Seguridad con llave público -privada para consultas GrapHQL vía WebService RestFull

Edwar Alonso Rojas Blanco, edwar.red@gmail.com, edrojasb@unal.edu.co, Universidad Nacional de Colombia – Maestría en ingeniería de sistemas y computación - Cyberseguridad – 2019 -I

***Resumen*—Los servicios web tipo RestFull son muy utilizados en actualidad por su sencillez de implementación y despliegue. Cada desarrollador establece los parámetros que quiere recibir y la forma en que retorna la información, sin embargo, a veces resulta difícil y costoso mantener estos proyectos pues existen muchos "endpoint" que tienen diferentes comportamientos; GrpaHQL es un estandar desarrollado por Facebook y una comunidad que consisten en tener un sólo un 'endpoint' y una estructura bien definida de los RestFull services. Cifrar las consultas grapHQL es una de las estrategias que se utilizan para garantizar la seguridad de los webservices Restfull.**

I. Introducción

Este documento tiene como propósito mostrar una estrategia para brindar seguridad desde la capa de aplicación a la exposición de webservices tipo rest encriptando los mensajes mediante llave publica-privata con algoritmo RSA. Como se utiliza el lenguaje GraphQL solo tenemos un "endpoint" expuesto para realizar las operaciones.

II. Rest Full Service

REST (Representational State Transfer), es uno de los principales autores del Protocolo de transferencia de hipertexto (HTTP) versiones.

Debido a que la tecnología REST utiliza HTTP, esto da la facilidad de que pueda ser utilizada prácticamente por cualquier lenguaje de programación y que sea fácil de testear, además es un requisito de un servicio REST que el cliente y el servidor sean independientes entre sí[1].

Este estilo de trabajo es un enfoque de las comunicaciones que se utiliza a menudo en el desarrollo de servicios Web. El uso de REST es mayormente preferida que con el estilo SOAP (Simple Object Access Protocol) que en comparación es más pesado, porque REST no aprovecha tanto ancho de banda, lo que hace que sea un mejor ajuste para su uso a través de Internet.

La arquitectura REST donde las comunicaciones son más ligeras entre productor y consumidor, mantenerles y escalables, hacen de REST un estilo de construcción popular para APIs basadas en la nube, como las proporcionadas por Amazon, Microsoft y Google. Cuando los servicios Web utilizan la arquitectura REST, se denominan API RESTful (Interfaces de programación de aplicaciones) o API REST.

REST se utiliza a menudo en aplicaciones móviles, sitios Web de redes sociales, procesos empresariales automatizados, entre otros. El estilo REST hace énfasis en que las interacciones entre los clientes y los servicios se mejoran al tener un número limitado de operaciones (verbos). La flexibilidad se obtiene asignando recursos a sus propios identificadores de recursos universales únicos (URI). Debido a que cada verbo tiene un significado específico (GET, POST, PUT y DELETE), REST evita la ambigüedad.

- ✓ Se usa GET para obtener un recurso
- ✓ Se usa POST para crear un recurso en el servidor
- ✓ Se usa PUT para cambiar el estado de un recurso o actualizarlo
- ✓ Se usa DELETE para eliminar un recurso

III. GraphQL

"GraphQL es un protocolo de consulta de datos originalmente elaborado por Facebook para uso interno. Desde 2015 existe en una versión open source que ha impulsado su desarrollo y que ha generado muchas ventajas y beneficios prácticos.

Si está familiarizado con el funcionamiento de las aplicaciones, sabrá que los protocolos de consulta de datos son esenciales para obtener toda clase de reportes e información. REST es el mecanismo más utilizado, pero ahora GraphQL ha surgido como una alternativa que pretende adaptarse mejor a las necesidades actuales.

El objetivo esencial de GraphQL es ofrecer a los clientes una forma más directa, sencilla y eficiente para obtener exactamente los datos que requieren, a través de un protocolo potente y dinámico. [2].

Para entender el funcionamiento de GraphQL, es necesario explicar los tres componentes elementales que caracterizan al proyecto de código abierto:

Lenguaje de consulta: en primer lugar, el concepto GraphQL describe un lenguaje de consulta (Query Language) que facilita a los programas el acceso a una API. Mientras otras arquitecturas de interfaz solo permiten consultas estrictas que a menudo solo garantizan acceder a un solo recurso, las consultas GraphQL se distinguen por una gran flexibilidad. Esta se demuestra, en concreto, por el hecho de no existir un límite para el número de recursos consultados y porque se pueden definir exactamente los campos de datos que se van a consultar. GraphQL permite tanto consultas de lectura como de escritura.

Sistema de tipos: GraphQL trabaja con un sistema propio de tipos que permite describir a una API con tipos de datos. Las

[1*] Revista Argentina de Trabajos Estudiantiles. Patrocinada por la IEEE.

estructuras de datos definidas de este modo son las que crean el marco para las consultas. Cada tipo consta de uno o varios campos que, a su vez, contienen sus propios tipos de datos. Este sistema individual sirve a GraphQL como punto de orientación para validar consultas y poder rechazar las erróneas.

Entorno de tiempo de ejecución: por último, GraphQL también ofrece distintos entornos de tiempo de ejecución para servidores para llevar a cabo consultas GraphQL. Este es el propósito de las bibliotecas disponibles para distintos lenguajes de programación, por ejemplo, Go, Java, JavaScript, PHP, Python o Ruby, que otorgan una gran libertad al usuario a la hora de elegir el lenguaje de su API

GraphQL. No obstante, el entorno de tiempo de ejecución es exclusivamente responsable de la conversión (análisis) y validación de las consultas, además de la serialización de las respuestas (conversión del objeto en un orden correspondiente de bits). Guardar y calcular los datos (por ejemplo, en un banco de datos) corresponde al ámbito de funciones de tu aplicación web.

La interacción del lenguaje de consulta, el sistema de tipos y el entorno de tiempo de ejecución genera una estructura API extremadamente versátil no solo para cualquier plataforma o aplicación, sino que también puede ajustarse perfectamente a las propiedades de tu aplicación web: puedes integrar la interfaz de GraphQL sin problemas en el código de tu proyecto, independientemente del marco de trabajo que utilices, sea Django, Rails o Node.js “ [3].

IV. Encriptación Asíncrona con RSA

Este método se basa en la dificultad de hallar los factores primos de un número n que sea producto de dos primos muy grandes. En principio es lo mismo tener los dos números primos que tener su producto: dados dos números es fácil multiplicarlos, y dado un número es teóricamente posible factorizarlo en producto de números primos. Pero cuando hablamos de un número de tamaño 1024bits (309 cifras en sistema decimal), intentar factorizarlo es computacionalmente impracticable.

Los ordenadores encuentran con rapidez números primos grandes y también pueden multiplicarlos en una milésima de segundo para obtener un resultado de doscientas cifras. Por ejemplo, esto se hace cada vez que se visita una página segura, el navegador genera sobre la marcha una nueva clave de usar y tirar. Pero, ¿cuánto cuesta descomponer en factores primos un número con doscientas cifras? Es prácticamente imposible, esto es lo que garantiza la seguridad del sistema.

El RSA es un algoritmo asimétrico que cifra en bloque y que utiliza una clave distribuida públicamente y otra privada guardada en secreto por su propietario. Si dos personas A y B se quieren comunicar con este método es el receptor B quien tiene toda la clave; el emisor A conoce la parte pública de la clave, que sirve para cifrar mensaje. El receptor guarda cuidadosamente la parte privada de la clave, que sirve para descifrar [4].

V. DESARROLLO DEL SERVIDOR

Se desplegó un servicio REST en una aplicación web con un único “endpoint” en el que se envía como parámetro la consulta en graphQL.

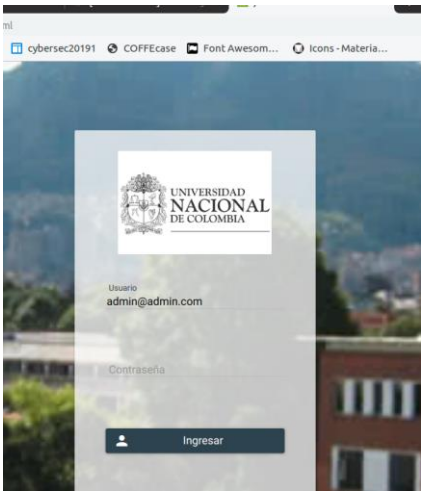

Fig 1. Pantalla de ingreso

La aplicación que despliega el servicio web tiene las siguientes opciones:

- **Generar Llaves:** Genera los archivos que contiene las llaves pública y privada.
- **Desactivar seguridad:** Sirve para probar que pasa con el servicio cuando no se valida la seguridad.

Si la seguridad está desactivada el servicio Rest responde a peticiones de cualquier origen, como la siguiente:

http://localhost:8080/tncity-war/webresources/generic/graphql/**query{listPerson{idpersona,nombres,apellidos,email,numdocumento}}**

Esto traerìa la lista de todas las personas registradas en el sistema. Lo que està en negrilla es la consulta graphQL

Si la seguridad se activa, la petición deberá hacerla parecido a lo siguiente:

http://dev.sobre5.com:8080/tncity-war/webresources/generic/graphql/**B7ZbuGYzww8Ta7bqPKgtt%2B3GMZVcT3cII%2FZoGOpuuBweQ2ESyABrqB%2BqkLQtoRkhhtLeqSkRs8ThMQUuvU42otGPKAYu6UOJWR9JgtybDm41r00P03OZS2WBh%2BJS5zmaRdQ5libd53ITjPsRh95vUn6r1FDmsAQilSr2Mm%2BgSyE%3D**

En donde la parte de la consulta va cifrada con llave privada aplicando RSA y posteriormente codificada en base64 para poder ponerla en la URL.

VI. DESARROLLO DEL CLIENTE

Cuando el servidor tiene la seguridad activada, se desarrolló un cliente StandAlone que permite cifrar el mensaje de modo que pueda descifrarse en el servidor con la llave pública

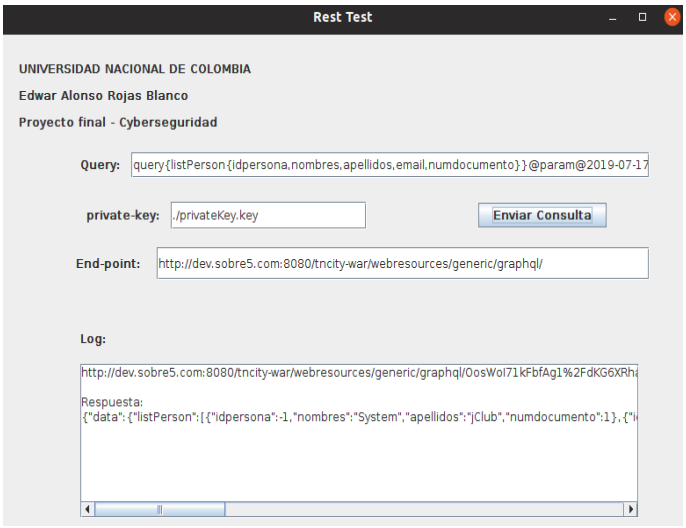

Fig 2. Programa cliente que aplica el cifrado.

El programa cifra la consulta con la llave privada y agrega una estampa de tiempo para que el valor enviado sólo pueda utilizarse una vez, y para hacer más difíciles los análisis de frecuencias.

VII. CONCLUSIONES

- Los servicios REST son sencillos de implementar y de consumir, sin embargo, deben aplicárseles mecanismos de seguridad pues podrían constituir brechas de seguridad.
- El lenguaje graphQL facilita el mantenimiento de los RestService pues centraliza el punto de acceso.
- Una buena estrategia de seguridad es cifrar los parámetros de los RestService, sin embargo, debe aplicarse variabilidad por ejemplo con estampa de tiempo para evitar violaciones en el acceso mediante análisis de frecuencias.

VIII. REFERENCIAS

[1] RestFULL Web Services, Leonar Richardson & Sam Ruby, ISBN: 9780596529260 , 2007.
[2] GraphQL, https://www.aplyca.com/es/blog/graphql, viditado, 10 julio de 2019.
[3] GraphQL, Official Site, https://graphql.org/ , Visitado 12 Julio de 2019.
[4] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. Introduction to Algorithms, Second Edition. MIT Press and McGraw-Hill, 2001.