mensaje implementado en el servidor IdP, para eliminar el código de estado 307. Se recomienda el uso del mensaje de estado 303, porque es el único que descarta el cuerpo de una solicitud HTTP POST (Argyriou, 2017).

Cross Site Request Forgery (CSRF): en este tipo de amenaza el atacante toma ventaja de la sesión establecida entre el navegador del usuario y la OR. Luego de la autenticación exitosa a un IdP, el RO crea un mensaje local que contiene información sobre la cuenta del usuario y el IdP administrador de la cuenta.

Ataque: comienza cuando el atacante obliga al navegador del usuario a iniciar una solicitud al RO y el resultado deseado es hacer que el RO ejecute acciones sin involucrar al usuario. Se atrae a la víctima a visitar un sitio malicioso o seguir el enlace del sitio objetivo.

Corrección: crear funciones que producen valores de longitudes aleatorias para que el valor del estado no sea adivinable (Argyriou, 2017).

Mixing IdP Redirect End-Point: este ataque, se ejecuta en el modo de autorización implícita. El objetivo de este ataque es obtener el token de autorización y deslegitimar al usuario a través del redireccionamiento del RO a cualquier IdP.

Ataque: en este escenario, el ataque se produce tanto como RO y IdP. El usuario inicia la sesión a través de la petición POST para seleccionar en IdP veraz. El atacante intenta enviar y recibir mensajes con un IdP veraz y ganar acceso a recursos protegidos del usuario.

Corrección: en este ataque es posible debido a la pobre identificación de transmisión de valores. Este problema se resuelve agregando el código y estado transmitido al cuerpo de los paquetes de intercambio (Argyriou, 2017).

Session Wrapping Attack: el atacante manipula el RO para usar recursos protegidos. Este método es suplementario al ataque CSRF, cuando todas las verificaciones están hechas en el lugar, el atacante todavía puede encontrar los valores de código de estado.

Ataque: después de validar las credenciales del usuario, el IdP contiene el URI de los recursos protegidos que fueron solicitados en primer lugar. El atacante manipula esta URI con otro que pone a un malicioso sitio web para extraer de la cabecera los valores del estado y código.

Corrección: una posible solución a este ataque podría ser restringir las políticas de direccionamiento en particular la política de cruce de origen, especificando qué valores son enviados cuando se hace una petición en el mismo origen (Argyriou, 2017).

Brute Force Attack Against the IdP: el atacante es parte de la red y está habilitado para poder contaminar el tráfico y confundir al servidor IdP. Para hacerlo, el atacante utiliza el protocolo HTTP y no HTTPs.

Ataque: inspeccionando el intercambio de tráfico, el atacante puede encontrar parámetros de peticiones en el intercambio entre dos entidades, lanzando ataques de fuerza bruta en contra del servidor IdP.

Corrección: para evitar ataques de fuerza bruta, se debe sustituir el uso del protocolo HTTP con HTTPS y con la creación de tokens extensos y al azar (Argyriou, 2017).

CAPÍTULO III

3. DISEÑO DE LA INVESTIGACIÓN

3.1. Tipo y diseño de la investigación

El tipo de la investigación será investigativa experimental en base a las características definidas en el estudio, con la finalidad de monitorear el control de acceso a recursos web para plataformas clouding a través del protocolo OAuth 2.0.

El diseño tendrá diferentes métodos y técnicas con bases teóricas y resultados medibles.

3.2. Métodos y técnicas de investigación

Para el desarrollo de la investigación se van a utilizar los siguientes métodos:

- **Científico:** la Formulación del problema, recopilar datos, probar la hipótesis y concluir, con instrumentos que nos permiten realizar la presente investigación.

- **Experimental:** la presente investigación monitorea el control de acceso a recursos web en un ambiente controlado con el objetivo de evaluar y examinar los efectos que se manifiestan al implementar el protocolo OAuth 2.0, es decir observar sus consecuencias, en este caso el mejoramiento del control de acceso.

- **Analítico:** se realizará un análisis de los controles de acceso existentes, el nivel de seguridad, su funcionalidad y características para determinar el punto de inicio de la investigación.

- **Inductivo:** a partir de los controles de acceso existentes, se implementará el nuevo modelo basado en el protocolo de autorización OAuth 2.0 que mejore el control de acceso de la información a recursos web.

- **Deductivo:** Se emplea en la investigación, a partir de la observación en diferentes casos de uso, se formula entonces la correspondiente hipótesis y posteriormente generamos los conocimientos previos del tema de estudio para obtener conclusiones que luego serán verificadas mediante la experimentación.

3.3. Enfoque de la investigación

El estudio considera un enfoque cualitativo debido a que se verificará el nivel de mejora en el control acceso a recursos web en un entorno web en la nube.

3.4. Alcance de la investigación

El alcance de la investigación es de tipo descriptivo y está en relación al grado de control de acceso a recursos y la seguridad que brindan los estándares del protocolo de autorización definidos, en los que se puede observar los elementos más importantes y así obtener datos de los casos de estudio.

3.5. Población de estudio

Se considera que la población para el estudio son los accesos a recursos de servicios web en un entorno computacional en la nube dentro de un ambiente de pruebas.

3.6. Unidad de análisis

La unidad de análisis serán los controles de acceso implementados como son: roles, flujos de comunicación de protocolo, autorización de permisos, token de acceso, versión TLS, redirecciones HTTP.

3.7. Selección de la muestra

La selección de la muestra se tomará de forma aleatoria de la población en estudio en un entorno computacional en la nube dentro de un ambiente de pruebas.

3.8. Tamaño de la muestra

La selección se obtendrá mediante la fórmula estadística que determina su tamaño en consideración al parámetro de estudio y el tipo de población.

$$n = \frac{z^2 * p * q}{e^2};$$

En dónde:

e: Error o Precisión => (0.05)
n: Tamaño de la Muestra
z: Nivel de Confianza => (1.96)
p: Variabilidad Positiva => (0.5)

q: Variabilidad Negativa => (0.5)

$$n = \frac{z^2 * p * q}{e^2}$$

$$\text{n} = \frac{1.96^2 * 0.5 * 0.5}{0.05^2}$$

$$n = \frac{0.9604}{0.0025} = 384{,}16$$

n=384.16 por lo tanto el tamaño de la muestra es 385.

3.9. Técnicas de recolección de datos

Las siguientes técnicas se utilizarán en la actual investigación:

Primarias: información original obtenida de primera en la investigación mediante la lectura, revisión de documentos y observación en el ambiente de pruebas implementado con el fin de contrastar la hipótesis.

Secundarias:

- Libros especializados referentes al tema de estudio.
- Artículos publicados en revistas científicas.
- Trabajos de investigación publicados y afines al tema de investigación.
- Revistas científicas informáticas.
- Páginas de internet seguras con información confiable y especializada.

3.10. Instrumentos para la recolección de datos

En el protocolo de autorización OAuth 2.0 como la variable independiente, se utilizará el sistema de autorización planteado para evaluar los indicadores.

El control de acceso como como la variable dependiente, se usará un aplicativo web en una plataforma en la nube para probar el rendimiento de cada sistema de autorización.

3.11. Instrumentos para procesar datos recopilados

Los datos recolectados en la investigación serán procesados y representados gráficamente en el lenguaje de programación estadístico R.

3.12. Variables e indicadores

Variable Independiente. Protocolo de autorización OAuth 2.0.

Variable Dependiente. Control de acceso a recursos web.

3.13. Operacionalización conceptual de variables

Tabla 3-1. Operacionalización de variables

Variable	**Tipo**	**Definición**
Protocolo de autorización OAuth 2.0	Independiente	Procedimientos para el proceso de autorización en la cloud computing
Control de acceso a recursos web	Dependiente	Características esenciales para garantizar el control de acceso a recursos web.

Elaborado por: Guillermo Valencia, 2021

3.14. Operacionalización metodológica de variables

Tabla 3-2. Operacionalización metodológica de variables

Variable	**Indicadores**	**Técnica**	**Instrumento**
Protocolo de autorización OAuth 2.0	Autenticación Autorización Acceso Limitado	• Observación	Aplicación web en una plataforma en la nube
Control de acceso a recursos web	Mecanismos de autorización Roles y alcance Flujos de comunicación Cifrado del tráfico, encriptación Alertas y monitoreo Protección frente a amenazas	• Experimentación • Observación • Test de rendimiento • Análisis de rendimiento • Comprobación	RESTClient Firefox Comando Curl Linux Lenguajes de programación: PHP, R OWASP ZAP

Elaborado por: Guillermo Valencia, 2021

3.15. Procesamiento y análisis

Para el propósito de estudio de este trabajo investigativo, el marco para identificar el manejo de riesgos en recursos en la nube y describir servicios REST API consumidos por el cliente, se toman

de varios criterios y aspectos de seguridad basados en “National Institute of Standards and Technology” (NIST), que son:

- Controles de Seguridad.
- Autorización y Monitoreo.
- Categorización de la Seguridad (NIST, 2020).

3.16. Definición de las variables

Basado en las métricas para determinar el manejo de riesgos en recursos en la nube mencionados anteriormente se tiene:

Tabla 3-3. Variables para análisis de datos

Variable	**Unidad**	**Marco**	**Definición**
Control de flujos de comunicación	Tiempo tomado en la transmisión de datos(ms)	Controles de Seguridad	Tiempo de respuesta en flujo de transmisión de datos
Autenticación y autorización	Número de mecanismos	Autorización y Monitoreo	Mecanismos de autorización y autenticación
Roles y alcance	Disponibilidad y alcance	Categorización de la Seguridad	Tipos de accesos a los recursos y su alcance
Cifrado del tráfico y encriptación	Métodos utilizados	Controles de Seguridad	Tipo de cifrado y encriptación
Alertas y monitorización	Número de canales	Autorización y Monitoreo	Alertas permitidas por el Servicio en la nube
Protección frente a amenazas	Número de mecanismos	Controles de Seguridad	Medidas frente a amenazas y ataques

Elaborado por: Guillermo Valencia, 2021

3.17. Diseño de escenarios

Para el estudio de control de acceso a recursos compartidos, se diseñaron dos escenarios en plataformas clouding en los cuales, cada escenario debe solicitar un recurso al Servidor de Recursos Heroku (SRH) que hospeda a un Sistema de Inventario (SI) de tipo monolítico propietario del recurso. Este SI utiliza el motor de base de datos PostgreSQL ubicado en el servidor de base de datos de Amazon Web Service (AWS).

El recurso de este SI es un listado de productos de acceso solo lectura. El SRH es el escenario base para los dos escenarios propuestos.

3.17.1. Escenario base de Servidor de Recursos

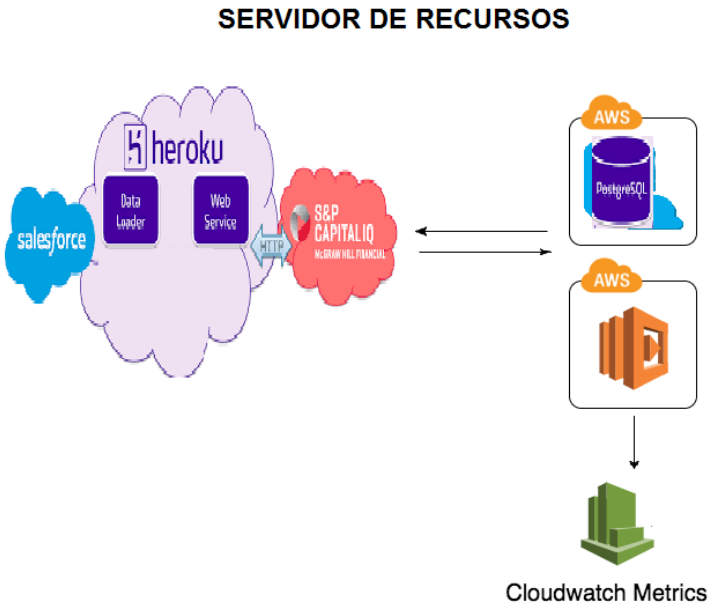

Figura 3-1. Servidor de Recursos en Heroku
Elaborado por: Guillermo Valencia, 2021

3.17.1.1. Heroku

La plataforma Heroku es de tipo Software as a Service (SAAS) y es utilizada como Servidor de recursos. Toma las peticiones del cliente y verifica sus credenciales para devolver el recurso solicitado. El servidor de recursos y su configuración de hardware y software es la siguiente:

- Description: Canonical, Ubuntu, 18.04 LTS, amd64 bionic image build on 2019-07-22
- Status: available
- Platform details: Linux/UNIX
- Platform: Ubuntu
- Image Size: 8GB
- Visibility: Public
- Network interfaces: eth0

Number of vCPUs: 1

3.17.1.2. Amazon AWS

Infraestructura cloud computing de tipo Infrastructure as a Service (IAAS) donde toma la función de servidor de base de datos, que utiliza el motor de base de datos Postgres para almacenar la información del sistema de inventario ubicado en la plataforma de Heroku. Para este caso utiliza la infraestructura AWS como Data as a Service (DAAS).

3.17.1.3. Motor de base de datos PostgreSQL

La gestión de bases de datos es realizada a través del motor de bases de datos relacionales y de código libre PostgreSQL. Éste permite almacenar la información del sistema de inventario. El recurso compartido se encuentra en la tabla producto.

3.17.1.4. Sistema de Inventario

Es un aplicativo web de manejo de inventario de productos, que es utilizado para realizar los accesos a recursos compartidos. El aplicativo web es un sistema de tipo monolítico desarrollado con el framework de alto rendimiento YII basado en componentes PHP, que toma el patrón de diseño del tipo Modelo Vista Controlador (MVC).

3.17.2. Escenario para análisis de acceso a recursos a través de autenticación básica

Este escenario, la aplicación local cliente hace una solicitud HTTP Request con autenticación básica de credenciales de usuario previamente creados en la base de datos de PostgreSQL, para solicitar el listado de productos con permisos de acceso solo de lectura del sistema de inventario ubicado en el servidor de recursos Heroku, como muestra la Figura 3-1.

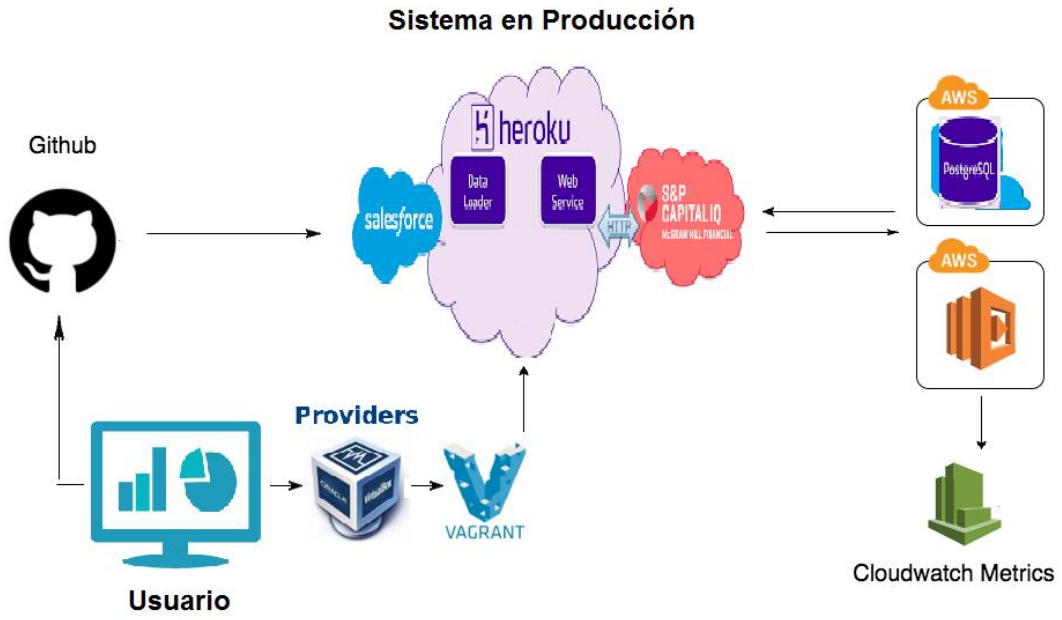

Figura 3-2. Recursos compartidos con autenticación básica
Elaborado por: Guillermo Valencia, 2021

3.17.2.1. Equipo cliente con Vagrant

El cliente utiliza la herramienta de entornos virtualizados Vagrant que usa como base de imagen a una instancia de VirtualBox y configura los recursos de software y hardware a través del archivo Vagrantfile como muestra en el anexo 1.

3.17.2.2. Aplicación cliente para petición de recursos

La aplicación local cliente utiliza el lenguaje de programación PHP y consta principalmente del archivo de clase Aws.php que toma las credenciales de usuario y hace la petición HTTP request al servidor de recursos mediante la utilización de librerías cURL Guzzle, el archivo test_basic.php que crean una instancia de la clase Aws.php y toman los 385 datos de muestra para el respectivo estudio. La implementación, se muestra en el anexo 2.

3.17.2.3. Repositorio de control de versiones GitHub

Utiliza el sistema de control de versiones Git para alojar la aplicación cliente y el sistema de inventario para el acceso a los recursos compartidos. El software que opera GitHub, está escrito en Ruby on Rails.

3.17.3. Escenario para análisis de acceso a recursos a través de autorización con token de acceso

En este segundo escenario la aplicación local cliente solicita un token de autorización de acceso al servidor de acceso de CEDIA a través de un HTTP Request con credenciales de cliente previamente creados en la base de datos del motor SQLite alojado en el mismo servidor de CEDIA, para solicitar el listado de productos con permisos de acceso solo de lectura del sistema de inventario ubicado en el servidor de recursos Heroku, como muestra la Figura 3-2.

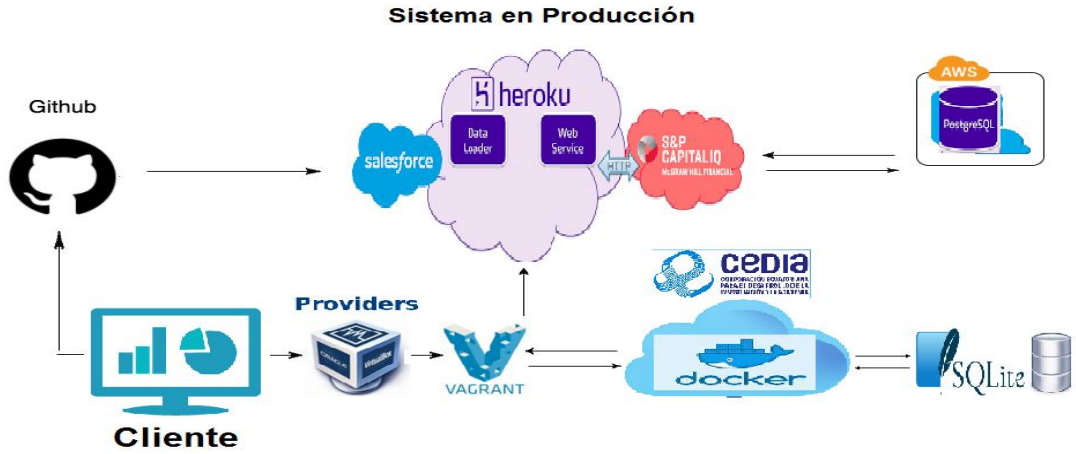

Figura 3-3. Recursos compartidos con token de acceso
Elaborado por: Guillermo Valencia, 2021

3.17.3.1. Equipo cliente con Vagrant

El cliente utiliza la herramienta de entornos virtualizados Vagrant que usa como base de imagen a una instancia de VirtualBox y configura los recursos de software y hardware a través del archivo Vagrantfile como muestra en el anexo 1.

3.17.3.2. Aplicación cliente para petición de recursos

La aplicación local cliente utiliza el lenguaje de programación PHP y consta principalmente del archivo de clase Aws.php que ejecuta dos peticiones:

- Toma las credenciales del cliente (aplicación) y hace la petición HTTP request de solicitud de token al servidor de acceso mediante la utilización de librerías cURL Guzzle
- Con el token de acceso hace otra solicitud HTTP request al servidor de recursos de Heroku para acceder al listado de productos solo de lectura.

El archivo test_oauth.php que crea 385 instancias de la clase Aws.php de muestra para el respectivo estudio. La implementación, se muestra en el anexo 2.

3.17.3.3. CEDIA

Infraestructura cloud computing de tipo: Platform as a Service (PAAS) donde se aloja el servidor de acceso. La máquina virtual tiene la siguiente configuración:

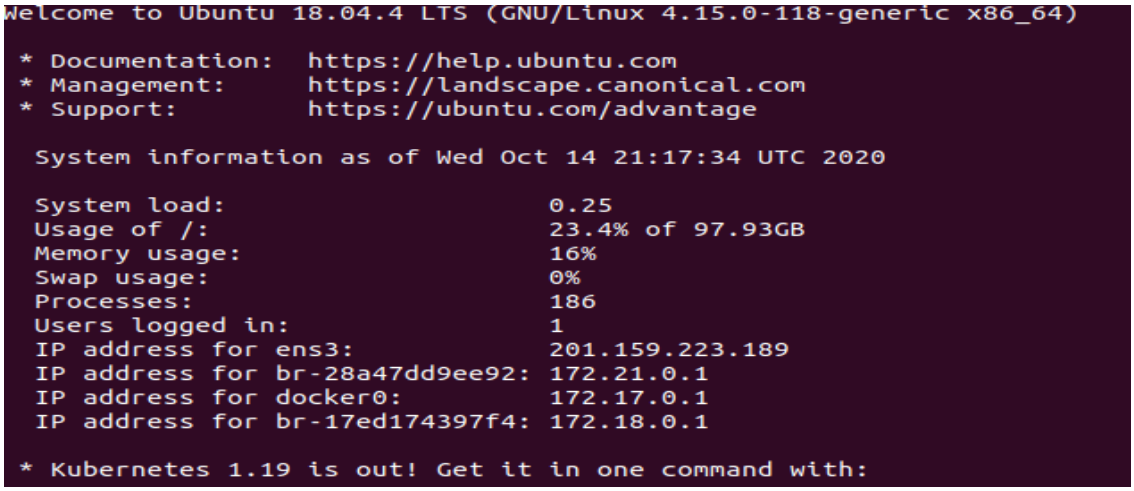

Figura 3-4. Recursos de CEDIA
Elaborado por: Guillermo Valencia, 2021

3.17.3.4. Docker

Contenedor de software de código abierto que proporciona una capa adicional de abstracción y automatización, donde se ejecuta el servidor de acceso que otorga un token de acceso de tipo JWT a la aplicación cliente. La configuración, se realiza a través del archivo doker-compose.yml como se muestra a continuación:

```
version: '3.7'

volumes:
    logs:
        driver: local

services:
    slim:
        image: php:7-alpine
        working_dir: /var/www
        command: php -S 0.0.0.0:8080 -t public
```

```
environment:
  docker: "true"
ports:
  - 8080:8080
volumes:
  - .:/var/www
  - logs:/var/www/logs
```

3.17.3.5. Motor de base de datos SQLite

Es un sistema de gestión de base de datos relacional compatible con ACID y de dominio público que, se utiliza para almacenar la información del sistema de control de acceso. La configuración de la base de datos muestra el anexo 3.

3.17.3.6. Repositorio de control de versiones GitHub

Utiliza el sistema de control de versiones Git para alojar la aplicación cliente y el sistema de inventario para el acceso a los recursos compartidos. El software que opera GitHub, está escrito en Ruby on Rails.

3.18. Plan de pruebas para acceso a recursos compartidos

El plan de pruebas implementa el modelo básico de intercambio de recursos para medir el rendimiento, el alcance y verificar la seguridad implementada en cada escenario.

3.18.1. Medir el rendimiento y el alcance con la aplicación cliente

La aplicación cliente está diseñada para que tome 385 muestras del intercambio de peticiones y respuestas realizadas al servidor de recursos. La configuración, se muestra en el anexo 2.

3.18.2. Analizar las vulnerabilidades y ethical hacking

El análisis, se realiza a través del proyecto Open Web Application Security Project (OWASP) con el uso de la herramienta de análisis de vulnerabilidades de seguridad de software web OWASP ZAP.

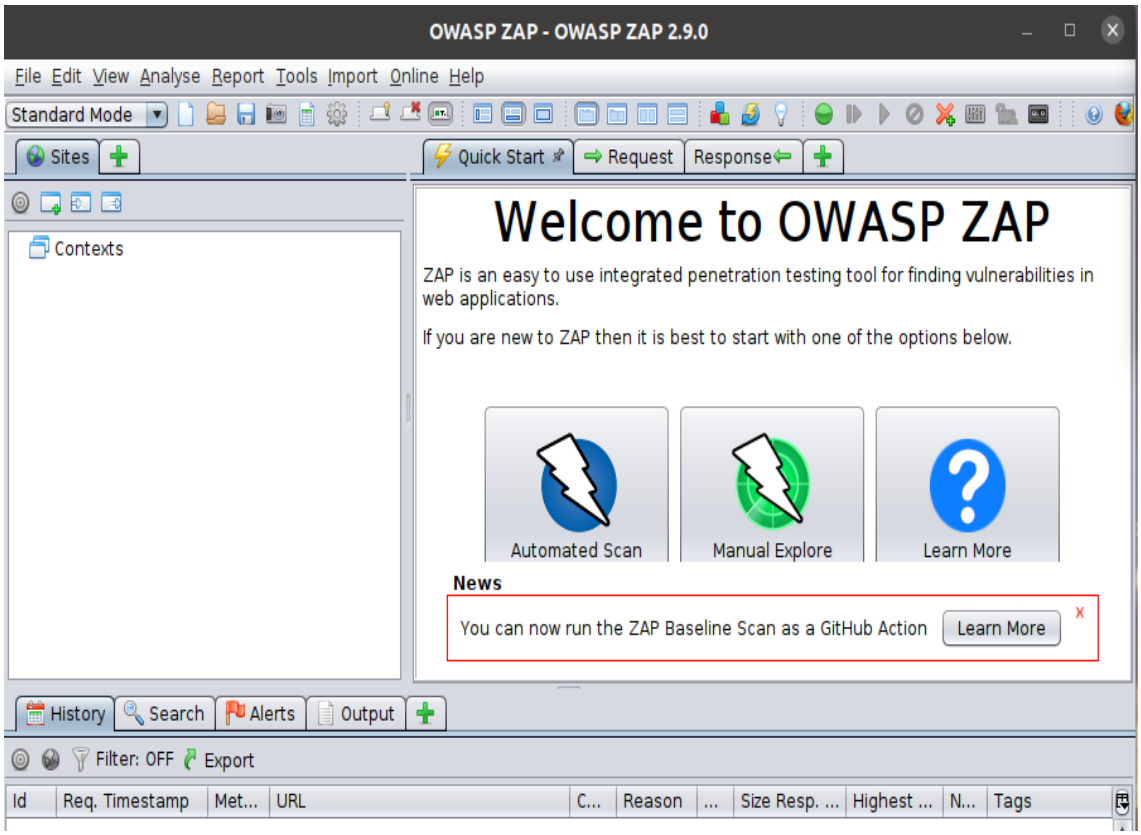

Figura 3-5. Owasp zap
Elaborado por: Guillermo Valencia, 2021

Para configuración, se selecciona archivo y se selecciona nueva sesión y en el cuadro de diálogo seleccionar sesión no persistente para este caso.

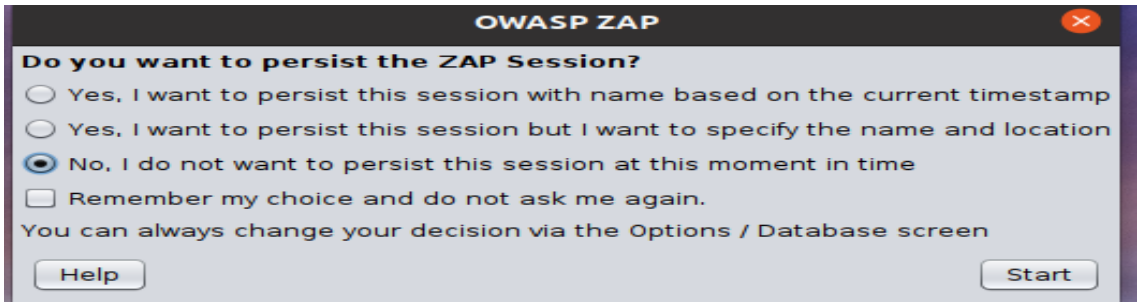

Figura 3-6. Owasp zap configuración
Elaborado por: Guillermo Valencia, 2021

Luego seleccionar autoscan para este caso y en la opción de URL to attack ingresar la dirección del endpoint, el resto de opciones, se deja por defecto para después presionar el botón Attack.

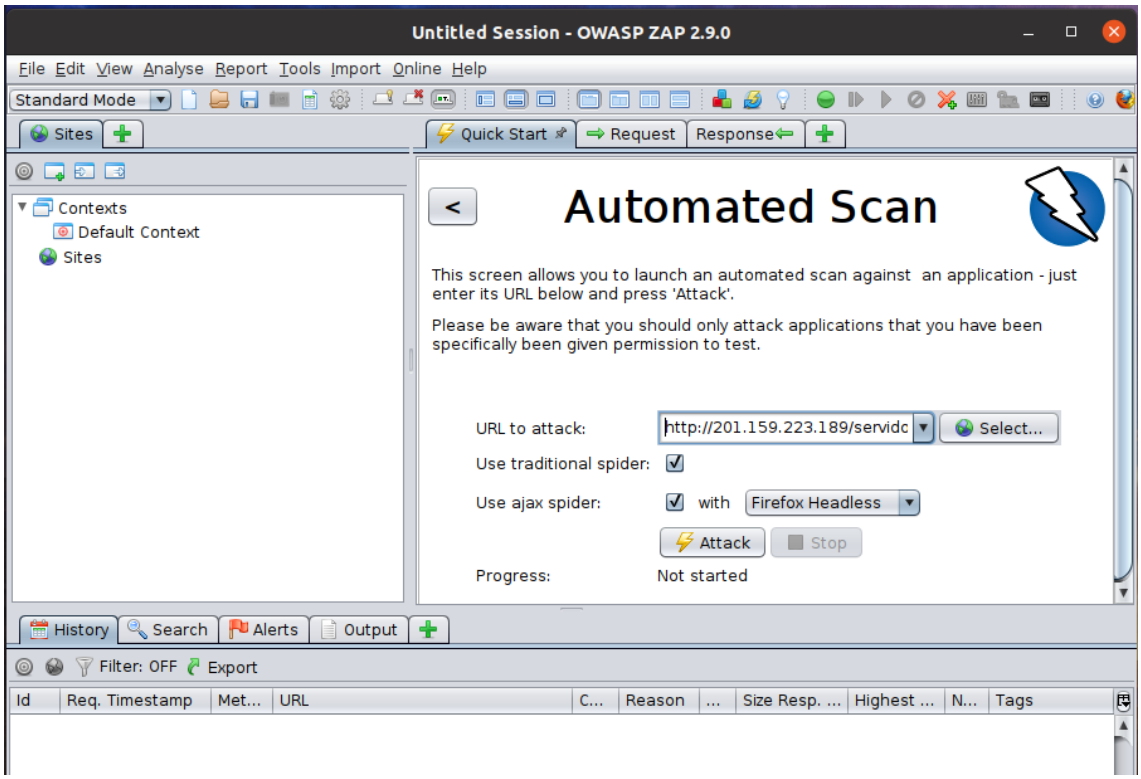

Figura 3-7. Owasp zap autoscan
Elaborado por: Guillermo Valencia, 2021

3.19. Plan para el análisis de las variables

Para el cumplimiento de las bases de la Seguridad de la Información cuyos pilares son Confidencialidad, Integridad y Disponibilidad de la información (CID), se procede a realizar el plan de pruebas como se menciona a continuación:

- Identificar técnicas de autorización y autenticación
- Determinar el tiempo de respuesta en la transmisión de datos en su aporte a la disponibilidad del servicio.
- Verificar el uso de roles en los recursos y sus alcances.
- Indicar métodos de cifrado de tráfico y encriptación para mantener la integridad de los datos.
- Conocer el uso de alertas y monitoreo.
- Analizar las vulnerabilidades y ethical hacking.
- Identificar la disponibilidad de mecanismos de protección frente a amenazas y su aporte a la Seguridad de datos.

3.19.1. Implementación del código para el análisis de las variables en R

Para el análisis de las variables primero, se procede a cargar el archivo de datos de las muestras obtenidas en cada escenario. El código de programación del lenguaje estadístico R, su configuración y ejecución en línea de comandos, se muestra a continuación:

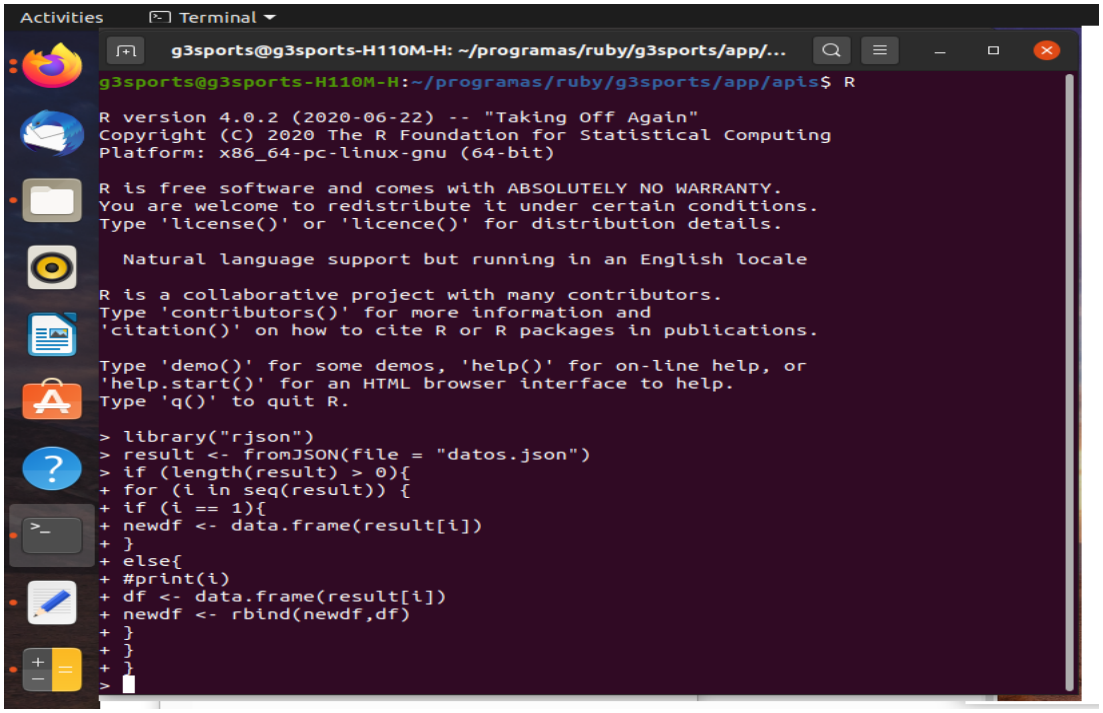

Figura 3-8. Creación del marco de datos
Elaborado por: Guillermo Valencia, 2021

Una vez creado el marco de datos para el análisis, se selecciona la variable de estudio y se realiza las operaciones estadísticas como se muestra en el código R a continuación:

```
> # Obtener el vector tiempo del data frame
> tiempo <- newdf$Request_time
> #tiempo
>
> #tomar cinco decimales y cambiar a dato numérico
> options(digits=5)
> t <- as.double(tiempo)
>
>
> #Histograma
> #density(t)
> plot(density(t))
> hist(t, freq = F, main="Histograma Autorización por Token", xlab = "Tiempo res
puesta", ylab = "Frecuencia")
> lines(density(t), col = "red", lty = 2, lwd = 3)
>
>
> # Uso de plot para visualizar la normalidad
> # QQ-plots: Quantile-Quantile plots - R Base Graphs
> #qqnorm(t)
> qqnorm(t, main = "Normal Q-Q Plot Autorización por Token ",
+         xlab = "Theoretical Quantiles", ylab = "Sample Quantiles")
>
> qqline(t, col = "red")
>
> # Kolmogorov-Smirnov test
> #ks.test(t, "pnorm", mean=mean(t), sd=sd(t))
> # Shapiro test
> #shapiro.test(t)
>
> # mediantiempo
> mean(t)
```

Figura 3-9. Marco de datos de análisis
Elaborado por: Guillermo Valencia, 2021

CAPÍTULO IV

4. RESULTADOS Y DISCUSIÓN

4.1. Presentación de resultados

Los resultados obtenidos en la presente investigación, se discuten de acuerdo al plan de pruebas realizado a cada escenario para realizar las comparativas en relación a los objetivos y la hipótesis planteada. El presente estudio hace referencia a las pruebas realizadas en el intercambio de mensajes para acceder a los recursos web utilizando las herramientas propuestas para cada caso.

4.1.1. Tiempo de respuesta

Se desea comprobar la normal distribución de los datos en el indicador que hace referencia al Tiempo de respuesta. Según el teorema de límite central, sin importar el contenido de la distribución, la distribución de la muestra tiende a ser normal si el tamaño de muestra es lo suficientemente grande (n>30) (Sheldon, 2009). Se puede ignorar la distribución de los datos y usar un test paramétrico, sin embargo, para ser consistente la normalidad, se hace una inspección visual con un plot de densidad y un plot de cuantil con el uso de herramientas de gráficas ggplot y qqplot de R.

4.1.1.1. Escenario con autenticación básica

Los datos de la muestra para este escenario se presentan en el anexo 4

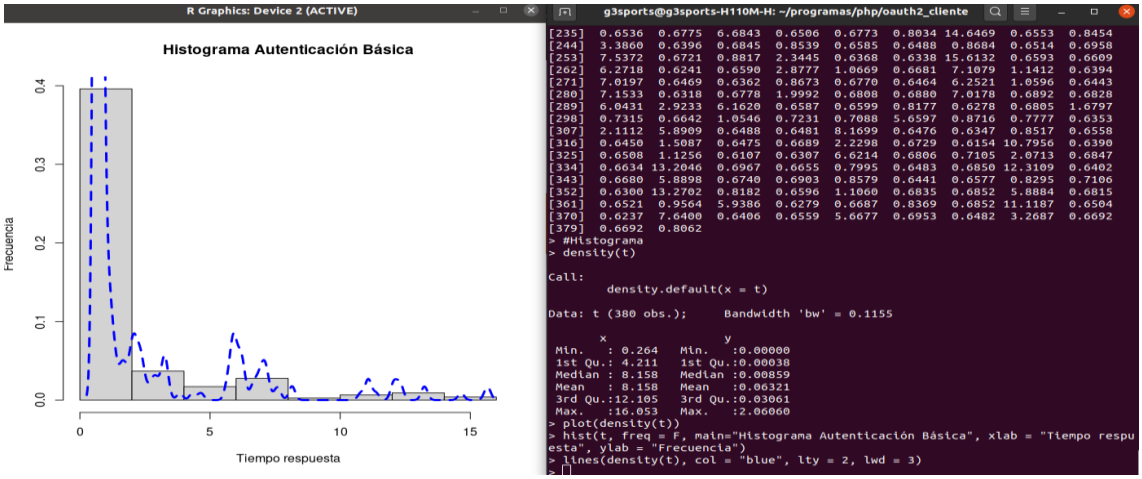

Figura 4-1. Histograma de densidad en función del tiempo
Elaborado por: Guillermo Valencia, 2021

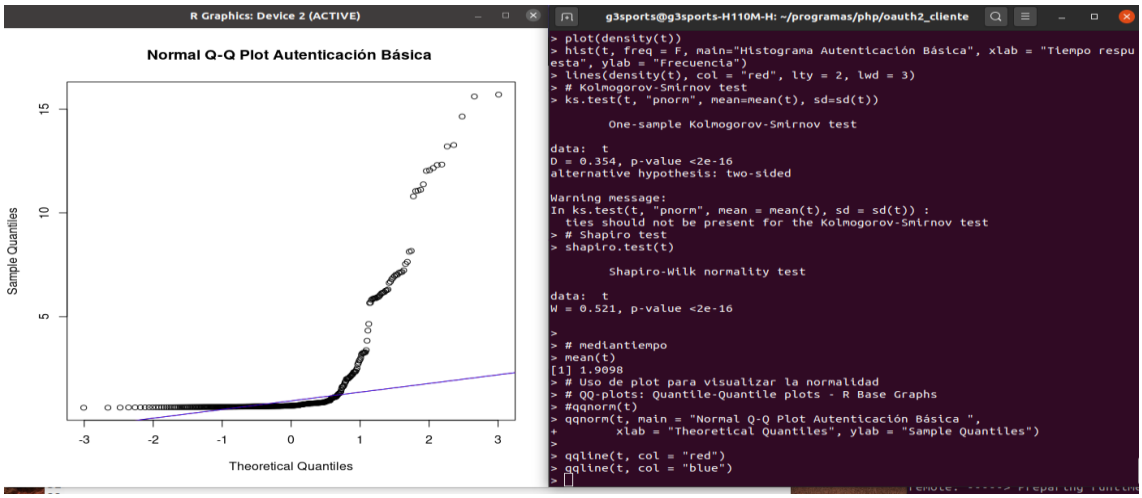

Figura 4-2. Cuantil en función del tiempo
Elaborado por: Guillermo Valencia, 2021

Es posible el uso de un test de significancia comparando la distribución de la muestra con una normal para determinar si los datos presentan una seria desviación de la normalidad. Existen varios métodos para realizar test de normalidad como por ejemplo Kolmogorov-Smirnov (K-S) normality y Shapiro-Wilk's.

Shapiro-Wilk's es el método más ampliamente usado para test de normalidad, y provee de mejores resultados que K-S Este se basa en la correlación entre los datos y sus resultados normales (STHDA, 2020).

```
> # Kolmogorov-Smirnov test
> ks.test(t, "pnorm", mean=mean(t), sd=sd(t))

        One-sample Kolmogorov-Smirnov test

data:  t
D = 0.354, p-value <2e-16
alternative hypothesis: two-sided

Warning message:
In ks.test(t, "pnorm", mean = mean(t), sd = sd(t)) :
  ties should not be present for the Kolmogorov-Smirnov test
> # Shapiro test
> shapiro.test(t)

        Shapiro-Wilk normality test

data:  t
W = 0.521, p-value <2e-16
```

Figura 4-3. Test Shapiro y Ks
Elaborado por: Guillermo Valencia, 2021

De acuerdo a los resultados p-value < 0.05, lo que implica que los datos de la muestra tienen una significancia de la distribución normal, es decir se asume que los datos no tienen una distribución normal.

4.1.1.2. Escenario con autorización por token

Los datos de la muestra para este escenario se presentan en el anexo 5

Figura 4-4. Histograma de densidad en función del tiempo
Elaborado por: Guillermo Valencia, 2021

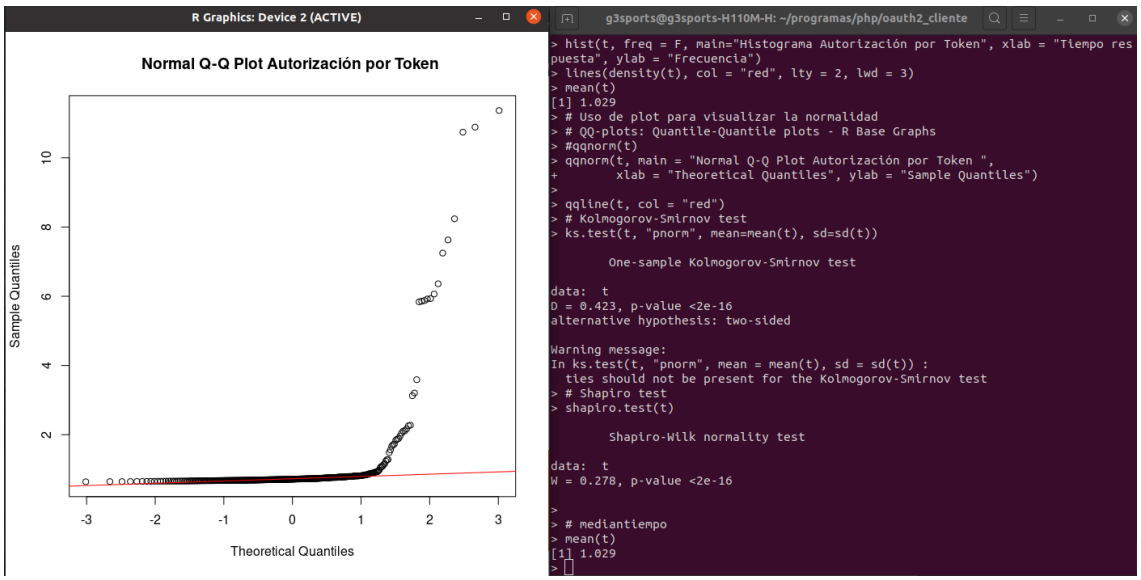

Figura 4-5. Cuantil en función del tiempo
Elaborado por: Guillermo Valencia, 2021

Es posible el uso de un test de significancia comparando la distribución de la muestra con una normal para determinar si los datos presentan una seria desviación de la normalidad. Existen varios métodos para realizar test de normalidad como por ejemplo Kolmogorov-Smirnov (K-S) normality y Shapiro-Wilk's.

Shapiro-Wilk's es el método más ampliamente usado para test de normalidad, y provee de mejores resultados que K-S Este se basa en la correlación entre los datos y sus resultados normales (STHDA, 2020).

```
> # Kolmogorov-Smirnov test
> ks.test(t, "pnorm", mean=mean(t), sd=sd(t))

        One-sample Kolmogorov-Smirnov test

data:  t
D = 0.423, p-value <2e-16
alternative hypothesis: two-sided

Warning message:
In ks.test(t, "pnorm", mean = mean(t), sd = sd(t)) :
  ties should not be present for the Kolmogorov-Smirnov test
> # Shapiro test
> shapiro.test(t)

        Shapiro-Wilk normality test

data:  t
W = 0.278, p-value <2e-16
```

Figura 4-6. Test Shapiro AWS
Elaborado por: Guillermo Valencia, 2021

De acuerdo a los resultados p-value < 0.05, lo que implica que los datos de la muestra tienen una significancia de la distribución normal, es decir se asume que los datos no tienen una distribución normal.

Debido a que ningún modelo cumple con una distribución normal, se procede a crear una comparativa en función del tiempo.

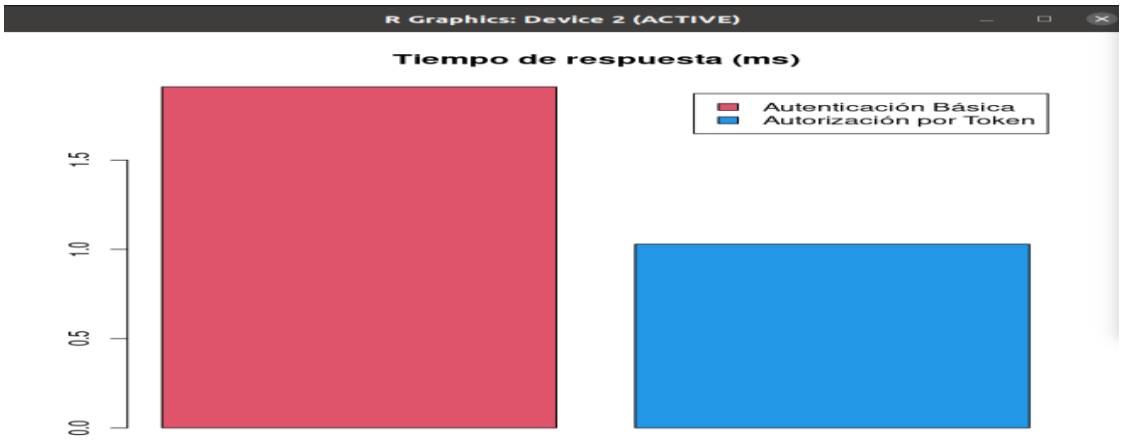

Figura 4-7. Comparativa en función del tiempo
Elaborado por: Guillermo Valencia, 2021

Tabla 4-1. Tabla de tiempo de respuesta

No.	Escenario con autenticación básica	Escenario con autorización por token
Tiempo de respuesta (ms)	**1.9098**	**1.0290**

Actualmente Google considera que el tiempo de respuesta del servidor web debe estar por debajo de 200 milisegundos, sin embargo 500 milisegundos en el tiempo de respuesta es aceptable (Google Developers, 2020).

Para medir este indicador se utiliza la escala de Likert con valores promedio de los resultados obtenidos en los escenarios.

Tabla 4-2. Tabla de escala de valores

Calidad de transmisión en milisegundos	Código
< =500	4
<= 1000	3
<= 2000	2
> 3000	1

Tabla 4-3. Tabla de ponderación tiempo

No.	Escenario con autenticación básica	Escenario con autorización por token
Ponderación	**2**	**3**

4.1.2. Autenticación y autorización

Para acceder a los recursos compartidos, se ha creado una aplicación REST API para cada escenario, que permita establecer la conexión y el intercambio de mensajes a través de métodos de autenticación y autorización. A continuación, se analiza cada porción de código implementado en cada aplicación para determinar el tipo de acción realizada.

Tabla 4-4. Tabla de autenticación y autorización básica

Código	**Acción**
$user = 'user_'.$i; $pass = 'pass_'.$i; $tempArray = []; $aws = new Aws($user, $pass);	Solicita usuario/password en el sistema de inventario.

$session = Yii::$app->session;
 	
// the following code will NOT work
$session['captcha']['number'] = 5;
$session['captcha']['lifetime'] = 3600;
 	
// the following code works:
$session['captcha'] = [
'number' => 5,
'lifetime' => 3600,
];
 	
// the following code also works:
echo $session['captcha']['lifetime'];
	Inicio de sesión crea idSession

Tabla 4-5. Tabla de autenticación y autorización por token

Código	**Acción**
$path = $_SERVER['HOME'] . '/.aws_php7.json'; $jsonCredentials = file_get_contents($path); $credentials = json_decode($jsonCredentials, true); $aws = new Aws($credentials['key'], $credentials['secret']);	Carga el archivo json con las credenciales proporcionadas al registrar la aplicación(idcliente/secret)
private function requestAccessToken() { $encodedString = base64_encode(	Si la autenticación y las credenciales son correctas, obtiene el token de acceso JWT

$this->key . ':' . $this->secret); $headers = ['Authorization' => 'Basic ' . $encodedString, 'Content-Type' => 'application/x-www-form-urlencoded;charset=UTF-8']; $options = ['headers' => $headers, 'body' => 'grant_type=client_credentials']; $response = $this->client->post(self::AWS_API_BASE_URI, $options);	

public function obtenerUsuarios(string $nombre, int $cont): array {
$options = [
'headers' => $this->getAccessTokenHeaders(),
];
try {
$response = $this->clientApi->get(self::GET_PRODUCTOS, $options);
	Envía el token JWT de acceso al servidor recursos donde verifica y si es válido realiza la solicitud.

Tabla 4-6. Tabla de mecanismos utilizados en autenticación y autorización

No.	**Escenario con autenticación básica**	**Escenario con autorización por token**
	• Autenticación básica (usuario/password) • Id sesión cookie token	• base64_encode (aplicación/Secret) • Token de acceso JWT • JWT con alcance y caducidad
Total mecanismos	**2**	**3**

Para medir este indicador se utiliza la escala de Likert con valores promedio de los resultados obtenidos en los escenarios.

Tabla 4-7. Tabla de escala de valores

No. Mecanismos	**Código**
>= 4	4
3	3

2	2
1	1

Tabla 4-8. Tabla de ponderación autenticación autorización

No.	**Escenario con autenticación básica**	**Escenario con autorización por token**
Ponderación	**2**	**3**

4.1.3. Roles y alcance de acceso a recursos compartidos

Luego del proceso de autenticación correcta de los distintos usuarios, comienza el proceso de autorización mediante la utilización de roles de cada método de autorización implementado.

Tabla 4-9. Tabla de roles y alcance con autenticación básica

Código	**Acción**
public function behaviors() { return ['access' => ['class' => AccessControl::className(), 'only' => ['delete','update', 'create', 'index', 'view'], #'only' => ['delete','update', 'create', 'view'], 'rules' => [[//'actions' => ['delete','update', 'create'], 'allow' => true, 'roles' => ['@'],],],],	Rol básico creado en el controlador

Tabla 4-10. Tabla de roles y alcance con autorización por token

Código	**Acción**
\League\OAuth2\Server\Entities\AccessTokenEntityInterface persistNewAccessToken() : void	Rol basado en token JWT
\League\OAuth2\Server\Entities\AccessTokenEntityInterface getIdentifier() : string	Genera randómicamente un único identificador
\League\OAuth2\Server\Entities\AccessTokenEntityInterface getExpiryDateTime() : \DateTime	Tiempo de expiración de Token
\League\OAuth2\Server\Entities\AccessTokenEntityInterface	Alcance de acceso a recurso lectura/escritua

getScopes() : ScopeEntityInterface[]	

Tabla 4-11. Tabla de mecanismos utilizados en roles y alcance

No.	Escenario con autenticación básica	Escenario con autorización por token
	• Rol de acceso por Controlador	• Rol único basado en token de acceso OAuth2.0 • Rol basado en caducidad de token JWT • Rol basado en el alcance del token JWT
Total mecanismos	**1**	**3**

Para medir este indicador se utiliza la escala de Likert con valores promedio de los resultados obtenidos en los escenarios.

Tabla 4-12. Tabla de escala de valores

No. Mecanismos	Código
>= 4	4
3	3
2	2
1	1

Tabla 4-13. Tabla de ponderación roles y alcance

No.	Escenario con autenticación básica	Escenario con autorización por token
Ponderación	**1**	**3**

4.1.4. Cifrado de tráfico y encriptación

La valoración de cifrado y encriptación en función a la cantidad y calidad de mecanismos implementados en el intercambio de mensajes para cada uno de los escenarios.

Tabla 4-14. Tabla de intercambio de mensajes en autenticación básica

Método	Estado	Autenticación básica
GET -H "Authorization: Basic user:password, Content-Type: application/json" -IL -s -w 'Total: %{time_total}s\nEstado: %{http_code}\n\n' 'https:/laliquorstore.herokuapp.com/web/productos/lista'];	200 (OK)	Cifrado de código base64 de autorización básica

Tabla 4-15. Tabla de intercambio de mensajes con autenticación por token

Método	**Estado**	**Autenticación básica**
POST -H 'Authorization: Basic MjhiY29jMDIwaWU1cjB1bGxibm8xbjltcnQ6MXF0bWFkZG0xZGI2dW4zYmlrdnZyMmJyMWttMzBoNWFwdTBoNnJxdTE3MHRjZTEzcDQyZ==' -d "grant_type=client_credentials" -i http://201.159.223.189:8080/servidor	200 (OK)	Cifrado de código base64 de autorización básica
GET -H 'Authorization: Bearer krpCqqDu9LYwoc1NxYvyS0iZabBH1dbLLE1X7Pbx1vNSVJjWqCNI2QfuhWXg3nN_7bZQ-qLad80Epnu1-Xj2rPIQvmvY0DY9U1cO7qfMi8r1JsfDllF_I2XJHNGmXReu9GCRnOpnav1G_rWhZ08eSoW7v1CHjHUAC6gm-_9Wb3znk9YVyDVbKw8jVXNvlC2lcmEemMahNbDcO6W-unRCLu3r9Jo9MS7PsRq9mNcljfijtV2mHDHHV3yoCEGlhgrbt9rladNLhu1fvFhGBoXrgM2eRM_LdObPHzXl7QaiLbyLAj2U13JQMSte9nqtRYZKIEvfBc6KixX_4h-jyvAFCQ -i 'https://laliquorstore.herokuapp.com/web/index.php?r=producto%2Flista'	302 (redireccionamiento)	Cifrado de código de 32bits
GET / HTTP/1.1" 200 306314 1.4632 http://201.159.223.189:8080/servidor?access_type=offline&client_id=230060729598-ti5nndplskr1a8dubuatmmj5fi14dg8d.apps.googleusercontent.com&redirect_uri=http%3A%2F%2Flocalhost%3A4567%2Foauth2callback&response_type=code&scope	200 (OK)	Cifrado de estado y comunicación https

Tabla 4-16. Tabla de mecanismos utilizados en cifrado y encriptación

No.	**Escenario con autenticación básica**	**Escenario con autorización por token**
	• Base 64 encoded cifrado de credenciales de usuario	• Algoritmo SHA256 de cabecera • Base 64 encoded de payload • Firma HMACSHA256
Total mecanismos	**1**	**3**

Para medir este indicador se utiliza la escala de Likert con valores promedio de los resultados obtenidos en los escenarios.

Tabla 4-17. Tabla de escala de valores

No. Mecanismos	**Código**
>= 4	4
3	3
2	2

1	1

Tabla 4-18. Tabla de ponderación cifrado y tráfico

No.	**Escenario con autenticación básica**	**Escenario con autorización por token**
Ponderación	**1**	**3**

4.1.5. Alertas y monitoreo en recursos compartidos

Valoración de instrumentos de alertas y monitoreos implementados en los escenarios.

Tabla 4-19. Tabla de alertas y monitoreo con autenticación básica

Código	**Acción**

CREATE TRIGGER session_log_after_insert
AFTER INSERT ON sesion
BEGIN
INSERT INTO logs (new_id, id_usuario, DATETIME('NOW'));
END;
	Graba el acceso al sistema

Tabla 4-20. Tabla de alertas y monitoreo con autorización por token

Código	**Acción**
\League\OAuth2\Server\Entities\AccessTokenEntityInterface isAccessTokenRevoked() : Boolean	Verifica el estado del token en la base de datos
composer require slim/csrf // Add middleware to the application $app = new \Slim\App; $app->add(new \Slim\Csrf\Guard);	Evitar la falsificación de petición del sitio. CSRF
// uri = http// $app->response->redirect($app->urlFor('uri'), 303);	Redireccionar con Código de estado

La Tabla 4-10 muestra la presencia o no del indicador y el tipo de cifrado y encriptación en base a los datos procesados al ejecutar la tarea.

Tabla 4-21. Tabla de instrumentos utilizados en alertas y monitoreo

No.	**Escenario con autenticación básica**	**Escenario con autorización por token**
	• Trigger after new session SQL en base de datos	• Monitoreo de actividades recientes por token de acceso • Configuración de alertas y notificaciones En el middleware • Códigos de estado HTTP

Total mecanismos	**1**	**3**

Para medir este indicador se utiliza la escala de Likert con valores promedio de los resultados obtenidos en los escenarios.

Tabla 4-22. Tabla de escala de valores

No. Mecanismos	**Código**
>= 4	4
3	3
2	2
1	1

Tabla 4-23. Tabla de ponderación alertas y monitoreo

No.	**Escenario con autenticación básica**	**Escenario con autorización por token**
Ponderación	**1**	**3**

4.1.6. Vulnerabilidades y mecanismos de protección frente a amenazas

Esta tarea identifica los mecanismos de seguridad implementados en cada escenario en el intercambio de mensajes para acceder a recursos compartidos y hacer frente a las amenazas y vulnerabilidades más comunes presentes en aplicaciones web y específicas en el protocolo de comunicación HTTP el cual es usado por REST API. La Tabla 4-13 muestra el top 10 de las vulnerabilidades presentes en aplicaciones web según OWAST.

Tabla 4-24. Vulnerabilidades según OWAST

Vulnerabilidad detectada de acuerdo a OWASP Top 10 Web Application Security Risks 2017	**Código de vulnerabilidad**
Injection	A1
Broken Authentication	A2
Sensitive Data Exposure	A3
XML External Entities (XXE)	A4
Broken Access Control	A5
Security Misconfiguration	A6
Cross-Site Scripting (XSS)	A7
Insecure Deserialization.	A8

Using Components with Known Vulnerabilities	A9
Insufficient Logging&Monitoring	A10

4.1.6.1. Análisis de vulnerabilidades con autenticación básica

De acuerdo al análisis de vulnerabilidades y ethical hacking realizado en la herramienta OWASP ZAP, se toma el resumen de los resultados del informe del anexo 6.

Tabla 4-25. Vulnerabilidades autenticación básica OWAST ZAP

Risk Level	Number of Alerts
High	0
Medium	1
Low	5
Informational	2

En este análisis, se puede observar que el nivel de riesgo de las vulnerabilidades encontradas es bajo, pero, se debe tomar las debidas acciones para proteger los recursos.

El siguiente análisis hace uso del comando curl para identificar otras vulnerabilidades como, se muestra en la a continuación:

```
g3sports@g3sports-H110M-H:~$ curl -X GET -H 'Authorization: Basic dXNlcl8xOnBhc3
NfMQ==' -i 'https://laliquorstore.herokuapp.com/web/index.php?r=producto%2Flista
'
HTTP/1.1 200 OK
Connection: keep-alive
Date: Fri, 06 Nov 2020 20:15:50 GMT
Server: Apache
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Set-Cookie: PHPSESSID=usn499ig02qqpvtte3rbkdemu1sho8t6; path=/; HttpOnly
Transfer-Encoding: chunked
Content-Type: text/html; charset=UTF-8
Via: 1.1 vegur
```

Figura 4-8. Análisis por comando curl en el servidor de recursos
Elaborado por: Guillermo Valencia, 2021

La información muestra datos sensibles del servidor (Apache: PHP)

A continuación, se muestra un resumen de vulnerabilidades encontradas:

Tabla 4-26. Vulnerabilidades detectadas con autenticación básica

Vulnerabilidad detectada	Código de vulnerabilidad
Débil encriptación Base64_decode	A2
PHPSESSID puede suplantar la sesión	A5
Insuficiente log y monitoreo	A10
Muestra datos sensibles: Server: Apache, PHP	A3

4.1.6.2. Análisis de vulnerabilidades con autorización por token

De acuerdo al análisis de vulnerabilidades y ethical hacking realizado en la herramienta OWASP ZAP, se toma el resumen de los resultados del informe del anexo 7.

Tabla 4-27. Vulnerabilidades Autorización por token OWAST ZAP

Risk Level	Number of Alerts
High	0
Medium	1
Low	4
Informational	1

En este análisis, se puede observar que el nivel de riesgo de las vulnerabilidades encontradas es bajo, pero, se debe tomar las debidas acciones para proteger los recursos.

El siguiente análisis hace uso del comando curl para identificar otras vulnerabilidades como se muestra a continuación:

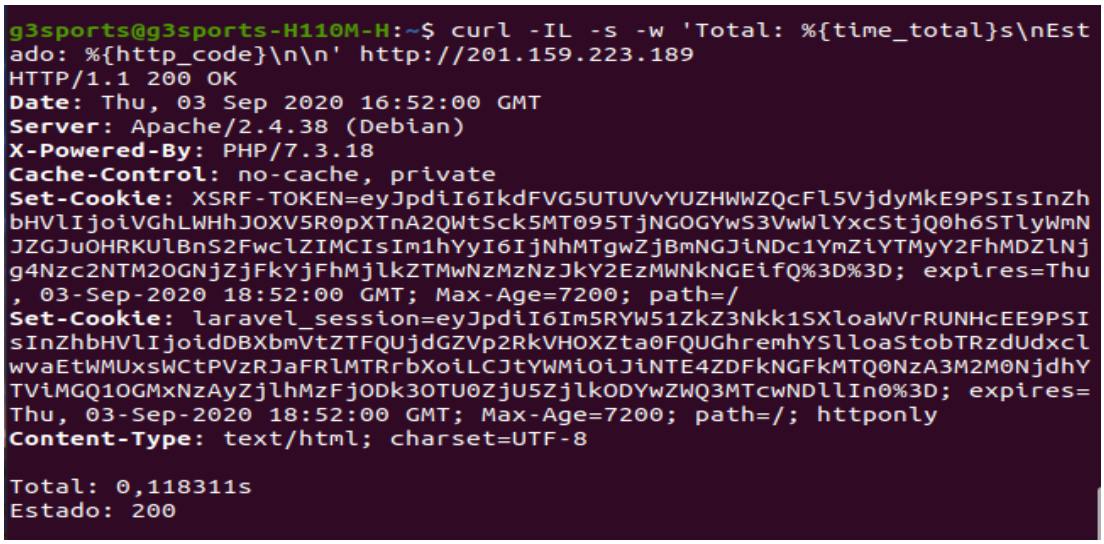

Figura 4-9. Análisis por comando curl en el servidor de acceso
Elaborado por: Guillermo Valencia, 2021

También se puede observar que no dispone de DNS y por consiguiente no se puede implementar la seguridad en la capa de transporte (TLS) y muestra demasiada información del servidor (Apache2.4: PHP7.3).

A continuación, se muestra un resumen de vulnerabilidades encontradas:

Tabla 4-28. Vulnerabilidades detectadas con autorización por token

Vulnerabilidad detectada	Código de vulnerabilidad
Man in the middle en HTTP	A2
Muestra datos sensibles: Server: Apache2.4., Powered-by:PHP7.3.18, OS:Debian	A3
No utiliza encriptación origen destino (TLS)	A5
Dirección IP en la URI	A6

Tabla 4-29. Tabla de mecanismos de seguridad implementados

No.	Escenario con autenticación básica	Escenario con autorización por token
	• A1 • A4 • A6 • A7 • A8 • A9	• A1 • A4 • A7 • A8 • A9 • A10
Total mecanismos observados	6	6

Para medir este indicador se utiliza la escala de Likert con valores promedio de los resultados obtenidos en los escenarios.

Tabla 4-30. Tabla de escala de valores

No. Mecanismos	**Código**
>= 7	4
4-6	3
3-4	2
1-2	1

Tabla 4-31. Tabla de ponderación vulnerabilidades

No.	**Escenario con autenticación básica**	**Escenario con autorización por token**
Ponderación	**3**	**3**

4.2. Análisis de interpretación

Los datos obtenidos en las mediciones anteriores, se representan en una escala nominal como muestra la matriz de impacto de la tabla 4-27.

Tabla 4-32. Tabla de matriz de impacto

No.	**Indicadores**	**Escenario con autenticación básica**	**Escenario con autorización por token**
1	Control de flujos de comunicación	2	3
2	Autenticación y autorización	2	3
3	Roles y alcance	1	3
4	Cifrado del tráfico y encriptación	1	3
5	Alertas y monitorización	1	3
6	Protección frente a amenazas	3	3
TOTAL		**10**	**18**

4.3. Comprobación estadística de la hipótesis

Para la prueba de la hipótesis planteada, en datos nominales, se utiliza el test de independencia Chi-cuadrado Test que considera hipótesis nula Ho y la hipótesis de investigación Hi.

- Hi: La evaluación de sistemas de autorización en plataformas clouding a través del protocolo OAuth 2.0 mejorará el control de acceso a recursos web.
- Ho: La evaluación de sistemas de autorización en plataformas clouding a través del protocolo OAuth 2.0 no mejorará el control de acceso a recursos web.

Para determinar si acepta la hipótesis, se toman los datos nominales obtenidos de los dos escenarios y, se ingresan en el software estadístico R para determinar los resultados.

```
> #Chi-cuadrado
> tabla<-matrix(c(2,3,2,3,1,3,1,3,1,3,3,3), nrow=2)
> colnames(tabla) = c("CF", "AA", "RA", "CT", "AM", "PA")
> rownames(tabla) = c("Autenticación básica","Autorización por token")
> tabla
                      CF AA RA CT AM PA
Autenticación básica   2  2  1  1  1  3
Autorización por token  3  3  3  3  3  3
> Test_chi <- chisq.test(tabla, correct=FALSE)
Warning message:
In chisq.test(tabla, correct = FALSE) :
  Chi-squared approximation may be incorrect
> Test_chi

        Pearson's Chi-squared test

data:  tabla
X-squared = 1.21, df = 5, p-value = 0.94
```

Figura 4-10. Test Chi-cuadrado
Elaborado por: Guillermo Valencia, 2021

El valor crítico de distribución de Chi-cuadrado con significancia de 0.05% y 5 grados de libertad, se obtiene en R como se muestra en la figura 4-11.

```
> #Calcular el valor crítico
> qchisq(0.95,5)
[1] 11.0705
>
```

Figura 4-11. Valor crítico
Elaborado por: Guillermo Valencia, 2021

$x^2\ crítico = 11.070$

x-scuared = 1.21

En base a estos datos, **Ho** debe aceptarse si:

$x^2\ calculado \leq\ x^2\ crítico$

Caso contrario rechaza **Ho** y acepta **Hi**

1.21 <= 11.070

En consecuencia, concluye el rechazo de la Hipótesis nula (**Ho**) y, se acepta a la Hipótesis alternativa (**Hi**) para un nivel de confianza del 95% y un nivel de significancia del 5%.

CAPÍTULO V

5. PROPUESTA DISEÑO E IMPLEMENTACIÓN

5.1. Sistema de autorización OAuth 2.0

5.1.1. Introducción

La implementación de un sistema de autorización con el protocolo OAuth 2.0, permite mejorar el control de acceso a los recursos web de manera ágil y segura. La configuración del servidor OAuth 2.0 permite proteger las APIs con el uso de tokens de acceso.

El servidor de autorización implementa los siguientes permisos:

- Permiso de código de autorización
- Permiso implícito
- Permiso de credenciales de cliente
- Permiso de propietario de recursos con password de credenciales
- Actualizar permisos

5.1.2. Escenario implementado

El Escenario implementado para el sistema de acceso a recursos a través de autorización con token de acceso se muestra a continuación:

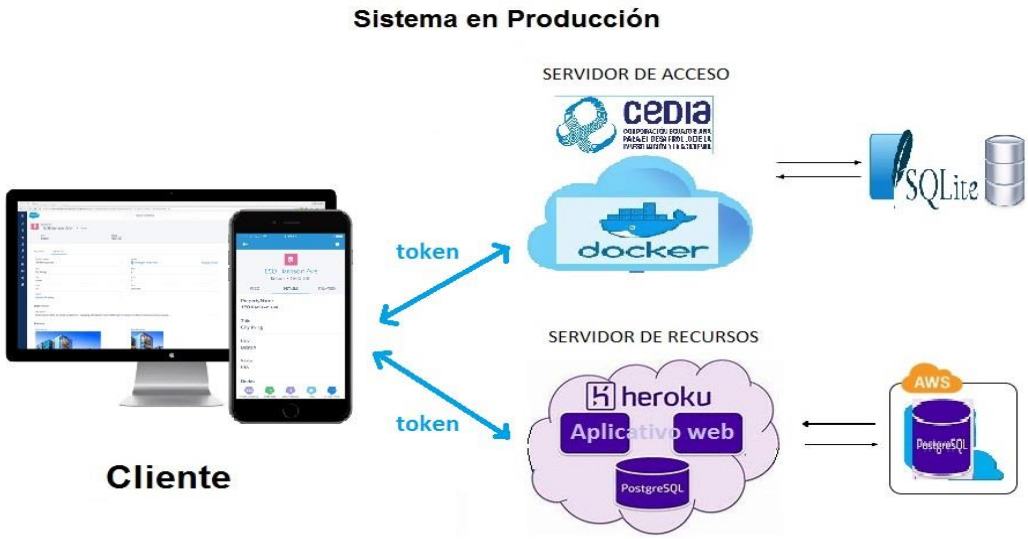

Figura 5-1. Sistema implementado
Elaborado por: Guillermo Valencia, 2021

5.1.3. Requerimientos

El sistema de autorización hace uso de los siguientes requerimientos:

- Ubuntu 18.0
- Docker
- PHP 7.4
- Composer
- Slim framework
- Openssl
- Json
- PSR-7 complaint para todos los mensajes HTTP
- Para prevenir ataques de man-in-the-middle, el servidor de autorización implementa TLS certificate en el intercambio de mensajes de autorización.
- Librerías encriptación para proteger la integridad de las firmas.
- SQLITE como motor de base de datos
- Conexión a internet

5.1.4. Instalación

Toda la instalación utiliza línea de comandos en el terminal de Linux con los siguientes pasos:

- Instalar Docker:

  $ sudo apt-get update

  $ sudo apt-get install \

    apt-transport-https \

    ca-certificates \

    curl \

    gnupg-agent \

    software-properties-common

  #Agregar GPG key:

  $ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add –

```
# set up the stable repository

$ sudo add-apt-repository \

  "deb [arch=amd64] https://download.docker.com/linux/ubuntu \

  $(lsb_release -cs) \

  stable"

# Install Docker Engine

$ sudo apt-get update

$ sudo apt-get install docker-ce docker-ce-cli containerd.io
```

- Clonar el archivo https://github.com/guillovale/oauth2_server.git del repositorio github.com.
- Ir al directorio donde se aloja la aplicación y verificar si existe el archivo de configuración docker-compose.yml que tiene la siguiente configuración:

```
version: '3.7'

volumes:
  logs:
    driver: local

services:
  slim:
    image: php:7-alpine
    working_dir: /var/www
    command: php -S 0.0.0.0:8080 -t public
    environment:
      docker: "true"
    ports:
      - 8080:8080
    volumes:
      - .:/var/www
      - logs:/var/www/logs
```

- Ejecutar el comando:

```
$ path/miapp/ docker-compose up
```

5.1.5. Clases implementadas

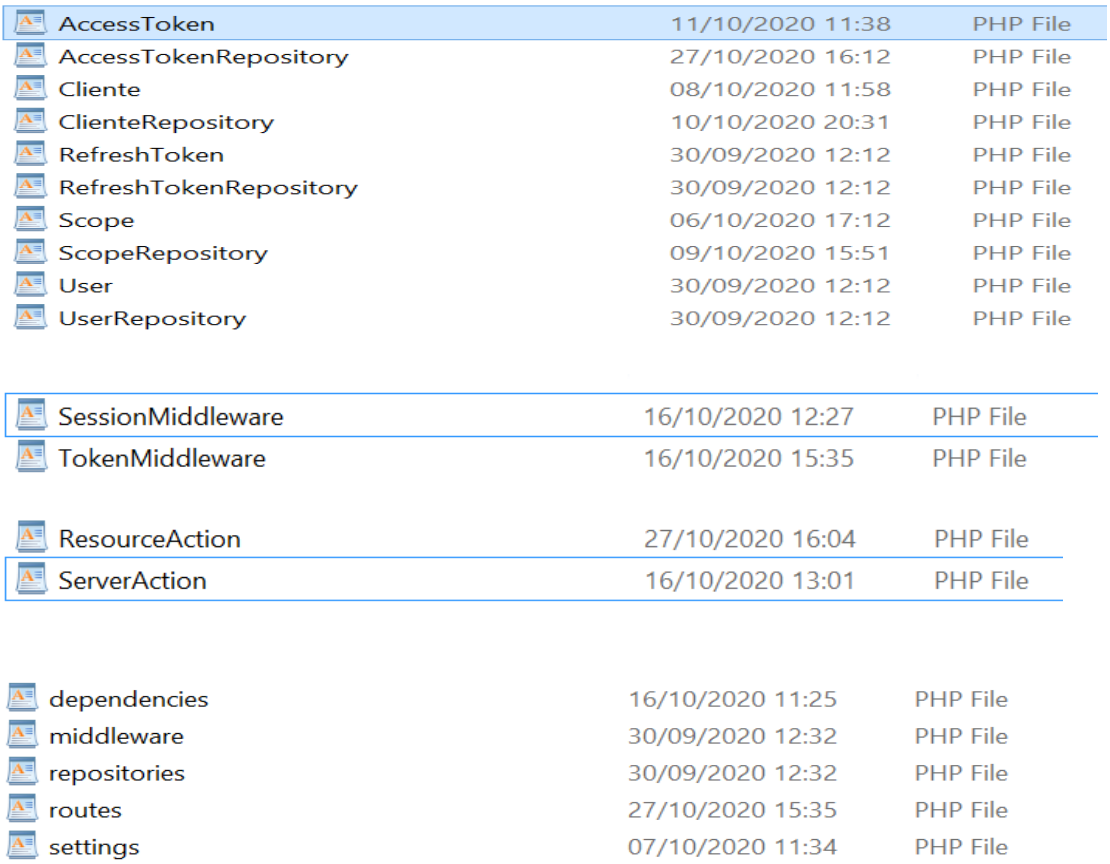

Figura 5-2. Clases implementadas
Elaborado por: Guillermo Valencia, 2021

5.1.6. Verificar el servicio

Abrir un terminal y ejecutar el comando curl como se muestra en la figura a continuación:

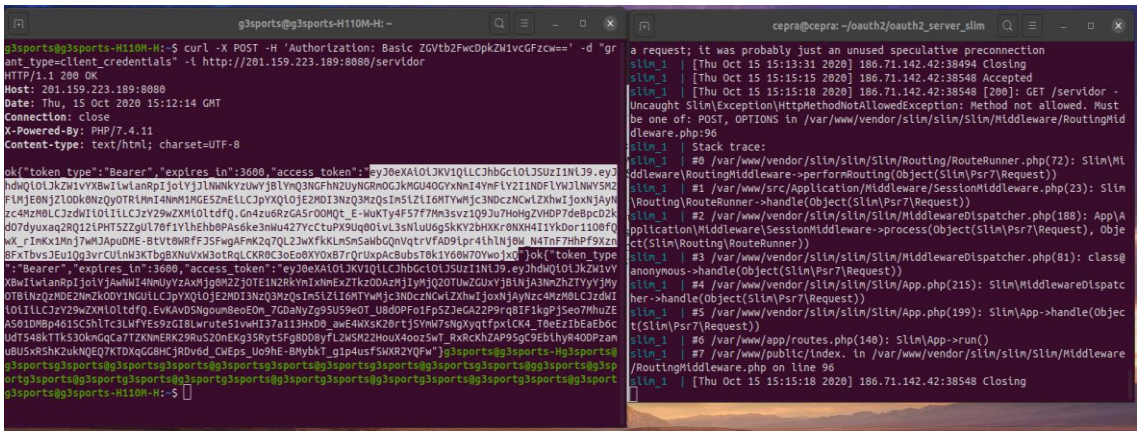

Figura 5-3. Token de acceso del servidor de acceso
Elaborado por: Guillermo Valencia, 2021

Comprobar el alcance del token generado por el servidor de acceso:

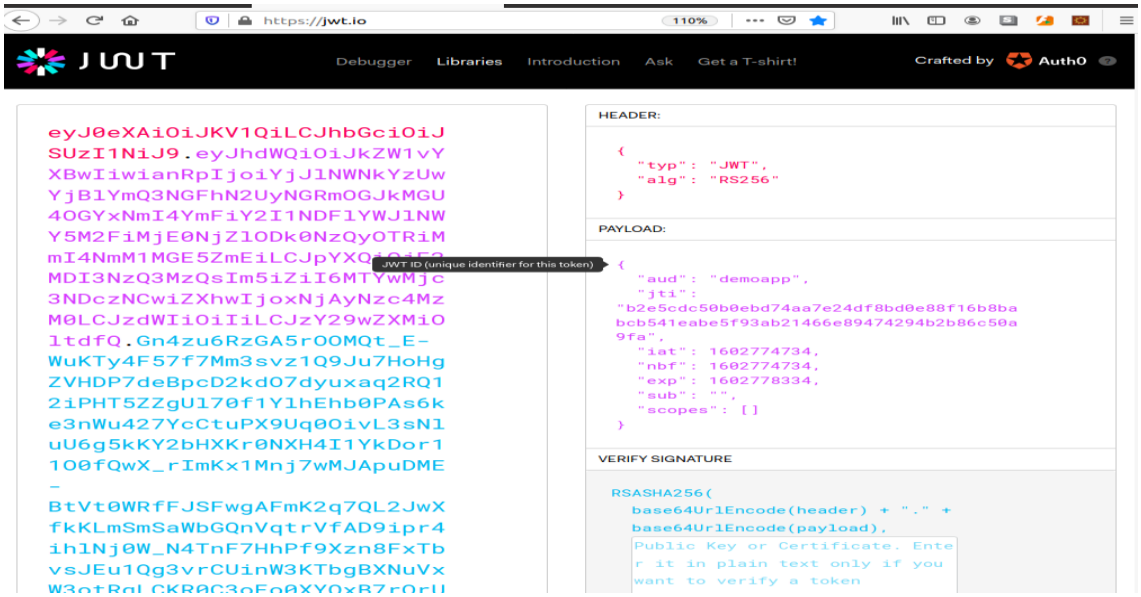

Figura 5-4. Verificar token de acceso del servidor
Elaborado por: Guillermo Valencia, 2021

Por último, verificar el acceso no permitido a través del navegador:

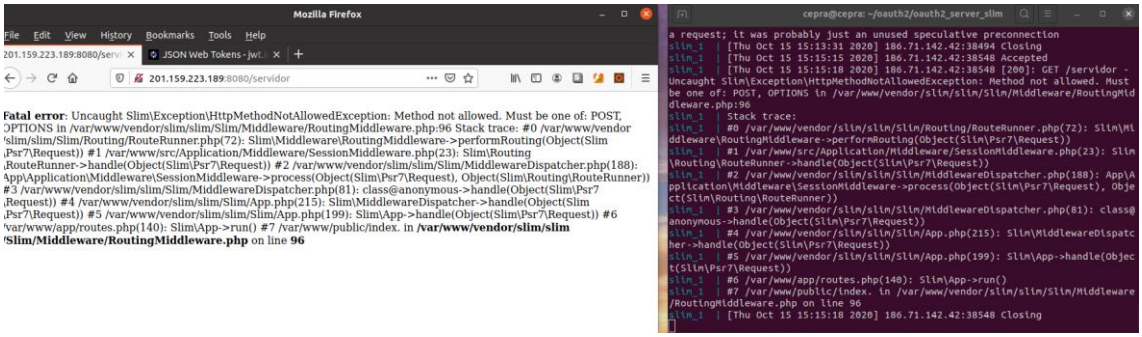

Figura 5-5. Control de acceso del servidor
Elaborado por: Guillermo Valencia, 2021

5.1.7. Configuraciones adicionales

El Sistema de Control de Acceso utiliza el modelo Client credentials grant que implementa las siguientes interfaces:

- Access Token Repository Interface documentation
- Client Repository Interface documentation
- Refresh Token Repository Interface documentation
- Scope Repository Interface documentation

Si se desea utilizar otro modelo, se debe implementar adicionalmente las siguientes interfaces:

- Auth Code Repository Interface documentation
- User Repository Interface documentation

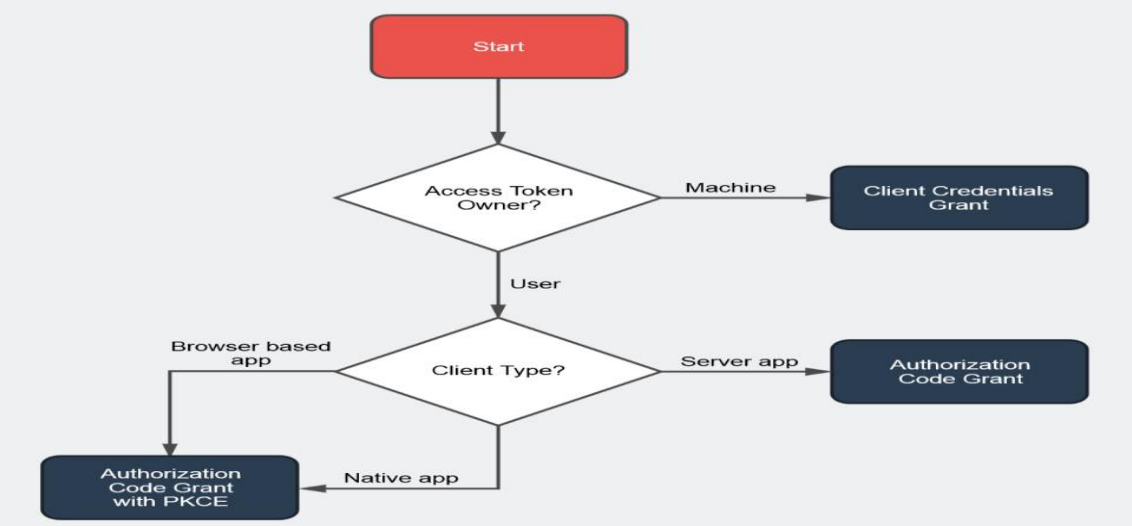

Figura 5-6. Diagrama de tipos de acceso OAuth2
Source: (Reinink, 2020)
Elaborado por: Guillermo Valencia, 2021

Es necesario crear claves pública y privada propias y clave de encriptación con los comandos:

$ openssl genrsa -out private.key 2048

$ openssl rsa -in private.key -pubout -out public.key

$ php -r 'echo base64_encode(random_bytes(32)), PHP_EOL;'

Por defecto está creada la base de datos oauth.db con las tablas e información necesarias. En caso de modificaciones o actualizaciones de la base de datos, se puede volver a ejecutar el archivo de configuración como, se muestra:

$ php path/de /la/aplicación/data/rebuild_db.php

CONCLUSIONES

- Después del análisis de los escenarios cloud computing estudiados, se selecciona al framework de autorización OAuth2.0 con estándar RFC 6749, como modelo base para la implementación de Sistemas de Token de Seguridad (STS) en el intercambio de mensajes que mejoran el control de acceso a recursos web.
- El estudio de vulnerabilidades permitió determinar los posibles riesgos que puedan poner en peligro la autenticación, autorización y a las propiedades de integridad de sesión en el intercambio de mensajes y, se evidencio procedimientos que permiten mitigar estas amenazas.
- De acuerdo al escenario propuesto con el modelo de autorización OAuth 2.0, se diseñó e implementó el servidor de autorización de acceso y servidor de recursos que permitió mejorar el control de acceso a recursos web en plataformas clouding.
- En el sistema de autenticación básica con inicio de sesión, se observó que el tiempo de respuesta comenzó a decaer a medida que aumentaron las peticiones de recursos, esto, se debe a que almacenan los estados de sesión en el servidor. El sistema de autorización por token no tiene estado, por lo que, se obtiene un mejor tiempo de respuesta.
- La solución REST APIs implementada en el servidor de acceso, permitió determinar el comportamiento efectivo del framework OAuth 2.0 en el intercambio de mensajes.

RECOMENDACIONES

- La implementación del framework OAuth 2.0 es bastante extenso a la hora de crear el código de programación para la aplicación cliente y los servidores de acceso y de recursos, y por consiguiente susceptible a cometer errores de codificación, por lo que es necesario disponer de unidades de test.

- Las buenas prácticas a la hora de crear endpoints tomando en cuenta que la ruta debe tener consistencia con un recurso específico, por ejemplo /clientes hace referencia a una lista de clientes.

- Los mensajes de código de estado son indispensables en las respuestas HTTP, para determinar el estado de cada petición de recursos y también pueden ser utilizados como código de respuesta en caso de ingresar un URL mal formado por error o mal intencionado.

- Es posible implementar el framework OAuth 2.0 como base para otros sistemas de token de seguridad, como ejemplo, se puede usar al servidor web NGINX también como servidor de control de acceso, lo que podría disminuir los pasos para la autenticación y autorización.

- Es imperativo implementar seguridad en la capa de transporte (TLS) para evitar a man in the middle e incluso en los dos puntos de transmisión a través de mutual TLS (mTLS).

BIBLIOGRAFÍA

Argyriou, M. &. (2017). *Security Flows in OAuth 2.0 Framework: A Case Study.* Cham: Springer.

Boyd, R. (2012). *Getting Started with OAuth 2.0.* CA.: O'Reilly Media, Inc.

Castro Millán, B. (2019). *Security in API and API managers.* Universitat Oberta de Catalunya (UOC).

DigitalOcean. (2020). *Una introducción a OAuth 2*. Obtenido de https://www.digitalocean.com/community/tutorials/una-introduccion-a-oauth-2-es

Docker. (2020). *Use containers to Build, Share and Run your applications.* Obtenido de Docker: https://www.docker.com/resources/what-container

Google Cloud. (2019). *Google Cloud named a Leader in the 2019 Gartner Magic Quadrant for Full Life Cycle API Management for the fourth consecutive time*. Obtenido de https://cloud.google.com/blog/products/apigee/google-cloud-named-a-leader-in-2019-gartner-magic-quadrant-full-life-cycle-api-management-for-fourth-consecutive-time

Google Developers. (2020). *PageSpeed Insights*. Obtenido de https://developers.google.com/speed/pagespeed/insights/?hl=es

Hardt, E. (2012). *The OAuth 2.0 Authorization Framework*. Obtenido de https://tools.ietf.org/html/rfc6749#section-1.1

HashiCorp. (2020). *Introduction to Vagrant*. Obtenido de https://www.vagrantup.com/intro

Hill, R. H. (2012). *Guide to cloud computing: principles and practice.* Springer Science & Business Media.

Kiani, K. (2016). Four Attacks on OAuth – How to Secure Your OAuth Implementation. (SANS, Ed.) *SANS Institute Reading Room, Mar-2011.[Online].*

Koponen, A. (2016). *A secure OAuth 2.0 implementation model.*

LinuxFoundationX. (2020). *Introduction to Cloud Foundry and Cloud Native Software Architecture (LFS132)*. Obtenido de https://training.linuxfoundation.org/training/introduction-to-cloud-foundry-and-cloud-native-software-architecture/

Lopez, A. (2016). *Learning PHP 7.*

Mozilla. (2020). *Generalidades del protocolo HTTP*. Obtenido de https://developer.mozilla.org/es/docs/Web/HTTP/Overview

NIST. (2020). *National Institute of Standards and Tecnology*. Obtenido de https://www.nist.gov/

NIST. (2020). *Security and Privacy Controls for Federal Information Systems and Organizations*. Obtenido de https://csrc.nist.gov/Projects/risk-management/rmf-overview/security-controls

O'Raw. (2015). Security Evaluation of the OAuth 2.0 Framework. *Information & Computer Security.*

OWAST. (2020). *Top 10 Web Application Security Risks*. Obtenido de https://owasp.org/

Ping Identity. (2012). The Essential OAuth Primer: Understanding OAuth for Securing Cloud APIs. *Understanding OAuth for Securing Cloud APIs*.

Reinink, J. (2020). *OAuth 2.0 Server*. Obtenido de https://oauth2.thephpleague.com/authorization-server/which-grant/

Richer, J. &. (2017). *OAuth 2 in Action.* Simon and Schuster.

Seitz, L. S. (2018). *Authentication and authorization for constrained environments (ACE) using the OAuth 2.0 framework (ACE-OAuth).*

Sheldon, M. R. (2009). *INTRODUCTION TO PROBABILITY AND STATISTICS FOR ENGINEERS AND SCIENTISTS.*

Shernan, E. C. (2015). *More guidelines than rules: CSRF vulnerabilities from noncompliant OAuth 2.0 implementations.*

Siriwardena, P. (2019). *Advanced API Security: OAuth 2.0 and Beyond.* Apress.

STHDA. (2020). *Normality Test in R*. Obtenido de http://www.sthda.com/english/wiki/normality-test-in-r

Tomas. (2014). *Entendiendo Vagrant*. Obtenido de http://tomasdelvechio.github.io/old/260/

ANEXOS

ANEXO A. Configuración de archivo Vagrant

```
# -*- mode: ruby -*-

 Vagrant.configure("2") do |config|

 config.vm.box = "hashicorp/bionic64"

 config.vm.network "forwarded_port", guest: 80, host: 8080

 config.vm.provision "shell", path: "provisioner.sh"

end
```

El archivo de configuración provisioner.sh tiene la siguiente configuración:

```
#!/bin/bash

sudo apt-get update

sudo apt-cache search sqlite

sudo apt install -y sqlite3

sudo apt install -y lynx

sudo add-apt-repository ppa:ondrej/php -y

sudo apt-cache show php

sudo apt install php7.4 -y

sudo apt install php7.4-fpm php7.4-curl php7.4-json php7.4-cgi php7.4-xsl php7.4-pdo php7.4-zip -y

sudo apt install php7.4-cli php7.4-gd php7.4-mbstring php7.4-sqlite3 zip unzip -y

sudo apt-get --purge autoremove -y

sudo service php7.4-fpm restart

sudo systemctl disable --now apache2

sudo service nginx restart

cd /vagrant
```

```
php -r "copy('https://getcomposer.org/installer', 'composer-setup.php');"

php -r "if (hash_file('sha384', 'composer-setup.php') === '795f976fe0ebd8b75f26a6dd68f78fd3453ce79f32ecb33e7fd087d39bfeb978342fb73ac986cd4f54edd0dc902601dc') { echo 'Installer verified'; } else { echo 'Installer corrupt'; unlink('composer-setup.php'); } echo PHP_EOL;"

php composer-setup.php

php composer.phar require slim/psr7

php composer.phar require nyholm/psr7 nyholm/psr7-server

php composer.phar require guzzlehttp/psr7 http-interop/http-factory-guzzle

php composer.phar require laminas/laminas-diactoros

#php composer.phar create-project slim/slim-skeleton:dev-master oauth2_server

# cd oauth2_server; php -S localhost:8080 -t public public/index.php
```

ANEXO B. Archivo de configuración de clase AWS y test de muestra

```
<?php
namespace AwsApp;
use Exception;
use GuzzleHttp\Client;
use GuzzleHttp\Exception\RequestException;
use GuzzleHttp\Pool;
use GuzzleHttp\Psr7\Request;
use GuzzleHttp\Psr7\Response;
use Firebase\JWT\JWT;
class Aws {
	const AWS_API_BASE_URI = 'http://201.159.223.189:8080/';
	const GET_PRODUCTOS = 'https://laliquorstore.herokuapp.com/web/';
	#const GET_ACCESO = 'http://localhost:8080/index.php?r=producto%2Flista';
	const GET_ACCESO =
'https://laliquorstore.herokuapp.com/web/index.php?r=producto%2Flista';
	private $key;
	private $secret;
	private $accessToken;
	private $client;
	private $clientApi;

	public function __construct(String $key, String $secret) {
		$this->key = $key;
		$this->secret = $secret;
		$this->client = new Client(['base_uri' => self::AWS_API_BASE_URI]);
		$this->clientApi = new Client(['base_uri' => self::GET_PRODUCTOS]);
	}

	private function requestAccessToken() {
		$encodedString = base64_encode($this->key . ':' . $this->secret);
		$headers = [
```

```
                'Authorization' => 'Basic ' . $encodedString,
                'Content-Type' => 'application/x-www-form-urlencoded;charset=UTF-8'
            ];
            $options = [
                'headers' => $headers,
                'body' => 'grant_type=client_credentials'
            ];
            try
            {
                #$response = $this->client->request('POST','servidor', $options);
                #$response = $this->client->requestAsync('POST','servidor', $options);
                $promise = $this->client->requestAsync('POST','servidor', $options)->then(function ($response)
                    {
                        $sr = substr($response->getBody(),9);
                        $sr = str_replace("sessionok", "","", $sr);
                        $sr = '['.$sr.']';
                        #$body = json_decode($response->getBody(), true);
                        $body = json_decode($sr, true);
                        if ($body)
                        {
                            #$this->accessToken = $body['access_token'];
                            $this->accessToken = $body[0]['access_token'];
                            echo $this->accessToken;
                        }
                    });
                $promise->wait();
            }
            catch (ClientException $e)
            {
```

```
                        if ($e->getCode() == 401) {
                                echo 'error';
                                throw $e;
                                #$this->requestAccessToken();
                                #$response = $this->client->get(self::GET_PRODUCTOS,
$options);
                        }
                        else
                        {
                                throw $e;
                        }
                }
        }

        private function getAccessTokenHeaders(): array {
                if (empty($this->accessToken)) {
                        $this->requestAccessToken();
                }

                # oauth2
                return ['Authorization' => 'Bearer ' . $this->accessToken,
                                                'cache-control: no-cache',
                                ];
        }

        public function obtenerAcceso()
        {
                $totaltime = 0;
                $ini = 0;
                $fin = 0;
                $tempArray = [];
```

```
        $ini = microtime(1);
        $options = [
                'headers' => $this->getAccessTokenHeaders(),
        ];

        try {
                $response = $this->clientApi->get(self::GET_ACCESO, $options);
                $fin = microtime(1);

        } catch (ClientException $e) {
                                                        throw $e;

        }

        $totaltime =      round($fin - $ini, 4);

        #var_dump($totaltime); exit;
        $repuesta = $response->getHeaders();
        foreach ($response->getHeaders() as $name => $values) {
                        $value = implode(', ', $values);
                        if ($value) $tempArray[$name] = $value;
        }

        $tempArray['Request_time'] = ".$totaltime;
        $json = json_encode($tempArray, JSON_PRETTY_PRINT);

        return $json;
}
public function obtenerBasic()
{
        $totaltime = 0;
        $ini = 0;
```

```
        $fin = 0;
        $tempArray = [];

        $encodedString = base64_encode(
                $this->key . ':' . $this->secret
        );
        $headers = [
                'Authorization' => 'Basic ' . $encodedString,
                #'Content-Type' => 'application/json;charset=utf-8'
                'Content-Type' => 'application/x-www-form-urlencoded;charset=UTF-
8'
        ];
        $options = [
                'headers' => $headers,
                'body' => 'grant_type=client_credentials'
        ];

        $ini = microtime(1);
        try {
                $response = $this->clientApi->get(self::GET_ACCESO, $options);

                /*$promise = $this->clientApi-
>requestAsync('GET',self::GET_ACCESO, $options)->then(function ($response)
                        {
                                $body = json_decode($response->getBody(),
true);
                                $repuesta = $response->getHeaders();
                });
                $promise->wait();
                */
                $fin = microtime(1);
                $totaltime =    round($fin - $ini, 4);
                #$repuesta = $response->getHeaders();
```

```
                        foreach ($response->getHeaders() as $name => $values) {
                                $value = implode(', ', $values);
                                if ($value) $tempArray[$name] = $value;
                        }
                        $tempArray['Request_time'] = ".$totaltime;
                        $json = json_encode($tempArray, JSON_PRETTY_PRINT);
                        return $json;
                } catch (ClientException $e) {
                                $ini = 0;
                                $fin = 0;
                                throw $e;
                }
        }
}

<?php

//namespace App\Application\Actions\Aws;
use AwsApp\Aws;
require __DIR__ . '/vendor/autoload.php';
$pathAws = $_SERVER['HOME'] . '/.aws_php7.json';
$pathGoogle = $_SERVER['HOME'] . '/programas/ruby/g3sports/app/apis/credentials.json';
$jsonCredentials = file_get_contents($pathAws);
#$jsonCredentials = file_get_contents($pathGoogle);
$credentials = json_decode($jsonCredentials, true);

#$google = new Aws($credentials["installed"]['client_id'],
$credentials["installed"]['client_secret']);

for ($i=1; $i<= 385; $i++)
{
        $aws = new Aws($credentials['key'], $credentials['secret']);
```

```
		$dato = $aws->obtenerAcceso();
		$inp = file_get_contents('datos_oauth.json');

		if (!$inp)
		{
			file_put_contents('datos_oauth.json', '['.$dato, FILE_APPEND);
		}
		else
		{
			file_put_contents('datos_oauth.json', ','.$dato, FILE_APPEND);
		}
		//sleep for 3 seconds
		#sleep(3);
}

file_put_contents('datos_oauth.json', ']', FILE_APPEND);

<?php

//namespace App\Application\Actions\Aws;
use AwsApp\Aws;
require __DIR__ . '/vendor/autoload.php';

for ($i=1; $i<= 3; $i++)
{
		$user = 'user_'.$i;
		$pass = 'pass_'.$i;
		$tempArray = [];
		$aws = new Aws($user, $pass);
		$dato = $aws->obtenerBasic();
		#print_r($dato);
```

```
        $inp = file_get_contents('datos_basic.json');

        if (!$inp)
        {
                file_put_contents('datos_basic.json', '['.$dato, FILE_APPEND);
                #$tempArray[] = json_decode($inp);
        #       array_push($tempArray, $inp);
        }
        else
        {
                file_put_contents('datos_basic.json', ','.$dato, FILE_APPEND);
        }

        //sleep for 3 seconds
        sleep(3);
}

file_put_contents('datos_basic.json', ']', FILE_APPEND);
```

ANEXO C. Configuración de tablas en la base de datos de control de acceso Oauth2

```
CREATE TABLE oauth_clients (
  client_id            VARCHAR(80)   NOT NULL,
  client_secret        VARCHAR(80),
  redirect_uri         VARCHAR(2000),
  grant_types          VARCHAR(80),
  scope                VARCHAR(4000),
  user_id              VARCHAR(80),
  PRIMARY KEY (client_id)
);

CREATE TABLE oauth_access_tokens (
  access_token         VARCHAR(40)    NOT NULL,
  client_id            VARCHAR(80)    NOT NULL,
  user_id              VARCHAR(80),
  expires              TIMESTAMP      NOT NULL,
  scope                VARCHAR(4000),
  PRIMARY KEY (access_token)
);

CREATE TABLE oauth_authorization_codes (
  authorization_code  VARCHAR(40)     NOT NULL,
  client_id            VARCHAR(80)    NOT NULL,
  user_id              VARCHAR(80),
  redirect_uri         VARCHAR(2000),
  expires              TIMESTAMP      NOT NULL,
  scope                VARCHAR(4000),
  id_token             VARCHAR(1000),
  PRIMARY KEY (authorization_code)
);

CREATE TABLE oauth_refresh_tokens (
```

```
  refresh_token       VARCHAR(40)     NOT NULL,
  client_id           VARCHAR(80)     NOT NULL,
  user_id             VARCHAR(80),
  expires             TIMESTAMP       NOT NULL,
  scope               VARCHAR(4000),
  PRIMARY KEY (refresh_token)
);

CREATE TABLE oauth_users (
  username            VARCHAR(80),
  password            VARCHAR(80),
  first_name          VARCHAR(80),
  last_name           VARCHAR(80),
  email               VARCHAR(80),
  email_verified      BOOLEAN,
  scope               VARCHAR(4000),
  PRIMARY KEY (username)
);

CREATE TABLE oauth_scopes (
  scope               VARCHAR(80)     NOT NULL,
  is_default          BOOLEAN,
  PRIMARY KEY (scope)
);

CREATE TABLE oauth_jwt (
  client_id           VARCHAR(80)     NOT NULL,
  subject             VARCHAR(80),
  public_key          VARCHAR(2000)   NOT NULL
);
```

ANEXO D. Datos para estudio de autenticación básica en función del tiempo

Autenticación básica en milisegundos (ms)								
tiempo (ms)	**tiempo (ms)**	**tiempo (ms)**	**tiempo (ms)**	**tiempo (ms)**	**tiempo (ms)**	**tiempo (ms)**	**tiempo (ms)**	**tiempo (ms)**
0,7612	0,6502	2,0542	0,7344	0,6954	0,8187	0,7249	0,6656	0,6925
0,7864	0,6566	0,6449	0,828	0,6755	0,6686	6,1452	0,7136	0,6851
0,8271	0,6867	0,6764	0,8295	0,6557	0,6733	3,1708	0,7049	0,6547
11,3771	0,6379	0,7034	0,8857	0,662	3,8381	3,0114	0,6973	0,8025
0,792	0,8057	0,7255	6,8003	0,6743	0,6771	1,6303	0,6542	0,6477
5,9731	0,7571	0,7141	2,0091	0,8244	0,6496	11,0749	0,6625	0,6754
0,784	0,6981	0,6412	0,941	0,6874	0,6619	0,7836	0,6363	0,7671
0,9239	0,723	0,6588	1,8469	0,6952	0,684	1,6131	0,7102	0,6842
1,3534	0,6704	0,7697	0,921	0,6715	0,7023	12,3334	0,6723	0,8506
0,644	0,6636	1,28	0,6907	0,6297	1,2862	0,6677	0,6469	1,7542
1,3095	0,6852	12,0221	0,7119	0,6789	0,7926	0,682	0,646	0,9314
0,6456	0,662	3,249	0,6356	0,7499	1,2173	0,6673	2,3099	0,9589
0,6263	0,6608	0,7676	0,6653	0,8632	6,313	0,654	0,6256	1,2466
0,728	0,8617	6,8786	1,6245	0,6985	7,1404	0,9473	0,877	2,3585
0,8082	0,8465	1,0892	1,9899	1,1093	3,2283	1,114	2,6975	0,7931
0,6215	0,6603	4,6495	0,6882	2,2103	0,6307	0,6422	12,0575	0,6234

0,6625	0,8373	0,6895	0,6987	6,1062	0,6701	0,6826	0,8475	0,6966
0,6932	0,8073	0,6761	0,6828	5,8023	0,6576	6,1864	0,6517	0,6586
1,3963	0,6382	0,6459	0,7607	0,6857	0,6291	6,9502	0,6328	0,6109
2,7801	0,6724	0,7303	2,4297	0,6559	0,6683	2,1676	0,6807	0,6462
15,7061	0,6963	0,7313	0,8203	0,6539	0,6456	2,0031	0,6591	4,3286
0,8182	0,697	0,7689	2,3382	0,8271	1,0255	2,1411	0,9598	0,817
5,9361	2,5149	0,9099	0,8355	0,6437	0,6463	5,8563	0,6464	0,6759
0,8314	0,7117	0,663	1,8715	0,6789	0,6565	11,0538	0,6309	1,1289
0,6405	0,6677	3,2834	0,8198	0,6604	5,8482	0,6748	0,6536	3,2595
0,7194	0,6421	7,2339	0,7195	0,6395	1,181	0,6736	0,6819	8,1284
0,6536	0,6775	6,6843	0,6506	0,6773	0,8034	14,6469	0,6553	0,8454
3,386	0,6396	0,6845	0,8539	0,6585	0,6488	0,8684	0,6514	0,6958
7,5372	0,6721	0,8817	2,3445	0,6368	0,6338	15,6132	0,6593	0,6609
6,2718	0,6241	0,659	2,8777	1,0669	0,6681	7,1079	1,1412	0,6394
7,0197	0,6469	0,6362	0,8673	0,677	0,6464	6,2521	1,0596	0,6443
7,1533	0,6318	0,6778	1,9992	0,6808	0,688	7,0178	0,6892	0,6828
6,0431	2,9233	6,162	0,6587	0,6599	0,8177	0,6278	0,6805	1,6797
0,7315	0,6642	1,0546	0,7231	0,7088	5,6597	0,8716	0,7777	0,6353
2,1112	5,8909	0,6488	0,6481	8,1699	0,6476	0,6347	0,8517	0,6558

0,645	1,5087	0,6475	0,6689	2,2298	0,6729	0,6154	10,7956	0,639
0,6508	1,1256	0,6107	0,6307	6,6214	0,6806	0,7105	2,0713	0,6847
0,6634	13,2046	0,6967	0,6655	0,7995	0,6483	0,685	12,3109	0,6402
0,668	5,8898	0,674	0,6903	0,8579	0,6441	0,6577	0,8295	0,7106
0,63	13,2702	0,8182	0,6596	1,106	0,6835	0,6852	5,8884	0,6815
0,6521	0,9564	5,9386	0,6279	0,6687	0,8369	0,6852	11,1187	0,6504
0,6237	7,64	0,6406	0,6559	5,6677	0,6953	0,6482	3,2687	0,6692
0,6692	0,8062	0,6215	0,6603	4,6495	0,6882	2,2103		

ANEXO E. Datos para estudio de autorización por token en función del tiempo

Autorización por token en milisegundos (ms)								
tiempo (ms)	**tiempo (ms)**	**tiempo (ms)**	**tiempo (ms)**	**tiempo (ms)**	**tiempo (ms)**	**tiempo (ms)**	**tiempo (ms)**	**tiempo (ms)**
1,7101	0,6666	0,65	0,7177	0,7857	0,7641	0,7367	0,7823	0,6847
0,695	0,6497	0,6832	0,7276	11,3655	0,7156	0,74	0,7661	0,9333
0,7929	0,8126	0,8087	0,8423	0,9296	0,7774	0,8047	0,888	0,8751
0,6916	0,6799	0,7049	0,6563	0,6457	0,6883	0,707	0,6731	0,7149
0,6574	0,7071	0,681	0,87	5,9326	0,6854	0,6558	0,6593	0,6829
0,6701	0,6957	0,7927	0,6615	0,6721	0,7181	0,6957	0,7472	0,7574
6,3601	0,7573	0,8854	0,8486	0,6783	0,6794	0,6807	0,6471	0,668
0,6937	0,6586	0,7203	0,6882	0,6404	1,4846	0,6783	0,6803	0,6642
0,73	0,6859	0,6624	0,708	0,6727	0,683	0,6783	7,6271	0,7034
0,8077	0,7258	0,6983	0,7081	0,7293	0,7537	0,7914	0,9049	0,7766
0,7626	0,8157	1,0786	0,817	0,7772	0,7508	0,7324	0,688	0,7311
0,7138	0,7231	0,7817	0,7566	0,7238	0,7179	2,1154	0,706	0,7004
0,7252	0,7781	0,7205	0,757	0,732	0,7148	0,7151	0,7154	0,7321
0,726	7,2469	0,7731	0,7376	0,7647	0,7113	0,7288	0,8086	0,7772
0,8133	0,7453	0,7808	0,8289	1,8895	0,7425	0,702	0,7349	0,7193
0,7076	0,713	0,6796	0,7147	0,7193	0,7094	0,7237	0,9837	0,7765

10,8865	0,7414	0,748	0,737	0,7084	0,7673	0,7843	0,7933	0,8008
0,8057	0,7414	0,7369	0,7011	1,8351	0,7032	0,7524	0,8095	0,7822
0,754	0,7016	0,7353	0,7937	0,7837	0,7228	0,7451	1,2814	6,0648
0,7202	0,7194	0,6912	0,7283	0,7022	0,6922	0,6998	0,7727	0,732
0,683	0,6885	1,9591	1,2343	0,7577	0,7051	0,7601	0,6774	0,8929
0,8298	0,7065	0,7093	0,7112	0,6764	0,7734	0,7643	5,9194	0,728
0,7229	0,7417	0,7504	0,6975	0,715	0,767	0,8103	0,7276	2,0469
0,7464	5,8768	0,7395	0,9584	0,7915	0,7936	0,6571	0,6939	0,6758
0,6752	0,7026	0,6814	0,7653	0,681	1,5604	0,9263	0,6925	0,7443
0,7219	0,6905	0,7028	0,6594	0,7104	0,6905	0,6889	0,6929	0,7047
0,6841	0,8409	0,6703	1,1602	0,7376	0,6835	0,6616	0,7045	0,6663
0,6918	0,673	0,6756	0,684	0,6724	0,6941	10,7407	0,7281	2,2747
0,7918	0,6747	0,7	0,701	0,6598	0,7016	0,8517	0,682	0,6977
0,6891	0,6716	0,6825	0,7079	1,7309	0,729	0,7216	0,7082	0,6829
3,5899	5,8551	0,6916	0,7055	0,682	0,7626	0,7031	0,6822	0,7048
0,6868	0,7208	0,6918	1,2766	0,7649	5,839	0,698	0,7006	0,6679
0,7052	0,7236	0,6794	0,7082	0,7222	0,7931	0,7646	0,7817	0,8651
8,2377	0,7324	0,9058	0,6904	0,6885	2,2584	0,7177	0,7007	0,7052
0,7007	0,6887	2,1713	0,7003	0,7296	0,7168	0,7244	0,6715	0,6681

1,0851	0,6692	0,6541	0,6962	0,672	1,665	0,8209	0,6815	0,6848
0,6815	0,673	0,6701	1,134	0,681	0,6871	0,6975	0,689	0,6801
0,7145	0,706	3,1269	1,8719	0,7653	0,8053	0,7485	0,7567	0,672
0,6796	0,7257	0,6526	0,6786	0,6992	1,0581	0,7003	0,6747	0,6518
0,8348	0,688	0,6656	0,6704	0,6741	0,6608	0,6675	0,7225	0,6797
0,6587	0,7289	2,1049	0,6963	0,7035	0,6808	0,6804	0,6792	0,6769
0,6602	0,6651	0,7107	0,6617	0,6779	0,6767	0,6947	3,2052	0,6759
0,6969	0,6618	0,7361	0,6752	0,7209	0,7321	0,6468		

ANEXO F. Informe OWASP ZAP de autenticación básica

ZAP Scanning Report

Summary of Alerts

Risk Level	Number of Alerts
High	0
Medium	1
Low	5
Informational	2

Alert Detail

Medium (Medium)	X-Frame-Options Header Not Set
Description	X-Frame-Options header is not included in the HTTP response to protect against 'ClickJacking' attacks.
URL	https://laliquorstore.herokuapp.com/web/index.php?r=producto%2Flista
Method	GET
Parameter	X-Frame-Options
Instances	1
Solution	Most modern Web browsers support the X-Frame-Options HTTP header. Ensure it's set on all web pages returned by your site (if you expect the page to be framed only by pages on your server (e.g. it's part of a FRAMESET) then you'll want to use SAMEORIGIN, otherwise if you never expect the page to be framed, you should use DENY. ALLOW-FROM allows specific websites to frame the web page in supported web browsers).
Reference	https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Frame-Options
CWE Id	16

WASC Id	15
Source ID	3
Low (Medium)	**X-Content-Type-Options Header Missing**
Description	The Anti-MIME-Sniffing header X-Content-Type-Options was not set to 'nosniff'. This allows older versions of Internet Explorer and Chrome to perform MIME-sniffing on the response body, potentially causing the response body to be interpreted and displayed as a content type other than the declared content type. Current (early 2014) and legacy versions of Firefox will use the declared content type (if one is set), rather than performing MIME-sniffing.
URL	https://content-signature-2.cdn.mozilla.net/chains/remote-settings.content-signature.mozilla.org-2020-12-14-15-04-19.chain
Method	GET
Parameter	X-Content-Type-Options
Instances	1
Solution	Ensure that the application/web server sets the Content-Type header appropriately, and that it sets the X-Content-Type-Options header to 'nosniff' for all web pages. If possible, ensure that the end user uses a standards-compliant and modern web browser that does not perform MIME-sniffing at all, or that can be directed by the web application/web server to not perform MIME-sniffing.
Other information	This issue still applies to error type pages (401, 403, 500, etc.) as those pages are often still affected by injection issues, in which case there is still concern for browsers sniffing pages away from their actual content type. At "High" threshold this scanner will not alert on client or server error responses.
Reference	http://msdn.microsoft.com/en-us/library/ie/gg622941%28v=vs.85%29.aspx https://owasp.org/www-community/Security_Headers
CWE Id	16
WASC Id	15

Source ID	3

Low (Medium)	**Incomplete or No Cache-control and Pragma HTTP Header Set**
Description	The cache-control and pragma HTTP header have not been set properly or are missing allowing the browser and proxies to cache content.
URL	https://firefox.settings.services.mozilla.com/v1/buckets/main/collections/partitioning-exempt-urls/changeset?_expected=1592906663254
Method	GET
Parameter	Cache-Control
URL	https://firefox.settings.services.mozilla.com/v1/buckets/monitor/collections/changes/records?collection=cfr-fxa&bucket=main
Method	GET
Parameter	Cache-Control
Evidence	max-age=60
URL	https://firefox.settings.services.mozilla.com/v1/buckets/monitor/collections/changes/records?collection=message-groups&bucket=main
Method	GET
Parameter	Cache-Control
Evidence	max-age=60
URL	https://firefox.settings.services.mozilla.com/v1/buckets/monitor/collections/changes/records?collection=cfr&bucket=main

Method	GET
Parameter	Cache-Control
Evidence	max-age=60
URL	https://firefox.settings.services.mozilla.com/v1/buckets/main/collections/message-groups/changeset?_expected=1595616291726
Method	GET
Parameter	Cache-Control
URL	https://firefox.settings.services.mozilla.com/v1/buckets/monitor/collections/changes/records?collection=partitioning-exempt-urls&bucket=main
Method	GET
Parameter	Cache-Control
Evidence	max-age=60
URL	https://firefox.settings.services.mozilla.com/v1/buckets/main/collections/cfr/changeset?_expected=1603719109121
Method	GET
Parameter	Cache-Control
URL	https://firefox.settings.services.mozilla.com/v1/buckets/main/collections/cfr-fxa/changeset?_expected=1585237415718

Method	GET
Parameter	Cache-Control
Instances	8
Solution	Whenever possible ensure the cache-control HTTP header is set with no-cache, no-store, must-revalidate; and that the pragma HTTP header is set with no-cache.
Reference	https://cheatsheetseries.owasp.org/cheatsheets/Session_Management_Cheat_Sheet.html#web-content-caching
CWE Id	525
WASC Id	13
Source ID	3

Low (Medium)	**Cookie Without SameSite Attribute**
Description	A cookie has been set without the SameSite attribute, which means that the cookie can be sent as a result of a 'cross-site' request. The SameSite attribute is an effective counter measure to cross-site request forgery, cross-site script inclusion, and timing attacks.
URL	https://laliquorstore.herokuapp.com/web/index.php?r=producto%2Flista
Method	GET
Parameter	PHPSESSID
Evidence	Set-Cookie: PHPSESSID
URL	https://laliquorstore.herokuapp.com/web/index.php?r=producto%2Flista
Method	GET
Parameter	_csrf

Evidence	Set-Cookie: _csrf
Instances	2
Solution	Ensure that the SameSite attribute is set to either 'lax' or ideally 'strict' for all cookies.
Reference	https://tools.ietf.org/html/draft-ietf-httpbis-cookie-same-site
CWE Id	16
WASC Id	13
Source ID	3
Low (Medium)	**Cookie Without Secure Flag**
Description	A cookie has been set without the secure flag, which means that the cookie can be accessed via unencrypted connections.
URL	https://laliquorstore.herokuapp.com/web/index.php?r=producto%2Flista
Method	GET
Parameter	PHPSESSID
Evidence	Set-Cookie: PHPSESSID
URL	https://laliquorstore.herokuapp.com/web/index.php?r=producto%2Flista
Method	GET
Parameter	_csrf
Evidence	Set-Cookie: _csrf
Instances	2
Solution	Whenever a cookie contains sensitive information or is a session token, then it should always be passed using an encrypted channel. Ensure that the secure flag is set for cookies containing such sensitive information.

Reference	https://owasp.org/www-project-web-security-testing-guide/v41/4-Web_Application_Security_Testing/06-Session_Management_Testing/02-Testing_for_Cookies_Attributes.html
CWE Id	614
WASC Id	13
Source ID	3
Low (Medium)	**X-Content-Type-Options Header Missing**
Description	The Anti-MIME-Sniffing header X-Content-Type-Options was not set to 'nosniff'. This allows older versions of Internet Explorer and Chrome to perform MIME-sniffing on the response body, potentially causing the response body to be interpreted and displayed as a content type other than the declared content type. Current (early 2014) and legacy versions of Firefox will use the declared content type (if one is set), rather than performing MIME-sniffing.
URL	https://laliquorstore.herokuapp.com/web/index.php?r=producto%2Flista
Method	GET
Parameter	X-Content-Type-Options
Instances	1
Solution	Ensure that the application/web server sets the Content-Type header appropriately, and that it sets the X-Content-Type-Options header to 'nosniff' for all web pages. If possible, ensure that the end user uses a standards-compliant and modern web browser that does not perform MIME-sniffing at all, or that can be directed by the web application/web server to not perform MIME-sniffing.
Other information	This issue still applies to error type pages (401, 403, 500, etc.) as those pages are often still affected by injection issues, in which case there is still concern for browsers sniffing pages away from their actual content type. At "High" threshold this scanner will not alert on client or server error responses.

Reference	http://msdn.microsoft.com/en-us/library/ie/gg622941%28v=vs.85%29.aspx https://owasp.org/www-community/Security_Headers
CWE Id	16
WASC Id	15
Source ID	3

Informational (Low)	**Timestamp Disclosure - Unix**
Description	A timestamp was disclosed by the application/web server - Unix
URL	https://firefox.settings.services.mozilla.com/v1/buckets/main/collections/cfr/changeset?_expected=1603719109121
Method	GET
Evidence	86400000
URL	https://firefox.settings.services.mozilla.com/v1/buckets/main/collections/cfr/changeset?_expected=1603719109121
Method	GET
Evidence	537001984
URL	https://firefox.settings.services.mozilla.com/v1/buckets/main/collections/message-groups/changeset?_expected=1595616291726
Method	GET
Evidence	86400000

URL	https://firefox.settings.services.mozilla.com/v1/buckets/main/collections/cfr/changeset?_expected=1603719109121
Method	GET
Evidence	172800000
URL	https://firefox.settings.services.mozilla.com/v1/buckets/main/collections/cfr/changeset?_expected=1603719109121
Method	GET
Evidence	604800000
Instances	5
Solution	Manually confirm that the timestamp data is not sensitive, and that the data cannot be aggregated to disclose exploitable patterns.
Other information	86400000, which evaluates to: 1972-09-26 19:00:00
Reference	http://projects.webappsec.org/w/page/13246936/Information%20Leakage
CWE Id	200
WASC Id	13
Source ID	3

Informational (Low)	**Timestamp Disclosure - Unix**
Description	A timestamp was disclosed by the application/web server - Unix
URL	https://laliquorstore.herokuapp.com/web/index.php?r=producto%2Flista
Method	GET

Evidence	78601390
Instances	1
Solution	Manually confirm that the timestamp data is not sensitive, and that the data cannot be aggregated to disclose exploitable patterns.
Other information	78601390, which evaluates to: 1972-06-28 12:43:10
Reference	http://projects.webappsec.org/w/page/13246936/Information%20Leakage
CWE Id	200
WASC Id	13
Source ID	3

ANEXO G. Informe OWASP ZAP de autorización por token

ZAP Scanning Report

Summary of Alerts

Risk Level	Number of Alerts
High	0
Medium	1
Low	4
Informational	1

Alert Detail

Medium (Medium)	X-Frame-Options Header Not Set
Description	X-Frame-Options header is not included in the HTTP response to protect against 'ClickJacking' attacks.
URL	http://localhost:8080/
Method	GET
Parameter	X-Frame-Options
Instances	1
Solution	Most modern Web browsers support the X-Frame-Options HTTP header. Ensure it's set on all web pages returned by your site (if you expect the page to be framed only by pages on your server (e.g. it's part of a FRAMESET) then you'll want to use SAMEORIGIN, otherwise if you never expect the page to be framed, you should use DENY. ALLOW-FROM allows specific websites to frame the web page in supported web browsers).
Reference	https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Frame-Options
CWE Id	16
WASC Id	15
Source ID	3

Low (Medium)	**Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s)**
Description	The web/application server is leaking information via one or more "X-Powered-By" HTTP response headers. Access to such information may facilitate attackers identifying other frameworks/components your web application is reliant upon and the vulnerabilities such components may be subject to.
URL	http://localhost:8080/
Method	GET
Evidence	X-Powered-By: PHP/7.4.3
Instances	1
Solution	Ensure that your web server, application server, load balancer, etc. is configured to suppress "X-Powered-By" headers.
Reference	http://blogs.msdn.com/b/varunm/archive/2013/04/23/remove-unwanted-http-response-headers.aspx http://www.troyhunt.com/2012/02/shhh-dont-let-your-response-headers.html
CWE Id	200
WASC Id	13
Source ID	3

Low (Medium)	**X-Content-Type-Options Header Missing**
Description	The Anti-MIME-Sniffing header X-Content-Type-Options was not set to 'nosniff'. This allows older versions of Internet Explorer and Chrome to perform MIME-sniffing on the response body, potentially causing the response body to be interpreted and displayed as a content type other than the declared content type. Current (early 2014) and legacy versions of Firefox will use the declared content type (if one is set), rather than performing MIME-sniffing.
URL	http://localhost:8080/
Method	GET
Parameter	X-Content-Type-Options

Instances	1
Solution	Ensure that the application/web server sets the Content-Type header appropriately, and that it sets the X-Content-Type-Options header to 'nosniff' for all web pages. If possible, ensure that the end user uses a standards-compliant and modern web browser that does not perform MIME-sniffing at all, or that can be directed by the web application/web server to not perform MIME-sniffing.
Other information	This issue still applies to error type pages (401, 403, 500, etc.) as those pages are often still affected by injection issues, in which case there is still concern for browsers sniffing pages away from their actual content type. At "High" threshold this scanner will not alert on client or server error responses.
Reference	http://msdn.microsoft.com/en-us/library/ie/gg622941%28v=vs.85%29.aspx https://owasp.org/www-community/Security_Headers
CWE Id	16
WASC Id	15
Source ID	3
Low (Medium)	**X-Content-Type-Options Header Missing**
Description	The Anti-MIME-Sniffing header X-Content-Type-Options was not set to 'nosniff'. This allows older versions of Internet Explorer and Chrome to perform MIME-sniffing on the response body, potentially causing the response body to be interpreted and displayed as a content type other than the declared content type. Current (early 2014) and legacy versions of Firefox will use the declared content type (if one is set), rather than performing MIME-sniffing.
URL	https://content-signature-2.cdn.mozilla.net/chains/remote-settings.content-signature.mozilla.org-2020-12-14-15-04-19.chain
Method	GET
Parameter	X-Content-Type-Options
URL	https://content-signature-2.cdn.mozilla.net/chains/onecrl.content-signature.mozilla.org-2020-12-14-15-04-14.chain

Method	GET
Parameter	X-Content-Type-Options
URL	https://content-signature-2.cdn.mozilla.net/chains/pinning-preload.content-signature.mozilla.org-2020-12-14-15-04-17.chain
Method	GET
Parameter	X-Content-Type-Options
Instances	3
Solution	Ensure that the application/web server sets the Content-Type header appropriately, and that it sets the X-Content-Type-Options header to 'nosniff' for all web pages. If possible, ensure that the end user uses a standards-compliant and modern web browser that does not perform MIME-sniffing at all, or that can be directed by the web application/web server to not perform MIME-sniffing.
Other information	This issue still applies to error type pages (401, 403, 500, etc.) as those pages are often still affected by injection issues, in which case there is still concern for browsers sniffing pages away from their actual content type. At "High" threshold this scanner will not alert on client or server error responses.
Reference	http://msdn.microsoft.com/en-us/library/ie/gg622941%28v=vs.85%29.aspx https://owasp.org/www-community/Security_Headers
CWE Id	16
WASC Id	15
Source ID	3

Low (Medium)	**Incomplete or No Cache-control and Pragma HTTP Header Set**
Description	The cache-control and pragma HTTP header have not been set properly or are missing allowing the browser and proxies to cache content.
URL	https://firefox.settings.services.mozilla.com/v1/buckets/blocklists/collections/addons-

	bloomfilters/changeset?_expected=1604515098569&_since=%221602268629760%22
Method	GET
Parameter	Cache-Control
URL	https://firefox.settings.services.mozilla.com/v1/buckets/monitor/collections/changes/records?collection=partitioning-exempt-urls&bucket=main
Method	GET
Parameter	Cache-Control
Evidence	max-age=60
URL	https://firefox.settings.services.mozilla.com/v1/buckets/blocklists/collections/gfx?_expected=1480349135384
Method	GET
Parameter	Cache-Control
Evidence	no-cache, no-store
URL	https://firefox.settings.services.mozilla.com/v1/buckets/main/collections/sites-classification?_expected=1544035467383
Method	GET

Parameter	Cache-Control
Evidence	no-cache, no-store
URL	https://firefox.settings.services.mozilla.com/v1/buckets/main/collections/hijack-blocklists?_expected=1572620201554
Method	GET
Parameter	Cache-Control
Evidence	no-cache, no-store
URL	https://firefox.settings.services.mozilla.com/v1/buckets/monitor/collections/changes/records?collection=cfr&bucket=main
Method	GET
Parameter	Cache-Control
Evidence	max-age=60
URL	https://firefox.settings.services.mozilla.com/v1/buckets/main/collections/message-groups/changeset?_expected=1595616291726
Method	GET
Parameter	Cache-Control

URL	https://firefox.settings.services.mozilla.com/v1/buckets/main/collections/top-sites/changeset?_expected=1603813848271&_since=%221599820752079%22
Method	GET
Parameter	Cache-Control
URL	https://firefox.settings.services.mozilla.com/v1/buckets/pinning/collections/pins/changeset?_expected=1485794868067
Method	GET
Parameter	Cache-Control
URL	https://firefox.settings.services.mozilla.com/v1/buckets/main/collections/language-dictionaries?_expected=1569410800356
Method	GET
Parameter	Cache-Control
Evidence	no-cache, no-store
URL	https://firefox.settings.services.mozilla.com/v1/buckets/main/collections/pioneer-study-addons-v1/changeset?_expected=1604359324355
Method	GET

Parameter	Cache-Control
URL	https://firefox.settings.services.mozilla.com/v1/buckets/main/collections/search-default-override-allowlist?_expected=1595254618540
Method	GET
Parameter	Cache-Control
Evidence	no-cache, no-store
URL	https://firefox.settings.services.mozilla.com/v1/buckets/blocklists/collections/plugins/changeset?_expected=1603126502200&_since=%221594658825158%22
Method	GET
Parameter	Cache-Control
URL	https://firefox.settings.services.mozilla.com/v1/buckets/security-state/collections/onecrl?_expected=1600115137321
Method	GET
Parameter	Cache-Control
Evidence	no-cache, no-store
URL	https://firefox.settings.services.mozilla.com/v1/buckets/main/collections/fx-monitor-breaches/changeset?_expected=1598625907365

Method	GET
Parameter	Cache-Control
URL	https://firefox.settings.services.mozilla.com/v1/buckets/main/collections/url-classifier-skip-urls/changeset?_expected=1594765026508&_since=%221582750412799%22
Method	GET
Parameter	Cache-Control
URL	https://firefox.settings.services.mozilla.com/v1/buckets/monitor/collections/changes/records?collection=message-groups&bucket=main
Method	GET
Parameter	Cache-Control
Evidence	max-age=60
URL	https://firefox.settings.services.mozilla.com/v1/buckets/main/collections/public-suffix-list/changeset?_expected=1575468539758
Method	GET
Parameter	Cache-Control

URL	https://firefox.settings.services.mozilla.com/v1/buckets/monitor/collections/changes/records?collection=cfr-fxa&bucket=main
Method	GET
Parameter	Cache-Control
Evidence	max-age=60
URL	https://firefox.settings.services.mozilla.com/v1/buckets/monitor/collections/changes/records
Method	GET
Parameter	Cache-Control
Evidence	max-age=60
Instances	27
Solution	Whenever possible ensure the cache-control HTTP header is set with no-cache, no-store, must-revalidate; and that the pragma HTTP header is set with no-cache.
Reference	https://cheatsheetseries.owasp.org/cheatsheets/Session_Management_Cheat_Sheet.html#web-content-caching
CWE Id	525
WASC Id	13
Source ID	3

Informational (Low)	**Timestamp Disclosure - Unix**
Description	A timestamp was disclosed by the application/web server - Unix

URL	https://firefox.settings.services.mozilla.com/v1/buckets/main/collections/fxmonitor-breaches/changeset?_expected=1598625907365
Method	GET
Evidence	49038354
URL	https://firefox.settings.services.mozilla.com/v1/buckets/main/collections/fxmonitor-breaches/changeset?_expected=1598625907365
Method	GET
Evidence	26183992
URL	https://firefox.settings.services.mozilla.com/v1/buckets/main/collections/fxmonitor-breaches/changeset?_expected=1598625907365
Method	GET
Evidence	49467477
URL	https://firefox.settings.services.mozilla.com/v1/buckets/main/collections/fxmonitor-breaches/changeset?_expected=1598625907365
Method	GET
Evidence	36395491
URL	https://firefox.settings.services.mozilla.com/v1/buckets/main/collections/fxmonitor-breaches/changeset?_expected=1598625907365
Method	GET

Evidence	23927853
URL	https://firefox.settings.services.mozilla.com/v1/buckets/main/collections/fxmonitor-breaches/changeset?_expected=1598625907365
Method	GET
Evidence	12218087
URL	https://firefox.settings.services.mozilla.com/v1/buckets/main/collections/fxmonitor-breaches/changeset?_expected=1598625907365
Method	GET
Evidence	24853850
URL	https://firefox.settings.services.mozilla.com/v1/buckets/main/collections/fxmonitor-breaches/changeset?_expected=1598625907365
Method	GET
Evidence	43423561
URL	https://firefox.settings.services.mozilla.com/v1/buckets/main/collections/fxmonitor-breaches/changeset?_expected=1598625907365
Method	GET
Evidence	29396116
URL	https://firefox.settings.services.mozilla.com/v1/buckets/main/collections/fxmonitor-breaches/changeset?_expected=1598625907365

Method	GET
Evidence	137272116
URL	https://firefox.settings.services.mozilla.com/v1/buckets/main/collections/fxmonitor-breaches/changeset?_expected=1598625907365
Method	GET
Evidence	58843488
URL	https://firefox.settings.services.mozilla.com/v1/buckets/main/collections/fxmonitor-breaches/changeset?_expected=1598625907365
Method	GET
Evidence	13338833
URL	https://firefox.settings.services.mozilla.com/v1/buckets/main/collections/fxmonitor-breaches/changeset?_expected=1598625907365
Method	GET
Evidence	26892897
URL	https://firefox.settings.services.mozilla.com/v1/buckets/monitor/collections/changes/records
Method	GET
Evidence	10583668

URL	https://firefox.settings.services.mozilla.com/v1/buckets/blocklists/collections/addons-bloomfilters/changeset?_expected=1604515098569&_since=%221602268629760%22
Method	GET
Evidence	30315127
URL	https://firefox.settings.services.mozilla.com/v1/buckets/main/collections/fxmonitor-breaches/changeset?_expected=1598625907365
Method	GET
Evidence	28364826
URL	https://firefox.settings.services.mozilla.com/v1/buckets/main/collections/fxmonitor-breaches/changeset?_expected=1598625907365
Method	GET
Evidence	68648009
URL	https://firefox.settings.services.mozilla.com/v1/buckets/main/collections/cfr/changeset?_expected=1603719109121
Method	GET
Evidence	537001984
URL	https://firefox.settings.services.mozilla.com/v1/buckets/main/collections/fxmonitor-breaches/changeset?_expected=1598625907365
Method	GET

Evidence	37217682
URL	https://firefox.settings.services.mozilla.com/v1/buckets/main/collections/fxmonitor-breaches/changeset?_expected=1598625907365
Method	GET
Evidence	359420698
Instances	96
Solution	Manually confirm that the timestamp data is not sensitive, and that the data cannot be aggregated to disclose exploitable patterns.
Other information	49038354, which evaluates to: 1971-07-22 08:45:54

Reference	http://projects.webappsec.org/w/page/13246936/Information%20Leakage
CWE Id	200
WASC Id	13
Source ID	3

ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO

DIRECCIÓN DE BIBLIOTECAS Y RECURSOS DEL APRENDIZAJE

DBRAI

UNIDAD DE PROCESOS TÉCNICOS

REVISIÓN DE NORMAS TÉCNICAS, RESUMEN Y BIBLIOGRAFÍA

Fecha de entrega: *10 / 09 / 2021*

INFORMACIÓN DEL AUTOR/A (S)
Nombres – Apellidos: *Guillermo Valencia Petroche*
INFORMACIÓN INSTITUCIONAL
Instituto de Posgrado y Educación Continua
Título a optar: *Magister en Seguridad Telemática*
f. Analista de Biblioteca responsable: *Lic. Luis Caminos Vargas Mgs.*

LUIS ALBERTO CAMINOS VARGAS

Firmado digitalmente porLUIS ALBERTO CAMINOS VARGAS
DN: cn=LUIS ALBERTO CAMINOS VARGAS c=EC l=RIOBAMBA
Motivo:Soy el autor de este documento
Ubicaci- n:
Fecha:2021-09-10 17:12-05:00

UNIDAD DE PROCESOS TÉCNICOS DBRAI ESPOCH